THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS

ANALOG COMPUTING

T.M.

JULY 1988
ISSUE 62

U.S.A. $3.50
CANADA $4.75

Telecommunications Coverage

XE Banks

Boot Camp
End User
Database Delphi
Panak Strikes!

When you want to talk Atari

XL/XE HARDWARE

INTERFACES

ICD
P:R Connection 59.99
Printer Connection 39.99
Supra
1150 . . . . . 38.99
1151 (1200 XL) 41.99
Xetec
Graphix Interface 38.99
Atari
850 Interface 109.00

COMPUTERS

Atari 130XE $135

Atari
65 XE 99.99

XL/XE ENHANCEMENTS

Axlon 32K Mem. Board (400/800) . 19.99
Atari 80 Column Card 74.99

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 89.99
2400 . . . . . 179.00
Supra
2400 Baud XL, XE 169.00
2400 Baud ST 169.00
2400 Baud (no software) 149.00

MONITORS

Magnavox
CM8502 13" Comp. & cables . . . 179.00

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 969.00
1040 Monchrome System 789.00
1040 Computer (no monitor) . . . 649.00
520ST FM Monochrome System (Includes: 520 ST, internal drive, modulator, mouse, Basic and monochrome monitor) . . . . . Call
SM124 Monochrome Monitor . . 169.00
SM1224 Color Monitor 329.00

DRIVES

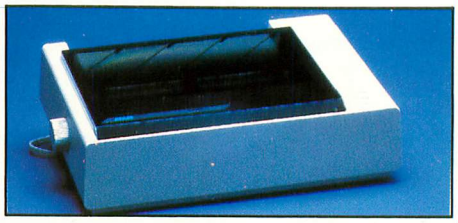
Atari
XF551 Drive (XL/XE) 189.00
AA314 DS/DD Disk (ST) 209.00
AA354 SS/DD Disk (ST) 129.00

Atari SHD204 20 Meg for ST $559

I.B.
5¼ ST Drive 209.00
Indus
GTS 100 3½" Drive (ST) 189.00
GT Drive (XL/XE) 189.00
Supra
20 Meg Hard Drive (XL/XE) 649.00
20 Meg Hard Drive (ST) 559.00
30 Meg Hard Drive (ST) 689.00

PRINTERS

Atari
1020 XL/XE Plotter 35.99



Atari XM-M801 XL/XE Dot Matrix $189

XM-M804 ST Dot Matrix 184.00
XDM 121 Letter Qlty. XL/XE . . . 199.00

Brother
M-1109 100 cps Dot Matrix 179.00
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 269.00
FX-86E 240 cps, 80 col. . . . . Call
FX-286E 240 cps, 132 col. . . . . Call
LQ-500 180 cps, 24-wire . . . . . Call
LQ-850 330 cps, 80 col. . . . . Call
EX-800 300 cps, 80 col. . . . . Call

NEC
P2200 pinwriter 24-wire 369.00
P660 pinwriter 24-wire 459.00
P760 pinwriter 132 col. 649.00

Okidata
Okimate 20 color printer 129.00
ML-182 120 cps, 80 column 239.00
ML-192+ 200 cps, 80 column . . 359.00
ML-193+ 200 cps, 132 col. 459.00

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 . . . 179.00
NX-15 120 cps, 132 column 319.00

Toshiba
P321-SL 216 cps, 24-wire 499.00

WE SHIP 90% OF ALL ORDERS WITHIN 24 HOURS

SELECT FROM OVER 3000 PRODUCTS

COMPUTER MAIL ORDER

CIRCLE #101 ON READER SERVICE CARD.

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

XL/XE SOFTWARE

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

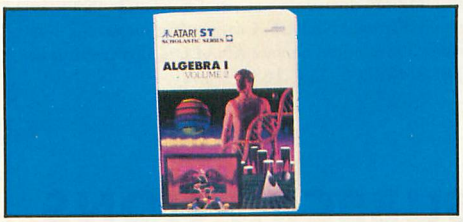
ACCESS
Leaderboard Golf **$13.99**

Atari Program Exchange
Misc. Programs (cassettes) .... at 1.99
Broderbund
Printshop 25.99
Karateka 14.99
Cygnus
Starfleet I 32.99
Datasoft
Alternate Reality (dungeon) 25.99
221 Baker St. 21.99
Electronic Arts
Pinball Construction 9.99
Firebird
Silicon Dreams 21.99
Jewels of Darkness 21.99
Microprose
Top Gunner 14.99
F-15 Eagle Strike 21.99
Silent Service 21.99
Origin Systems
Ultima 4 ..... 36.99
Roklyn SPECIAL
Anti-Sub/Journey to Planet .. (ea.) 3.99
Strategic Simulations
Gemstone Warrior 11.99
Sublogic
Flight Simulator II 31.99
Scenery CALIF. 14.99
X-Lent
Typesetter 22.99
Printshop Interface 21.99

ACCESSORIES

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 29.99
Multi I/O Board 256K 169.00
Multi I/O Board 1 Meg 299.00
US Doubler 28.99

ST SOFTWARE

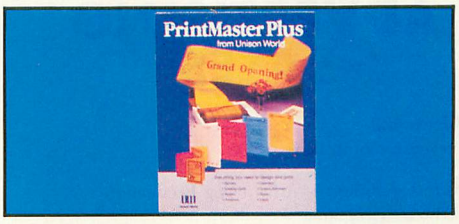


ATARI
Algebra I Vol. 1
GDE 5 **$16.99**

Access
Leaderboard Golf 22.99
Activision
Hacker II Doomsday 29.99
Antic
Stereo CAD 3-D 54.99
Avant Garde
PC Ditto 64.99
Batteries Included
Degas Elite 38.99

ST SOFTWARE

Comnet
ST Term 21.99
Dac
Easy Accounting 59.99
Electronic Arts
Gridiron Fooball/Auto Duel.(ea.) .30.99
Isgur Portfolio 129.00
Firebird
Jewels of Darkness 21.99
Infocom
Beyond Zork 34.99
Michtron
Major Motion 25.99
Microprose
Gunship 32.99
F-15 Strike Eagle 24.99
Miles Software
ST Wars 25.99
Mark Williams
C 119.00
Paradox
Wanderer (3D) 25.99
War Zone/Fireblaster 26.99
Psygnosis
Barbarian/Deep Space ..... (ea.) 26.99
Soft Logik Corp.
Publishing Partner 54.99
Strategic Simulations
Rings of Zilfin 23.99
Sublogic
Flight Simulator II 32.99
Timeworks
Swiftcalc/Wordwriter ..... (ea.) 47.99
Partner ST 37.99



UNISON WORLD
Printmaster Plus **$27.99**

Word Perfect Corp
Word Perfect 4.1 ..... 189.00

In the U.S.A. and in Canada

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

Outside the U.S.A. call 717-327-9575, Fax 717-327-1217

Educational, Governmental and Corporate Organizations call toll-free 1-800-221-4283

CMO. 101 Reichard Ave., 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. We are not responsible for typographic or photographic errors.

B707

ANALOG COMPUTING

WHERE TO WRITE

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

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

PERMISSIONS

ADVERTISING SALES

Address all advertising materials to:

Janice Rosenblum — Advertising Production
ANALOG Computing
9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

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**

SUBSCRIPTIONS

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

AUTHORS

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

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

ANALOG COMPUTING STAFF

Publisher
LEE H. PAPPAS

Executive Editor
CLAYTON WALNUM

Art Director
E. HERCH

Managing Editor
DEAN BRIERLY

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
CHARLES F. JOHNSON

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

Entertainment Editors
ARNIE KATZ; BILL KUNKEL; JOYCE WORLEY

Cover Illustration
TIM ALT FOR DIGITAL ART

Illustrations
DAVID MARTIN

Copy Chief
KATRINA VEIT

Copy Editors
SARA BELLUM
ANNE DENBOK
PAT ROMERO

Typographers
DAVID BUCHANAN; KLARISSA CURTIS; JUDY VILLANUEVA

Contributers
BILL BODENSTEIN; FRANK COHEN; GREG KNAUSS; MARIAN LORENZ; ALLAN MOOSE; KEVIN PECK; DAVID PLOTKIN; MICHAEL SCHOENBACH; DAVID SCOFIELD

Production Director
DONNA HAHNER

Production Assistant
STEVE HOPKINS

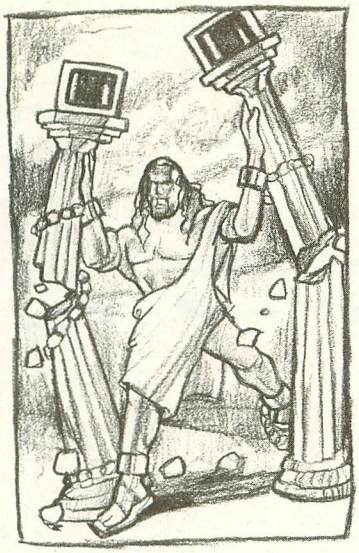
National Advertising Director
JE PUBLISHERS REPRESENTATIVE
(213) 467-2266

Advertising Production Director
JANICE ROSENBLUM

Advertising Manager
PAULA THORNTON

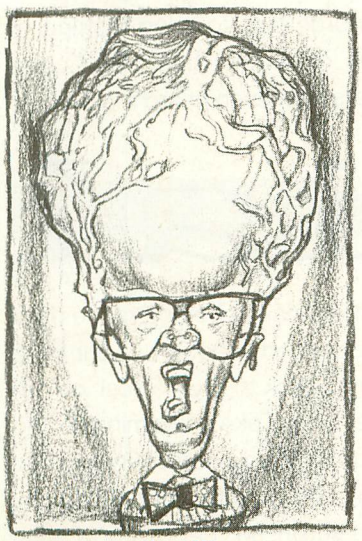
Subscriptions Director
IRENE GRADSTEIN

Vice-President Sales
JAMES GUSTAFSON



Page 6.

Page 20.



Page 23.

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.

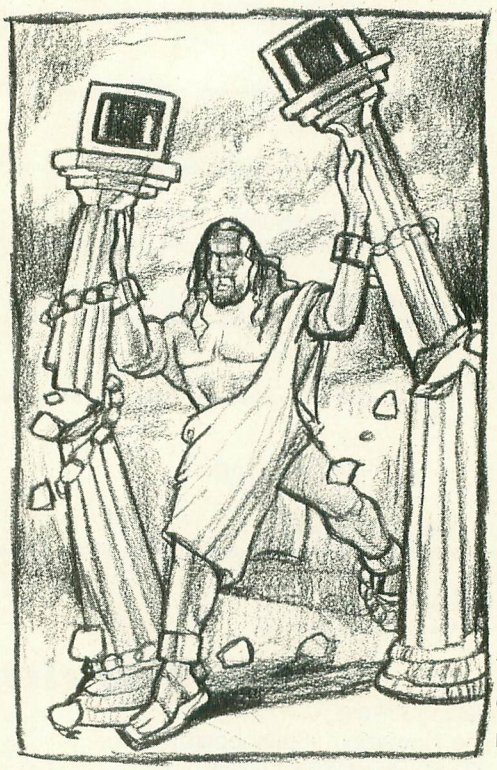
U.S. subscription: $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscription: Add $7 per year. Single copy $3.50 (add $1 for postage). Change of address: Six weeks advance notice, and both old and new addresses are needed. POSTMASTER: Send change 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.

by Lee Pappas

Editorial

▽ ▽ ▽ ▽ ▽ ▽

"What's new in the 8-bit world?"



Nothing. I have seen so few product releases for the "little" Atari it's amazing. Sure, there's some new software from time to time, but mostly from small companies (more often than not one-man operations . . . real Atari enthusiasts). The big software institutions have totally forgotten about this market of ours . . . No, I take that back: Springboard recently released Newsroom for the offspring of the ol' 400/800 line — and it's a great program. Of course, Atari is busy, producing game cartridges for the XE game machine. Stuff like Blue Max. Remember that game, originally produced in disk format by Synapse? What, maybe four years ago or so? Of course, I don't mean to knock the XEGS, this 8-bit offshoot; it's probably the last hope for our aging 6502 machines—what with most of the great 8-bit programmers having moved to the ST or whatever. What really puzzles me is I KNOW there are tens of thousands of do-or-die 8-bitters out there. The fact that we sell a zillion copies of ANALOG Computing every month proves that. So, what are you doing with your 400? Or 800? Or XL or XE? It can't just be sitting in the closet or you wouldn't be reading this right now. Are you playing games? Are you still using your computer for word processing? Telecommunicating? Or (gasp, dare I say it?), are you using your wonderful little machine for business? The readers of this magazine average in the mid-30s age-wise. That's a fairly mature group to be just playing games. No. You're out there using your computer for productive causes. And that leads me to the question . . . WHAT? I would love to know what you loyal Atari users are doing out there. It's a known fact that Atari computer owners are one of the most (if not THE most) loyal groups around in regards to supporting their machine. To an Atarian the common phrase "but I thought Atari just makes games" is fightin' words sure to bring a string of defensive remarks. I heard that comment years ago, even heard it last week. The 8-bit line has to be one of the most least understood products of our time. But getting back to my interest in Atari computer uses. Take a few minutes and write in. Even if you don't think you do anything special with your 8-bit, you might be surprised. Let us know what your computer does for you. We'll publish the most interesting responses . . . and you and your computer will be famous! A

CIRCLE #102 ON READER SERVICE CARD.

CIRCLE #103 ON READER SERVICE CARD.

CIRCLE #104 ON READER SERVICE CARD.

Utility

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 appolgies to those with cassette systems.

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

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

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

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

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

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

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

When you finish a line, M/L Editor will compare the entries' 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).

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),N$(4),A$(1),B$(1),F$(15)
   ,F1$(15)
LF 11 DIM MOD$(4)
BN 20 LINE=1000:RETRN=155:BACK5P=126:CHK5
   UM=0:EDIT=0
GO 30 GO5UB 450:POSITION 10,6:? "Start or
    Continue? ";:GO5UB 500:? CHR$(A)
ZG 40 POSITION 10,8:? "FILENAME";:INPUT F
   $:POKE 752,1:? " "
FE 50 IF LEN(F$)<3 THEN POSITION 20,10:?
   "  ";:GOTO 40
NF 60 IF F$(1,2)<>"D:" THEN F1$="D:":F1$(
   3)=F$:GOTO 80
KL 70 F1$=F$
TN 80 IF CHR$(A)="5" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$:TRAP 110
HQ 100 FOR X=1 TO 16:GET #2,A:NEXT X:LINE
   =LINE+10:GOTO 100
WH 110 CL0SE #2:OPEN #2,9,0,F1$:GOTO 170
VT 120 TRAP 160:OPEN #2,4,0,F1$:GO5UB 440
   :POSITION 10,10:? "FILE ALREADY EXI5T5
   !!":POKE 752,0
ZU 130 POSITION 10,12:? "ERA5E IT? ";:GO5
   UB 500:POKE 752,1:? CHR$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
    CL0SE #2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
   HEN 130
BH 160 CL0SE #2:OPEN #2,8,0,F1$
IE 170 GO5UB 450:POSITION 10,1:? "NOW ON
   LINE: ";LINE:CHK5UM=0
GH 180 L1=3:FOR X=1 TO 16:POSITION 13*(X<
   10)+12*(X>9),X+2:POKE 752,0:? "BYTE #"
   ;X;": ";:GO5UB 310
KH 190 IF EDIT AND L=0 THEN BYTE=BF(X):GO
   TO 210
FY 200 BYTE=VAL(N$)
OZ 201 MOD$=N$
BU 210 POSITION 22,X+2:? BYTE;"  "
YZ 220 BF(X)=BYTE:CHK5UM=CHK5UM+BYTE*X:IF
    CHK5UM>9999 THEN CHK5UM=CHK5UM-10000
M5 230 NEXT X:CHK5UM=CHK5UM+LINE:IF CHK5U
   M>9999 THEN CHK5UM=CHK5UM-10000
IG 240 POSITION 12,X+2:POKE 752,0:? "CHEC
   K5UM: ";:L1=4:GO5UB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
5Y 270 POSITION 22,X+2:? C;"   "
IL 280 IF C=CHK5UM THEN 300
DI 290 GO5UB 440:EDIT=1:CHK5UM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GO5UB 500:IF (A=A5C("Q") OR A=A5C(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACK5P AND (A<4
   8 OR A>57) THEN 320
DX 331 IF A=RETRN AND N$="" THEN N$=MOD$
TD 335 IF A=RETRN AND L=0 AND X>1 THEN 35
   0
JR 340 IF ((A=RETRN AND  NOT EDIT) OR A=B
   ACK5P) AND L=0 THEN 320
DW 350 IF A=RETRN THEN POKE 752,1:? " ":R
   ETURN
GG 360 IF A<>BACK5P THEN 400
5A 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
A5 380 N$=""
RE 390 ? CHR$(BACK5P);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WX 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHIC5 0:END
YT 430 GO5UB 440:POSITION 10,10:? "NO 5UC
   H FILE!":FOR X=1 TO 1000:NEXT X:CL0SE
   #2:GOTO 30
FD 440 POKE 710,48:50UND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:50UND 0,0,0,0:RETURN
MY 450 GRAPHIC5 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 5TEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 5TEP 2:POKE DL+X,0
   :NEXT X
ZW 480 POKE DL+41,65:POKE DL+42,PEEK(560)
   :POKE DL+43,PEEK(561):POKE 87,0
AC 490 POSITION 2,0:? "analog ml editor":
   POKE 559,34:RETURN
WZ 500 OPEN #1,4,0,"K:":GET #1,A:CL0SE #1
   :RETURN
```

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

HARD TO FIND SOFTWARE TITLES

SPINNAKER SOFTWARE - $9.95 EACH
(all titles of cartridge)

ADVENTURE CREATOR (age 8-adult)
Children use their own imaginations to create their own adventures while learning about computer programming design and logic.

ALF IN THE COLOR CAVES (age 3-6)
As children maneuver Alf through the mazes they improve their routing and predictions skills and it helps them to understand cause-and-effect relationships.

ALPHABET ZOO (ages 3-8)
Strengthens letter recognition skills as children associate letters of the alphabet with the sounds they represent.

DELTA DRAWING (age 4-adult)
An easy and friendly introduction to computer learning.Builds an understanding of procedural thinking and encourages children to exercise their creativity.

FACEMAKER (age 3-8)
Helps children become comfortable with computer fundamentals such as: menus, cursors, graphics, and improves memory and concentration.

KINDERCOMP (ages 3-8)
Helps children improve their reading readiness and counting skill and familiarizes them with the keyboard.

KIDS ON KEYS (age 3-9)
Familiarizes children with the computer keyboard as they learn to identify numbers, letters, and words and associate them with images on the screen.

STORY MACHINE (age 5-9) *Won't work on XL/XE*
Helps children learn to write correctly and develop a positive attitude towards writing.

HAYDEN SOFTWARE - $9.95 EACH
(all titles on disk)

DON'T SHOOT THAT WORD (all ages)
A fun way to practise recognizing words by associating them with pictures.

MONKEY SEE, MONKEY SPELL (all ages)
Children will play for hours as they race the monkey up and down the Letter Tree to spell correctly the objects pictured at the bottom of the screen.

MICRO ADD/SUBTRACT (pre-school)
Colorful animation and familiar childhood tunes teach pre-schoolers the basic principles of addition and subtraction.

MICRO SUBTRACTION (pre-school)
An animated series of exercises to teach children number values and the basics of subtraction.

MICRO MULTIPLICATION (pre-school)
A colorful animated series of exercises to teach children number values and the basics of multiplication.

MICRO DIVISION (pre-school)
A colorful series of exercises to teach children number values and the basics of division.

MATCH UP! (pre-school)
Colorful pictures engage pre-schoolers in matching and sorting a series of familiar objects.

SHAPE UP! (pre-school)
Pleasing animation and amusing sound effects will delight your pre-schoolers as they learn shape discrimination.

FISHER PRICE SOFTWARE $8.95

MEMORY MANOR (on cartridge age 4-8)
Captivating pictures and sound help improve memory skills and concentration. (limited quantity)

THUNDER MOUNTAIN - $8.95 EACH OR ALL 6 FOR $39.95
(all titles on disk - ages 4-8)

ABC's WITH THE TINK TONKS
Five educational activities help children learn ABC order and become familiar with the computer keyboard. Offers multiple levels of difficulty.

COUNT AND ADD WITH THE TINK TONKS
An entertaining program that teaches basic math concepts, practice counting and presents simple problems to solve.

SUBTRACTION WITH THE TINK TONKS
Strengthen math skills by solving basic subtraction problems in context with multiple levels of difficulty. Lively animation and original music throughout.

SPELLING WITH THE TINK TONKS
An interactive story with five multilevel educational activities including spelling and vocabulary building.

BEING A SMART THINKER WITH THE TINK TONKS
Bold, high-resolution graphics help to strengthen concentration, memory, and visual discrimination skills.

DEVELOP THINKING SKILLS WITH THE TINK TONKS
Strengthens reasoning skills, logic, critical thinking, concentration, memory, and visual discrimination with multiple levels of difficulty.

SIERRA ON-LINE SOFTWARE

MR. COOL - $15.95 (cartridge - arcade)
With Mr. Cool you'll have a blast in the furnace, being cool as a cube, dodging fireballs, and side stepping springs. He'll hop into the hearts of game players young and old.

BC's QUEST FOR TIRES - $9.95 (disk - arcade)
This fast moving game features Thor zipping through the comic strip world of B.C. on his prehistoric unicycle in an effort to rescue Cute Chick from the hungry dinosaur.

CROSS FIRE - $9.95 (disk - arcade)
The aliens have you surrounded and laser shots fly from all directions. If you are to live, you'll have to concentrate on where the shots are coming from and where you're going so you don't get caught in the Cross Fire.

JAWBREAKER - $9.95 (disk - arcade)
This arcade game takes you to the candy store for a wild game of tag. If you can eat all the sweets, it's off to the store for another game. A merry maze chase game that will leave you hungry for more.

OIL'S WELL - $9.95 (disk - arcade)
You've hit oil and struck it rich but other oil barons have sabotaged your operation. Watch out for the steel-munching oozies and explosive land mines they've planted. Don't let them cheat you out of your millions!

SESAME STREET - $9.95 EACH
(all titles on disk)

ERNIE'S MAGIC SHAPES (pre-school)
Provides a playful setting featuring lively colorful graphics in which children practice visual discrimination.

BIG BIRDS SPECIAL DELIVERY (pre-school)
Improves a childs understanding of the attributes of form, class, and function while having fun!

ASTROGROVER (pre-school)
Offers experience with early number concepts in counting, adding and subtracting while having fun in outer space.

ADVENTURE INTERNATIONAL

SCOTT ADAM'S TEXT ADVENTURES - $14.95
Includes all twelve text adventure stories and hint book. (disk)

THE SORCERER OF CLAYMORGUE CASTLE - $8.95
(disk - graphics and text adventure)

SPIDERMAN - $7.95
(disk - graphics and text adventure)

THE HULK - $6.95
(disk - graphics and text adventure)

OLIN IN EMERALD - $9.95
(disk - educational graphics & text adv. age 7-14)

MIDWAY BATTLES - $7.95
(disk - simulation)

PREPPIE I and II $9.95
(disk - arcade includes both games)

STRATOS - $7.95 (disk - arcade)

C'EST LA VIE - $7.95 (disk - arcade)

REAR GUARD - $7.95 (disk - arcade)

TUTTI FRUTTI - $7.95 (disk - arcade)

RECONDITIONED HARDWARE
(30 DAY WARRANTY)

1030 MODEM WITH EXPRESS - $29.95

ATARI TRAK BALL - $9.95

CX-85 NUMERIC KEYPAD - $7.95
(includes handler disk)

ATARI 400 COMPUTER - $29.95

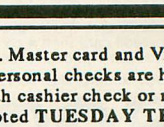
ATARI 600XL COMPUTER - $49.95

ATARI 800 COMPUTER (48K)
$69.95 B or $79.95 A

ATARI 810 DISK DRIVE
$110.00 B or $120.00 A
(B has some scratches, etc. - A is like new)

ATARI 850 INTERFACE - $89.95

ATARI 1020 COLOR PRINTER - $29.95
(40 column width)



SHIPPING INFORMATION: Prices do not include shipping and handdling - $5.00 minimum. Calif. residents include 7% sales tax. Master card and Visa accepted if your telephone number is listed in your local phone directory. Orders may be pre-paid with money order, ca shier 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 in cash. 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 - All items are fully warranteed - prices are subject to change. Phone orders accepted **TUESDAY THROUGH FRIDAY** from 10:00 am to 6:00 pm PST.

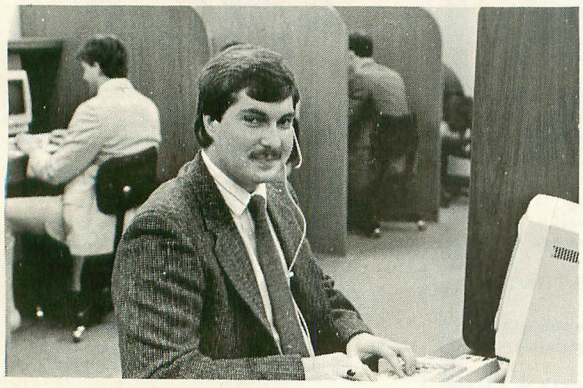
We carry a complete line of ATARI products and have a large public domain library for both 8-bit and ST computers.
Write or call for free catalogue.

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 $219.95
4160 CGA $254.95
4460 EGA $349.95
GB 200 Super Card $184.95
4570 ..... $CALL

Magnavox:
BM7652 ..... $79.95
BM7622 ..... $79.95
7BM-613 ..... $74.95
7BM-623 ..... $79.95
CM8502 ..... $199.95
CM8505 ..... $209.95
CM8562 ..... $239.95
CM8762 ..... $239.95
8CM-515 ..... $249.95

Blue Chip:
BCM 12" Green TTL $64.95
BCM 12" Amber TTL $69.95

NEC:
Multisync II $599.95

ATI Technologies:
Graphics solution $129.95
EGA Wonder $199.95
VIP $299.95

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:
Music Studio $19.95
Solid Gold Vol.#1 ..... $10.95

Batteries Included:
Paperclip 80 Col $31.95

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
'87 Team 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
Phantasie $22.95
Wargame Construc. $16.95
Wizards Crown $22.95
Phantasie II $22.95
Shiloh $22.95
Eternal Dagger $22.95

Sublogic:
Flight Simulator II $31.49

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

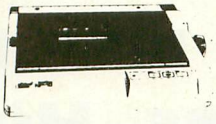
Tac 3 ..... $9.95
Tac 2 ..... $10.95
Tac 5 ..... $12.95
Tac 1 + IBM/AP $26.95
Economy $5.95
Slik Stick $6.95
Black Max $10.95
Boss $11.99
3-Way $19.99

1-800-233-8760

CIRCLE #106 ON READER SERVICE CARD.

star micronics · inc NX-1000

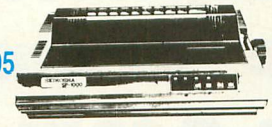
- 144 cps Draft
- 36 cps NLQ
- EZ Operation Front Panel Control



$179.95

SEIKOSHA Sp 180Ai

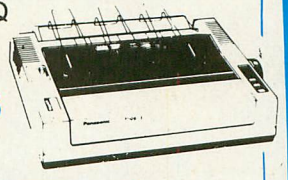
- 100 cps draft
- 20 cps NLQ



$129.95

Panasonic 1091 Model II

- 192 cps Draft
- 32 cps NLQ



$199.95

PRINTERS

star micronics · inc

NX-1000	$179.95
NX-1000C	$179.95
NX-1000 Color	$225.95
NX-1000C Color	$229.95
NX-15	$309.95
NR-10	$339.95
NR-15	$439.95
NB-15 24 Pin	$699.95
NB24-10 24 Pin	$425.95
NB24-15 24 Pin	$579.95
Laser 8	$CALL

BROTHER

M1109	$195.95
M1409	$299.95
M1509	$335.95
M1709	$475.95
Twinwriter 6 Dot & Daisy	$899.95
M1724L	$599.95
HR20	$339.95
HR40	$569.95
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
MP5420FA	$995.95
SP Series Ribbon	$7.95
SK3000 Ai	$339.95
SK3005 Ai	$419.95
SPB 10	$CALL
SL 130Ai	$599.95

Toshiba

321SL	$489.95
341 SL	$659.95
P351 Model II	$899.95
351 SX 400 cps	$1019.95

EPSON

LX800	$199.95
FX86E	$279.95
FX286E	$424.95
EX800	$399.95
LQ500	$339.95
LQ1000 w/Tractor	$549.95
LQ2500	$819.95
GQ3500	$LOW
LQ850	$525.95
LQ1050	$715.95

OKIDATA

Okimate 20	$119
Okimate 20 w/cart	$179.95
120	$189.95
180	$219.95
182	$209.95
182 +	$225.95
183	$249.95
192 +	$339.95
193 +	$449.95
292 w/interface	$449.95
293 w/interface	$585.95
294 w/interface	$819.95
393	$955.95

Panasonic

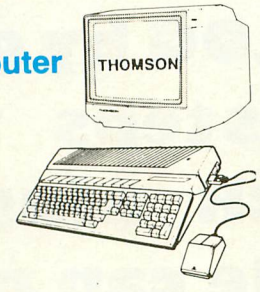
1080i Model II	$179.95
1091i Model II	$199.95
1092i	$319.95
1592	$409.95
1595	$459.95
3131	$299.95
3151	$479.95
KXP 4450 Laser	$CALL
1524 24 Pin	$559.95
Fax Partner	$589.95

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
Tribute 124	$469.95

ATARI 520 ST Computer

- Built-in Drive
- Thomson 4120 Monitor

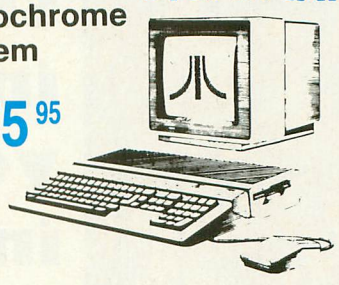


$729.95

ATARI HARDWARE

520 ST FM Mono	$675.95
520 ST FM Color	$729.95
1040 ST Mono	$739.95
1040 ST Color	$955.95
130XE Computer	$135.95
SX551 Drive	$174.95
SF 314 Disk Drive	$219.95
Indus GT Atari Drive	$169.95
SHD 204 20 MEG Drive	$579.95
XM301 Modem	$42.95
SX212 Modem	$89.95
GTS 100 (3.5" DSDD ST)	$195.95
GTS 1000 5¼ DSDD ST	$CALL

ATARI 520 ST-FM Monochrome System



$675.95

Internal drive included

ATARI 1040 ST Color System

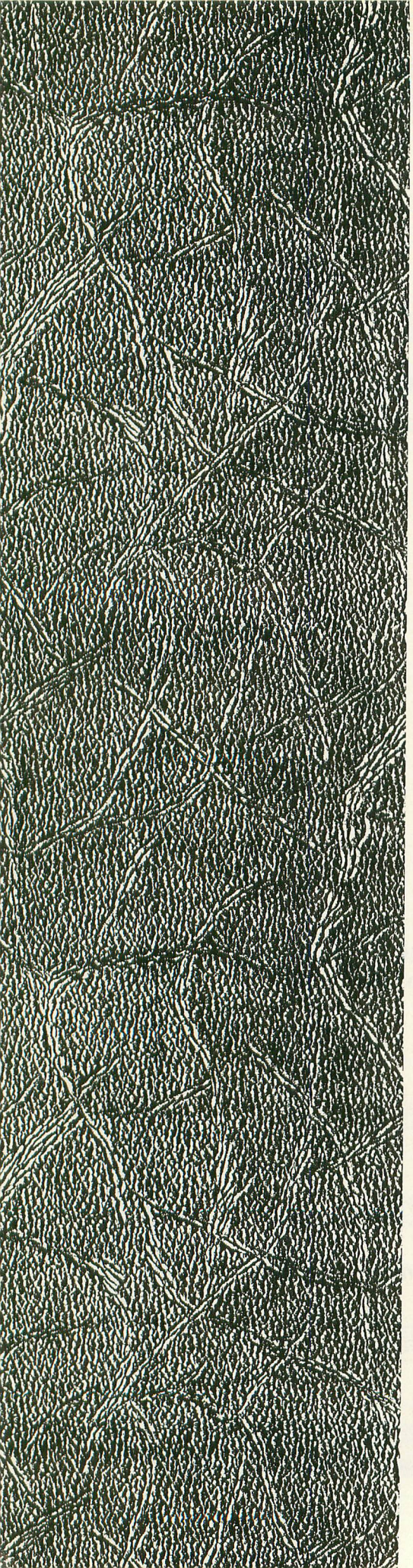
$955.95

Attention Educational Institutions:

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

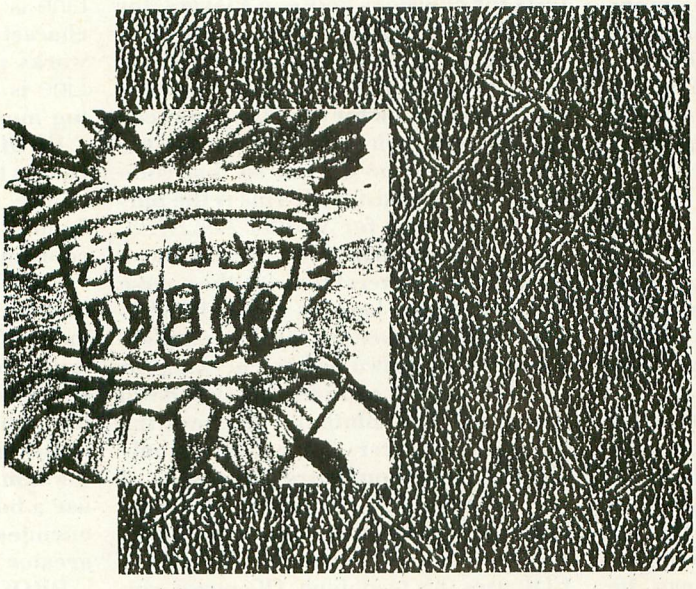
ATARI 1040 Monochrome System

$789.95



ANALOG Man is the editor of the famous ANALOG Computing Magazine, the premier magazine for Atari users. His job is to assemble the pages of each issue, which he does by running over the pages, causing them to fall to the level below in the girder-like offices of ANALOG Mag. You must help Man do his job of assembling nine issues of ANALOG by guiding his footsteps with your joystick plugged into port 1. He can climb up and down ladders, and falling down the holes left by runover pages doesn't hurt a bit...Man is tough.

48k disk or cassette



ANALOG Man

by David Plotkin

Of course, there is far more to it than just happily showing up at the office every day. The other personal computers are getting more and more nervous with the success of ANALOG and Atari, and they have decided the way to finish Atari for good is to prevent ANALOG from reaching its loyal readers. So one day, they showed up at ANALOG's offices and began chasing poor Man. Their touch deprives Man of one of his five lives. But Man is not defenseless. To combat the evils of the enemy personal computers, Man carries five bombs. Pressing the button on your joystick sets off a bomb, and any enemy who touches a bomb is instantly frozen and can do no further harm until he unfreezes.

There are nine different levels to ANALOG Man, and everything gets faster after you complete the first nine screens. Getting through all nine screens earns you two additional bombs, up to a maximum of ten. Oh, yes—the enemies stay frozen a shorter length of time in the upper levels . . . So get busy, loyal readers of ANALOG, and help ANALOG Man get the issues of your favorite magazine out on time.

Running ANALOG Man

ANALOG Man is too long to compile from memory. Punch it in exactly as listed (using D:CHECK IN ACTION! from issue 44 to check your typing), then save it to disk (using the SHIFT-CNTRL-W command). Go to the monitor (SHIFT-CNTRL-M) and reboot the system to clear memory (B). Reenter the monitor and type: C "D:FILENAME." When the compile is done, simply type R to run the program.

Program Take-apart

Some of the more interesting procedures are listed below, with a word of explanation on how they work. Much can be learned from studying the structured Action! listing.

PROC DOWNLOAD: The screens for this game are constructed using a redefined character set in Antic mode 4, the multicolored character mode. This procedure steps back the top of memory and moves the character set from ROM into RAM so it can be modified.

PROC DLINT: ANALOG Man uses a display list interrupt (DLI) to get extra color on the screen. The numbers with dollar signs in front of them are hex codes for the machine language equivalent of the commands to put the contents of the accumulator, X and Y registers on the stack and pop them back off. The balance of this procedure is simply to wait for the horizontal synch, then change the contents of the text window color register and the intensity of the text in the window.

PROC SCORELINE: Setting up the DLI defined in DLINT places the address of DLINT into the card variable Vdslst, which resides at locations 512 and 513. Whenever a DLI is required, the Atari checks the contents of these locations to find the address of the routine to execute for the DLI. It will now use Dlint. Byte array Dlist was "pointed" to the same place in memory as the display list, so changing one of the elements of Dlist will change the display list, thus calling the DLI at the required line. The DLI is actually turned on by placing hex $C0 into location NMIEN ($D40E).

PROC MOVEIT: Byte array Adres is pointed to the address defined by the PmAdr function, offset by the y coordinate of the Player in question. Then num bytes of array Shape are moved to this address using the built-in MOVEBLOCK command. Finally, the x coordinate of the Player is set by changing one of the elements of byte array PmHpos, which has been defined to reside at the memory locations that the Atari uses to set the horizontal locations of the Players ($D000).

PROC TESTCOL: This procedure tests for collisions between Players, for use in PROC PMHIT. Testing for collisions in a language as fast as Action! can be a little tricky. Whenever it becomes necessary to look for a collision between two Players, you must wait for the entire screen to be drawn, so that collisions will be registered. This is the purpose of waiting for Vcount AND *128*. The problem is that if you need to check for collisions several times in the course of one program loop, as you do in ANALOG Man, the waiting for the complete screen to be drawn before checking for the collision will considerably slow down the game. The solution is to check the hardware registers for collisions only *once* in each loop, store the results of the check in temporary holding registers, and use the temporary registers for all further work. TESTCOL uses this technique. Of course, you must clear the temporary registers before each collision check, and clear the hardware registers (PmHitClr = 1) after each check.

PROC TITLE: The rolling colors of the title screen are created by storing colors directly into the hardware color registers. The color to store is based on the timer located at memory register 20, which "ticks" every 1/60 of a second. Since 60 times per second is too fast to change the color (it doesn't look very nice), the number in the timer is divided by *4* (RSH 2). The result is then added to the scan line counter, Vcount, so that each scan line is a different color, and the rolling rainbow effect is based on the timer. By subtracting one of the two numbers generated by the above method from *128*, the colors of that register appear to roll backward. By avoiding the use of the DLI, you can have multiple colors within each letter—something most people will tell you can't be done on the Atari.

PROC GR4INIT: This procedure sets up the necessary information for use in the custom PLOT and LOCATE routines to come later. The elements of card array Linept are equated to the address of the beginning of each screen line. Then byte array Dlist is pointed to the Display list by equating Dlist to Sdlst, which is a card variable residing at locations 560 and 561, the registers which contain the address of the display list. Finally, the display list is modified to Antic mode 4 by changing the elements of Dlist.

PROC PLOT4: This is a custom PLOT routine, far faster than the one built into the Action! cartridge. Byte array Line is equated to an element of card array Linept. Then an element of Line is modified to place the required character on the screen. LOCATE4 works similarly, except the element of Line is simply returned instead of being modified.

PROC SQUASHED: This procedure checks to see if a falling level has hit one of the enemies. Note the conversion from Playfield coordinates to Player coordinates in order to do this check.

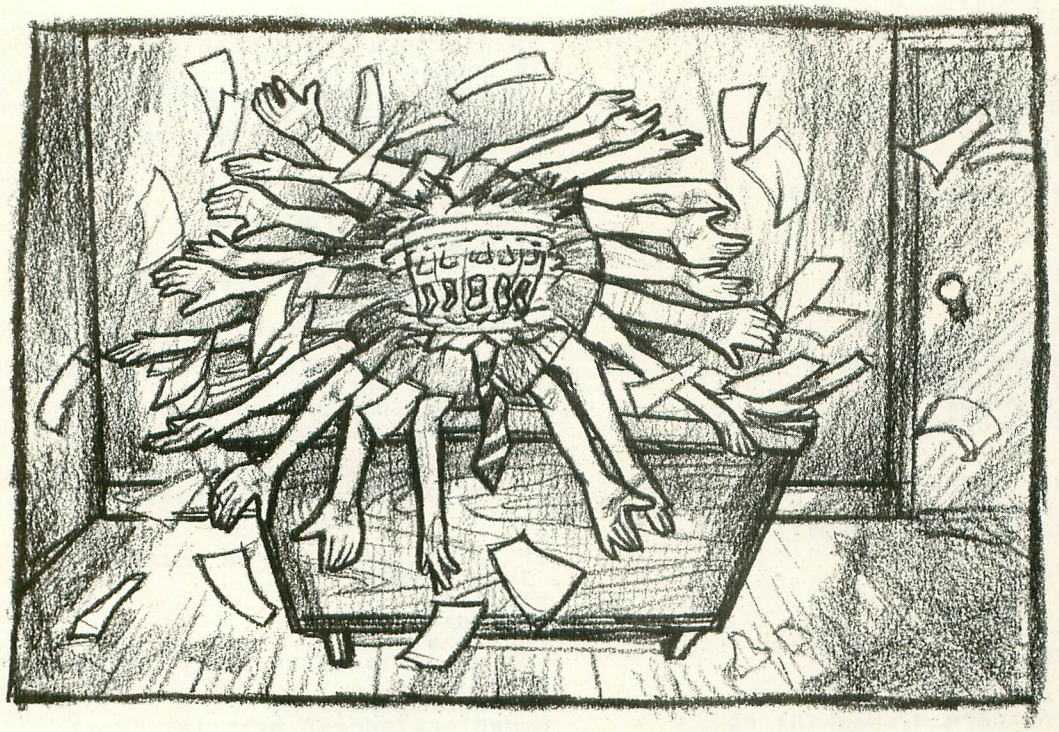
PROC NOCHASE: This procedure and PROC CHASE control the movement of the enemy Players. If the distance between ANALOG Man and his enemies is too great, they will not "see" him, and will move randomly. However, if they get close, they will begin to follow him, and the only escape may be to use a bomb. The distance at which the enemies will begin to follow Man gets greater as you get to higher levels.

PROC VECTOR: This procedure doesn't seem to do anything, since it contains nothing but a RETURN. In fact, it is very important in determining what level will appear on the screen. The problem that I faced was that if you get killed in the middle of a game (highly likely!), it is very unwieldy to get back to level 1 if you decide to play again.

In fact, the whole coding scheme was unwieldy, looking something like this: Screen1 (), Play (), Screen2 (), Play (), etc. . . Instead, the address of each procedure to draw a screen (Screen1, etc.) is stored into the elements of card array Sc in the last procedure of the program, Main. Then, Vector is simply equated to the appropriate element of Sc, so now Vector points to the procedure to draw a screen instead of to the dummy procedure that does nothing. Calling Vector now executes the procedure to draw a screen.

Summary

ANALOG Man is a rather long program, but it would have been considerably longer and more confusing if the powerful capabilities to relocate arrays and even procedures had not been used. I think you can see that Action! is one of the most powerful languages ever developed for *any* home computer. I recommend that if you are serious about your Atari, you support the developers of Action! and purchase a copy of this outstanding language.



Listing 1: Action!

```
; ANALOG MAN by David Plotkin
;
;       COPYRIGHT 1988
;     BY ANALOG COMPUTING
;
;        CHECKSUM DATA
;[56 57 F8 33 3E 56 EC CC
; B7 CD B7 6D FA 21 D4 D9
; 2F D7 9A 8B CB 11 75 44
; D9 E9 43 C6 BB EB D0 11
; CB 34 7F 69 98 7D C4 05
; 0D 2A 95 C7 8A 67 19 F1
; 1C 79 6F 6C BF C8 9D 92
; 44 BC CA 83 95 B9 61 0B
; 0C 40 E8 FA 15 63 C1 43
; 57 BE 36 37 69 D0 9F D6
; BD 22 31 70 06 33 3C 2A
; E4 2D 8D 6C 09 C9 73 ]

MODULE

BYTE ChrBase=756,Bkgrnd=710,X,Y,
     Fate=53770,Level=[5],CursIn=752,
     Stick0=632,Ps,Loud=[0],Indx=[0],
     Snd1=$D208,Snd2=$D20F,Freq=[169],
     Wsync=$D40A,Colbk=$D018,
     Nmien=$D40E,Consol=53279,
     Colints=$D017,X0,Y0,Ft=[200],
     Lv=[5],Ld=[0],Ld2=[0],Atrt=77,
     PmHitClr=$D01E,Dmactl=$22F,
     Gractl=$D01D,PmBase=$D407,
     Priority=$26F,Vcount=54283,
     Loud1=[0],Tone=[8],Flg=[1],
     Mstatus=[0],Pep=[5],My=[0]

CARD Scrn=88,RamSet,HiMem=$2E5,
     Score=[0],Sdlst=560,
     Vdslst=512,Max=[0],
     PM_BaseAdr,Adres,AdresB

INT  Xdir,Ydir

INT ARRAY Pxdr=[0 0 0 0],
          Pydr=[0 0 0 0]

CARD ARRAY Linept(24),Sc(10)

BYTE ARRAY Charset,Dlist,Stacky(9),
     PmHpos(8)=$D000,Stack(9),
     Stackx(7)=[0 9 26 9 26 9 26],
     Px(4)=[0 0 0 0],Py(4)=[0 0 0 0],
     Begx(4)=[0 64 124 184],
     Begy(4)=[0 18 90 42],
     Pm_Width(5)=$D008,Plptr,
     Pm_Mismask(4)=[$FC $F3 $CF $3F],
     Pcolr(4)=704,Pmtopf(8)=$D000,
     Pmtop(8)=$D008,Pfcol(8),Pcol(8),
  Chmp1(0)=[0 0 0 0 28 42 54 28 73 127
  28 20 22 48 0 0 0 0],
  Chmp2(0)=[0 0 0 0 28 42 54 28 8
  127 93 20 52 6 0 0 0 0],
  Ibm(0)=[0 0 0 0 88 84 88 84 88 0 68
  108 84 68 0 0 0 0],
  Cmdore(0)=[0 0 0 0 224 176 151 134
  128 128 134 151 176 224 0 0 0 0],
  Apple(0)=[0 0 0 0 48 8 8 62 127 127
  127 127 62 28 0 0 0 0],
  Estat(4),
  Ms11(0)=[170 85 170 85 170 85 170 85
  170 85 170 85 170 85 170 85 170 85],
  Ms12(0)=[85 170 85 170 85 170 85 170
  85 170 85 170 85 170 85 170 85 170],
  ShapeTable(0)=[
  85 17 17 68 68 17 17 85;1-GIRDER
  160 170 160 160 160 160 170 160;
  10 170 10 10 10 10 170 10;3 RT LDR
  170 51 85 51 170 51 85 51;4 LV1
  0 0 0 0 170 85 170 85;5 LV1 CRUNCH
  85 51 170 51 85 51 170 51;6 LV2
  0 0 0 0 85 170 85 170;7 LV2 CRUNCH
  255 34 85 34 255 34 85 34;8 LV3
  0 0 0 0 255 85 255 85;9 LV3 CRUNCH
  165 51 90 51 165 51 90 51;10 LV4
  0 0 0 0 165 90 165 90;11 LV4 CRUNCH
    190 235 215 195 195 215 235 190;12
    235 130 150 170 170 150 130 235;13
    ]

PROC Pause() ;TEST
WHILE Consol<>6 DO OD RETURN

PROC Download()
;Step back HiMem and move the
;character set into RAM
RamSet=(HiMem-$400)&$FC00;1K boundary
ChrBase=RamSet RSH 8
HiMem=RamSet
MoveBlock(RamSet,57344,1024)
Charset=RamSet
```

```
RETURN

PROC Modify()
;Modify the RAM character set
CARD xx
FOR xx=0 TO 103
DO
  Charset(xx+8)=ShapeTable(xx)
OD
RETURN

PROC Pmgraphics()
Zero(PmHpos,8)
Zero(Pm_Width,5)
Dmactl=$2E Pcolr(0)=52
Pm_BaseAdr=(HiMem-$400)&$FC00
PmBase=Pm_BaseAdr RSH 8
HiMem=Pm_BaseAdr+384
Priority=&$C0%1 Gractl=3
RETURN

CARD FUNC PmAdr(BYTE n)
IF n>=4 THEN n=0 ELSE n=+1 FI
RETURN(Pm_BaseAdr+384+(n*$80))

PROC PmClear(BYTE n)
CARD ctr
BYTE ARRAY playadr
playadr=PmAdr(n)
IF n<4 THEN Zero(playadr,$80)
 ELSE n=-4
FOR ctr=0 TO $80-1
DO playadr(ctr)=&Pm_Mismask(n) OD
FI
RETURN

PROC Dlint()
;the display list interrupt routine
[$48 $8A $48 $98 $48]
Wsync=1 Colbk=50 Colints=12
[$68 $A8 $68 $AA $68 $40]

PROC ScoreLine()
;set up the dli
Vdslst=Dlint Dlist(27)=132 Nmien=$C0
RETURN

PROC Moveit(BYTE ARRAY shape BYTE
            which,num,xx,yy)
Adres=PmAdr(which)+yy
MoveBlock(Adres,shape,num)
PmHpos(which)=xx
RETURN

PROC Putman()
;Clear PM space/ put Players onscreen
BYTE lp
FOR lp=0 TO 3
DO
  Estat(lp)=0 PmClear(lp)
OD Mstatus=0 Ld=0 Ld2=0 SndRst()
X0=76 Y0=66 Moveit(Chmp1,0,18,X0,Y0)
FOR lp=1 TO 3
DO
 Px(lp)=Begx(lp) Py(lp)=Begy(lp)
 IF lp=1 THEN
  Moveit(Ibm,lp,18,Px(lp),Py(lp))
  ELSEIF lp=2 THEN
  Moveit(Cmdore,lp,18,Px(lp),Py(lp))
  ELSE
  Moveit(Apple,lp,18,Px(lp),Py(lp))
 FI
OD
RETURN

PROC Testcol()
BYTE ll
FOR ll=0 TO 7 DO
Pfcol(ll)=0 Pcol(ll)=0 OD
DO UNTIL Vcount&128 OD
FOR ll=0 TO 7 DO
Pfcol(ll)=Pmtopf(ll)
Pcol(ll)=Pmtop(ll) OD
PmHitClr=1
RETURN

BYTE FUNC PmHit(BYTE n,cnum)
 IF n<4 THEN n=+4 ELSE n=-4 FI
IF cnum<4 THEN
 RETURN((Pcol(n) RSH cnum)&1)
 ELSE cnum=&3
 RETURN((Pfcol(n) RSH cnum)&1)
FI RETURN(0)

PROC Msldrop()
;put Pepper on screen
BYTE trig=644,lp,tt=[0]
IF Ld>1 THEN Ld=-2
  Sound(2,Ld LSH 3,10,Ld) ELSEIF
  Mstatus>0 THEN
  Sound(2,Mstatus LSH 2,10,4)
FI
IF Mstatus>0 THEN tt=1-tt Mstatus=+1
  IF tt=0 THEN
    MoveBlock(AdresB,Ms12,18) ELSE
    MoveBlock(AdresB,Ms11,18)
  FI
  IF Mstatus=50 THEN Zero(AdresB,18)
    Mstatus=0 Sound(2,0,0,0)
  FI
FI
IF trig=1 OR Pep=0 OR Mstatus>0
 THEN RETURN
FI
Mstatus=1
FOR lp=0 TO 3
DO PmHpos(lp+4)=X0-3+(lp LSH 2) OD
My=Y0
AdresB=PmAdr(4)+My
MoveBlock(AdresB,Ms11,18) Ld=12
Pep=-1
Position(36,23) Print("   ")
Position(36,23) PrintB(Pep)
RETURN

PROC Gotbumped()
BYTE lq,lq1
IF Ld2>0 THEN Ld2=-1 FI
Sound(3,Ld2 LSH 3,8,Ld2)
FOR lq=0 TO 3 DO FOR lq1=1 TO 3 DO
IF PmHit(lq+4,lq1)=1 AND Estat(lq1)=0
 THEN Ld2=14 Estat(lq1)=1 Score=+5
 PmHpos(lq+4)=0
FI OD OD
FOR lq=1 TO 3 DO
IF Estat(lq)>0 THEN Estat(lq)=+1
 Pcolr(lq)=((Rand(14)+1) LSH 4)+10
FI
IF Estat(lq)=Ft THEN Estat(lq)=0
 PmClear(lq)
 Pcolr(lq)=((Rand(14)+1) LSH 4)+10
 Px(lq)=Begx(lq) Py(lq)=Begy(lq)
 IF lq=1 THEN
  Moveit(Ibm,lq,18,Px(lq),Py(lq))
  ELSEIF lq=2 THEN
  Moveit(Cmdore,lq,18,Px(lq),Py(lq))
  ELSE
  Moveit(Apple,lq,18,Px(lq),Py(lq))
 FI
FI OD RETURN

PROC Title()
BYTE colpf0=53270,colpf1=53271,
     colpf3=53273,rtclock=20
Graphics(18)
Position(5,4) PrintD(6,"ANALOG MAN")
Position(8,5) PrintD(6,"BY")
Position(3,7)
PrintD(6,"david plotkin")
Position(3,9)
PrintD(6,"PRESS select")
WHILE Consol<>5
DO colpf3=Fate Atrt=0 Wsync=0
  colpf0=128-Vcount+rtclock RSH 2
  colpf1=Vcount+rtclock RSH 2
OD
RETURN

PROC Gr4Init()
;Set up the address of each screen
;line,initialize and set up Gr. 4
CARD xx
BYTE clr1=709
Graphics(0) CursIn=1 Print(" ")
```

```
FOR xx=0 TO 23
DO Linept(xx)=Scrn+(40*xx) OD
Dlist=Sdlst Dlist(3)=68
FOR xx=6 TO 27
DO Dlist(xx)=4 OD clr1=68
RETURN

PROC Update()
;print data on the text line
Position(0,23) Print("Score: ")
Position(7,23) PrintC(Score)
Position(13,23) Print("Lives: ")
Position(20,23) PrintB(Lv)
Position(22,23) Print("Hi: ")
Position(26,23) PrintC(Max)
Position(32,23) Print("SB: ")
Position(36,23) Print("   ")
Position(36,23) PrintB(Pep)
RETURN

PROC Plot4(BYTE x,y,ch)
;Plot a char at location x,y
BYTE ARRAY line
line=Linept(y) line(x)=ch
RETURN

BYTE FUNC Locate4(BYTE x,y)
;Returns the value of the char at x,y
BYTE ARRAY line
line=Linept(y)
RETURN(line(x))

PROC Hline(BYTE x1,y1,x2,ch)
;draw a line of ch characters from
;x1,y1 to x2,y1 (horizontal line)
BYTE ARRAY line
BYTE lp
line=Linept(y1) lp=x1
DO line(lp)=ch lp==+1 UNTIL lp=x2+1 OD
RETURN

INT FUNC HStick(BYTE port)
BYTE ARRAY ports(4)=$278
INT ARRAY value(4)=[0 1 $FFFF 0]
port==&3
RETURN (value((ports(port)&$C) RSH 2))

INT FUNC VStick(BYTE port)
BYTE ARRAY ports(4)=$278
INT ARRAY value(4)=[0 1 $FFFF 0]
port==&3
RETURN (value(ports(port)&3))

PROC EndGame()
;game over
BYTE trig=644,wsync=$D40A,rtclock=20,
     lm=53271,vcount=54283
SndRst() Bkgrnd=0 Dlist(10)=2
IF Score>Max THEN Max=Score FI
Put(125) Update()
Position(7,5)
Print("All DONE  Press FIRE")
DO vcount=0  lm=vcount+rtclock RSH 2
  Atrt=0 UNTIL trig=0
OD
Bkgrnd=148 Dlist(10)=4 Put(125)
Lv=5 Pep=5 Indx=0 Level=5 Ft=200
Score=0 Update() PmHitClr=0
RETURN

PROC Meltdown()
BYTE lp,lq,time=20
BYTE ARRAY melt
SndRst() melt=PmAdr(0)+Y0+4
FOR lp=0 TO 30
DO lq=Rand(10) melt(lq)=Fate
 Sound(0,Fate,8,8)
 time=0 DO UNTIL time=3 OD
OD
FOR lp=0 TO 9
DO melt(lp)=0 Sound(0,lp*10,10,8)
 time=0 DO UNTIL time=2 OD
OD Sound(0,0,0,0)
RETURN

PROC Ouch()
BYTE lc,ld
IF Pcol(4)=0 THEN RETURN FI
FOR lc=1 TO 3
DO IF PmHit(0,lc)=1 AND Estat(lc)>0
 THEN RETURN FI
OD
Meltdown()
FOR lc=0 TO 7 DO PmClear(lc) OD
Lv==-1 Position(20,23) PrintB(Lv)
IF Lv=0 THEN EndGame() ELSE Putman()
PmHitClr=0 FI RETURN

PROC InitLev()
;Set initial stack values, call Putman
BYTE lp
FOR lp=1 TO 8 DO Stack(lp)=0 OD
Stacky(1)=4 Stacky(2)=4 Stacky(3)=10
Stacky(4)=10 Stacky(5)=16 Stacky(6)=16
Stacky(7)=0 Stacky(8)=0 Putman()
RETURN

PROC Girders()
;draw the main four lines of girders
;clear screen and init new level
SndRst() Zero(Scrn,960) Loud=0
Hline(2,22,37,1) Hline(2,16,37,1)
Hline(2,10,37,1) Hline(2,4,37,1)
Hline(9,4,13,4)  Hline(9,10,13,6)
Hline(9,16,13,8) Hline(26,4,30,4)
Hline(26,10,30,6) Hline(26,16,30,8)
InitLev()
RETURN

PROC Screen1()
;draw screen 1
BYTE lp
Girders();now the ladders
FOR lp=4 TO 21
DO Plot4(2,lp,2) Plot4(3,lp,3)
  Plot4(19,lp,2) Plot4(20,lp,3)
  Plot4(36,lp,2) Plot4(37,lp,3)
OD Position(15,23)
Print ("Beginners Luck          ")
RETURN

PROC Screen2()
;draw screen 2
BYTE lp
Girders() FOR lp=4 TO 21
DO Plot4(19,lp,2) Plot4(20,lp,3) OD
FOR lp=10 TO 15
DO Plot4(2,lp,2) Plot4(3,lp,3) OD
Position(15,23)
Print ("Where are the ladders?")
RETURN

PROC Screen3()
;draw screen 3
BYTE lp
Girders() FOR lp=4 TO 21
DO Plot4(19,lp,2) Plot4(20,lp,3) OD
FOR lp=4 TO 9
DO Plot4(2,lp,2) Plot4(3,lp,3) OD
FOR lp=16 TO 21
DO Plot4(36,lp,2) Plot4(37,lp,3) OD
Position(15,23)
Print ("Side to Side           ")
RETURN

PROC Screen4()
;draw screen 4
BYTE lp
Girders() Hline(16,4,23,0)
Hline(16,16,23,0) FOR lp=4 TO 21
DO Plot4(14,lp,2) Plot4(15,lp,3)
  Plot4(24,lp,2) Plot4(25,lp,3)
OD Position(15,23)
Print ("First Holes             ")
RETURN

PROC Screen5()
;draw screen 5
BYTE lp
Girders() Hline(16,10,23,0)
Hline(16,16,23,0) FOR lp=4 TO 21
DO Plot4(19,lp,2) Plot4(20,lp,3) OD
FOR lp=4 TO 9
DO Plot4(14,lp,2) Plot4(15,lp,3)
```

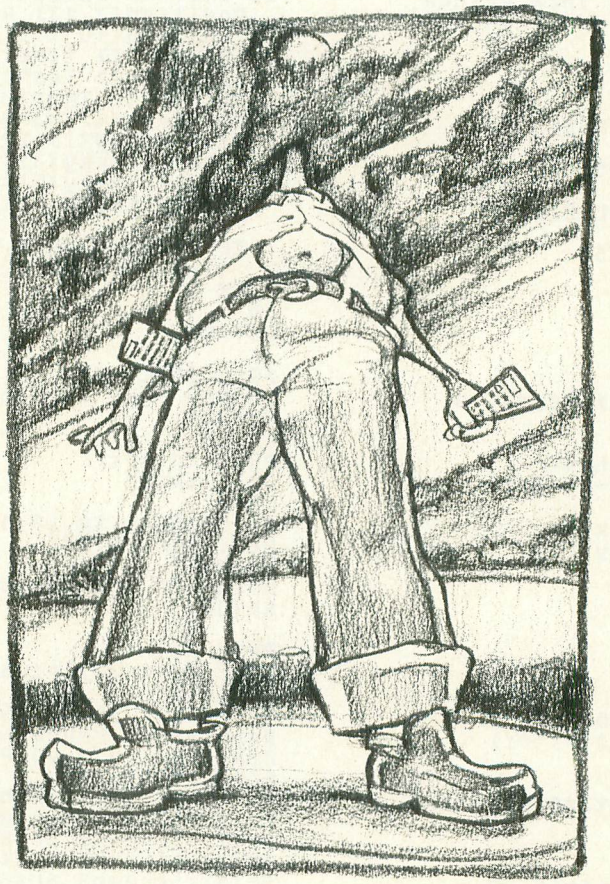
```
  Plot4(24,lp,2) Plot4(25,lp,3)
OD FOR lp=16 TO 21
DO Plot4(14,lp,2) Plot4(15,lp,3)
  Plot4(24,lp,2) Plot4(25,lp,3)
OD Position(15,23)
Print("Up and Down                ")
RETURN

PROC Screen6()
;draw screen 6
BYTE lp
Girders() Hline(16,4,23,0)
Hline(16,10,23,0) Hline(16,16,23,0)
FOR lp=4 TO 21
DO Plot4(14,lp,2) Plot4(15,lp,3)
  Plot4(24,lp,2) Plot4(25,lp,3)
OD Position(15,23)
Print ("All Holes                ")
RETURN

PROC Screen7()
;draw screen 7
BYTE lp
Girders() Hline(16,10,23,0)
Hline(16,16,23,0) FOR lp=4 TO 21
DO Plot4(19,lp,2) Plot4(20,lp,3) OD
Position(15,23)
Print ("Time at the TOP          ")
RETURN

PROC Screen8()
;draw screen 8
BYTE lp
Girders()
Hline(16,10,23,0) Hline(16,16,23,0)
FOR lp=4 TO 21
DO Plot4(2,lp,2) Plot4(3,lp,3)
  Plot4(36,lp,2) Plot4(37,lp,3)
OD Position(15,23)
Print ("Use the Stairs           ")
RETURN

PROC Screen9()
;draw screen 9
BYTE lp
Girders() Hline(16,4,23,0)
Hline(16,10,23,0) Hline(16,16,23,0)
FOR lp=4 TO 21
DO Plot4(2,lp,2) Plot4(3,lp,3)
  Plot4(36,lp,2) Plot4(37,lp,3)
OD Position(15,23)
Print ("Elevator Shaft           ")
RETURN

PROC Falling(BYTE tt)
;keep track of level status
BYTE lp
IF tt=4 THEN
  IF X0<120 THEN Stack(1)==+1 ELSE
    Stack(2)==+1 RETURN
  FI
FI
IF tt=6 THEN
  IF X0<120 THEN Stack(3) ==+1 ELSE
    Stack(4)==+1 RETURN
  FI
FI
IF tt=8 THEN
  IF X0<120 THEN Stack(5)==+1 ELSE
    Stack(6)==+1 RETURN
  FI
FI
RETURN

PROC Squashed(BYTE wh)
BYTE lk,xx,yy
xx=(Stackx(wh) LSH 2)+48
yy=(Stacky(wh) LSH 2)+16-14
FOR lk=1 TO 3
DO IF Px(lk)>=xx-8 AND Px(lk)<=xx+16
   AND Py(lk)=yy THEN Estat(lk)=1
   Score==+5 Ld=14
  FI
OD RETURN

RETURN

PROC DropLevel()
;make levels fall, keep track of y pos
BYTE lp,lev
BYTE ARRAY wh(7)=[0 5 5 7 7 9 9]
FOR lp=1 TO 6
DO IF Stack(lp)>=5 THEN Stack(lp)==+1
   FI
  IF Stack(lp)>=7 THEN
  Hline(Stackx(lp),Stacky(lp),
        Stackx(lp)+4,0) Score==+1
  Stacky(lp)==+1 lev=Stacky(lp)
  IF lev=10 OR lev=16 THEN Stack(lp)=0
    Hline(Stackx(lp),lev,Stackx(lp)+4,
          wh(lp)-1)
    IF Stacky(lp+2)=lev THEN
      Stack(lp+2)=7 Stacky(lp+2)=lev+1
      Hline(Stackx(lp+2),lev+1,
        Stackx(lp+2)+4,wh(lp+2))
    FI ELSE
    Hline(Stackx(lp),lev,Stackx(lp)+4,
          wh(lp))
    IF lev=22 THEN Stack(lp)=0 FI
  FI
  IF lev=10 OR lev=16 OR lev=22 THEN
    Squashed(lp)
  FI
FI OD
RETURN

PROC Check()
;Look ahead-see whats there and move
BYTE xt1,xt2,yt1,yt2,t1,t2,t3,t4
BYTE ARRAY pstn
xt1=(X0-48) RSH 2 yt1=(Y0-16+14) RSH 2
t1=Locate4(xt1,yt1)
t2=Locate4(xt1+1,yt1)
IF t1=0 AND t2=0 THEN;falling
  Y0==+4 Moveit(pstn,0,18,X0,Y0)
  Tone=10 Loud=10
  RETURN
FI
IF Stick0=15 THEN RETURN ELSE
  Tone=8 Flg=1-Flg
  IF Flg=0 THEN pstn=Chmp1 ELSE
    pstn=Chmp2
  FI
FI
IF Stick0=7 THEN;move right
  t1=Locate4(xt1+2,yt1) Loud=6
  IF X0<192 THEN X0==+4 FI
  Moveit(pstn,0,18,X0,Y0)
  IF (t1=4 OR t1=6 OR t1=8) THEN
    Plot4(xt1+2,yt1,t1+1) Falling(t1)
  FI
FI
IF Stick0=11 THEN;move left
  t1=Locate4(xt1-1,yt1) Loud=6
  IF X0>56 THEN X0==-4 FI
  Moveit(pstn,0,18,X0,Y0)
  IF (t1=4 OR t1=6 OR t1=8) THEN
    Plot4(xt1-1,yt1,t1+1) Falling(t1)
  FI
FI
IF Stick0=14 THEN;move up
  t1=Locate4(xt1,yt1)
  t2=Locate4(xt1+1,yt1)
  t3=Locate4(xt1,yt1-1)
  t4=Locate4(xt1+1,yt1-1)
  IF ((t1=2 AND t2=3) OR
    (t3=2 AND t4=3))
    THEN Y0==-4 Loud=6
    Moveit(pstn,0,18,X0,Y0)
  FI
```

```
FI
IF Stick0=13 THEN;move down
  t1=Locate4(xt1,yt1)
  t2=Locate4(xt1+1,yt1)
  IF (t1=2 AND t2=3) THEN Y0==+4
    Moveit(pstn,0,18,X0,Y0) Loud=6
  FI
FI
RETURN

PROC Noise()
;the sound effects
IF Loud>0 THEN Loud==-1
  Sound(1,Y0,Tone,Loud)
FI
RETURN

PROC NoChase(BYTE d1,dr,du,dd,lp)
BYTE sel
IF (du=0 AND dd=0) THEN
 IF (Pxdr(lp)<0 AND d1=1) THEN RETURN
  ELSEIF (Pxdr(lp)>0 AND dr=1) THEN
  RETURN
 FI
FI
IF (d1=0 AND dr=0) THEN
 IF (Pydr(lp)<0 AND du=1) THEN RETURN
  ELSEIF (Pydr(lp)>0 AND dd=1) THEN
  RETURN
 FI
FI sel=Rand(4)
IF (sel=0 AND d1=1) THEN
 Pxdr(lp)=-4 Pydr(lp)=0 ELSEIF
 (sel=1 AND dr=1) THEN
 Pxdr(lp)=4 Pydr(lp)=0 ELSEIF
 (sel=2 AND du=1) THEN
 Pxdr(lp)=0 Pydr(lp)=-4 ELSEIF
 (sel=3 AND dd=1) THEN
 Pxdr(lp)=0 Pydr(lp)=4 ELSE
 Pxdr(lp)=0 Pydr(lp)=0
FI
RETURN

PROC Chase()
;the creatures move
BYTE lp,xt1,xt2,yt1,yt2,t1,t2,t3,t4,
     dir,dl,dr,du,dd
INT delx,dely,dx,dy
FOR lp=1 TO 3; for each chaser
DO delx=X0-Px(lp) dely=Y0-Py(lp)
 dx=delx dy=dely
 IF delx<0 THEN delx=-delx FI
 IF dely<0 THEN dely=-dely FI
 delx==RSH 2 dely==RSH 2
 xt1=(Px(lp)-48) RSH 2
 yt1=(Py(lp)-16+14) RSH 2
 t1=Locate4(xt1,yt1)
 t2=Locate4(xt1+1,yt1)
 t3=Locate4(xt1,yt1-1)
 t4=Locate4(xt1+1,yt1-1)
 dir=0 dl=0 dr=0 du=0 dd=0
 IF (t1=2 AND t2=3 AND Py(lp)<91)
  THEN dd=1
 FI
 IF ((t1=2 AND t2=3) OR (t3=2 AND t4=3
))
  THEN du=1
 FI
 IF (yt1=4 OR yt1=10 OR yt1=16 OR
  yt1=22) THEN dir=1
 FI
 IF (dir=1 AND Px(lp)>56) THEN dl=1 FI
 IF (dir=1 AND Px(lp)<192)
      THEN dr=1 FI
 IF (dely<=Level AND delx<=Level) THEN
  IF (dx<0 AND dl=1) THEN
   Pxdr(lp)=-4 Pydr(lp)=0
   ELSEIF (dx>0 AND dr=1) THEN
   Pxdr(lp)=4 Pydr(lp)=0
   ELSEIF (dy<0 AND du=1) THEN
   Pxdr(lp)=0 Pydr(lp)=-4
   ELSEIF (dy>0 AND dd=1) THEN
   Pxdr(lp)=0 Pydr(lp)=4
   ELSE Pxdr(lp)=0 Pydr(lp)=0
  FI ELSE NoChase(dl,dr,du,dd,lp)
 FI
 IF Estat(lp)<>0 THEN Pxdr(lp)=0
  Pydr(lp)=0;killed!
 FI
 IF t1=0 AND t2=0 THEN Pxdr(lp)=0
  Pydr(lp)=4
 FI; falling!
 Px(lp)==+Pxdr(lp) Py(lp)==+Pydr(lp)
 IF lp=1 THEN
  Moveit(Ibm,lp,18,Px(lp),Py(lp))
  ELSEIF lp=2 THEN
  Moveit(Cmdore,lp,18,Px(lp),Py(lp))
  ELSE
  Moveit(Apple,lp,18,Px(lp),Py(lp))
 FI
OD
RETURN

PROC Play()
;the play game loop
BYTE lp,time=20
DO Check() Chase() Msldrop() Atrt=0
  Position(7,23) PrintC(Score)
  FOR lp=0 TO 2
  DO Noise() time=0 DO UNTIL time=1 OD
  OD Noise() Testcol() Gotbumped()
  Ouch() IF Indx=0 THEN EXIT FI
  DropLevel();make levels fall
  IF (Stacky(1)=22 AND Stacky(2)=22
    AND Stacky(3)=22 AND Stacky(4)=22
    AND Stacky(5)=22 AND Stacky(6)=22)
    THEN EXIT;test for level finished
  FI
  IF Level=5 THEN Check()
   time=0 DO UNTIL time=2 OD
  FI
OD
RETURN

PROC Vector()
;Dummy PROC for the screens
RETURN

PROC Intro()
BYTE tm=20
tm=0
DO Sound(0,tm,10,4) UNTIL tm=100 OD
Position(15,23)
Print("                    ")
Update() Sound(0,0,0,0)
RETURN

PROC Main()
BYTE time=20,lp,ch=764
Title()
Gr4Init() Snd1=0 Snd2=3
Download() Pmgraphics()
FOR lp=0 TO 7 DO PmClear(lp) OD
FOR lp=1 TO 3
DO Pcolr(lp)=((Rand(14)+1)LSH 4)+10 OD
Pcolr(0)=56 Modify() ScoreLine()
Sc(1)=Screen1 Sc(2)=Screen2
Sc(3)=Screen3 Sc(4)=Screen4
Sc(5)=Screen5 Sc(6)=Screen6
Sc(7)=Screen7 Sc(8)=Screen8
Sc(9)=Screen9
DO Indx==+1 Vector=Sc(Indx)
  FOR lp=0 TO 7 DO PmClear(lp) OD
  Vector() Intro() Play()
  IF Indx=9 THEN Indx=0 Level==+4
    IF Pep<8 THEN Pep==+2 FI
    IF Ft>100 THEN Ft==-20 FI Update()
  FI
OD
RETURN
```

General

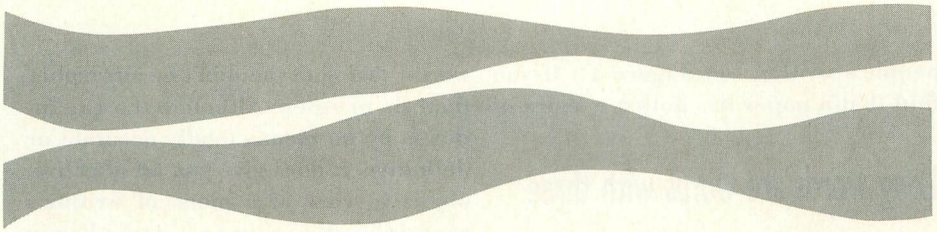
48K disk or cassette

Lost In The Fog



A Computerization of the Fog Index of Readability

by Greg Knauss



Have you ever wondered why some people's writing has all the order of an 11-car pileup? How can these writers, normally very clear individuals, make such a mess of simple ideas in a tangle of run-on sentences, misplaced punctuation, and needlessly long words?

Why does it give you a headache to read even the shortest paper by these folks? Simple. People with this problem, and we all run into it at one time or another, don't proofread. They write a sentence, one that may be long and complex, but never take the time to read it. That sentence sits there with all the grace of a tractor, when several shorter sentences would have done the job much more effectively.

A reader may have to read a sentence several times just to understand it. It's an annoying, often inconvenient, problem. And one that shows no sign of going away anytime soon. There is a simple solution, however. To stop writing this type of 'foggy' or unclear work, there are several things that you can do: 1. Re-read what you've written, both after you've written it and after you've let it sit for awhile. 2. Re-read the paper like someone who knows nothing about the subject. Did you cover all the bases? Is there anything that the layman wouldn't understand? 3. Get someone else to read the paper.

One of the best things you can do for yourself is to get an honest opinion from someone who knows absolutely nothing about the subject. I let my mom read all my stuff about computers.

The Fog Index

And although all of these steps are useful and very helpful to eliminate foggy writing, there is only one real way to mechanically, objectively determine if a piece of writing is clear or not: the Fog Index. Robert Gunning, creator of the Fog Index, coined the phrase foggy writing to describe text with a low readability.

The Index is supposed to be an objective description of writing's clarity: A high number indicates poorly or complexly written work, a low number is simplistic and easily understood writing. The Index number corresponds to the grade level needed to understand the text. To determine the Fog Index of a paper, word, long word, and sentence counts are taken and a simple formula used to determine the result.

Long words are words with three syllables or more. Independent phrases, such as what follows a semicolon, are counted as separate sentences. Lost in the Fog is a computerized version of the Fog Index for the 8-bit Atari computers. To use Lost, first type in Program 1, and save it to disk. Be sure to save a copy before using it, because it contains machine language that will crash the computer if typed incorrectly. Now RUN Lost and insert a disk with a text file on it into drive one. Though Lost in the Fog was designed for use with files created by AtariWriter, it should work with any word processor that stores its text as a standard DOS file.

One of the best things you can do for yourself is to get an honest opinion from someone who knows absolutely nothing about the subject.

AtariWriter Plus files must be saved with the ASCII Save option to be checked correctly. Lost will request the filename of the text to be fogged. Enter one. To get a directory of drive one, just press RETURN in place of a filename. Once you've entered the file's name, the program will start its fogging. The entire text file will be printed on the bottom half of the screen, so you can reread your writing as it goes by. To interrupt a Fog session, just press the ESC key. The results so far will be immediately displayed.

What People Like To Read

Researchers, using the Fog Index as a guide, have found that most people like to read below their grade level. Even college professors are uncomfortable above a Fog Index of 12. Most popular magazines, such as ANALOG, publish between the sixth and eighth grade. (For a fogging of some famous people's writing, see Figure 1.) If you find that a paper has gotten a score of ten or higher it would probably benefit from shorter sentences and shorter words; dialogue shouldn't be any higher than six or seven. Although the Fog Index is by no means totally accurate or definitive, it does give you an effective, objective view of a piece of writing's readability. By using Lost in the Fog in conjunction with the other steps mentioned above, your papers can take on a clarity they never had before. It really can help make you a better writer. By the way, this article has a Fog Index of eight. Not bad, eh?

Long words are words with three syllables or more.

Lost In The Fog

Listing 1: Basic

```
SR 10 DIM A$(100),B$(100),F$(20):A$=" "
NH 20 GRAPHICS 0:POKE 559,0:POKE 709,4:PO
   KE 710,6:POKE 712,6:POKE 82,1
LZ 30 OPEN #1,4,0,"K"
XG 40 DL=(PEEK(106)-4)*256:TOP=(PEEK(106)
   -12)*256:SCR=PEEK(88)+PEEK(89)*256
XQ 50 FOR I=0 TO 32:READ A:POKE DL+I,A:NE
   XT I
NN 60 POKE DL+4,TOP-INT(TOP/256)*256:POKE
    DL+5,TOP/256
YI 70 A=SCR+480:POKE DL+17,A-INT(A/256)*2
   56:POKE DL+18,A/256
RL 80 POKE 560,DL-INT(DL/256)*256:POKE 56
   1,DL/256
NP 90 FOR I=1536 TO 1556:READ A:POKE I,A:
   NEXT I:POKE 512,0:POKE 513,6:POKE 5428
   6,192
HL 100 POKE 88,PEEK(DL+4):POKE 89,PEEK(DL
   +5)
EC 110 POSITION 3,2:? "lost in the___":PO
   SITION 28,2:? "fog"
PD 120 POSITION 7,4:? "Copyright 1988 Mag
   num Opus":POSITION 13,6:? "By Greg Kna
   uss"
IZ 130 POKE 88,SCR-INT(SCR/256)*256:POKE
   89,SCR/256
KT 140 FOR I=1 TO 15:? :NEXT I:POSITION 1
   ,13:? "Please enter the filename and e
   xtender"
XR 150 ? "of the text to be fogged, place
    the"
WA 160 ? "disk containing it in drive one
   , and"
TB 170 ? "press RETURN. For a directory
    of the"
QJ 180 ? "disk in drive one, press RETUR
   N"
YA 190 ? "alone.":?
KJ 200 POKE 752,0:? "FILENAME.EXT? D:";:I
   =0
QW 210 POKE 559,34
BW 220 GET #1,A
FN 230 IF A=155 THEN 290
QC 240 IF A=126 AND I>0 THEN A$(I)="":? C
   HR$(A);:I=I-1:GOTO 220
HI 250 IF A>128 THEN A=A-128
FQ 260 IF A>96 AND A<123 THEN A=A-32
LY 270 IF  NOT ((A>64 AND A<91) OR (A>47
    AND A<58) OR A=46) AND I<12) THEN 220
HW 280 I=I+1:A$(I)=CHR$(A):? CHR$(A);:GOT
   O 220
HR 290 POKE 752,1:IF I>0 THEN F$="D:":F$(
   3)=A$:GOTO 400
ZC 300 POKE 752,1:? :? :? "Directory...":
   A=0
NN 310 TRAP 370
QT 320 OPEN #2,6,0,"D:*.*"
EP 330 INPUT #2;A$:A=A+1:POSITION 1,23:IF
    INT(A/2)=A/2 THEN POSITION 20,23
JN 340 ? A$;:IF INT(A/2)=A/2 THEN ?
GN 350 IF INT(A/14)=A/14 THEN ? :? :POSIT
   ION 8,22:? "Press RETURN for more.":
   POKE 764,255:GET #1,I:?
NV 360 GOTO 330
SG 370 IF INT(A/2)<>A/2 THEN ?
JO 380 CLOSE #2:? :? " Press RETURN to
   enter a filename."
PZ 390 POKE 764,255:GET #1,A:GOTO 140
XF 400 TRAP 640:POKE 764,255:L=0:FOR I=1
   TO 15:? :NEXT I:POSITION 1,12
SY 410 OPEN #2,4,0,F$:I=0:N=0:L=0:W=0:S=0
   :LW=0
MM 420 GET #2,A:IF (A>64 AND A<123) OR A=
   16 THEN A$=CHR$(A):N=1:GOTO 440
NP 430 GOTO 420
VQ 440 N1=N:B$=A$:N=1:I=0
WF 450 GET #2,A:A$(N)=CHR$(A):IF A<>32 AN
   D A<>155 THEN I=1:N=N+1
DG 460 IF A<>32 AND A<>155 THEN 450
XM 470 IF I=0 THEN 450
CE 480 IF N1>9 THEN LW=LW+1
PI 490 IF LEN(A$)>1 THEN I=ASC(A$(LEN(A$)
   -1)):IF I=46 OR I=33 OR I=63 OR I=58 O
   R I=59 THEN S=S+1
ZE 500 IF I=34 OR I=39 OR I=41 OR I=93 TH
   EN I=ASC(A$(LEN(A$)-2)):IF I=46 OR I=3
   3 OR I=63 OR I=58 OR I=59 THEN S=S+1
BE 510 W=W+1:? B$;:L=L+N1:IF L+N>38 THEN
   L=0:?
BY 520 IF B=155 THEN L=0
NE 530 B=A:POKE 77,0:IF PEEK(764)<>28 THE
   N 440
BZ 540 ? B$:IF N1>9 THEN LW=LW+1
PY 550 ? :? :? "Analysis of ";F$(3,LEN(F$
   ));"...":?
JY 560 ? "Number of words: ";W
TN 570 ? "Number of sentences: ";S
TE 580 ? "Number of long words: ";LW
IQ 590 IF S=0 THEN ? :? "Need at least o
   ne sentence for":? "analysis!":GOTO 63
   0
MQ 600 ? "Number of words/sentence: ";INT
   (W/S)
NF 610 ? "Number of long words/sentence:
   ";INT(LW/S)
NL 620 ? "Fog Index rating: ";INT((0.4*(W
   /S+LW*100/W)*100+0.5)/100)
PV 630 GOTO 380
DS 640 A=PEEK(195):IF A=136 THEN 540
XW 650 ? :? :? "ERROR! Number ";A;".":
   ? :? "Press RETURN to restart."
AW 660 FOR I=1 TO 7:? :NEXT I:POKE 764,25
   5:GET #1,A:CLOSE #2:GOTO 140
LE 670 DATA 112,112,112,66,0,0,2,6,112,7,
   2,2,2,2,130,112,66,0,0,2,2,2,2,2,2,2,2
   ,2,2,2,0,0,65
ZL 680 DATA 72,169,0,141,10,212,141,26,20
   8,169,146,141,24,208,169,10,141,23,208
   ,104,64
```

130 XE only

XE Banks

by Dave Schofield

You'll notice, if you keep up with all the popular Atari specific publications, that the general consensus about the extra 64k in the 130XE is: BASIC programmers, keep out! This is actually pretty sound advice, but does lead one to think that the matter is an open-and-shut case, which is not necessarily true. Let's first take a look at why BASIC can't access the extra banks of RAM.

XE Banks

BASIC and the extra RAM

The first thing to know is that BASIC can indeed access the extra RAM. It's as simple as a single POKE statement! Hmmm, there must be more to it than that. Well, there is. Since the XE has 64K of main RAM, we can think of it as four 16K blocks. The first block goes from memory locations 0 to 16383. The second from 16384 to 32767, third 32768 to 49151, and the fourth is 49152 to 65535. The extra 64K is divisible exactly the same way. We'll call each of these divisions a "bank." The special memory management chip in the XE can let you access all four banks, but only one at a time. It does this by laying the bank you specify in place of the main RAM's second bank, from 16384 to 32767. So, whenever you execute that special POKE, the bank you specify shows up in place of the main RAM's bank.

What happens to the main RAM's bank two? Ah, this is where the problem arises. The normal bank two section of RAM becomes completely invisible, as though it didn't exist. Now suppose you had a fairly good sized BASIC program in memory. Chances are it will extend well into the second bank of RAM. As soon as you enable an extra bank in place of the normal one, your BASIC program goes into la-la land. You can't re-enable the normal bank because you no longer have the program in RAM!

When does the good news come?

Right now! Even though BASIC can't hack enabling and disabling the extra RAM banks, machine language can. Here's how: Most everyone knows that a USR statement transfers control to a machine language routine. Let's say the routine is located in Page 6, from 1536 to 1791. When the machine language program has control, it doesn't care what you do with the second bank of RAM, since it lies within the safety of the first bank. So, the secret is to give control to the machine language, let it enable the extra banks, perform the operation involving the extra RAM, then re-enable the normal bank two, and return control to BASIC. Sound simple? Actually, it really is.

Our application

Listing 1 is a BASIC program that will load up to eight graphics screens (in an UNcompressed, 62-sector format) and save them into the extra memory. After they are all there, you can load any one of them back into the screen area, one by one, almost instantly! Here's how to use it:

1) Round up several (up to eight) screen files on a single disk.
2) Type in Listing 1.
3) In Line 140, set the variable PICS equal to the number of picture files you have on the disk.
4) Type the picture files' names on Line 9100 as DATA elements (erasing the ones already there).
5) Lines 9201 thru 9208 are DATA statements describing the picture file. (9201 describes picture one, 9202 describes picture two, etc.) Here's what the program expects to find there:

Data element 1: The graphics mode (plus 16) of the picture. (Can be 31, 24, 9, 10 or 11.) Data elements 2-6: The values to POKE into the color registers from 708-712.

6) Save the program before running it.
7) After you type RUN, the files will load onto the screen, and be saved into the extra RAM. The saving to extra RAM is done right after the picture loads, and you'll notice there is virtually no delay at all.
8) The last screen loaded will remain visible. Now you can press a number between one and eight instantly.

If you only loaded five pictures, pressing 6, 7 or 8 will put garbage on the screen. No harm, it's just the data that's in the extra RAM at powerup.
9) Hit 0 to clear the screen, ESCAPE to quit the program.

Technically speaking

All the machine language program does is decide which bank should be enabled and at what address it will find the correct data (determined from the variable BANK in the USR calls). It then enables the correct bank. Next, it does a very simple memory move routine to move $1E00 (7680) bytes into the screen RAM area or from the screen RAM area based on the first argument in the USR call (0 means save from screen to extra RAM, 1 means from extra RAM to screen area). Lastly, it re-enables the normal configuration and returns to BASIC.

Using the machine code in your own programs is easy.

Follow these rules:

POKE the data found in Lines 9005 thru 9030 into locations 1536 thru 1637

Call it with: A = USR (1536, OPER, SCRN, BANK) where OPER equals 0 to move data from the screen to RAM, and 1 to move RAM to the screen, and SCRN equals the address of the screen, found by SCRN = PEEK (88) + PEEK (89)* 256 (after you execute your graphics command). Bank equals a number between one and eight signifying which bank (actually half-bank) to store to or load from. If you use only one or two banks, it doesn't matter which ones you choose; they are all exactly the same.

Listing 2 is the assembly source code written with Mac/65 from OSS.

Lastly - - -

This program is simply a way to demonstrate the ability to access the extra RAM, even from BASIC (sort of). The routine in Page 6 isn't completely useless though. If you're writing a drawing program for the XE, you can easily use this routine to enable you to have eight screens in the computer at once! Switching from one to the other would be a piece of cake. Or you could set aside one of the banks for an Undo feature. This simply means that each time a new "tool" or color is chosen to draw with, you save the current screen into the extra RAM. If the user decides he liked it better before his changes, he hits the Undo key, the program loads the extra RAM back to the screen, and it looks like all the changes disappear. See if you can find some other uses, too.

XE Banks

Listing 1: BASIC

```
VE 10 REM XEBANK5 FROM ANALOG COMPUTING
RU 15 ? CHR$(125):POKE 752,1:POSITION 2,1
   0:? "Insert disk with pictures. Hit RE
   TURN"
UF 16 IF PEEK(764)=255 THEN 16
RZ 17 POKE 764,255:POKE 752,0:? CHR$(125)
HC 20 IF PEEK(1637)<>237 THEN GOSUB 9000
PY 100 ICCOM=834:ICBADR=836:ICBLEN=840
QZ 110 CGBINR=7:X1=16
UF 120 DIM FILE$(20),T$(20)
MQ 130 GRAPHICS 8:SC=PEEK(88)+PEEK(89)*25
   6
OT 140 PICS=8:RESTORE 9100
LI 150 FOR BANK=1 TO PICS
EV 160 READ T$:FILE$="D:":FILE$(3)=T$
BG 170 CLOSE #1:OPEN #1,4,0,FILE$
VX 180 POKE ICCOM+X1,CGBINR
UE 190 POKE ICBADR+X1,PEEK(88):POKE ICBAD
   R+1+X1,PEEK(89)
UO 200 POKE ICBLEN+X1,0:POKE ICBLEN+1+X1,
   30
VN 205 POKE 752,1:POKE 656,1:? CHR$(156);
   CHR$(127);"Loading ";FILE$;"..."
HA 210 A=USR(ADR("hhh■LV■"),X1):CLOSE #1
QS 220 A=USR(1536,0,SC,BANK)
AG 230 NEXT BANK
XR 235 CLOSE #1:OPEN #1,4,0,"K:"
AY 240 ? CHR$(125):POKE 656,1:POKE 657,10
   :? "Hit 1-8, ESC to end";
SX 250 GET #1,BANK:BANK=BANK-48
OY 260 IF BANK=0 THEN ? #6;CHR$(125):GOTO
    250
OU 280 IF BANK=-21 THEN END
TX 290 IF BANK<1 OR BANK>8 THEN 250
FL 300 RESTORE 9200+BANK:READ MODE,C0,C1,
   C2,C3,C4:GRAPHICS MODE
TN 310 POKE 708,C0:POKE 709,C1:POKE 710,C
   2:POKE 711,C3:POKE 712,C4
OY 320 SC=PEEK(88)+PEEK(89)*256:A=USR(153
   6,1,SC,BANK)
OB 330 GOTO 250
JF 9000 FOR I=1536 TO 1637:READ D:POKE I,
   D:NEXT I:RETURN
KT 9005 DATA 104,104,104,208,18,104,133,2
   25,104,133,224,104,104,32,65,6,134,226
   ,132,227
GL 9010 DATA 76,38,6,104,133,227,104,133,
   226,104,104,32,65,6,134,224,132,225,16
   2,30
YO 9015 DATA 160,0,177,224,145,226,200,20
   8,249,202,240,7,230,225,230,227,76,40,
   6,169
GS 9020 DATA 253,141,1,211,96,170,24,106,
   144,8,32,86,6,162,0,160,64,96,32,86
RU 9025 DATA 6,162,0,160,94,96,202,189,94
   ,6,141,1,211,96,225,225,229,229,233,23
   3
ZG 9030 DATA 237,237
QD 9100 DATA PIC.1,PIC.2,PIC.3,PIC.4,PIC.
   5,PIC.6,PIC.7,PIC.8
ZI 9200 REM SCREEN SPECIFIC DATA
UX 9201 DATA 31,20,18,0,0,70
TP 9202 DATA 31,50,10,4,0,0
ZJ 9203 DATA 31,36,12,68,0,0
QP 9204 DATA 31,40,202,148,0,0
HO 9205 DATA 31,2,4,38,0,0
AQ 9206 DATA 31,0,10,66,0,148
BB 9207 DATA 24,0,10,2,0,2
XZ 9208 DATA 24,0,10,0,0,0
```

Listing 2: Assembly

```
10        .OPT NO LIST
20 ;SAVE#D:XEBANK5.M65
30 ;ASM,,#D:XEBANK5.OBJ
40 ;
50 ;--------------------------------
60 ; Save up to 8 Gr.7+,8,9,10 or 11
70 ; screens into the XE's extra
80 ; RAM. Won't interfere with
90 ; BASIC!
0100 ;
0110 ; Call with:
0120 ; A=USR(1536,oper,scrn,bank)
0130 ; Where:
0140 ; 1536 is addr of routine
0150 ; oper = 0 for save to XRAM
0160 ;         1 for load from XRAM
0170 ; scrn = address of screen RAM
0180 ; bank = a number from 1-8
0190 ;         signifying which "bank"
0200 ;         to load/save to/from.
0210 ;--------------------------------
0220 ;
0230 FROM =  $E0
0240 TO   =  $E2
0250 PORTB = $D301
0260 ;
0270      *=  $0600
0280 START
0290      PLA           # of args
0300      PLA           ignore high byte
0310      PLA           this is the one
0320      BNE LOAD      make SCRN the TO
0330 ;else make SCRN the FROM
0340 SAVE
0350      PLA
0360      STA FROM+1
0370      PLA
0380      STA FROM
0390      PLA           high of BANK
0400      PLA           low of BANK
0410      JSR WHERE
0420      STX TO
0430      STY TO+1
0440      JMP ACTION
0450 LOAD
0460      PLA
0470      STA TO+1
0480      PLA
0490      STA TO
0500      PLA           high of BANK
0510      PLA           low of BANK
0520      JSR WHERE
0530      STX FROM
0540      STY FROM+1
0550 ACTION
0560 ;here we actually move the data
0570      LDX #$1E      # of pages
0580 LOOP1
0590      LDY #0
0600 LOOP2
0610      LDA (FROM),Y
0620      STA (TO),Y
0630      INY
0640      BNE LOOP2
0650      DEX
0660      BEQ DONE
0670      INC FROM+1    next page of RAM
0680      INC TO+1
0690      JMP LOOP1
0695 DONE
0700      LDA #$FD      restore it to
0710      STA PORTB     normal status.
0720      RTS           back to BASIC
0730 WHERE
0740 ; decide if the address is $4000
0750 ; or $5E00. Enable accordingly.
0760      TAX           save it for later
0770      CLC
0780      ROR A         odd or even?
0790      BCC EVEN
0800 ODD
0810 ; was an odd number, so the bank
0820 ; is 1, 3, 5 or 7. That means
0830 ; it's located at $4000.
0840      JSR ENABLE
0850      LDX #$00
0860      LDY #$40
0870      RTS
0880 EVEN
0890 ; set up for pointing to $5E00.
0900      JSR ENABLE
0910      LDX #$00
0920      LDY #$5E
0930      RTS
0940 ENABLE
0950      DEX           1-8 now 0-7
0960      LDA INSTR,X
0970      STA PORTB
0980      RTS
0990 INSTR
1000      .BYTE $E1,$E1,$E5,$E5
1010      .BYTE $E9,$E9,$ED,$ED
```

Tips on Modem Use

Modems aren't particularly difficult to set up and use. On the other hand, it's relatively easy to foul up your modem's operation through carelessness or lack of knowledge. This article will serve as a guide in setting up and using your modem. If you observe the procedures and precautions mentioned herein, you'll save yourself a lot of trouble later on.

odem

by Michael A. Banks

Making Connections by the Book

As with any electronic device, it is important that you connect your modem properly and use it under practical operating conditions. Study the manual that comes with your modem to assure proper connection, and consult with your computer store and the modem manufacturer's customer-support department if necessary.

Connectors and Cables

The cables (and the connectors used with those cables) that connect a computer's serial port with a modem are obviously very important elements in the data transfer chain. Like serial ports, connectors and cables used with serial ports must conform to the RS-232C standard.

Connectors

There are two types of RS-232 connectors—nine- and 25-pin—and these may be male or female. Nine- and 25-pin connectors are known as DB-9 and DB-25 connectors, respectively. Each type has numbered pins (very important if you intend to make your own cables—saves a lot of messing around with a continuity tester).

With some equipment you may find that you have what is called a "gender problem."

DB connectors can be found on the serial port of your micrcomputer and modem, and at either end of the connecting cable. (A connecting cable is typically a "ribbon cable"—a flat cable with multiple connectors.)

Mixing Connectors

DB-25 and DB-9 connectors can be used at opposite ends of a cable if necessary (as when a computer's serial port has a DB-9 connector and its modem has a DB-25 connector). All that's required for the connection to be successful is that the pins on each connector be properly wired (i.e., the wire on each numbered pin on the DB-9 connector should be connected to the correspondingly numbered pin on the DB-25 connector).

Connector and Cable "Gender"

Incidentally, there's a standard that dictates that the female version of a DB connector should be used only on modems, while the male version should be used on computer serial ports. Thus, a "standard" RS-232C cable will have a male connector on one end (to connect with the modem), and a female connector on the other end (to connect with the computer).

Unfortunately, not all manufacturers follow this standard regarding the gender of their serial ports. So, with some equipment you may find that you have what is called a "gender problem." (No sex-change jokes, please—this is serious stuff!) When this is the case, you'll have to buy or make an appropriate cable with both female or both male connectors. Or, you can obtain what are called "gender changes" to change the "sex" of one end of the cable.

Telephone Plug/Jack Types

Before plugging a telephone line's modular plug into a modem, make sure of the plug's type. Most modems are designed with modular jacks, but modular (also designated "RJ") jacks and plugs come in more than one variety. Some are cross-compatible, and some aren't.

Generally, a home or single-line business telephone system uses RJ-11 plugs, and these present no problem—even if the plugs are set up to provide dial light power to a "Trimline" phone or other lighted-dial telephone sets. (The only danger in using an RJ-11 plug that provides power for a lighted dial with a modem is if the modem is set up to operate with RJ-12 or RJ-14 plugs. See below for more information on these plugs.)

RJ-41 and RJ-45S plugs are also "safe" to use with most all modems; the

exceptions may be modems which have RJ-12 or RJ-14 plugs. See your modem's documentation for details.

If your telephone system is a multiple-line or "key" telephone system, you must have a modem that is capable of interfacing with RJ-12 or RJ-13 plugs (such as a Hayes Smartmodem 2400). The modem you use must also be software-switchable to RJ-12/RJ-14 operation.

Getting "Set" DIP Switch Settings

Most modems have user-accessible DIP switches (although the relative accessibility varies from modem to modem). DIP switches are used to set various attributes of a modem, such as whether it waits for a carrier detect before going online, etc.

Some software packages require that certain modem attributes be set to a specific state. If your software has decent documentation, it will tell you which states must be set; in which case all you have to do is refer to your modem's documentation to find out which DIP switches are used to set the attributes in question. (Some software manuals will even tell you how to set each DIP switch on the more popular modem brands. Too, some modem manuals provide specific instructions on DIP switch settings for certain software packages.)

Telephone Company Regulations

Local telephone company regulations may vary, but in general the following rules are in effect:

Your telephone company should be notified that you will be connecting an FCC-registered device to your telephone line before you connect it, and that you will be disconnecting the modem when you disconnect it permanently.

You cannot connect a direct-connect modem to a pay telephone, nor to a party line.

Ventilation and Heat

Don't use a modem as a bookshelf or repository for other materials. While some external modems are designed to serve as a resting place for a telephone set, they aren't designed to be smothered by papers, disks, etc. A modem's electronic components generate heat, which must be dissipated; too much heat buildup can interfere with proper operation of the modem. Therefore, heat vents—as well as most of the top of the modem—should not be covered.

Use the Switch

If your modem is equipped with a power switch, use it to turn the modem off and on. Leaving the switch in the "ON" position and just plugging and unplugging the modem's power supply is not a good idea; this can occasionally create power surges or current overload.

Don't use a modem as a bookshelf or repository for other materials.

When changing the battery in a battery-powered modem, the power switch should be in the "OFF" position, for the same reasons.

Overloading Circuits

Don't plug your modem into an overloaded or faulty circuit. Aside from the fire hazard this creates, overloaded circuits often have low voltage, and low voltage can cause excess heat and poor performance in your modem. (Overloaded circuits are typically those with too many electrical devices plugged into them.)

Surge Protectors

Surge protectors (also called "spike protectors") are an excellent investment. The purpose of a surge protector is to protect an electronic device from surges in a power or telephone line. Such surges are common during thunderstorms and during periods when electrical power consumption is particularly heavy. Power-line surge protectors come in a variety of styles, but all operate in the same manner. Placed in the circuit between your computer and/or modem and the wall outlet, they contain capacitors which absorb and then bleed off excess power. Note that power-line surge protectors come in several configurations. Some are simply small cylinders or cubes and offer only one receptacle. Others are large rectangular boxes which mount on the wall in place of the wall receptacle's cover. These usually offer more than one receptacle. Some of the better surge protectors not only provide protection against power surges, but also filter "line noise," and provide a circuit breaker for protection against current overload. Telephone-line surge protectors operate on the same principle as power-line surge protectors. Installed between a modem and its telephone line, a telephone-line surge protector absorbs then slowly discharges potentially damaging voltage spikes.

Weather Conditions

Never use your modem during a severe thunderstorm, nor at any time you observe lightning. Lightning is a guaranteed source of power surges in both AC power lines and telephone lines and, unless you have a surge protector on both your modem's telephone line and power line, there's an excellent chance that your modem and computer will be "zapped" by a current surge. (Even with surge protectors, there's no guarantee that lightning won't damage your equipment.)

References

There are several excellent books on using modems that expand upon these topics with technical information. These include:

The Modem Book, by Michael A. Banks (Brady Books, 1988); *Understanding Data Communications*, by George E. Friend, et. al., (Howard W. Sams & Co., 1987); and *Communications and Networking for the IBM PC & Compatibles*, by Larry Jordan and Bruce Churchill (Brady Books, 1987).

Check with your local computer store for information on ordering these books.

48k disk or cassette

The Magic Of Tesselations -Part II

by Allan Moose
and Marian Lorenz

A tesselation or tiling is the complete covering of a flat surface by one or more figures in a pattern, with no overlapping of the figures and no open spaces. As we discussed last month, tesselations are commonly found in linoleum patterns, parquet floors or fabrics.

The theme of distorting a basic tile shape as it is drawn in successive rows forms the basic idea behind many of M.C. Escher's drawings.

All of the tilings we discussed, and many of the tilings found around one's home, share the property of being periodic. A tesselation is periodic if you can shift the drawing without rotation or reflection to a new position where all outlines again fit exactly. While there are an infinite number of shapes that will tesselate periodically, periodic tilings by no means exhaust all of the possible ways to cover a plane surface. In this article we will present two programs that illustrate more intricate tilings. The first draws a non-periodic tiling that has rotational symmetry. The second covers the screen with tiles that are gradually deformed as each successive row is drawn on the screen.

One of the charming things about the study of tilings is that it bridges the gap between art and science. Mathematicians have related tilings to group theory, which is the abstract study of symmetry. Physicists study tesselations to gain insights into the formation of crystals, and, of course, the famous artist M. C. Escher made frequent use of tilings in his work. In developing our programs we drew upon ideas from all of these fields. In particular, we have made use of the ideas of translation and rotation of coordinates as a way of writing short programs that you may easily modify.

In order to illustrate tesselations with rotational symmetry, the basic tile used is a diamond which is based on a 30-60-degree right triangle.

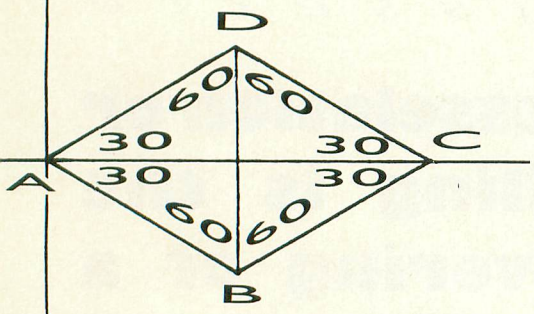


Figure 1

Previously, we introduced the use of a "local coordinate system" (LCS). The idea behind a LCS is that if you want to repeatedly draw the same figure on the CRT screen, the most efficient way to do it is to represent the coordinates of the vertices of the figure (points A,B,C,D in Figure 1) in terms of a hypothetical coordinate system. Then to draw the figure on the screen all you do is position the origin of the LCS where you want it and draw. In this way, the same subroutine can draw all the tiles you need.

In the LCS, the coordinates of the vertices of Figure 1 are:

A = 0, 0
B = 17.32, -10
C = 34.64, 0
D = 17.32, 10

To make a tesselation with rotational symmetry, we want to draw this diamond in a circular pattern so that the first row will look like:

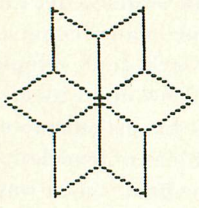


Figure 2

The next circular row must fit around this design. The algorithm for generating the pattern can be simply stated as:

That is, for each diamond, no matter where it is, the program first reads the data numbers, then rotates the figure into the proper position, and then translates the vertex of the diamond to the proper location.

With this background, you should be able to follow the program of Listing 1. This program creates three rows of diamonds (Figure 3). You may easily modify it to add a fourth. Be sure to include a clipping routine to avoid the dreaded "cursor out of range" error!

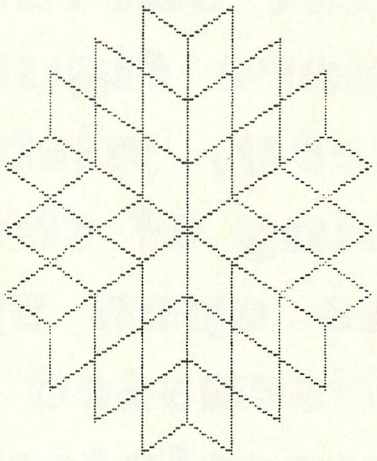
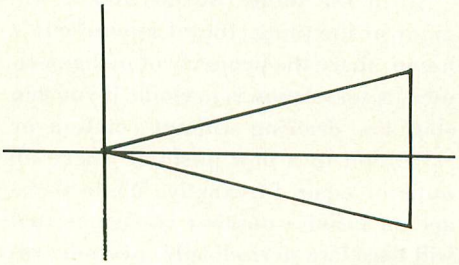
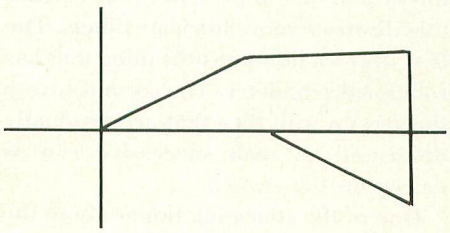


Figure 3

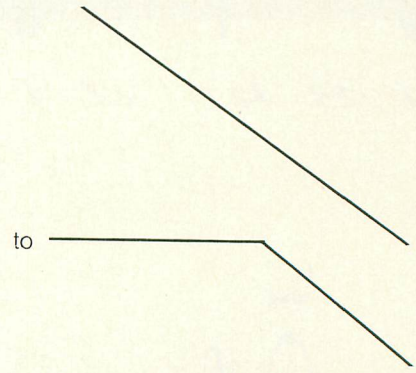
Of course, you will want to experiment with more than just diamond tiles. An excellent place to start is with triangle tilings. The reason is that each tile can be changed by distorting the legs. For example, change



to



by bending each leg from



to

Actually you can be more imaginative than this because a triangle can be distorted in an infinite number of ways to yield a figure capable of being used in a rotational tesselation.

The theme of distorting a basic tile shape as it is drawn in successive rows forms the basic idea behind many of M. C. Escher's drawings. For example, birds might gradually lose their shapes and become checkerboarded fields of hay or even metamorphose completely into fish as in "Sky and Water1." Our second program illustrates this gradual metamorphosis of one shape into another. In addition to being indebted to M. C. Escher for inspiration, we also must credit Douglas Hofstadter who, several years ago, devoted a column of "Metamagical Themas" in *Scientific American* to "Parquet Deformations."

The basic idea is that simple geometric shapes which can tile the plane are slowly deformed as they move across or down the plane. Deformations may be created with a number of simple techniques such as:

1. Lengthening or shortening a line.
2. Introducing a "hinge" into a line segment so that it can flex.
3. Rotating a line or a group of lines that form a natural sub-unit.
4. Introducing a small "bump" or tooth into a line segment.

By using one of these techniques and allowing it to continue long enough, such deformations can have unexpected results; one outcome being that tiles at the end of the work bear little or no resemblance to those at the beginning.

In order to keep our program simple, we restricted it to drawing a diamond and flexing the sides of the tile "in" or "out" to deform it. There are, of course, many other methods of deform-

ing tiles, just as there are many other shapes that lend themselves to deformation. Here is a chance to exercise your creativity by building upon the ideas in this program.

It is evident that drawing a tesselation in which the shape of the tile is changed before each row is drawn is more involved than simply drawing the same tile many times. We have approached this problem by introducing what at first

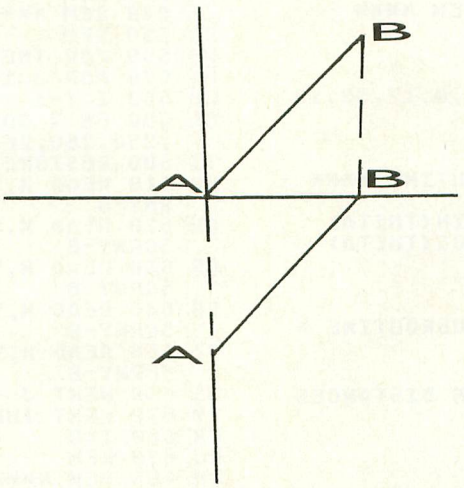
Physicists study tesselations to gain insights into the formation of crystals.

glance may seem like an unnecessary complication. First, we note that many shapes which can tile a surface have some sort of rotational symmetry. This means that if you rotate the shape around its center through some fraction of a circle (½, ⅓, ⅙, etc.) you get back the original shape. If this is the case, and it certainly is with the diamond, then we need only specify part of the shape, say two sides, and let the computer rotate that part as often as necessary to close the shape. You should visualize this rotation as taking place in the LCS. The rotation routine necessary to do this is the same as the rotation subroutine in Listing 1, lines 140 and 150. If we must rotate the part of the shape N times to produce a closed figure, then the angles through which we must rotate it are multiples of 360/N. Having constructed the tile in a LCS, it may be easily translated to the proper positions and plotted on the screen.

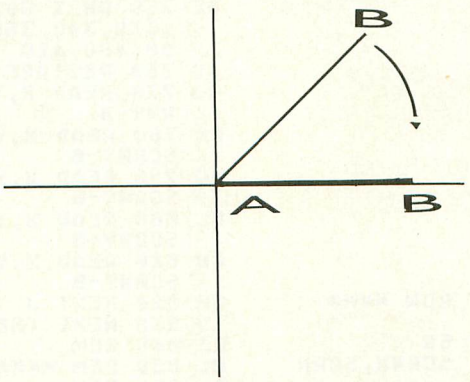
Introducing the extra step of putting a tile together by rotation gives us a way to deform the tiles. Conceptually, deforming a tile is rather simple. Just take each side of the tile in turn, keeping the end points stationary, move the midpoint alternately in toward the center or out away from the center one unit at a time. The problem is in determining where to move the midpoint to. That is, given the coordinates of the end points, what are the coordinates of the point one, two, or three units closer to, or farther from the tile's center than the line's midpoint? If the line happens to be horizontal or vertical, then there is no problem. For example, a horizontal line's midpoint has the same y-coordinate as its end points. It has an x-coordinate equal to the average of the x-coordinates of the ends. Moving the midpoint toward or away from the center is a matter of moving it up or down. That is, the x-coordinate stays the same and the y-coordinate increases or decreases by one. If the line is diagonal we can still find the midpoint easily enough. However, moving it is the hard part, as the distance and direction of movement depend entirely on the inclination of the line segment.

It would be much easier for the purposes of this exercise if we could make each side horizontal long enough to deform it and then put it back where it belongs. Fortunately, we already have the tools available to do just that—local coordinates and rotation. In mathematical terms we want to:

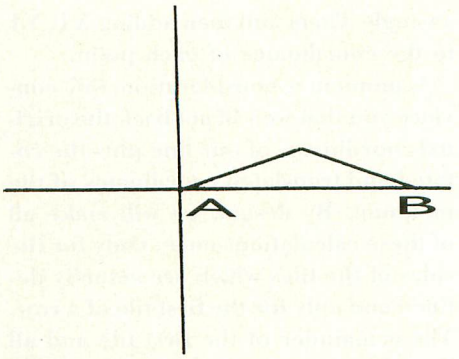
- Translate each line segment to the origin.



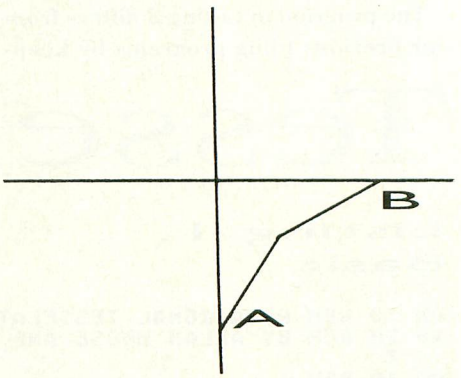
- Rotate it onto the positive x-axis.



- Deform it as explained above.



- Rotate and translate it back into position.



For us the procedure is as follows:

1. Take each of the defined sides in turn. Pick one end point and call it X1, Y1. Call the other end X2, Y2.
2. If we shift the origin of the LCS to the point at X1, Y1 then the other point will have the coordinates (X2-X1), (Y2-Y1).
3. Find the inclination of the line or the angle theta which it makes with the x-axis. If the line is vertical, then THETA = 90. If the line slopes towards the right (x2 > x1), then THETA = ARCTAN ((Y2-Y1)/(X2-X1)). If the line slopes towards the left (X2 < X1), then THETA = 180 + ARCTAN((Y2-Y1)/(X-2-X1)).
4. Rotate both end points, in terms of their coordinates relative to the first point, through this angle. The point 0,0 will not move, of course, but the other point should now have a y-coordinate of 0, meaning the line is now lying horizontally on the x-axis.
5. Finding and deforming the midpoint is now trivial. Its x-coordinate will be half the length and its y-coordinate plus or minus 1, 2, 3, etc.
6. Now all we need do is move the deformed line back where it belongs. This is just a matter of rotating through

an angle Theta and then adding X1, Y1 to the coordinates of each point.

A moment's consideration will convince you that we will get back the original coordinates of our line plus the rotated and translated coordinates of the midpoint. By design, we will make all of these calculations once. Only for the sides of the tiles which are actually defined and only for the first tile of a row. The remainder of the first tile and all the other tiles in that row are derived from the defined sides by rotation and shifting as usual.

The program in Listing 2 differs from our previous tiling programs by keeping the coordinate values of the vertices in an array. Two arrays are maintained. The first stores the original vertex coordinates. The second, larger one, holds the coordinates of the deformed tile.

The data necessary for drawing the tiles is given in lines 140 and 150. This means that to change the shape of your basic tile you need only change one or two program lines. In fact, it turns out that you don't have to change the tile shape in order to change the design drawn by the computer. Try specifying the diamond tile by two vertices and four rotations:

```
140 DATA 2, 4
150 DATA 0, -8, 8, 0
```

rather than by three vertices and two rotations. When the number of vertices is odd, line 850 will flex the sides alternately "in" or "out." When the number of vertices specified is even, all sides will be flexed "in."

It is evident that changing the tiling produced by the second program is a simple task. Because of this the program is great for experimentation! Some suggestions are to try square, rectangular, or hexagonal tiles and change the type of deformations used.

Tesselations

Listing 1. Basic

```
DX 10 REM ROTATIONAL TESSELATION
NV 20 REM BY ALLAN MOOSE AND MARIAN LOREN
   Z
BB 40 REM
JN 50 REM **** INITIALIZE SYSTEM ****
BD 60 REM
RX 70 GRAPHICS 24:COLOR 1
CS 80 DEG :A=0:B=0
OI 90 DATA 0,0,17.32,-10,34.64,0,17.32,10
   ,0,0
ON 100 GOTO 450
QO 110 REM
PJ 120 REM **** ROTATION SUBROUTINE ****
QS 130 REM
AH 140 XPRIME=X*COS(THETA)-Y*SIN(THETA)
WF 150 YPRIME=X*SIN(THETA)+Y*COS(THETA)
IG 160 SCRNX=160+XPRIME
VX 170 SCRNY=96-YPRIME
ZN 180 RETURN
WM 200 REM **** END ROTATION SUBROUTINE *
   ***
QP 210 REM
QJ 220 REM **** SET TRANSLATION DISTANCES
    FOR SECOND ROW ****
UX 230 A=17.32:B=10:RETURN
OT 240 A=0:B=20.32:RETURN
WJ 250 A=-17.32:B=10:RETURN
WO 260 A=-17.32:B=-10:RETURN
MD 270 A=0:B=-20.32:RETURN
TR 280 A=17.32:B=-10:RETURN
ET 290 REM **** SET TRANSLATION DISTANCES
    FOR THIRD ROW ****
XN 300 A=34.64:B=0:RETURN
WZ 310 A=34.64:B=20:RETURN
WA 320 A=17.32:B=30:RETURN
EE 330 A=0:B=40:RETURN
XO 340 A=-17.32:B=30:RETURN
YU 350 A=-34.64:B=20:RETURN
XO 360 A=-34.64:B=0:RETURN
ZC 370 A=-34.64:B=-20:RETURN
YB 380 A=-17.32:B=-30:RETURN
WF 390 A=0:B=-40:RETURN
UJ 400 A=17.32:B=-30:RETURN
VL 410 A=34.64:B=-20:RETURN
QT 420 REM
CF 430 REM **** PLOT THE FIRST ROW ****
QX 440 REM
VY 450 FOR THETA=0 TO 360 STEP 60
US 460 READ X,Y:GOSUB 140:PLOT SCRNX,SCRN
   Y
QI 470 READ X,Y:GOSUB 140:DRAWTO SCRNX,SC
   RNY
QK 480 READ X,Y:GOSUB 140:DRAWTO SCRNX,SC
   RNY
QM 490 READ X,Y:GOSUB 140:DRAWTO SCRNX,SC
   RNY
PV 500 READ X,Y:GOSUB 140:DRAWTO SCRNX,SC
   RNY
TJ 510 RESTORE 90
TN 520 NEXT THETA
QW 530 REM
ZE 540 REM **** PLOT THE SECOND ROW ****
RA 550 REM
WB 560 FOR THETA=0 TO 360 STEP 60
MS 570 FOR J=1 TO 2
QU 580 I=I+1
CA 590 ON I GOSUB 280,230,230,240,240,250
   ,250,260,260,270,270,280
TI 600 RESTORE 90
QB 610 READ X,Y:GOSUB 140:PLOT SCRNX+A,SC
   RNY-B
CN 620 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
CP 630 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
CR 640 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
CT 650 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
GS 660 NEXT J
TY 670 NEXT THETA
FX 680 I=0
RJ 690 REM
NH 700 REM **** PLOT THE THIRD ROW ****
QU 710 REM
VV 720 FOR THETA=0 TO 360 STEP 60
NC 730 FOR J=1 TO 3
QO 740 I=I+1
QE 750 ON I GOSUB 410,300,310,310,320,330
   ,330,340,350,350,360,370,370,380,390,3
   90,400,410
TV 760 RESTORE 90
QO 770 READ X,Y:GOSUB 140:PLOT SCRNX+A,SC
   RNY-B
DA 780 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
DC 790 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
CL 800 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
CN 810 READ X,Y:GOSUB 140:DRAWTO SCRNX+A,
   SCRNY-B
GM 820 NEXT J
TS 830 NEXT THETA
RB 840 REM
MK 850 REM **** SCREEN DUMP ****
RF 860 REM
MZ 870 DIM GRAF$(200)
RS 875 GOSUB 1000
```

```
AC 880 GRAF$(1)=CHR$(0):GRAF$(200)=CHR$(0
   ):GRAF$(2)=GRAF$
MO 890 LPRINT CHR$(27);CHR$(65);CHR$(8)
GR 900 SCRNMEM=PEEK(88)+PEEK(89)*256
LU 910 MEMLOC=SCRNMEM+40*191
TF 920 HIBYTE=INT(ADR(GRAF$)/256)
EW 930 LOBYTE=ADR(GRAF$)-HIBYTE*256
EP 940 POKE 203,LOBYTE:POKE 204,HIBYTE
AQ 950 FOR SCRNCOL=MEMLOC TO MEMLOC+39
CL 960 DUMP=USR(1536,SCRNCOL)
EH 970 LPRINT CHR$(27);CHR$(75);CHR$(200)
   ;CHR$(0);GRAF$
KB 980 NEXT SCRNCOL
OQ 990 END
JE 1000 RESTORE 1040
HD 1010 FOR K=0 TO 43
BY 1020 READ ML:POKE 1536+K,ML
FU 1030 NEXT K
NZ 1040 DATA 104,104,141,15,6,104,141,14,
   6,160,4,162,192,173,0,0,202,240,24
EF 1050 DATA 145,203,200,216,173,14,6,56,
   233,40,141,14,6
ZX 1060 DATA 144,3,76,13,6,206,15,6,76,13
   ,6,96
AU 1070 RETURN
```

Listing 2.

Basic

```
SS 10 REM **** ESCHER VERSION 5 ****
NV 20 REM BY ALLAN MOOSE AND MARIAN LOREN
   Z
BB 40 REM
NI 50 GOTO 140
FW 60 REM **** ROTATION SUBROUTINE ****
BE 70 REM
MN 80 XPRIME=X*COS(THETA)-Y*SIN(THETA)
IL 90 YPRIME=X*SIN(THETA)+Y*COS(THETA)
YX 100 RETURN
QO 110 REM
XK 120 REM **** INITIALIZE SYSTEM, VARIAB
   LES, ARRAYS ****
QS 130 REM
DL 140 DATA 3,2
BD 150 DATA 0,-8,8,0,0,8
NT 160 GRAPHICS 24:DEG :COLOR 1
IJ 170 READ NUMVERTS,NUMROTS
NP 180 DIM ARRAY1(NUMVERTS,2),ARRAY2(NUMV
   ERTS*2-1,2)
LE 190 THETAINC=360/NUMROTS
HN 200 FOR VERTEX=1 TO NUMVERTS
QV 210 READ XCOORD,YCOORD
IV 220 ARRAY1(VERTEX,1)=XCOORD:ARRAY1(VER
   TEX,2)=YCOORD
ZX 230 NEXT VERTEX
SP 240 REM **** DETERMINE HEIGHT AND WIDT
   H OF TILE ****
HX 250 FOR VERTEX=1 TO NUMVERTS
CD 260 IF ABS(ARRAY1(VERTEX,1))>XMAX THEN
    XMAX=ABS(ARRAY1(VERTEX,1))
II 270 IF ABS(ARRAY1(VERTEX,2))>YMAX THEN
    YMAX=ABS(ARRAY1(VERTEX,2))
AH 280 NEXT VERTEX
BJ 290 HEIGHT=2*YMAX:WIDTH=2*XMAX
SI 300 MAXCOLS=INT(320/WIDTH)-1
UJ 310 MAXROWS=INT(192/HEIGHT)-1
SR 320 INITIALX=XMAX+5:INITIALY=YMAX+5
QU 330 REM
YF 340 REM **** PLOT THE FIRST ROW OF TIL
   ES ****
QY 350 REM
HD 360 FOR COL=1 TO MAXCOLS
UP 370 FOR THETA=0 TO 360 STEP THETAINC
IE 380 FOR VERTEX=1 TO NUMVERTS
ZU 390 X=ARRAY1(VERTEX,1):Y=ARRAY1(VERTEX
   ,2)
VO 400 GOSUB 80
LV 410 SCRNX=INITIALX+(COL-1)*WIDTH+XPRIM
   E
RM 420 SCRNY=INITIALY-YPRIME
KH 430 IF VERTEX=1 THEN PLOT SCRNX,SCRNY:
   GOTO 450
PL 440 DRAWTO SCRNX,SCRNY
AD 450 NEXT VERTEX
TU 460 NEXT THETA
UN 470 NEXT COL
RF 480 REM
LY 490 REM **** DRAW SUCCEEDING ROWS OF T
   ILES ****
QQ 500 REM
NA 510 FOR ROW=2 TO MAXROWS
TQ 520 YOFFSET=YOFFSET+1
UU 530 GOSUB 730
HB 540 FOR COL=1 TO MAXCOLS
UN 550 FOR THETA=0 TO 360 STEP THETAINC
SP 560 FOR VERTEX=1 TO NUMVERTS*2-1
BJ 570 X=ARRAY2(VERTEX,1):Y=ARRAY2(VERTEX
   ,2)
WF 580 GOSUB 80
MM 590 SCRNX=INITIALX+(COL-1)*WIDTH+XPRIM
   E
EQ 600 SCRNY=INITIALY+(ROW-1)*HEIGHT-YPRI
   ME
PI 610 IF VERTEX=1 THEN PLOT SCRNX,SCRNY:
   GOTO 660
YA 620 REM CLIPPING ROUTINE
KP 630 IF SCRNX<0 OR SCRNX>319 THEN GOTO
   940
JU 640 IF SCRNY<0 OR SCRNY>191 THEN GOTO
   940
PP 650 DRAWTO SCRNX,SCRNY
AH 660 NEXT VERTEX
TY 670 NEXT THETA
UR 680 NEXT COL
FP 690 NEXT ROW
QG 700 GOTO 940
QU 710 REM
GL 720 REM **** SUBROUTINE TO CREATE AN A
   RRAY OF DEFORMED TILES ****
QY 730 REM
HQ 740 FOR I=1 TO NUMVERTS-1
TP 750 X1=ARRAY1(I,1):ARRAY2(2*I-1,1)=ARR
   AY1(I,1)
MP 760 X2=ARRAY1(I+1,1)
XP 770 Y1=ARRAY1(I,2):ARRAY2(2*I-1,2)=ARR
   AY1(I,2)
NR 780 Y2=ARRAY1(I+1,2)
CH 790 X=X2-X1:Y=Y2-Y1
UE 800 IF X=0 THEN THETA=(-1)*SGN(Y)*90:G
   OTO 850
QO 810 THETA=ATN(Y/X)
OA 820 IF X<0 THEN THETA=180+THETA
FM 830 THETA=-THETA:REM ROTATE TOWARD X-A
   XIS
WA 840 GOSUB 80
XL 850 X=XPRIME/2:Y=YOFFSET*(-1)^(I+1)
LX 860 THETA=-THETA:REM ROTATE BACK INTO
   POSITION
WG 870 GOSUB 80
FD 880 ARRAY2(2*I,1)=XPRIME+X1:ARRAY2(2*I
   ,2)=YPRIME+Y1
GQ 890 NEXT I
IV 900 REM **** COMPLETE ARRAY2 ****
SU 910 FOR J=1 TO 2:ARRAY2(2*NUMVERTS-1,J
   )=ARRAY1(NUMVERTS,J):NEXT J
ZJ 920 RETURN
MH 930 REM **** SCREEN DUMP ****
UY 940 GOSUB 1080
MW 950 DIM GRAF$(200)
ZZ 960 GRAF$(1)=CHR$(0):GRAF$(200)=CHR$(0
   ):GRAF$(2)=GRAF$
ML 970 LPRINT CHR$(27);CHR$(65);CHR$(8)
HH 980 SCRNMEM=PEEK(88)+PEEK(89)*256
MK 990 MEMLOC=SCRNMEM+40*191
MZ 1000 HIBYTE=INT(ADR(GRAF$)/256)
FW 1010 LOBYTE=ADR(GRAF$)-HIBYTE*256
MY 1020 POKE 203,LOBYTE:POKE 204,HIBYTE
MP 1030 FOR SCRNCOL=MEMLOC TO MEMLOC+39
PW 1040 DUMP=USR(1536,SCRNCOL)
RG 1050 LPRINT CHR$(27);CHR$(75);CHR$(200
   );CHR$(0);GRAF$
AK 1060 NEXT SCRNCOL
FI 1070 END
JL 1080 RESTORE 1120
IB 1090 FOR K=0 TO 43
BU 1100 READ ML:POKE 1536+K,ML
FQ 1110 NEXT K
NV 1120 DATA 104,104,141,15,6,104,141,14,
   6,160,4,162,192,173,0,0,202,240,24
EB 1130 DATA 145,203,200,216,173,14,6,56,
   233,40,141,14,6
ZT 1140 DATA 144,3,76,13,6,206,15,6,76,13
   ,6,96
AQ 1150 RETURN
```

Boot Directory

by Bill Bodenstein

It finally happened! Sooner or later, I knew it would. It was inevitable that my 1050 disk drive would break down. Naturally, it quits on me between terms of my summer course (I'm taking COBOL—why, I'm not sure), when I'm a little short of money. And of course it conks out in the middle of various programming projects—most, I might add, involve disk I/O. Well, I remained undaunted in the heinous face of disaster, and thanks to the help of a friend (I'd name my friend here, but he'd no doubt sue me for libel or something), I completed one of these projects.

Utility

48k disk

The program is called **Boot-Directory**. You'll find the BASIC code in Listing 1 and the MAC/65 source code in Listing 2. It works on any single-density Atari DOS 2.0 or 2.5 disk, and takes up no space at all—it's stored in the currently unused third boot sector. Just type in and run Listing 1, insert the disk you want modified, hit RETURN, and *voila!* The **Boot-Directory** machine-language routine will load and run automatically every time you boot that disk. Re-run the BASIC program to modify other disks. To remove **Boot-Directory**, just go to DOS and use the Write DOS Files option.

What's Going On?

Oh, maybe I ought to explain exactly what **Boot-Directory** is and how to use it. When you boot a DOS 2.0/2.5 disk, the three boot sectors (1-3) are loaded into memory by the operating system. Then control is turned over to the boot routine beginning at the seventh byte from the first sector. From here, a short machine-language program loads the File Manager System contained in the DOS.SYS file into memory, and control is reverted back to the operating system. What we'll do is cleverly insert a jump instruction to the code beginning at the third sector (**Boot-Directory**) at the seventh byte in the first sector, then exit to $0714, the start of the DOS boot-load routine. For more info, read *Inside Atardos* by Bill Wilkinson.

How It Works.

Okay, now we know how **Boot-Directory** gets control, but what does it do? Simple. It lists all the files from the directory; it starts at sector 361 and prints the names of all undeleted, existing files.

If you hold down the SELECT key, **Boot-Directory** will, upon completion, loop until you hit RESET. Thus, if you're just looking for a file, insert a disk while holding down SELECT and wait until the directory is completely listed. If you do not see the file you want, insert another disk and press RESET. You could, of course, reboot by powering-down, but you never know when the computer might not want to power-up! (Remind me to tell you about the time my computer went on the blink....) The entire **Boot-Directory** routine is limited to one sector, just 128 bytes. Not much space for super-neato features, I'm afraid. But, nevertheless, as you can see from examining Listing 2, I did manage—just barely—to squeeze the entire code into sector 3. If you modify the MAC/65 source code, don't forget the 128-byte maximum size! As for me, I won't be modifying anything for a while. Anyone know a place that repairs disk drives cheap?

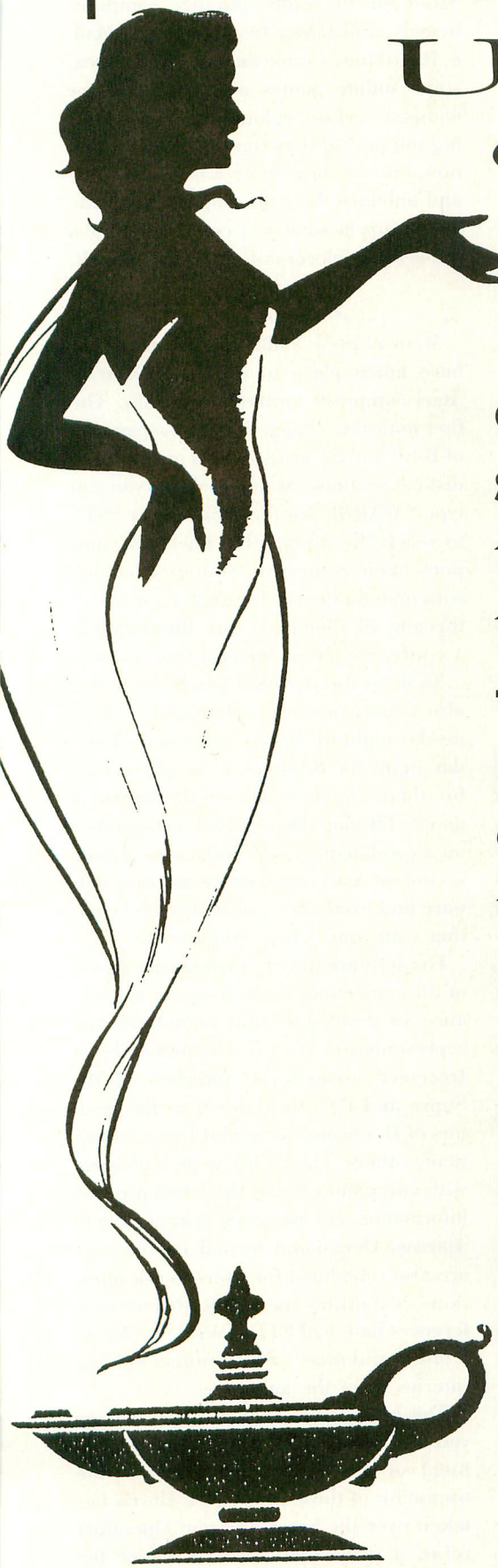
Listing 1.: Basic

```
JO 10 REM ** BOOT-DIRECTORY Maker **
GX 11 REM by Bill Bodenstein
JA 12 REM COPYRIGHT 1988 BY ANALOG COMPUT
   ING
BQ 20 DIM X$(25),A$(1):FOR X=1 TO 25:READ
    A:X$(X,X)=CHR$(A):NEXT X
YI 24 IF PEEK(1799)=20 AND PEEK(1800)=7 T
   HEN 30
HS 26 ? "Atari File Manager System not in
   ":? "memory. Reboot with DOS 2.0 or 2.
   5.":STOP
UD 30 ? "    *** BOOT-DIRECTORY Maker ***
   ":? :? "This program will write a shor
   t M/L"
GS 40 ? "routine (BOOT-DIRECTORY) to sect
   or 3":? "which will list all files nam
   es in":? "the directory at boot-up."
DP 50 ? :? "If just searching for a file,
    hold":? "down (SELECT) until BOOT-DIR
   ECTORY is"
UX 60 ? "executed, insert a new disk, and
   ":? "press (SYSTEM RESET) to reboot di
   sk."
EU 70 RESTORE 500:SUM=0:FOR X=1536 TO 153
   6+124:READ N:SUM=SUM+N:POKE X,N:NEXT X
IC 80 READ CHCKSUM:IF CHCKSUM<>SUM THEN ?
    "BAD DATA!":STOP
LN 100 ? :? "Insert a diskette to modify.
   ":? "HIT (RETURN)";:INPUT A$
YK 110 CMD=82:BUFF=(PEEK(15)+1)*256:SECT=
   1:X=USR(ADR(X$),CMD,BUFF,SECT)
BM 120 IF PEEK(BUFF+7)=0 AND PEEK(BUFF+8)
   =8 THEN ? "DISK ALREADY MODIFIED!":GOT
   O 100
RB 130 IF PEEK(BUFF+7)<>20 OR PEEK(BUFF+8
   )<>7 THEN ? "NOT A DOS 2.x DISK!":GOTO
    100
FD 140 POKE 1799,0:POKE 1800,8:? "Re-writ
   ing DOS.SYS to disk...":OPEN #1,8,0,"D
   :DOS.SYS":CLOSE #1
SE 150 ? "Writing new sector 3...":CMD=87
   :BUFF=1536:SECT=3:X=USR(ADR(X$),CMD,BU
   FF,SECT)
EC 160 ? "Disk now contains BOOT-DIRECTOR
   Y.":POKE 1799,20:POKE 1800,7:GOTO 100
BG 400 DATA 104,104,104,141,2,3,104,141,5
   ,3,104,141,4,3,104,141,11,3,104,141,10
   ,3,76,83,228
CT 500 DATA 169,104,141,10,3,169,1,141,11
   ,3,169,0,141,4,3,133,206
QL 510 DATA 169,6,141,5,3,133,207,133,84,
   238,10,3,173,10,3,201,113
MG 520 DATA 176,65,169,82,141,2,3,32,83,2
   28,162,0,161,206,240,51,48,36
BZ 530 DATA 160,5,169,11,157,72,3,196,85,
   176,4,160,23,169,9,132,85
QE 540 DATA 157,66,3,165,206,24,105,5,157
   ,68,3,165,207,157,69,3,32,86,228
UR 550 DATA 165,206,24,105,16,41,127,133,
   206,240,183,208,199
```

```
IX 560 DATA 173,31,208,201,5,240,254,169,
   0,133,84,169,20,141,7,7
QR 570 DATA 169,7,141,8,7,76,20,7
ME 580 DATA 12577
```

Listing 2: Assembly

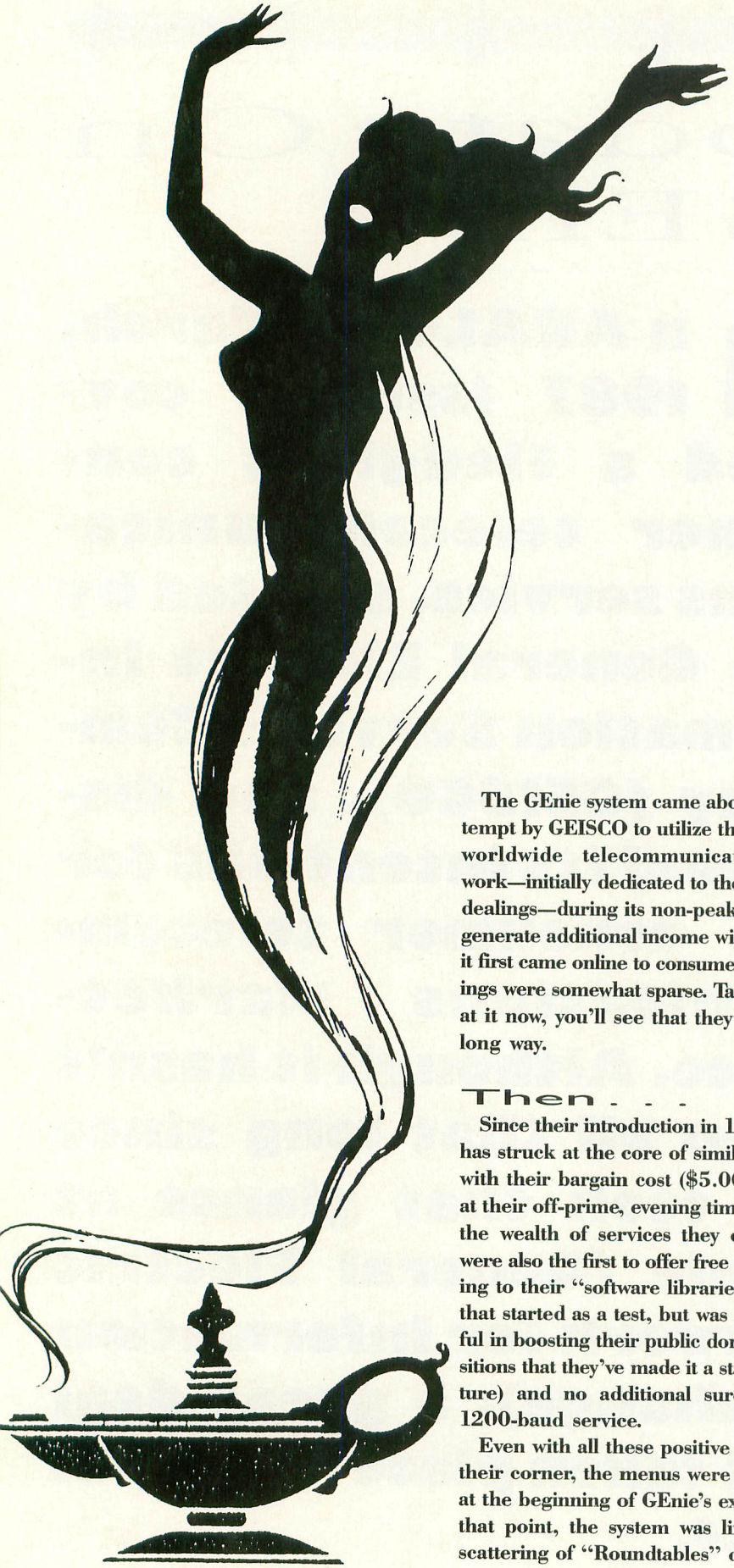
```
10 ************************************
20 **       BOOT-DIRECTORY           **
30 **       by Bill Bodenstein       **
40 **       COPYRIGHT 1988           **
45 **       BY ANALOG COMPUTING      **
50 ************************************
60 ;
70 ;This program creates a directory
80 ;lister which is executed when
90 ;disk is booted.
0100 ;
0110 ;
0120 ; EQUATES
0130 ;
0140 CURSORROW = $54
0150 CURSORCOL = $55
0160 CONSOLEPRESS = $D01F
0170 SELECTKEY = 5
0180 ;
0190 DIRSECT = $0169
0200 DIRBUFF = $0600
0210 FENTRYPTR = $CE
0220 ENTRYLENGTH = 16
0230 ;
0240 FNPLACE = 5
0250 PUTRECORD = 9
0260 PUTCHARS = 11
0270 ICCOM = $0342
0280 ICBADR = $0344
0290 ICBLEN = $0348
0300 CIO =   $E456
0310 ;
0320 READ =  82
0330 DCBCMD = $0302
0340 DCBBUF = $0304
0350 DCBSEC = $030A
0360 SIO =   $E453
0370 ;
0380 CONTBOOT = $0714 ;start of boot
0390 ;  record routine which loads
0400 ;  DOS.SYS into memory.
0410 ;
0420     *=  $0800
0430 ;
0440 ;The following code is stored in
0450 ;the 3rd boot sector and booted
0460 ;into memory from $800 to $87F.
0470 ;
0480 ;
0490 ;Set data control block to read
0500 ;sectors into directory buffer
0510 ;and set file entry pointer to
0520 ;point to the first file entry
0530 ;in directory.
0540 ;
0550 SETUPDCB
0560     LDA # <DIRSECT-1
0570     STA DCBSEC
0580     LDA # >DIRSECT
0590     STA DCBSEC+1
0600     LDA # <DIRBUFF
0610     STA DCBBUF
0620     STA FENTRYPTR
0630     LDA # >DIRBUFF
0640     STA DCBBUF+1
0650     STA FENTRYPTR+1
0660 SETROW STA CURSORROW ;start at
0670 ;  row 6, avoiding silly stuff
0680 ;  LOAD*IT puts on the screen.
0690 ;
0700 ;
0710 ;Skip to next sector and check
0720 ;if last sector in directory.
0730 ;
0740 READINSECTOR
0750     INC DCBSEC
0760     LDA DCBSEC
0770     CMP # <DIRSECT+8
0780     BCS EXITBOOTDIR
0790     LDA #READ
0800     STA DCBCMD
0810     JSR SIO
0820 ;
0830 ;
0840 ;Check status flag of entry:
0850 ;  =0 if no more files
0860 ;  >=128 if deleted file
0870 ;
0880 CHECKENTRY
0890     LDX #0
0900     LDA (FENTRYPTR,X)
0910     BEQ EXITBOOTDIR
0920     BMI NEXTENTRY
0930 ;
0940 ;
0950 ;Indent file name and put two on
0960 ;each line.
0970 ;
0980 WHERECURSOR?
0990     LDY #5
1000     LDA #PUTCHARS
1010     STA ICBLEN,X ;fname=11 chars
1020     CPY CURSORCOL
1030     BCS SETPOSITION
1040     LDY #23
1050     LDA #PUTRECORD ;causes a
1060 ;  carriage return after name.
1070 SETPOSITION STY CURSORCOL
1080 ;
1090 ;
1100 ;Tell the screen editor to print
1110 ;the 11-character file name from
1120 ;the file entry.
1130 ;
1140 PRINTFN
1150     STA ICCOM,X ;xreg=0
1160     LDA FENTRYPTR
1170     CLC
1180     ADC #FNPLACE
1190     STA ICBADR,X
1200     LDA FENTRYPTR+1
1210     STA ICBADR+1,X
1220     JSR CIO
1230 ;
1240 ;
1250 ;Skip to next 16-byte entry.
1260 ;
1270 NEXTENTRY
1280     LDA FENTRYPTR
1290     CLC
1300     ADC #ENTRYLENGTH
1310     AND #127
1320     STA FENTRYPTR
1330     BEQ READINSECTOR
1340     BNE CHECKENTRY
1350 ;
1360 ;
1370 EXITBOOTDIR
1380 ;
1390 ;If <SELECT> pressed, loop until
1400 ;<SYSTEM RESET> pressed.
1410 ;
1420 SELECTPRESSED?
1430     LDA CONSOLEPRESS
1440     CMP #SELECTKEY
1450 LOOPFOREVER BEQ LOOPFOREVER
1460 ;
1470 ;
1480 ;Fix patch made to sector 1:
1490 ;  JMP $800=>JMP $714
1500 ;and move cursor to top line
1510 ;(so LOAD*IT will be happy).
1520 ;
1530 RESETCURSOR
1540     LDA #0
1550     STA CURSORROW
1560 ;
1570 RESETJMP
1580     LDA # <CONTBOOT
1590     STA $0707
1600     LDA # >CONTBOOT
1610     STA $0708
1620 ;
1630 ;Return and let FMS take over.
1640 ;
1650 GOBOOTER JMP CONTBOOT
```



Update On GEnie

by Anay Eday

In ANALOG's March, 1987 issue, I covered a fledgling consumer telecommunications service, started by the General Electric Information Services Company (GEISCO), and discussed its intentions for the consumer telecommunications marketplace. Although it hasn't been all that long since we took that glance at GEnie (General Electric Network for Information Exchange), a great deal has taken place under its roof.



The GEnie system came about as an attempt by GEISCO to utilize their existing worldwide telecommunications network—initially dedicated to their business dealings—during its non-peak hours and generate additional income with it. When it first came online to consumers, its offerings were somewhat sparse. Taking a look at it now, you'll see that they've come a long way.

Then - - -

Since their introduction in 1985 GEnie has struck at the core of similar services with their bargain cost ($5.00 per hour at their off-prime, evening time rate) and the wealth of services they offer. They were also the first to offer free file uploading to their "software libraries" (a move that started as a test, but was so successful in boosting their public domain acquisitions that they've made it a standard feature) and no additional surcharge for 1200-baud service.

Even with all these positive features in their corner, the menus were pretty thin at the beginning of GEnie's existence. At that point, the system was limited to a scattering of "Roundtables" or RTs (the name GEnie uses to describe what many services call Special Interest Groups or SIGs) for the more popular computer brands, and basics like Electronic Mail, a Real-Time Conferencing (RTC) area, some online games and a section for computer-related columns and news. During the period between our last visit and now, they've done quite a bit to build up and enhance their offerings to bring the system up to what you could consider a full-service telecommunications network.

- - - And Now

Many of these additions and alterations have taken place to the satisfaction of Atari computer owners specifically. The first noticable change is in the separation of 8-bit and ST areas, which are now two distinct sections. At any prompt, you can type "ATARI8" for the 8-bit area or "ST" to reach the 16-bit SIG. Each SIG supports their respective computer handily with bulletin boards for ongoing message threads, file-filled software libraries and a conference area for real-time chats.

To keep the personal touch, there are also weekly meetings scheduled — Wednesday night for the ST users and Thursday night for 8-bitters. User attendance for these events is high with dozens of people filtering through each get-together on a good night. You'll find a strong cross section of Atari users, discussing new software and hardware, passing tips and chatting with Atari Corp. employees.

For software buyers a reassuring aspect of these meetings is the frequent appearance of developers and manufacturers' representatives from such companies as Intersect (makers of Interlink, QMI, Supra and FTL (prominent as the creators of Dungeon Master and Oids), among many others. The ability to go tete-a-tete with companies to get the latest product information and assistance is invaluable to Atarists. Occasional formal conferences are also scheduled for users to ask questions of industry members. Recent conferences have had FTL's president, Wayne Holder, and Atari's Sam Tramiel fielding queries from the gallery.

This brings us to another major change you'll discover, which is the active involvement of Atari Corp. personnel in the operation of these areas. Neil Harris has taken over the Sysop (System Operator) reins, a move that makes it easier for

GEnie users to get the straight scoop on Atari products and viewpoints.

To further the embracing of GEnie as their official home, Atari offers a developers' SIG, where questions and concerns of programmers can be tackled by the people who know best.

Manufacturers' Forums

Another attraction is Michtron's Roundtable (type MICHTRON from any prompt). As a strongly dedicated developer and distributor of ST software, Michtron is showing the focus of their support by linking with users to answer questions, pass on new product info (in the way of press releases and demo files within their software library) and as a follow-up on the sale of their products. Much of the latter covers the growing GFA line of software, particularly GFA BASIC. Many ST users are developing commercial software with GFA BASIC, and the Michtron RT lets them make contact with others for help.

Data Pacific, makers of the Magic Sac, a Macintosh emulation hardware/software combination for the ST also has a section on GEnie. Their new RT is a good source for compatible PD software, as well as the latest news on updates and other add-ons.

Other Additions

Outside the Atari-specific focus, GEnie has concerted their efforts toward building a well-rounded network. From their relatively bare bones start, they've consistently added to the menus to provide travel and shopping services, more online games and a wider variety of Roundtables to get people with similar interests together to discuss different computer brands and hobbies, and help manufacturers with product support. Some of these newer groups cater to such diverse subjects as scuba diving, photography (with a marketing service for professionals), taxes, working at home and writing.

The power of online services is expanding greatly, sliding away from just being a bulletin board or gathering place. Some of the offerings now provide admittance to functions that previously were accessible only to professionals.

One of these services, EAASY SABRE (American Airlines' own network), opens the airlines' reservations networks to telecomputing enthusiasts. While taking a bit of getting used to, these menu-driven services are being employed by many frequent travelers — particularly those who carry portable computers — enabling them to get quick transportation information from any phone and to stay on top of the volatile scheduling that flight traffic is governed by.

There are many other support groups and entertainment choices available, too many to cover thoroughly here. You can access the day's news headlines, send a paper mail letter, peruse movie and music reviews or even check the financial world through a gateway to the Dow Jones network. Some of these selections are surcharged, as is the case with the Dow Jones link, but you are told of that ahead of time: a "$" will precede any surcharged menu choice.

For recreation, online games are available. Some are single-player games and others, like Chess, are meant for head-to-head or group participation. One of the contests I tried, though it is still a little buggy, is an interactive Blackjack game. Factory Programming, programmers of some of Michtron's ST offerings, is creating software modules for different computer brands that assist in providing a graphics foundation to the Blackjack game.

In the ST version, you can move your mouse around to pick what table and seat to sit at, choose how to play your cards and how much to bet, all the while keeping up a conversation with any other players at the same table. Users without a compatible module can play a text version

Inside the GEnie's Bottle

One of my only complaints is directed at the surcharge for 2400 baud usage. Due to mass production and new integrated circuitry, there are many companies now offering 2400 modems for reasonable prices, making this technology accessible to the average telecommunications enthusiast. GEnie tacks on $7.50 an hour to the regular rate for this feature, which doesn't seem to make any sense on the surface, as it's more than double the cost of 1200-baud access.

As Bill Louden, GEnie's top man, explains it: "Our 2400-baud price is the same as CompuServe's. The price is more a function of its newness to the market: Costs of deployment are higher, and usage is quite low when compared to 1200 baud. Given our already low, price structure as well as the increased costs for this new technology, the argument 'twice the speed for twice the price' does not address all of the business cost issues. We are currently in over 60 cities with two 2400 baud. I cannot state what our expansion plans are; but I will state that we expect major expansion over the next two years."

Perhaps with that expansion, we can expect a trimming of the associated rates.

Where GEnie Stands

There's no question that GEnie has kept on a strongly upward pace. That is demonstrated by the fact that they've become the second most popular consumer-oriented service in the U.S. behind CompuServe (who claims a 400,000 user base), based on their 100,000 subscriber count. Looking ahead, they estimate that they add approximately 10,000 new users per month, which could bring them close to CompuServe's heels before too long. Also on the horizon, Louden figures to add more and more to GEnie's offerings: over 50 new products are slated for introduction during 1988.

In our last visit, GEnie was offering a "test drive" of the system via a toll-free number, and, doubtlessly, this sampling of the menus and operation got them to the subscriber level they're at now. Unfortunately it's no longer available, but they still offer an online sign-up, also by way of a free call.

If you dial 1-800-638-8369 from your terminal software (set it up with half duplex or local echo on), type HHH once you achieve connection, then enter GENIE at the U#= prompt, you'll enter their sign-up area. At that time, you'll see a short advertisement of what GEnie offers, then be able to enter information for initiating an account of your own.

The processing of the account was very quick when I first signed up; hopefully that's still the case. The initial sign-up costs $29.95. For that, you'll receive a copy of their new manual and two free hours of access time.

CompuServe's SIG*ATARI

by Mike Schoenbach

Eight years ago CompuServe introduced its first Atari Special Interest Group, which became known as SIG*ATARI. SIG*ATARI was the pioneer, being the first national interest group devoted exclusively to Atari computer owners. Over the past few years, CompuServe has grown to be the largest computer-information system available, having over 400,000 subscribers and offering more than 200 services to computer owners, professionals and hobbyists of all kinds.

Today, six services are available on CompuServe exclusively for the support of Atari computers and Atari computer owners. The original SIG*ATARI has been split off into four separate Atari Special Interest Groups (known as "Forums" on CompuServe) and two online databases to meet the changing needs of the Atari market. The expanded Atari coverage on CompuServe allows users to get the most information possible on any subject relating to Atari computers. It makes no difference whether you are a fanatical 8-bit computer owner, a brand-new computer owner who just purchased an Atari ST, a part-time software developer or just someone looking for help with a specific program, because CompuServe's SIG*ATARI has a lot to offer you. A whole community of people who share your same interest in Atari computing is just a phone call away!

Online Users Group

A CompuServe Forum is where people from all over the world gather electronically to discuss and learn more about a common interest. In fact, you can think of a Forum as a users group that meets 24 hours a day, seven days a week. The Atari 8-bit and Atari 16-bit Forums were both set up for users of Atari computers to communicate, share information, exchange tips, download programs and meet new people all over the world. Each Forum offers a message board for discussions, an electronic conference area for real-time global communication, and an extensive collection of files available for you to download.

The Data Libraries available in both Forums have files for all different types of interests. To help organize files better, each CompuServe Forum provides up to 18 specific Data Libraries for different file types. With upload time free of connect charges on CompuServe,

many members continue to regularly upload their newest creations for other Forum members to share.

Many Atari luminaries, including Bill Wilkinson, Steve Ahlstrom, Dan Moore, Tom Hudson, Keith Ledbetter, and more, continue to regularly visit both Forums to help answer questions and offer their knowledge to other Forum members. Many users feel embarrassed to ask what appears to them to be a "stupid" question. However, according to the Sysops, a stupid question has never been asked in the eight years since SIG*ATARI's inception.

"The friendly, helpful attitude of the entire membership base makes the new user comfortable and at home from the first time he or she signs on," says Dave Groves, an assistant Sysop of the Atari 16-bit Forum. "There is no problem too simple or too complex to get a solid solution from the experts and other users, who between them have used almost every program ever written for an Atari computer."

Participation is the key word. According to Groves,"The user base consists of members who are at the forefront of the Atari Market. We have the opinion leaders of the Atari community, the end-user public and a group genuinely concerned about the future of the Atari market. Many of our users write Atari-oriented periodicals, are leaders in major Atari Users Groups and are retail dealers. Online discussions generally lead to action."

SIG*ATARI members also provide constructive feedback to software developers and are very happy to lend a hand to other Forum members. For example, when Keith Ledbetter was ready to release his long-awaited **1030 Express** version 3.0, he sought the help of SIG*ATARI members to assist him with beta testing the program. Forum members provided Ledbetter with detailed bug descriptions as well as offered many suggestions for the final release version.

Any developers wishing to conduct a beta test online should contact the Sysops for more information.

Atari Vendor Support Forum

In February 1988, the Atari Vendors Support Forum was launched. The sole purpose of this new Forum was to create and maintain a direct link between many top third-party software manufacturers and their customers. Each participating vendor has his own message section which is used by the company and their customers to correspond with each other daily, a Data Library which offers product-help files, tutorials, patches and sometimes product updates, and an electronic conference room.

Current participants of the Vendors Forum include ICD, Inc., Intersect Software, Michtron, Regent Software, QMI, Data Pacific, Avant-Garde and *Atari Explorer Magazine*. By the time you read this, *ANALOG* and *ST Log* magazines will also have an official online support section in the Vendors Forum. Please note that other vendors maintain online support in the Atari 8- and 16-bit Forums as well. Ron Luks invites any vendors interested in setting up an official online support section to send an EasyPlex message to him (his User ID is 76703,254).

"The friendly, helpful attitude of the entire membership base makes the new user comfortable and at home from the first time he or she signs on."

Atari Programmers

Developing software for a complex machine such as the ST is no easy task. When a programmer undertakes a programming project—for fun or profit—he is mostly opening himself up for endless days of coding, more stress than anyone deserves and a great deal of hair loss (Ever wonder why programmers grow their hair so long? They know they're going to lose 25% of it per programming project). However, when the final product is released, most will agree that their time was well spent. In addition, with a minimal amount of psychotherapy, many of the side-effects of programming can also be relieved.

If you are developing a program for the Atari ST—whether you are a professional or first-time programmer—the Atari Programmers and Developers Forum on CompuServe can be a great asset to you! Participants in the Developers Forum include the entire cross section of programmers and developers in the Atari community. Professional programmers use the Developers Forum to exchange information, source code and tips with their colleagues. Amateur programmers will find a wealth of helpful information to assist them in turning the program that is in their mind to one that can be loaded into the computer.

"The Atari Developers Forum offers different things to different people," says Charles McGuinness, assistant Sysop of the Atari Developers Forum. "Software developers will find a chance to interact with each other and discuss methods and techniques for dealing with GEM and GEMDOS as well as every other aspect of the ST computer. For the amateur programmer, the Developers Forum offers the opportunity to discuss things with the pros, as well as being able to take advantage of the large library of source code that is available in the Developers Forum."

McGuinness adds that the Atari Developers Forum is the official site for obtaining updates to the Atari Developers Kit. A message section and Data Library has been set up for registered developers only (those who purchase the Atari Developers Kit). Registered developers who do not currently have access to section 7 ("Registered Developers") of the Forum should contact Cary Gee at Atari (his CompuServe User ID is 70007,2355) to gain admission. Once in, Developer Kit updates as well as other new development tools from Atari are readily available for you to download.

"The Developers Forum's usefulness does not necessarily end when you are finished writing your program," Ron Luks, primary sysop of the CompuServe Atari Forums, adds. "In addition to getting help with programming and product marketing, special restricted areas are available to developers who wish to beta

test preliminary versions of their products. These sections are set up and restricted to a small group of people who the Developers request to be admitted. This enables the developer to test and debug his software in the most efficient manner possible, and to limit the distribution of preproduction software."

The Atari Programmers and Developers Forum offers something to every Atari programmer. The help you receive here can mean the difference between forgetting or finishing your software product. And the Developers Forum is guaranteed to be more cost-effective than psychotherapy, so don't be shy about asking for help here!

Atari-Related Databases on Compuserve

In addition to the four Atari Forums, CompuServe also offers two online databases for Atari computer owners: Antic Online, the largest online magazine database available on CompuServe, and the Atari Users Network Database, which is a one-stop area for users to find out what's new in the various Atari Forums and to receive help and information on using the Forums. ATARINET's "What's new in SIG*ATARI" article is updated weekly and highlights new and noteworthy events in the four Atari Forums. The Atari Users Network Database also provides a listing of upcoming scheduled conferences in SIG*ATARI, and Forum help and information files.

More Info, Less $$$

CompuServe's standard daytime and nighttime rates are $6.00 an hour for 300/450 baud, and $12.50 an hour for 1200/2400 baud. Electronic communication can become addicting very quickly, so it is important that you try to use your online time as efficiently as possible. A number of tools have been designed to make interaction with CompuServe as cost effective as possible.

ST/FORUM (available in DL 13 of the Atari Developers Forum) is a program designed to minimize time spent on CompuServe. It does this by logging on, downloading all new messages as quickly as possible, and then logging off. The time it takes to download messages is probably only about half the time it takes to read them on line. The ST/FORUM user can read messages and compose his or her replies off line and then have ST/FORUM upload the replies the next time it logs on. According to Charles McGuinness, author of ST/FORUM, a number of significant enhancements are planned for future versions of the program.

"Presently, ST/FORUM just supports access to the message board. In the future, we hope to expand ST/FORUM to allow it to download files, so that in theory a user will never have to log on CompuServe 'in person.' By doing this we hope to make the users' bills the absolute smallest possible for the amount of usage they get from the service."

The Developers Forum is guaranteed to be more cost-effective than psychotherapy, so don't be shy about asking for help!

Bill Aycock's MCIS utility — for Atari 8-bit owners—is a similar program in that after a user captures new messages using their favorite terminal program, MCIS can be used to conveniently read the messages offline. MCIS is available in DL 5 of the Atari 8-bit Forum (BRO MCIS*.*).

Owners of **Flash 1.52** (and higher), **Interlink,** and **ST Talk Professional** can enjoy reduced download time by using CompuServe's new "Quick B" protocol, which was developed by CompuServe programmers specifically to maximize throughput in the multiuser CompuServe environment. Quick B protocol can be invoked by using the command DOW/PROTO:QB.

CompuServe's Forum software also makes it easy for you to retrieve information that is of interest to you. It allows you to select the message sections you wish to read; it will automatically notify you of any messages you have waiting in each Forum so you can retrieve them quickly; it provides for nine Sysop-written "Bulletin" files which will notify you of "hot items" in all the major areas of the Forum; there is a membership directory for you to use to find others who share your interests; it allows you to set the initial area of the Forum you wish to visit when logging in, and much more. Please consult the Forum Users Guide and online help files for more information on how to get the most out of the Forum software.

SIG*ATARI offers something for everyone. No matter where your Atari interests lie, you will find a whole supportive community that wants to share in your discoveries and help you learn new and exciting things about your computer waiting for you on CompuServe. If you're a new Forum member, the Sysops request that you post an "introduction" message on the message board so others can meet you. The Sysops also invite you to drop them a message any time. Their CompuServe User IDs are as follows:

Ron Luks	76703,254
Mike Schoenbach	76703,4363
Dave Groves	76703,4223
Keith Ledbetter	76701,124
Tom Hudson	76703,4224
Dick Brudzynski	76703,2011
Bill Aycock	76703,4061
Charles McGuinness	76701,11
Dan Rhea	76703,4364

How to Access SIG*ATARI

A menu of Atari services on CompuServe can be accessed by typing GO ATARI at any CompuServe system command prompt. However, these "Quick reference words" can be used to enter any of the following services directly:

The Atari 8-Bit Forum	(GO ATARI8)
The Atari 16-Bit Forum	(GO ATARI16)Atari
Developers Forum	(GO ATARIDEV)Atari
Vendor Support	(GO ATARIVEN) Atari
Users Network	(GO ATA-1)ANTIC Online
Magazine	(GO ANTIC)

Subscription Information:

CompuServe Information Service, Inc.
5000 Arlington Centre Blvd.
Columbus, OH 43220
(800) 848-8990

Reader Comment

Screen Dump Needed

I would like to thank you for the many useful programs and utilities that I have found in your magazine. I own an Atari 8-bit computer, and I have recently purchased an Epson-compatible printer. I need a quick graphics dump subroutine so that I can copy GRAPHICS 8 screens to my printer. I would also like the printed screen to be in a square format so that it looks the same as the screen. Can you help me?

—Scott Alter
Ft. Wayne, IN

Take a look at this issue's ***The Magic of Tesselations, Part 2****. The programs included there contain a screen dump subroutine that may be exactly what you're looking for.*

All That Glitters. . . .

It was with great excitement that I opened Issue No. 59, only to be presented with GLARE. After locating my polarized sunglasses, I read in the editorial that you planned to continue the use of this eye killer. Please! Please! Please do not do this to me and the rest of your readers who type in the listings. The matte-finished paper is easier to read, since it does not reflect the light. I hope you return to it quickly.

—Logan C. Kinnison
Columbus, NE 68601

We think that the largest number of ***ANALOG*** *readers would not agree with you. Many people associate the coated stock now used in* ***ANALOG*** *with an increase in class. "Slick" magazines include the most prestigious publications in the country, and as a result, the coated paper has practically become a status symbol. However, the editorial staff of* ***ANALOG Computing*** *actually agrees with you. The reasons you've stated for not liking the slick paper were exactly the reasons we switched to a different stock several years ago. Most people don't realize that the slick paper is actually a* cheaper *paper.*

Collector in Need

I desire information on the Atari 5200 game system that possibly your readers could supply. Since the 5200 is similar to the Atari 400 computer, I thought ***ANALOG Computing*** Magazine's readership an appropriate resource to pursue.

I wish information on: 1) any prototype cartridges that were not mass-produced and how to acquire them; 2) cartridges that were produced in limited quantities by small companies; 3) how to produce my own cartridges; and 4) inside or little-known facts of the 5200.

I have a large collection of the 5200 system cartridges and want to make it the most complete. My problem is compounded by limited contacts and knowledge in the industry. 5200 collectors are restricted by its limited abilities and being only available for about three to four years. I know this request is unusual, but I'm looking to you for much-needed help.

—Arthur Nestor
230 Arthur Street
Zelienople, PA 16063

Well, there you go, friendly readers. Does anyone out there have information on the 5200 to share?

Where's the Editor?

I like your magazine a lot, except for one thing, your **BASIC Editor II**. Why don't you print it in each issue like you do the **M/L Editor?** It would make it so much easier for people just starting with your magazine. The way it is now, you have to order the back issue. You put which issue to order, but not where to order it from or how much it is.

So, could you please tell me how much it is and where to order back issues from?

—Algeline Theriot
Houma, LA 70360

We print ***M/L Editor*** *in each issue because the* ***M/L Editor*** *data listings can not be typed any other way (well, if you really knew what you were doing, you could come up with a BASIC subroutine to read the data to the disk, but it'd be much easier to use the* ***M/L Editor)****, whereas BASIC programs may be typed in without the use of the* ***BASIC Editor II****, even though we don't recommend that you try it, since the possibilities for error are too great. In any case, you'll be happy to know that the* ***BASIC Editor II*** *will be reprinted in next month's* ***ANALOG****, so there's no need for you to order a back issue. If you'd still like to order a back issue, however, you may do so by writing to* ***ANALOG Computing****, P.O. Box 16927, N. Hollywood, CA 91615. The price for each back issue is $4. Also, you may wish to note that Issues 30 to 40 and Issues 44 to 61 are still available. Limited quantities of earlier issues are also in stock, and that information will be provided as soon as we get a chance to update our inventory list.*

80-Column Telecommunications

I have been a reader of **ANALOG** since Issue 12 and a subscriber since 17. Over the years I have seen a great many problems addressed and solved. I have a serious problem (as in serious business usage of an 8-bit Atari). Therefore, as a last-ditch effort, I am contacting your magazine.

Over the past seven years I have owned various 8-bit Ataris and have collected large amounts of hardware, software and magazines. During the past three months I have searched for 80-column modem software. I have purchased an ICD MIO board only to find that their 80-column adapter is "on hold." I have purchased an XEP80 adapter and spent many a long night leaving and retrieving messages on BBSs, searching for modem software that would work with it. I now have numerous editions of AMODEMS, XMODEMS, 850 Express, etc. I live in a rather small rural area without a users group, so moral support on a quest like this is hard to find.

Is there a public domain, commercial, freeware or shareware telecommunications program available for the 800XL with or without an MIO or XEP adapter that will allow 80-column viewing? All my office computer displays are 80-column, so it's a bit messy trying to use my trusty Atari as a home workstation reading 40 columns with word wrap. Any reasonable suggestion or offer will be acceptable. Maybe someone somewhere has solved this problem.

If any Atari user has any suggestions, please send them to me.

One last word: It was great to receive Issue 59 in the mail today. Thanks!

—Michael A. Reott Sr., D.D.S.
Hwy. 321 South
P.O. Box 615
Maiden, NC 28650

ST-Notes

Clash of the Monitors

by Frank Cohen

Astra Systems has introduced a new monitor switchbox that makes it easy to switch between using color and monochrome monitors with your ST. The Astra SW2 Monitor Switch is a small box that allows you to plug in a monitor's video cable and power cable. Pressing a switch on the front of the box toggles power and video from one monitor to another.

Another company, Practical Solutions, makes a similar box called Monitor Master. Their switchbox operates in a similar way to the Astra box. In the back of the box are sockets for color and monochrome monitor video cables. One cable runs out of the box and into your ST's video input jack. Monitor Master costs $49.99, as compared with the slightly expensive Astra box at $59.95.

Both boxes have a tricky problem when it comes to the ST's video input jack. Many ST users have damaged their STs when accidentally (or purposely) removing the video cable while the ST and monitor are turned on. The ST is sensitive to voltage spikes and has no protection for its video chip when a surge comes down the wires. Practical Solutions includes a small notice to users to be certain the power is turned off on your ST and monitor before operating the switch. Astra also comments that you should turn off your ST before operating their switch.

Only a small amount of the 520ST's have RF modulators. These modulators are needed to take the video output of an ST and send it into a videotape recorder. Both the Astra and Practical Solutions boxes have audio and composite video RCA jacks built in.

Monitor switching boxes are items of convenience, not necessity. However, at under $60 both the Astra SW2 and Monitor Master boxes are a happy accessory to your ST system.

CeBit News

In Hannover, West Germany, the CeBit trade show, probably the largest trade show in the world, saw some new product information from Atari. The Atari booth was crowded with companies offering software and hardware add-ons for the ST. It made quite a show for the public, however, the real news was being shown to only a select group of people.

In a hotel suite, Atari showed a preliminary version of its new 68030 high-end workstation computer system. The new system uses the new Motorola 68030 CPU, which is even more advanced than the previously reported 68020. The 68030 Atari machine is being designed to be an inexpensive Unix system. Unix is the most popular multi-user operating system among the scientific and research development communities. The Atari box will come with Unix System 5.31, which is the AT&T supported version of the Unix operating system.

Atari also showed the ABAQ Transputer and CD ROM units at the show. Both of these units have still not been completed at this writing.

Blow Up Your Blitter

It has always amazed me that a Macintosh running at 5 Megahertz is faster

than an Atari ST running at 8 Megahertz. You might think that because your ST is running quicker, then screen updates, text editing and object drawing would be vastly quicker than the Macintosh. All you have to do is look at each machine's screens to see the difference. Why?!

The Macintosh operating system—the collection of programs that allow a Mac programmer to write text on the screen, plot objects, etc.—was written for the 68000 16-bit chip in the Mac. The GEM operating system was written for 8086/8088 8-bit chip of the IBM PC. The GEM system for the Atari ST is the result of a slowly evolving set of programs which started as CP/M and is now a graphics-based operating system.

GEM is made up of two parts: Virtual Device Interface (VDI) and the Application Environment Services (AES). The VDI handles drawing text, lines, circles, etc. The AES draws drop-down menus, windows, dialog boxes, etc.

VDI is the reason the Mac runs faster. VDI was written mostly in the C Language. C is usually very inefficient when it comes to high-speed graphically oriented programming. Writing the same programs in 68000 assembly languages would be like adding a turbocharger to a 1988 Corvette engine. (Excuse the automotive metaphor, but it did make a nice segue).

Wayne Buckholdt at Softrek rewrote the text-drawing portions of the VDI in assembly language to develop Turbo ST. Turbo ST loads itself into your ST as a desk accessory on boot-up. The program intercepts all the text-plotting commands issued by VDI and processes them itself. The results can be speed improvements up to five times better than the normal VDI text-drawing speed.

Turbo even speeds up text drawing for TOS-based programs that don't use GEM. A popular text-editing program, Microemacs, scrolls incredibly quickly when Turbo ST is active.

The program is approximately 25,000 bytes long, so you won't notice much of a lag when booting your ST. Turbo ST carries a $49.95 suggested list price and is available now.

Apple Blasts Microsoft and Hewlett Packard

In 1986, Digital Research, Inc., the makers of the GEM system for the ST, caved in to a lawsuit filed by Apple Computers in which Apple alleged that DRI's operating system infringed on the look and feel of the Macintosh operating system. Both the GEM system and the Mac system are loosely based on research done at the Xerox Palo Alto Research Center (PARC). Xerox used the graphic operating system in its Star minicomputer systems. Later, under the leadership of Steven Jobs, Apple bought a share of the technology developed at Xerox PARC for Apple's new machine, the Macintosh.

Now, Apple is suing Microsoft and Hewlett Packard because the two companies have developed visual operating systems similar to the Mac's. Apple feels that its system of windows and mouse controls is theirs. DRI settled the suit out of court because DRI was having cash problems. As part of the settlement, DRI changed the GEM desktop to resemble a less friendly user-interface.

Hewlett Packard has developed a package called New Wave that works with Microsoft's Windows 2.03 operating system. Windows is an operating system for the IBM PC and compatibles that uses a mouse to manipulate windows and drop-down windows just like the Mac and GEM. HPS New Wave adds additional functions to windows to bring it even closer to the Xerox PARC system.

The suit was filed in March of this year. The announcement of the suit caused Microsoft stock to fall $5.625 per share. HP stock also fell down $2.125 after the announcement.

The strange thing about the suit is that Microsoft produces the most popular software for the Macintosh. With such a close working relationship, it is even odder that there was no discussion between Microsoft and Apple before the suits were filed.

By the way, in case you were wondering how much money might be involved in Apple's marketing of the Macintosh, Apple posted revenues for the last quarter of 1987 of $1.04 *billion*. That's a 52% increase over sales of 1986. Apple expects with continued support of the Mac II and other products that they will be doing $4 billion in sales every quarter of 1988.

WordPerfect Keeps Pitching

To follow up our reporting last month that WordPerfect Corp. (WPC) announced its intention of pulling out of the Atari ST market: They have changed their mind. WPC began making motions that they were removing their *word* processor, WordPerfect, from the ST market because they found complete copies on three pirate bulletin boards. At the time, WPC's representative said they were not having pirate problems with the Amiga or IBM PC versions as compared to the ST.

After a well-attended conference on CompuServe, WPC announced that it always intended on staying in the ST market, but was appalled at the apparent rampant piracy of software going on in the ST software industry. WPC effectively used its major clout in the ST community by making the announcements.

Companies Mentioned:

Astra Systems, 2500 S. Fairview, Unit L, Santa Ana, CA 92704; (714) 549-2141.

Practical Solutions, 1930 E. Grant Road, Tucson, AZ 85719; (602) 884-9612.

Softrek, 2628 Martz Court, Orlando, FL 32817; (305) 657-4611.

Eidersoft USA (800) 992-9198.

Apple, 20525 Mariani Avenue, Cupertino, CA 95014; (408) 996-1010.

WordPerfect Corp., 1555 N. Technology Way, Orem, UT 84057; (801) 227-4288.

About the author: Frank Cohen has been developing Atari programs since his first commercial product, Clowns & Balloons. When Atari Corp. began marketing the 16-bit ST computer, he founded Regent Software. Frank developed Regent Base, an SQL 4GL database, and is currently involved with several other ST related productivity and small business software packages. You may contact Frank directly on Delphi (REGENTWARE), Genie (FCOHEN) or CompuServe (72457, 3171).

Game

48k disk

Cryptogram Solver

by Kevin Peck

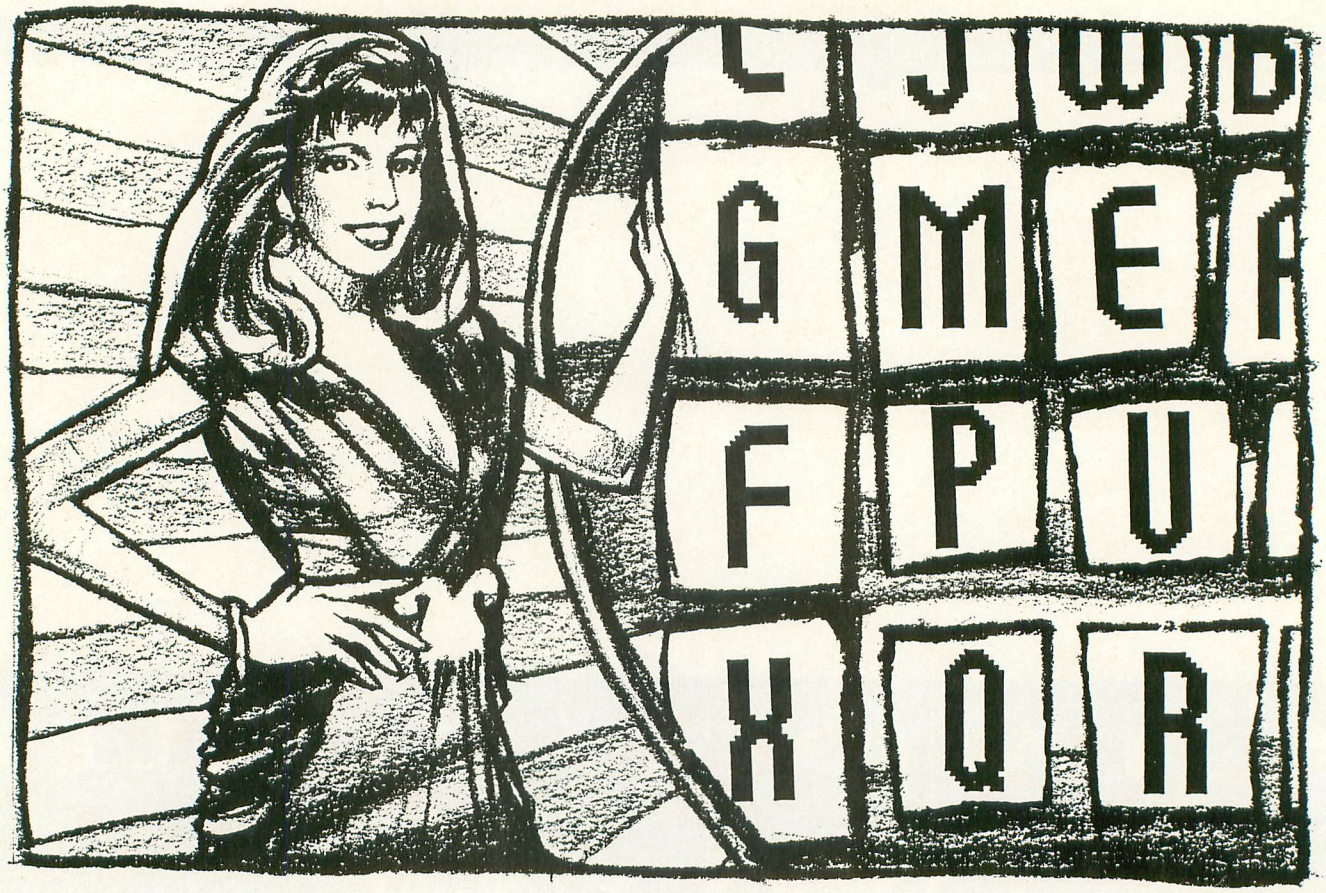
I have always had a lot of fun solving cryptograms out of puzzle books and magazines. However, some of the fun was lost through the multiple erasing and rewriting as guess after guess led to the solution of the puzzle.

But no more! The following program has put all the fun back into solving cryptograms—and has saved me a fortune in erasers.

After typing in the program using the ML Editor, you will be ready to use your computer to solve your first cryptogram. The program includes full on-screen prompts and command key summaries to make the program as easy to use as possible.

The first screen is the title screen. Press any key to begin the puzzle-solving process.

The main puzzle entry and solving screen now appears. The cursor is on the second line of the screen's blue area. This is the first position of puzzle text entry. You are now ready to start typing your puzzle text from the puzzle book. A summary of valid editing keys is shown in the grey area on the bottom of the screen. You are only allowed to enter text on every other line, giving you nine lines for puzzle text. The other nine lines in the blue area are for the solved puzzle text. All editing keys oper-



ate as they do in BASIC. You may insert and delete characters or whole lines. The one addition is a confirming prompt when you press SHIFT-CLEAR to clear the screen. This will save you from accidental loss of puzzle text. I usually like to type in all of the puzzle text, and then insert lines (SHIFT-INSERT) to center the puzzle on the screen.

When you have entered all of the puzzle text, press the START key to begin solving the puzzle. A new set of commands will appear in the grey area at the bottom of the screen. The top two lines of the screen— the green area—will come into play now. The first line is the alphabet in order, and the line below that will contain the replacement set of letters as you try and solve the puzzle. I included this so you can quickly see whether you are trying to substitute the same letter in two different places or to see which letters are still available for use in solving the puzzle.

You are now prompted for the puzzle letters you wish to replace on the red prompt line near the bottom of the screen. The cursor is an underline to let you know that the only text-editing keys available are the backspace, space bar and return keys. To use the special commands in the grey area at the bottom of the screen, press and hold CONTROL while pressing the highlighted letter key of the command that you wish to use. I will explain each command in detail for you in a moment.

To actually solve the cryptogram you must type at the "Replace" prompt the letters of the puzzle—the letters that you entered in the edit phase—that you wish to change. After pressing RETURN, you will be asked what to replace those letters with. Type the letters you want substituted and press RETURN at the "With" prompt. All changes will be reflected in the lines above your puzzle text in inverse video. The second line of the display will also be updated at this time. Notice that you may not type in more characters at the "With" prompt than you typed in at the "Replace" prompt. You may also not press RETURN without any input at the "Replace" prompt nor may you enter any spaces. You are allowed to press RETURN only at the "With" prompt. Doing so will cause the program to blank out all characters from the "Replace" prompt. Do this to clear out mistakes. You may selectively blank out characters by typing a space also. If you type less characters at the "With" prompt than you typed in at the "Replace" prompt, the extra characters will be changed to be spaces.

What about all of those command keys at the bottom of the screen? Well, the first command key listed is the Back key. If you notice a mistake in the letters at the "Replace" prompt while typing the "With" letters, press CNTRL-B to go back to the "Replace" prompt and fix the mistakes.

If you have made multiple misguesses while attempting to solve the puzzle, you may wish to use the Clear command. After a confirming prompt, the program clears out the changes to the puzzle only, not the puzzle text itself.

"Edit" allows you to go back to the edit screen, to make changes to the puzzle text. All changes up to this point are erased, but they will be restored when you return to puzzle solving. Make all changes necessary, and press START when you are done.

"New" allows you to start a new puzzle. There is a confirming prompt for

this command because it is rather drastic. It clears out all puzzle text along with all changes made to that puzzle and deposits you back into the puzzle text-editing screen. Only use this command when you have completely solved the current puzzle or when you are ready to give up on the current puzzle. Warning! There is no way to undo this command!

"Quit" is used only when you are completely done with the program. There is a confirming prompt. If you answer yes to the "Quit?" question, you will be returned to DOS.

Undo is a handy feature that allows you to undo the last change you made to the puzzle. If you use undo a second time, you will undo the undo. You can use this command to toggle between two possible changes to a puzzle to see which is going to work better.

I have included a sample puzzle to get you started. The puzzle appears in Figure 1. The puzzle includes a hint if you need it. The answers appear upside down in Figure 2. A list of common, easy-to-recognize words appear in Figure 3. I have tried to include as many oddball, easy-to-spot words from the word list as possible in the sample puzzles, so you can see how the word list will help you in solving your puzzles.

Writing Your First Machine Language Program

This project started out as a small, slow, BASIC program five years ago on a HeathKit H-89 computer. It had no editing commands, no fancy graphics and no mistake fixers. It did the job and was better than solving the puzzle on paper, but it was lacking as far as useful computer programs go. It was one of the few programs that I had converted from the H-89 to my new Atari. When I converted it two years back, it was identical in features and functions to the H-89 version. I decided to fix that.

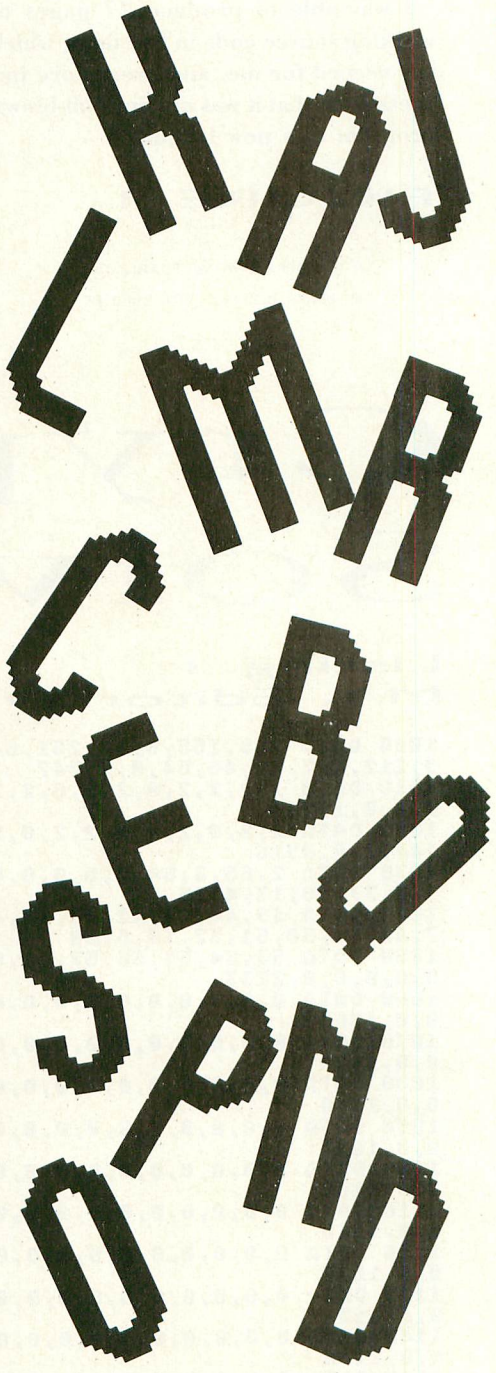
First, I wanted to use some of the fancy color graphics of the Atari. Since this is a text-oriented program I knew I would be using Graphics mode 0. The only way to add color was to use Display List Interrupts. I read all I could on them and wrote one for the program and had it up and running with BASIC. Next, I decided that the letter replacement routine had to be fast, so I wrote a short ML routine for that also. Then I started on the editing features. Unfortunately, BASIC was starting to drag. I looked at the program and said, "Hey! I have two machine-language routines already. Why not do the whole thing in assembly?" I had been trying to get myself to write an all-assembly language program for the longest time, and here it was half done (or so I thought at the time).

I ended up spending the next five days typing source code into Action! from OSS. Wait a minute! Action!, BASIC, assembly. What the heck is going on here? Time for a confession. I love MAC/65, but I hate line numbers, especially since MAC/65 never uses them except to keep source code in order. MAC/65 does have a nice feature that will take unnumbered ASCII text and append line numbers to it. Using this feature, I was able to type the source code in with the Action! editor. Action! allowed for source code manipulation, with the extra feature of scrolling around to find routine names and various labels without having to remember what line number each routine started at.

The next big help was my 130XE with SpartaDOS and its RAMdisk capabilities. I could never live without a RAMdisk again. It is a great place to keep temporary files, and it sure sped up assembly with INCLUDE files. It also helped when I entered the unnumbered text files into MAC/65.

The largest object file that I had ever compiled to date was 192 bytes. This program is just over 3,000 bytes, which just goes to show that you can make the big jump into major assembly language programming without going through byte by byte upgrades. What I am getting at is this: go for it and write that first big all assembly project. You can do it, and you will be proud and have a lot more confidence in yourself after you do.

Okay, I must admit that I did do one special thing in the program. I did bypass the normal way of reading the Atari keyboard. At first I opened the keyboard for input the normal way, but that caused problems with the display list interrupts, especially on the older Atari 800. The first place to look for a solution was De Re Atari, but that was no help. It said, "Another solution is to disable the OS keyboard service routine and provide your own keyboard routine." This would be a tedious job. *Oh, great!* I thought.



It turned out to be fairly simple. All I did was use the value in RAM location 764, the last key pressed value, to look up the ATASCII code in a table that sits in ROM. This table resides in different locations in the 400/800 than in the XL/XEs, but it was easy to have the pro-

gram adjust for that. The program's screens are crystal clear now. I only had to change one ten-line subroutine in the program to accomplish this, and now no annoying key click sounds occur on any of the Atari machines while running the program.

I was able to produce 27 pages of working source code in five days, which is a record for me, and even more impressive in that it was my first full-blown program in a new language.

FIGURE 1.

```
N WTBQDFKX FVNF AE FDXIKJ TMM
AE HDAFK DEKUKEE. CTD IKKJ FT
FDXI CTDX *NFNXA TI FT ETUZK
FVKEK QDOOUKE!
```

Words marked with "*" are proper nouns. The title of the puzzle may be of some help but. . .

HINT

The pattern FVNF can only be one word. Check the word list.

FIGURE 2.

```
A COMPUTER THAT IS TURNED OFF
IS QUITE USELESS. YOU NEED TO
TURN YOUR *ATARI ON TO SOLVE
THESE PUZZLES!
```

FIGURE 3. Word List

```
1. A I

2. AN AS AT BE BY DO GO HE HI IF IN IS
   IT ME MY NO OF OH ON OR SO TO US WE

3. ALL AND ANY ARE ATE BAD BIG BUT CAN
   CAT COW DAY DID FAT FOR HAD HAS HER
   HIS HIM HOW KID LAD LIE LOT MAY NOR
   NOT NOW OFF ONE OUR OUT SAD SAY SHE
   SIN SIX SON THE TOO TWO WAS WHO YOU

4. AWAY BOTH CALL COME DOES EVEN FEEL
   FIND FIVE FORM FOUR FROM FULL GOOD
   HAVE IDEA KNEW KEEP KEPT KNOW LESS
   LONG LOSE MANY MESS MORE MUST NEED
   ONCE REAL SAID THAN THAT THEM THEN
   THEY UPON WHEN WILL WITH YEAR YOUR

5. AFTER ALLOW AWAKE BRING EVERY GOING
   HAPPY KNOWN LIMIT LOCAL OFTEN OFFER
   PIZZA RULER SIDES STYLE THEIR THERE
   THESE THOSE THREE TOOTH VALUE WHERE

6. ACCEPT COFFEE LADDER LITTLE OFFICE
   PEOPLE PEPPER PLEASE REALLY MATTER

7. ELEMENT GENERAL HAPPINESS USELESS
```

Cryptogram Solver

Listing 1: M/L Editor Data

```
1000 DATA 255,255,0,64,251,64,76,158,6
9,112,112,66,46,64,0,2,747
1010 DATA 144,2,2,0,2,2,0,2,2,0,2,2,0,
2,2,0,1324
1020 DATA 2,2,0,2,2,0,2,2,0,2,2,144,2,
144,2,0,4916
1030 DATA 2,65,3,64,0,0,0,0,0,33,34,35
,36,37,38,13,4315
1040 DATA 39,40,41,42,43,44,45,13,46,4
7,48,49,50,51,52,13,6700
1050 DATA 53,54,55,56,57,58,0,0,0,0,0,
0,0,0,0,0,2233
1060 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1060
1070 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1070
1080 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1080
1090 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1090
1100 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1100
1110 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1110
1120 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1120
1130 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1130
1140 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1140
1150 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1150
1160 DATA 0,0,252,64,247,65,0,0,0,0,0,
35,50,57,48,52,7217
1170 DATA 47,39,50,33,45,0,51,47,44,54
,37,50,0,0,0,0,4478
1180 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1180
1190 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1190
1200 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1200
1210 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1210
1220 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1220
1230 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1230
1240 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1240
1250 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1250
1260 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,98
,121,0,4447
1270 DATA 43,101,118,105,110,0,48,101,
99,107,0,0,0,0,0,0,5944
1280 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1280
1290 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1290
1300 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1300
1310 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1310
1320 DATA 0,0,248,65,243,66,0,0,0,0,0,
0,0,0,0,0,3935
1330 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1330
1340 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1340
1350 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1350
1360 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,48
,114,101,5358
1370 DATA 115,115,0,97,110,121,0,107,1
01,121,0,116,111,0,98,101,2275
1380 DATA 103,105,110,0,0,0,0,0,0,0,0,
0,0,0,0,0,2023
1390 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1390
1400 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1400
1410 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1410
1420 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1420
1430 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1430
```

```
1440 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1440
1450 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1450
1460 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1460
1470 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1470
1480 DATA 0,0,244,66,239,67,0,0,0,0,0,0,
0,0,0,0,0,4073
1490 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1490
1500 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1500
1510 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1510
1520 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1520
1530 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1530
1540 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1540
1550 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1550
1560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1560
1570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1570
1580 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1580
1590 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1590
1600 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1600
1610 DATA 0,0,0,0,0,0,0,0,0,0,0,0,53,115
,101,0,128,7203
1620 DATA 163,172,165,161,178,128,0,12
8,169,174,179,165,178,180,128,0,9912
1630 DATA 128,164,165,172,165,180,165,
128,0,93,92,94,95,0,0,0,1658
1640 DATA 0,0,240,67,101,68,0,0,0,0,0,
128,162,171,128,179,4361
1650 DATA 176,128,0,128,178,165,180,18
1,178,174,128,0,116,111,0,101,6610
1660 DATA 110,116,101,114,0,116,101,12
0,116,0,0,0,0,0,128,163,696
1670 DATA 180,178,172,128,0,0,0,128,16
2,128,97,99,107,0,0,0,642
1680 DATA 0,128,163,128,108,101,97,114
,0,0,0,128,165,128,100,105,4327
1690 DATA 116,0,0,0,0,0,128,171,165,18
5,179,128,0,0,0,128,2958
1700 DATA 174,128,101,119,0,0,0,0,0,12
8,177,128,117,105,116,0,2403
1710 DATA 0,0,0,128,181,128,110,100,11
1,0,0,0,196,68,191,69,3933
1720 DATA 50,101,112,108,97,99,101,30,
55,105,116,104,30,48,114,101,3223
1730 DATA 115,115,0,128,179,180,161,17
8,180,128,0,119,104,101,110,0,5857
1740 DATA 100,111,110,101,35,108,101,9
7,114,0,48,117,122,122,108,101,4590
1750 DATA 0,52,101,120,116,31,35,108,1
01,97,114,0,33,108,108,0,1206
1760 DATA 35,104,97,110,103,101,115,31
,33,114,101,0,121,111,117,0,2338
1770 DATA 115,117,114,101,0,8,57,15,46
,9,0,0,0,0,128,165,8496
1780 DATA 228,233,244,233,238,231,128,
176,245,250,250,236,229,128,164,225,13
4
1790 DATA 244,225,128,0,0,0,0,49,117,1
05,116,12,0,37,120,105,781
1800 DATA 116,0,116,111,0,36,47,51,31,
51,116,97,114,116,0,46,732
1810 DATA 101,119,0,48,117,122,122,108
,101,31,0,44,97,115,116,0,1734
1820 DATA 35,104,97,110,103,101,0,53,4
6,36,47,46,37,14,69,58,8822
1830 DATA 155,30,31,255,128,48,2,72,13
8,72,173,11,212,201,40,176,7659
1840 DATA 5,162,255,142,119,69,174,119
,69,232,189,120,69,141,10,212,9554
1850 DATA 141,24,208,142,119,69,104,17
0,104,64,173,254,254,201,108,208,3919
1860 DATA 6,169,254,133,121,133,122,16
2,32,169,3,157,66,3,169,114,6205
1870 DATA 157,68,3,169,69,157,69,3,169
,12,157,74,192,69,203,69,6509
1880 DATA 3,169,0,157,75,3,32,86,228,7
6,154,70,204,69,199,70,7223
1890 DATA 162,39,189,198,67,157,118,67
,189,238,67,157,158,67,202,16,9108
1900 DATA 241,32,234,73,169,1,133,85,1
69,3,133,84,141,240,2,169,7610
1910 DATA 32,32,189,72,162,28,189,30,6
9,157,91,64,202,16,247,169,8621
1920 DATA 0,141,240,2,32,34,72,32,3,72
,162,22,189,209,68,157,5762
1930 DATA 86,67,202,16,247,173,31,208,
201,6,208,3,76,227,70,169,9087
1940 DATA 8,141,31,208,173,252,2,201,2
55,240,234,32,167,73,201,155,3495
1950 DATA 208,3,76,84,72,201,156,208,3
,76,209,72,201,157,208,3,8979
1960 DATA 76,24,73,201,254,240,45,201,
255,208,3,76,113,73,201,125,566
1970 DATA 208,3,76,14,74,201,126,240,2
7,41,127,201,28,144,182,201,9634
1980 DATA 32,144,2,176,3,76,107,72,201
,96,240,169,144,6,201,123,9182
1990 DATA 176,163,233,31,32,189,72,165
,85,201,38,208,162,165,84,201,1014
2000 DATA 19,208,10,169,37,133,85,32,3
,72,76,17,70,169,1,133,2181
2010 DATA 85,24,165,84,105,2,133,84,32
,3,72,76,17,70,169,0,872
2020 DATA 141,47,2,169,192,141,14,212,
169,123,141,0,2,169,69,141,6522
2030 DATA 1,2,169,3,141,48,2,169,64,14
1,49,2,169,0,141,200,4974
2040 DATA 2,169,192,141,198,2,169,34,1
41,47,2,169,200,70,195,71,7407
2050 DATA 46,133,88,169,64,133,89,169,
255,141,252,2,173,252,2,201,1919
2060 DATA 255,240,249,169,255,141,252,
2,32,111,74,169,1,141,240,2,7978
2070 DATA 169,19,133,84,169,38,133,85,
169,32,32,189,72,162,39,189,6970
2080 DATA 22,68,157,118,67,189,62,68,1
57,158,67,169,0,157,86,64,5898
2090 DATA 202,16,236,32,86,75,32,34,72
,162,7,189,196,68,157,80,6284
2100 DATA 67,202,16,247,162,0,169,78,1
57,88,67,142,195,68,32,180,7805
2110 DATA 71,174,195,68,201,2,240,243,
201,32,240,239,201,155,208,6,3663
2120 DATA 224,0,240,231,208,28,201,126
,208,12,224,0,240,221,169,0,816
2130 DATA 157,88,67,202,16,208,224,11,
240,209,233,31,157,88,67,232,1631
2140 DATA 208,196,142,195,68,169,0,157
,88,67,162,4,189,204,68,157,8693
2150 DATA 101,67,202,16,247,162,0,169,
78,157,106,67,142,168,68,32,6586
2160 DATA 180,71,174,168,68,201,32,240
,38,201,2,208,3,76,79,76,5740
2170 DATA 201,155,208,11,169,0,157,106
,67,32,195,75,76,14,71,201,5574
2180 DATA 126,208,12,224,0,240,213,169
,0,157,106,67,202,16,200,236,1103
2190 DATA 195,68,240,200,233,31,157,10
6,67,232,208,187,32,167,73,201,1859
2200 DATA 126,208,1,96,201,155,208,1,9
6,201,2,208,196,71,191,72,9479
2210 DATA 1,96,201,32,208,1,96,201,3,2
08,3,76,193,73,201,5,6138
2220 DATA 208,5,104,104,76,204,69,201,
14,208,3,76,81,74,201,17,5388
2230 DATA 208,3,76,173,74,201,21,208,3
,76,29,75,201,65,144,192,7512
2240 DATA 201,91,144,10,201,123,176,18
4,201,97,144,180,233,32,96,162,1574
2250 DATA 32,169,11,157,66,3,169,117,1
57,68,3,169,69,157,69,3,4080
2260 DATA 169,2,157,72,3,169,0,157,73,
3,32,86,228,96,169,0,4391
2270 DATA 162,39,157,78,67,202,16,250,
96,162,18,189,12,69,157,98,6947
2280 DATA 67,202,16,247,32,167,73,160,
0,41,127,201,78,240,14,201,8759
2290 DATA 110,240,10,201,89,240,8,201,
121,208,233,240,2,160,1,96,9692
2300 DATA 165,84,201,19,208,3,76,204,6
```

```
9,230,84,230,84,169,2,133,8755
2310 DATA 85,32,3,72,76,17,70,201,28,2
40,24,201,29,240,36,201,8157
2320 DATA 30,240,48,164,85,192,37,208,
61,160,2,132,85,32,3,72,3635
2330 DATA 76,17,70,164,84,192,3,208,42
,160,19,132,84,32,3,72,3071
2340 DATA 76,17,70,164,84,192,19,208,2
6,160,3,132,84,32,3,72,2873
2350 DATA 76,17,70,164,85,192,2,208,13
,160,37,132,85,32,3,72,3039
2360 DATA 76,17,70,32,189,72,76,112,70
,72,162,32,192,72,187,73,6606
2370 DATA 169,11,157,66,3,169,0,157,72
,3,157,73,3,104,76,86,2873
2380 DATA 228,32,127,73,166,84,165,203
,24,105,80,133,205,165,204,105,915
2390 DATA 0,133,206,232,232,224,21,240
,19,160,39,177,205,145,203,136,3013
2400 DATA 208,249,165,205,133,203,165,
206,133,204,208,218,160,39,169,0,2409
2410 DATA 145,203,136,208,251,169,2,13
3,85,32,127,73,160,2,177,203,8917
2420 DATA 133,93,32,3,72,76,17,70,166,
84,224,19,240,56,169,1,5823
2430 DATA 141,240,2,32,3,72,169,38,133
,203,169,67,133,204,162,21,8360
2440 DATA 202,202,228,84,240,32,165,20
3,56,233,80,133,205,165,204,233,5310
2450 DATA 0,133,206,160,39,177,205,145
,203,136,208,249,165,205,133,203,6547
2460 DATA 165,206,133,204,208,218,32,1
27,73,160,39,169,0,133,93,141,8067
2470 DATA 240,2,145,203,136,208,251,16
9,2,133,85,32,3,72,76,17,4124
2480 DATA 70,72,32,127,73,169,0,160,36
,145,203,104,76,112,70,166,7474
2490 DATA 84,165,88,133,203,165,89,133
,204,165,203,24,105,40,133,203,567
2500 DATA 165,204,105,0,133,204,202,20
8,240,96,169,8,141,31,208,173,1585
2510 DATA 31,208,201,6,208,244,96,169,
64,141,190,2,169,0,141,182,9436
2520 DATA 2,173,252,2,201,255,240,249,
168,169,255,141,188,73,183,74,4933
2530 DATA 252,2,177,121,96,32,171,75,3
2,34,72,162,17,189,250,68,7339
2540 DATA 157,79,67,202,16,247,32,45,7
2,192,1,208,6,32,183,75,5556
2550 DATA 76,180,71,32,234,73,32,159,7
4,104,104,76,227,70,169,126,8675
2560 DATA 133,203,169,64,133,204,162,9
,169,0,160,39,145,203,136,16,7729
2570 DATA 251,165,203,24,105,80,133,20
3,165,204,105,0,133,204,202,208,3039
2580 DATA 231,96,32,34,72,162,17,189,2
32,68,157,79,67,202,16,247,9532
2590 DATA 32,45,72,192,1,208,3,76,251,
69,165,88,24,105,120,133,7108
2600 DATA 203,165,89,105,0,133,204,162
,9,169,0,160,39,145,203,136,8791
2610 DATA 16,251,165,203,24,105,80,133
,203,165,204,105,0,133,204,202,1944
2620 DATA 208,231,76,204,69,32,171,75,
32,34,72,162,17,189,77,69,5158
2630 DATA 157,79,67,202,16,247,32,45,7
2,192,1,208,6,32,183,75,5046
2640 DATA 76,180,71,104,104,165,88,24,
105,40,133,203,165,89,105,0,6233
2650 DATA 133,204,162,19,169,0,160,39,
145,203,136,16,251,165,203,24,55
2660 DATA 105,40,133,203,165,204,105,0
,133,204,202,208,231,32,159,74,1815
2670 DATA 76,204,69,162,26,169,0,157,1
02,68,157,169,68,202,16,247,9666
2680 DATA 96,32,171,75,32,34,72,162,17
,189,59,69,184,74,179,75,6650
2690 DATA 157,79,67,202,16,247,32,45,7
2,192,1,208,6,32,183,75,5706
2700 DATA 76,180,71,104,104,162,16,169
,12,157,66,3,32,86,228,162,6793
2710 DATA 32,169,12,157,66,3,32,86,228
,169,64,141,14,212,173,229,551
2720 DATA 2,24,105,1,141,48,2,133,203,
173,230,2,141,49,2,133,5948
2730 DATA 204,160,4,177,203,133,88,200
,177,203,133,89,169,0,141,240,2309
2740 DATA 2,169,148,141,198,2,169,202,
141,197,2,169,2,133,85,169,9045
2750 DATA 12,141,252,2,96,32,171,75,32
,34,72,162,18,189,95,69,5050
2760 DATA 157,88,67,202,16,247,162,25,
189,169,68,72,189,102,68,157,9418
2770 DATA 169,68,104,157,102,68,202,16
,239,32,86,75,169,0,133,19,5288
2780 DATA 133,20,165,20,201,100,208,25
0,32,183,75,76,180,71,32,240,98
2790 DATA 75,169,126,133,203,169,64,13
3,204,169,9,141,168,68,160,2,8539
2800 DATA 24,165,203,105,40,133,205,16
5,204,105,0,133,206,177,205,201,3865
2810 DATA 0,240,24,201,33,176,7,9,128,
145,203,76,147,75,56,233,8784
2820 DATA 33,170,189,102,68,201,0,208,
238,240,238,200,192,38,208,221,6622
2830 DATA 24,165,203,105,80,133,203,16
5,204,105,0,133,204,206,168,68,1778
2840 DATA 208,188,96,162,39,189,78,67,
157,128,68,202,180,75,94,76,8652
2850 DATA 16,247,96,162,39,189,128,68,
157,78,67,202,16,247,96,162,117
2860 DATA 25,189,102,68,157,169,68,202
,16,247,174,195,68,202,138,168,3070
2870 DATA 189,106,67,141,168,68,56,189
,88,67,233,33,170,173,168,68,9849
2880 DATA 157,102,68,152,170,202,16,23
0,32,86,75,96,162,25,160,33,6576
2890 DATA 189,102,68,201,0,240,2,9,128
,153,86,64,202,136,192,27,8055
2900 DATA 208,238,169,13,153,86,64,136
,189,102,68,201,0,240,2,9,6375
2910 DATA 128,153,86,64,202,136,192,19
,208,238,169,13,153,86,64,136,9776
2920 DATA 189,102,68,201,0,240,2,9,128
,153,86,64,202,136,192,11,7829
2930 DATA 208,238,169,13,153,86,64,136
,189,102,68,201,0,240,2,9,6405
2940 DATA 128,153,86,64,202,136,192,4,
208,238,96,169,0,162,16,157,9446
2950 DATA 101,67,202,16,250,174,195,68
,76,30,71,224,2,225,2,0,5717
2960 DATA 64,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,3024
```

Listing 2: Assembly

```
10        .OPT NO LIST
20        .OPT OBJ
30        .TITLE       CRYPTOGRAM SOLVER
40 ;
50        *=  $8000
60 ;
70 START JMP OPNKYSCR
80 ;
90        .INCLUDE #D:CRYPTO2.M65
0100 ;
0110 ;
0120 ENTERTEXT
0130 ;
0140      LDX #39
0150 DSPELN
0160      LDA ELN1,X
0170      STA PMPTLINE+40,X
0180      LDA ELN2,X
0190      STA PMPTLINE+80,X
0200      DEX
0210      BPL DSPELN
0220      JSR CHNGCLRS
0230      LDA #1        ;set row and
0240      STA COLCRS    ;col for entry
0250      LDA #3        ;upper-left
0260      STA ROWCRS    ;pos of entry
0270      STA CRSINH
0280      LDA #32
0290      JSR PRNTCHR
0300      LDX #28
0310 W1   LDA EDTPMPT,X
0320      STA REPLINE+5,X
0330      DEX
0340      BPL W1
0350 ;
```

```
0360 SAYSTART
0370      LDA #0
0380      STA CRSINH
0390      JSR CLRPRMPT
0400      JSR MVECRS
0410      LDX #22
0420 W2   LDA ENTPMPT,X
0430      STA PMPTLINE+8,X
0440      DEX
0450      BPL W2
0460 GETKEY
0470      LDA CONSOL   ;read console
0480      CMP #6       ;if START
0490      BNE GETKEY2
0500      JMP ENTERDONE ;then DONE
0510 GETKEY2
0520      LDA #8       ;RESET console
0530      STA CONSOL
0540      LDA KEYPRS
0550      CMP #$FF
0560      BEQ GETKEY
0570 GETIT
0580      JSR GETLET
0590      CMP #$9B     ;RETURN?
0600      BNE G1
0610      JMP RTN
0620 G1   CMP #$9C     ;DEL LINE?
0630      BNE G2
0640      JMP DELINE
0650 G2   CMP #$9D
0660      BNE G3
0670      JMP INLINE
0680 G3   CMP #$FE     ;DEL CHR?
0690      BEQ PRNTIT
0700      CMP #$FF
0710      BNE G4
0720      JMP INSCHR
0730 G4   CMP #$7D     ;CLS issued
0740      BNE G5
0750      JMP CLREDT
0760 G5   CMP #$7E
0770      BEQ PRNTIT
0780      AND #$7F     ;turn off HI bit
0790      CMP #$1C
0800      BCC GETKEY
0810      CMP #$20
0820      BCC CC
0830      BCS G6
0840 CC   JMP CRSCTRL
0850 G6   CMP #$60
0860      BEQ GETKEY
0870      BCC PRNTIT
0880      CMP #$7B
0890      BCS GETKEY
0900      SBC #$1F
0910 PRNTIT JSR PRNTCHR
0920      LDA COLCRS
0930      CMP #38
0940      BNE GETKEY2
0950      LDA ROWCRS
0960      CMP #19
0970      BNE NXTLINE
0980      LDA #37
0990      STA COLCRS
1000      JSR MVECRS
1010      JMP GETKEY
1020 NXTLINE
1030      LDA #1
1040      STA COLCRS
1050      CLC
1060      LDA ROWCRS
1070      ADC #2
1080      STA ROWCRS
1090      JSR MVECRS
1100      JMP GETKEY
1110 ;
1120 ;-----------------------
1130 ; Set the Shown screen
1140 ; and the write to
1150 ; screen to the main
1160 ; puzzle entry screen
1170 ;-----------------------
1180 ;
1190 SETMAIN
1200      LDA #0       ;Screen off
1210      STA DMA
1220      LDA #192
1230      STA DLIENA
1240      LDA # <DLI1 ;Activate
1250      STA DLIVEC  ;DLI 1
1260      LDA # >DLI1
1270      STA DLIVEC+1
1280      LDA # <DLST1 ;and DLIST 1
1290      STA DLI
1300      LDA # >DLST1
1310      STA DLI+1
1320      LDA #0       ;reset colors
1330      STA BRD
1340      LDA #192
1350      STA BCK
1360      LDA #34      ;and screen
1370      STA DMA
1380      LDA # <MANSCR
1390      STA SCREEN
1400      LDA # >MANSCR
1410      STA SCREEN+1
1420      LDA #255
1430      STA KEYPRS
1440 WTKEY
1450      LDA KEYPRS
1460      CMP #255
1465      BEQ WTKEY
1470      LDA #255
1480      STA KEYPRS
1490      JSR DONEW
1500 ;
1510 ;-----------------------
1520 ; The START key has been
1530 ; pressed and we are ready
1540 ; to start solving the
1550 ; puzzle. We have to put
1560 ; the CNTRL prompts on the
1570 ; bottom of the screen and
1580 ; ask for the puzzle letters
1590 ; on the prompt line.
1600 ;-----------------------
1610 ;
1620 ENTERDONE
1630      LDA #1
1640      STA CRSINH
1650      LDA #19
1660      STA ROWCRS
1670      LDA #38
1680      STA COLCRS
1690      LDA #32
1700      JSR PRNTCHR
1710      LDX #39
1720 E2   LDA CMD1,X
1730      STA PMPTLINE+40,X
1740      LDA CMD2,X
1750      STA PMPTLINE+80,X
1760      LDA #0
1770      STA REPLINE,X
1780      DEX
1790      BPL E2
1800      JSR DOREP
1810 ;
1820 ;-----------------------
1830 ; The CNTRL commands are now
1840 ; on the screen. Time to
1850 ; clear the prompt line and
1860 ; put up the REPLACE> prompt
1870 ; for the user to start
1880 ; entering text.
1890 ;-----------------------
1900 ;
1910 PRNTREP
1920      JSR CLRPRMPT
1930      LDX #7
1940 E1   LDA REPPMPT,X
1950      STA PMPTLINE+2,X
1960      DEX
1970      BPL E1
1980      LDX #0
1990 PRNTCRS
2000      LDA #$4E
2010      STA REPLCHRS,X
2020 GETMORE
2030      STX REPLEN
2040      JSR GETEM
2050      LDX REPLEN
2060      CMP #$02
2070      BEQ GETMORE
2080      CMP #$20
```

```
2090      BEQ GETMORE
2100      CMP #$9B
2110      BNE GE24
2120      CPX #0
2130      BEQ GETMORE
2140      BNE GETWITH
2150 GE24
2160      CMP #$7E
2170      BNE SHOWREPCHR
2180      CPX #0
2190      BEQ GETMORE
2200 BCKREP
2210      LDA #0
2220      STA REPLCHRS,X
2230      DEX
2240      BPL PRNTCRS
2250 SHOWREPCHR
2260      CPX #11
2270      BEQ GETMORE
2280      SBC #$1F
2290      STA REPLCHRS,X
2300      INX
2310      BNE PRNTCRS
2320 ;
2330 ;-------------------------
2340 ; This will print the WITH>
2350 ; prompt on the screen and
2360 ; get the replacement chars
2370 ; up to the length of the
2380 ; puzzle chrs and then wait
2390 ; for the RETURN or BK SP
2400 ; key to be pressed. The
2410 ; CNTRL-B is now valid and
2420 ; will have to be dealt with
2430 ; too.
2440 ;-------------------------
2450 ;
2460 GETWITH
2470      STX REPLEN
2480      LDA #0
2490      STA REPLCHRS,X
2500      LDX #4
2510 PRNWITH LDA WTHPMPT,X
2520      STA WITHCHRS-5,X
2530      DEX
2540      BPL PRNWITH
2550      LDX #0
2560 PRNWCRS
2570      LDA #$4E
2580      STA WITHCHRS,X
2590 GETWLET
2600      STX ROWTMP
2610      JSR GETEM
2620      LDX ROWTMP
2630      CMP #$20
2640      BEQ WPRNT
2650      CMP #$02
2660      BNE GW5
2670      JMP BCKTOREP
2680 GW5
2690      CMP #$9B
2700      BNE GET2
2710 FINENT
2720      LDA #0
2730      STA WITHCHRS,X
2740      JSR STOREREP
2750      JMP PRNTREP
2760 GET2 CMP #$7E
2770      BNE WPRNT
2780      CPX #0
2790      BEQ GETWLET
2800 GETBACK
2810      LDA #0
2820      STA WITHCHRS,X
2830      DEX
2840      BPL PRNWCRS
2850 WPRNT
2860      CPX REPLEN
2870      BEQ GETWLET
2880      SBC #$1F
2890      STA WITHCHRS,X
2900      INX
2910      BNE PRNWCRS
2920 ;
2930 ;-------------------------
2940 ; This is where we enter the
2950 ; puzzle letters to be changed
2960 ;-------------------------
2970 ;
2980 GETEM
2990      JSR GETLET
3000      CMP #$7E
3010      BNE GE10
3020      RTS
3030 GE10 CMP #$9B
3040      BNE GE11
3050      RTS
3060 GE11 CMP #$02
3070      BNE GE12
3080      RTS
3090 GE12 CMP #$20
3100      BNE GE1
3110      RTS
3120 GE1 CMP #$03
3130      BNE GE2
3140      JMP CLRCHG
3150 GE2 CMP #$05
3160      BNE GE4
3170      PLA
3180      PLA
3190      JMP ENTERTEXT
3200 GE4 CMP #$0E
3210      BNE GE5
3220      JMP NEWPUZ
3230 GE5 CMP #$11
3240      BNE GE6
3250      JMP QUIT
3260 GE6 CMP #$15
3270      BNE GE8
3280      JMP UNDOIT
3290 GE8 CMP #$41
3300      BCC GETEM
3310      CMP #$5B
3320      BCC ISOK
3330      CMP #$7B
3340      BCS GETEM
3350      CMP #$61
3360      BCC GETEM
3370      SBC #$20
3380 ISOK
3390      RTS
3400 ;
3410 ;-------------------------
3420 ; Move the cursor so it will
3430 ; show up on the screen
3440 ; after inhibiting it.
3450 ;-------------------------
3460 ;
3470 MVECRS
3480      LDX #$20
3490      LDA #PUTCHR
3500      STA ICCMD,X
3510      LDA # <LFRT
3520      STA ICBAL,X
3530      LDA # >LFRT
3540      STA ICBAH,X
3550      LDA #2
3560      STA ICBLL,X
3570      LDA #0
3580      STA ICBLH,X
3590      JSR CIOV
3600      RTS
3610 ;
3620 ;-------------------
3630 ;    Clear the prompt
3640 ;    line on Main
3650 ;    entry screen.
3660 ;-------------------
3670 ;
3680 CLRPRMPT
```

```
3690      LDA #0
3700      LDX #39
3710 CLR1
3720      STA PMPTLINE,X
3730      DEX
3740      BPL CLR1
3750      RTS
3760 ;
3770 ;--------------------
3780 ; Confirm the Clear
3790 ; Screen command.
3800 ;--------------------
3810 ;
3820 CHKSURE
3830      LDX #18
3840 S1   LDA AYSPMPT,X
3850      STA PMPTLINE+20,X
3860      DEX
3870      BPL S1
3880 GETYN
3890      JSR GETLET
3900      LDY #0
3910      AND #127
3920      CMP #'N
3930      BEQ NOSURE
3940      CMP #'n
3950      BEQ NOSURE
3960      CMP #'Y
3970      BEQ ISSURE
3980      CMP #'y
3990      BNE GETYN
4000      BEQ ISSURE
4010 NOSURE
4020      LDY #1
4030 ISSURE
4040      RTS
4050 ;
4060 ;-------------------------
4070 ; We make it here if the
4080 ; RETURN key was pressed
4090 ; We will have to jump
4100 ; down 2 lines unless we
4110 ; are on the last line in
4120 ; which case we will have
4130 ; to return to the top line
4140 ;-------------------------
4150 ;
4160 RTN
4170      LDA ROWCRS
4180      CMP #19
4190      BNE R1
4200      JMP ENTERTEXT
4210 R1   INC ROWCRS
4220      INC ROWCRS
4230      LDA #2
4240      STA COLCRS
4250      JSR MVECRS
4260      JMP GETKEY
4270 ;
4280 ;-------------------------
4290 ; This is where we end up
4300 ; if one of the cursor
4310 ; control keys is pressed.
4320 ; We will move in the desired
4330 ; direction with full wrap-
4340 ; around where required.
4350 ;-------------------------
4360 ;
4370 CRSCTRL
4380      CMP #$1C
4390      BEQ CRSUP
4400      CMP #$1D
4410      BEQ CRSDN
4420      CMP #$1E
4430      BEQ CRSLT
4440 ;
4450 ; A cursor right will fall in
4460 ; to this part of routine.
4470 ;
4480      LDY COLCRS
4490      CPY #37
4500      BNE CRSDONE
4510      LDY #2
4520      STY COLCRS
4530      JSR MVECRS
4540      JMP GETKEY
4550 ;
4560 CRSUP
4570      LDY ROWCRS
4580      CPY #3
4590      BNE CRSDONED
4600      LDY #19
4610      STY ROWCRS
4620      JSR MVECRS
4630      JMP GETKEY
4640 ;
4650 CRSDN
4660      LDY ROWCRS
4670      CPY #19
4680      BNE CRSDONED
4690      LDY #3
4700      STY ROWCRS
4710      JSR MVECRS
4720      JMP GETKEY
4730 ;
4740 CRSLT
4750      LDY COLCRS
4760      CPY #2
4770      BNE CRSDONE
4780      LDY #37
4790      STY COLCRS
4800      JSR MVECRS
4810      JMP GETKEY
4820 ;
4830 CRSDONED
4840      JSR PRNTCHR ;Do double
4850 CRSDONE
4860      JMP PRNTIT  ;print.
4870 ;
4880 ;-------------------------
4890 ; This routine will print a
4900 ; single character stored in
4910 ; the Accumulator
4920 ;-------------------------
4930 ;
4940 PRNTCHR
4950      PHA
4960      LDX #$20
4970      LDA #PUTCHR
4980      STA ICCMD,X
4990      LDA #0
5000      STA ICBLL,X
5010      STA ICBLH,X
5020      PLA
5030      JMP CIOV    ;auto return
5040 ;
5050 ;-------------------------
5060 ; Delete a line of puzzle
5070 ; text from the screen.
5080 ; Move all lines below this
5090 ; line up one and erase the
5100 ; last puzzle text line
5110 ; completely.
5120 ;-------------------------
5130 ;
5140 DELINE
5150      JSR FINDLINE
5160      LDX ROWCRS
5170 D1   LDA ZSCR
5180      CLC
5190      ADC #80
5200      STA ZSCR1
5210      LDA ZSCR+1
5220      ADC #0
5230      STA ZSCR1+1
5240      INX
5250      INX
5260      CPX #21
5270      BEQ DBLNK
5280      LDY #39
```

```
5290 D2  LDA (Z5CR1),Y
5300     STA (Z5CR),Y
5310     DEY
5320     BNE D2
5330     LDA Z5CR1
5340     STA Z5CR
5350     LDA Z5CR1+1
5360     STA Z5CR+1
5370     BNE D1
5380 DBLNK LDY #39
5390     LDA #0
5400 D3  STA (Z5CR),Y
5410     DEY
5420     BNE D3
5430     LDA #2
5440     STA COLCR5
5450     JSR FINDLINE
5460     LDY #2
5470     LDA (Z5CR),Y
5480     STA OLDCHR
5490     JSR MVECR5
5500     JMP GETKEY
5510 ;
5520 ;-------------------------
5530 ; Insert a blank puzzle
5540 ; line on the screen by
5550 ; moving this line and all
5560 ; others below it down one
5570 ; Fill this line with blanks
5580 ;-------------------------
5590 ;
5600 INLINE
5610     LDX ROWCR5
5620     CPX #19
5630     BEQ IBLNK
5640     LDA #1
5650     STA CRSINH
5660     JSR MVECR5
5670     LDA # <LINE19
5680     STA Z5CR
5690     LDA # >LINE19
5700     STA Z5CR+1
5710     LDX #21
5720 I1  DEX
5730     DEX
5740     CPX ROWCR5
5750     BEQ IBLNK
5760     LDA Z5CR
5770     SEC
5780     SBC #80
5790     STA Z5CR1
5800     LDA Z5CR+1
5810     SBC #0
5820     STA Z5CR1+1
5830     LDY #39
5840 I2  LDA (Z5CR1),Y
5850     STA (Z5CR),Y
5860     DEY
5870     BNE I2
5880     LDA Z5CR1
5890     STA Z5CR
5900     LDA Z5CR1+1
5910     STA Z5CR+1
5920     BNE I1
5930 IBLNK JSR FINDLINE
5940     LDY #39
5950     LDA #0
5960     STA OLDCHR
5970     STA CRSINH
5980 I3  STA (Z5CR),Y
5990     DEY
6000     BNE I3
6010     LDA #2
6020     STA COLCR5
6030     JSR MVECR5
6040     JMP GETKEY
6050 ;
6060 ;-------------------------
6070 ; Insert a character at the
6080 ; cursor position. Delete
6090 ; the last character on the
6100 ; line also.
6110 ;-------------------------
6120 ;
6130 INSCHR
6140     PHA
6150     JSR FINDLINE
6160     LDA #0
6170     LDY #36
6180     STA (Z5CR),Y
6190     PLA
6200     JMP PRNTIT
6210 ;
6220 ;----------------------| --
6230 ; Find the current line of
6240 ; text on the screen. It
6250 ; will be found in Z5CR
6260 ; when we are done here.
6270 ;-------------------------
6280 ;
6290 FINDLINE
6300     LDX ROWCR5
6310     LDA SCREEN
6320     STA Z5CR
6330     LDA SCREEN+1
6340     STA Z5CR+1
6350 F1  LDA Z5CR
6360     CLC
6370     ADC #40
6380     STA Z5CR
6390     LDA Z5CR+1
6400     ADC #0
6410     STA Z5CR+1
6420     DEX
6430     BNE F1
6440     RTS
6450 ;
6460 ;-------------------------
6470 ; This routine will wait and
6480 ; do nothing until START is
6490 ; pressed.
6500 ;-------------------------
6510 ;
6520 PRSSTRT
6530     LDA #8       ;reset CONSOLE
6540     STA CONSOL   ;keys with 8
6550     LDA CONSOL   ;now read key
6560     CMP #6       ;6=START
6570     BNE PRSSTRT ;not yet!
6580     RTS          ;got it. RETURN
6590 ;
6600 ;-------------------------
6610 ; This routine locks on CAPS
6620 ; and turns off INVERSE and
6630 ; then gets one key from the
6640 ; keyboard. The calling
6650 ; routine will have to do
6660 ; the range checks.
6670 ;-------------------------
6680 ;
6690 GETLET
6700     LDA #64      ;UPPERCASE
6710     STA SHFLOK   ;lock
6720     LDA #0       ;INVERSE
6730     STA INVFLG   ;off
6740 NOPRS
6750     LDA KEYPRS
6760     CMP #$FF
6770     BEQ NOPRS
6780     TAY
6790     LDA #$FF
6800     STA KEYPRS
6810     LDA ($79),Y
6820     RTS
6830 ;
6840 ;-------------------------
6850 ; We come here when the user
6860 ; wants to clear out all
6870 ; changes to the puzzle.
6880 ;-------------------------
6890 ;
6900 CLRCHG
6910     JSR SVEPMPT
6920     JSR CLRPRMPT
6930     LDX #17
6940 CL  LDA CCGPMPT,X
6950     STA PMPTLINE+1,X
6960     DEX
6970     BPL CL
6980     JSR CHKSURE
6990     CPY #1
7000     BNE CP
7010     JSR RSTPMPT
7020     JMP GETEM
```

```
7030 CP  JSR CHNGCLRS
7040     JSR CLRBUF
7050     PLA
7060     PLA
7070     JMP ENTERDONE
7080 ;
7090 ; Clear the changes line
7100 ;
7110 CHNGCLRS
7120     LDA # <FRSTLINE
7130     STA ZSCR
7140     LDA # >FRSTLINE
7150     STA ZSCR+1
7160     LDX #9
7170 CP1 LDA #0
7180     LDY #39
7190 CP2 STA (ZSCR),Y
7200     DEY
7210     BPL CP2
7220     LDA ZSCR
7230     CLC
7240     ADC #80
7250     STA ZSCR
7260     LDA ZSCR+1
7270     ADC #0
7280     STA ZSCR+1
7290     DEX
7300     BNE CP1
7310     RTS
7320 ;
7330 ;---------------------------
7340 ; We make it here when the
7350 ; user wants to clear out the
7360 ; edit screen.
7370 ;---------------------------
7380 ;
7390 CLREDT
7400     JSR CLRPRMPT
7410     LDX #17
7420 CE10 LDA CPZPMPT,X
7430     STA PMPTLINE+1,X
7440     DEX
7450     BPL CE10
7460     JSR CHKSURE
7470     CPY #1
7480     BNE CE
7490     JMP SAYSTART
7500 CE  LDA SCREEN
7510     CLC
7520     ADC #120
7530     STA ZSCR
7540     LDA SCREEN+1
7550     ADC #0
7560     STA ZSCR+1
7570     LDX #9
7580 CLRLINE
7590     LDA #0
7600     LDY #39
7610 NL
7620     STA (ZSCR),Y
7630     DEY
7640     BPL NL
7650     LDA ZSCR
7660     CLC
7670     ADC #80
7680     STA ZSCR
7690     LDA ZSCR+1
7700     ADC #0
7710     STA ZSCR+1
7720     DEX
7730     BNE CLRLINE
7740     JMP ENTERTEXT
7750 ;
7760 ;---------------------------
7770 ; The user wants to start a
7780 ; new puzzle. We have to
7790 ; prompt the user to be sure
7800 ; of this.
7810 ;---------------------------
7820 ;
7830 NEWPUZ
7840     JSR SVEPMPT
7850     JSR CLRPRMPT
7860     LDX #17
7870 NP1 LDA NEWPMPT,X
7880     STA PMPTLINE+1,X
7890     DEX
7900     BPL NP1
7910     JSR CHKSURE
7920     CPY #1
7930     BNE DONEW
7940     JSR RSTPMPT
7950     JMP GETEM
7960 DONEW
7970     PLA
7980     PLA
7990     LDA SCREEN
8000     CLC
8010     ADC #40
8020     STA ZSCR
8030     LDA SCREEN+1
8040     ADC #0
8050     STA ZSCR+1
8060 ;
8070 ;-----------------------
8080 ; Clear the entire screen
8090 ;-----------------------
8100 ;
8110     LDX #19
8120 D01 LDA #0
8130     LDY #39
8140 D02 STA (ZSCR),Y
8150     DEY
8160     BPL D02
8170     LDA ZSCR
8180     CLC
8190     ADC #40
8200     STA ZSCR
8210     LDA ZSCR+1
8220     ADC #0
8230     STA ZSCR+1
8240     DEX
8250     BNE D01
8260     JSR CLRBUF
8270     JMP ENTERTEXT
8280 ;
8290 ;-----------------------
8300 ; Clear the replacement
8310 ; letter buffers.
8320 ;-----------------------
8330 ;
8340 CLRBUF
8350     LDX #26
8360     LDA #0
8370 D03 STA REPSET,X
8380     STA UNDOBUF,X
8390     DEX
8400     BPL D03
8410     RTS
8420 ;
8430 ;-------------------------------
8440 ; This routine will QUIT the
8450 ; puzzle after prompting the
8460 ; user and then exit to DOS
8470 ;-------------------------------
8480 ;
8490 QUIT
8500     JSR SVEPMPT
8510     JSR CLRPRMPT
8520     LDX #17
8530 Q1  LDA QUTPMPT,X
8540     STA PMPTLINE+1,X
8550     DEX
8560     BPL Q1
8570     JSR CHKSURE
8580     CPY #1
8590     BNE EXIT
8600     JSR RSTPMPT
8610     JMP GETEM
8620 EXIT
8630     PLA
8640     PLA
8650     LDX #$10
8660     LDA #$0C
8670     STA ICCMD,X
8680     JSR CIOV
8690     LDX #$20
8700     LDA #$0C
8710     STA ICCMD,X
8720     JSR CIOV
8730     LDA #64
8740     STA DLIENA
8750     LDA 741
8760     CLC
```

Cryptogra

```
8770      ADC #1
8780      STA 560
8790      STA Z5CR
8800      LDA 742
8810      STA 561
8820      STA Z5CR+1
8830      LDY #4
8840      LDA (Z5CR),Y
8850      STA 88
8860      INY
8870      LDA (Z5CR),Y
8880      STA 89
8890      LDA #0
8900      STA CR5INH
8910      LDA #$94
8920      STA BCK
8930      LDA #$CA
8940      STA LUM
8950      LDA #2
8960      STA COLCR5
8970      LDA #12
8980      STA KEYPR5
8990      RTS
9000 ;
9010 ;-----------------------------
9020 ; We need to UNDO the last
9030 ; change made by the user. We
9040 ; will have to swap the old
9050 ; rep buffer with the current
9060 ; buffer. This way you can
9070 ; UNDO the UNDO. Great uh??
9080 ;-----------------------------
9090 ;
9100 UNDOIT
9110      JSR SVEPMPT
9120      JSR CLRPRMPT
9130      LDX #18
9140 U1   LDA UNDPMPT,X
9150      STA PMPTLINE+10,X
9160      DEX
9170      BPL U1
9180      LDX #25
9190 U7   LDA UNDOBUF,X
9200      PHA
9210      LDA REP5ET,X
9220      STA UNDOBUF,X
9230      PLA
9240      STA REP5ET,X
9250      DEX
9260      BPL U7
9270      JSR DOREP
9280      LDA #0
9290      STA 19
9300      STA 20
9310 U2   LDA 20
9320      CMP #100
9330      BNE U2
9340      JSR R5TPMPT
9350      JMP GETEM
9360 ;
9370 ;-----------------------------
9380 ; Come in here to do the
9390 ; actual changes to the puzzle
9400 ; We have to check each screen
9410 ; location to see if there
9420 ; is a match in the REP5ET. We
9430 ; also need to move 'special'
9440 ; characters right on up with
9450 ; no changes. All 'SPACE's are
9460 ; left alone.
9470 ;-----------------------------
9480 ;
9490 DOREP
9500      JSR PRNTREP5ET
9510      LDA # <FR5TLINE
9520      STA Z5CR
9530      LDA # >FR5TLINE
9540      STA Z5CR+1
9550      LDA #9
9560      STA ROWTMP
9570 NXTLNE
9580      LDY #2
9590      CLC
9600      LDA Z5CR
9610      ADC #40
9620      STA Z5CR1
9630      LDA Z5CR+1
9640      ADC #0
9650      STA Z5CR1+1
9660 CHKNXT
9670      LDA (Z5CR1),Y
9680      CMP #0
9690      BEQ NXTCHR
9700      CMP #$21
9710      BC5 CHKREP
9720 INVIT
9730      ORA #$80
9740 CH5
9750      STA (Z5CR),Y
9760      JMP NXTCHR
9770 CHKREP
9780      SEC
9790      SBC #$21
9800      TAX
9810      LDA REP5ET,X
9820      CMP #0
9830      BNE INVIT
9840      BEQ CH5
9850 NXTCHR
9860      INY
9870      CPY #38
9880      BNE CHKNXT
9890      CLC
9900      LDA Z5CR
9910      ADC #80
9920      STA Z5CR
9930      LDA Z5CR+1
9940      ADC #0
9950      STA Z5CR+1
9960      DEC ROWTMP
9970      BNE NXTLNE
9980      RTS
9990 ;
010000 ;-----------------------------
010010 ; We need to save the PROMPT
010020 ; line in temp storage before
010030 ; we prompt the user with some
010040 ; kind of Yes/No question. We
010050 ; will then be able to restore
010060 ; the prompt line to it's
010070 ; original condition if they
010080 ; answer NO.
010090 ;-----------------------------
010100 ;
010110 SVEPMPT
010120   LDX #39
010130 SV1 LDA PMPTLINE,X
010140   STA PMPTLN,X
010150   DEX
010160   BPL SV1
010170   RTS
010180 ;
010190 ;-----------------------------
010200 ; Time to replace the SAVED
010210 ; prompt line back onto the
010220 ; screen.
010230 ;-----------------------------
010240 ;
010250 R5TPMPT
010260   LDX #39
010270 RS1 LDA PMPTLN,X
010280   STA PMPTLINE,X
010290   DEX
010300   BPL RS1
010310   RTS
010320 ;
010330 ;-----------------------------
010340 ; This will take the REPLACE
010350 ; letters from the prompt line
010360 ; and use them to point into
```

m Solver

```
010370 ; REPSET to store the WITH
010380 ; chars from the prompt line
010390 ; so that the DOREP routine can
010400 ; make changes to the puzzle
010410 ;
010420 ; Also going to save the old re
p
010430 ; set into the UNDO buffer.
010440 ;----------------------------
010450 ;
010460 STOREREP
010470   LDX #25
010480 TOUNDO LDA REPSET,X
010490   STA UNDOBUF,X
010500   DEX
010510   BPL TOUNDO
010520 ;
010530   LDX REPLEN
010540   DEX
010550 TORS TXA
010560   TAY
010570   LDA WITHCHRS,X
010580   STA ROWTMP
010590   SEC
010600   LDA REPLCHRS,X
010610   SBC #$21
010620   TAX
010630   LDA ROWTMP
010640   STA REPSET,X
010650   TYA
010660   TAX
010670   DEX
010680   BPL TORS
010690   JSR DOREP
010700   RTS
010710 ;
010720 ;----------------------------
010730 ; This routine will print the
010740 ; current replacement set on
010750 ; second line of the screen.
010760 ;----------------------------
010770 ;
010780 PRNTREPSET
010790   LDX #25
010800   LDY #33
010810 PR1
010820   LDA REPSET,X
010830   CMP #0
010840   BEQ PR11
010850   ORA #$80
010860 PR11 STA REPLINE,Y
010870   DEX
010880   DEY
010890   CPY #27
010900   BNE PR1
010910   LDA #13
010920   STA REPLINE,Y
010930   DEY
010940 PR2
010950   LDA REPSET,X
010960   CMP #0
010970   BEQ PR22
010980   ORA #$80
010990 PR22 STA REPLINE,Y
011000   DEX
011010   DEY
011020   CPY #19
011030   BNE PR2
011040   LDA #13
011050   STA REPLINE,Y
011060   DEY
011070 PR3
011080   LDA REPSET,X
011090   CMP #0
011100   BEQ PR33
011110   ORA #$80
011120 PR33 STA REPLINE,Y
011130   DEX
011140   DEY
011150   CPY #11
011160   BNE PR3
011170   LDA #13
011180   STA REPLINE,Y
011190   DEY
011200 PR4
011210   LDA REPSET,X
011220   CMP #0
011230   BEQ PR44
011240   ORA #$80
011250 PR44 STA REPLINE,Y
011260   DEX
011270   DEY
011280   CPY #4
011290   BNE PR4
011300   RTS
011310 ;
011320 ;-------------------------
011330 ; The user wants to go
011340 ; back to the REPLACE> 
011350 ; prompt. We have to clear
011360 ; out the WITH> area of
011370 ; the line and do it
011380 ;-------------------------
011390 ;
011400 BCKTOREP
011410   LDA #0
011420   LDX #16
011430 BR1 STA WITHCHRS-5,X
011440   DEX
011450   BPL BR1
011460   LDX REPLEN
011470   JMP PRNTCRS
011480   *=  $02E0
011490   .WORD START
011500   .END
```

Listing 3: Assembly

```
10 ; CRYPTO2.M65
20 ;
30 DLST1
40       .BYTE 112,112,66
50       .WORD MANSCR
60       .BYTE 0,2,144,2
70       .BYTE 2,0,2,2,0,2,2,0
80       .BYTE 2,2,0,2,2
90       .BYTE 0,2,2,0,2,2,0,2,2,0,2,2
0100     .BYTE 144,2,144,2,0,2,65
0110     .WORD DLST1
0120 ;
0130 ;-------------------------
0140 ; The MAIN text entry and
0150 ;  puzzle solving screen
0160 ;-------------------------
0170 ;
0180 MANSCR
0190     .SBYTE "     ABCDEF-GHIJKLM-"
0200     .SBYTE "NOPQRST-UVWXYZ      "
0210 REPLINE
0220     .SBYTE "                    "
0230     .SBYTE "                    "
0240 FRSTLINE
0250     .SBYTE "                    "
0260     .SBYTE "                    "
0270     .SBYTE "                    "
0280     .SBYTE "                    "
0290     .SBYTE "                    "
0300     .SBYTE "                    "
0310     .SBYTE "          CRYPTOGRA"
0320     .SBYTE "M SOLVER            "
0330     .SBYTE "                    "
0340     .SBYTE "                    "
0350     .SBYTE "                    "
0360     .SBYTE "                    "
0370     .SBYTE "                    "
0380     .SBYTE "                    "
0390     .SBYTE "            by Kevi"
0400     .SBYTE "n Peck              "
0410     .SBYTE "                    "
```

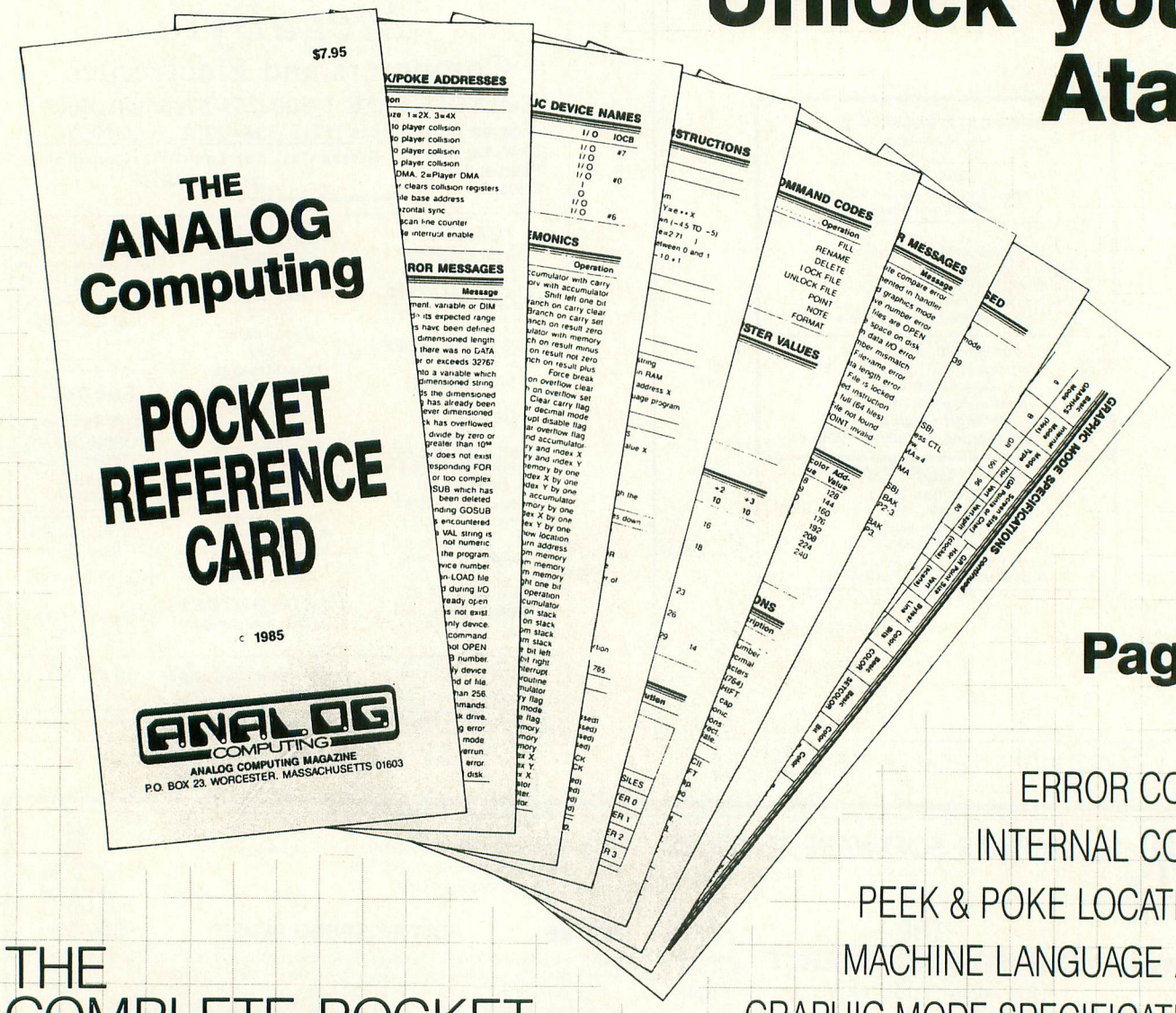
```
0420      .SBYTE "                          "
0430      .SBYTE "                          "
0440      .SBYTE "                          "
0450      .SBYTE "                          "
0460      .SBYTE "                          "
0470      .SBYTE "          Press any k"
0480      .SBYTE "ey to begin               "
0490      .SBYTE "                          "
0500      .SBYTE "                          "
0510      .SBYTE "                          "
0520      .SBYTE "                          "
0530      .SBYTE "                          "
0540      .SBYTE "                          "
0550      .SBYTE "                          "
0560      .SBYTE "                          "
0570      .SBYTE "                          "
0580      .SBYTE "                          "
0590 LINE19
0600      .SBYTE "                          "
0610      .SBYTE "                          "
0620 PMPTLINE
0630      .SBYTE "                          "
0640      .SBYTE "                          "
0650      .SBYTE "                          "
0660      .SBYTE "                          "
0670      .SBYTE "                          "
0680      .SBYTE "                          "
0690 ELN1
0700      .SBYTE "   Use  CLEAR   INSE"
0710      .SBYTE "RT   DELETE  ↓↑←→    "
0720 ELN2
0730      .SBYTE "          BK SP   RETU"
0740      .SBYTE "RN  to enter text    "
0750 REPLCHR$ = PMPTLINE+10
0760 WITHCHR$ = PMPTLINE+28
0770 CMD1
0780      .SBYTE "   CTRL      B ack    "
0790      .SBYTE "  C lear     E dit    "
0800 CMD2
0810      .SBYTE "   KEYS      N ew     "
0820      .SBYTE "  Q uit      U ndo    "
0830 ;
0840 ;-------------------------
0850 ; Define memory locations
0860 ;-------------------------
0870 ;
0880 LUM =    $02C5    ;Char luminance
0890 BCK =    $02C6    ;Background color
0900 BRD =    $02C8    ;Broder color
0910 DMA =    $022F    ;DMA enable
0920 DLI =    $0230    ;Dsply List Addr
0930 INVFLG = $02B6    ;Inv Video Flag
0940 SHFLOK = $02BE    ;Shft Lck 64=uppr
0950 COLORBK = $D018   ;Hrdwre Backgrnd
0960 WSYNC = $D40A     ;Line Sync
0970 DLIVEC = $0200    ;DLI vector
0980 DLIENA = $D40E    ;DLI enable=192
0990 CONSOL = $D01F
1000 SCREEN = $58
1010 VCOUNT = $D40B    ;Vert line count
1020 KEYPRS = $02FC    ;Last key press
1030 ROWCRS = $54      ;Cursor Row
1040 COLCRS = $55      ;Cursor Column
1050 OLDCHR = $5D      ;Chr under Crsr
1060 ZSCR =   $CB      ;Z-page scrn pntr
1070 ZSCR1 = $CD       ;2nd Z scrn pntr
1080 CRSINH = $02F0    ;Cursor ON/OFF
1090 OPEN =   $03      ;CIO open
1100 CLOSE = $0C       ;CIO close
1110 GETCHR = $07      ;CIO get
1120 PUTCHR = $0B      ;CIO put
1130 ICCMD = $0342     ;CIO cmd
1140 ICBAL = $0344     ;CIO buff HI
1150 ICBAH = $0345     ;CIO buff LO
1160 ICBLL = $0348     ;CIO len buf
1170 ICBLH = $0349     ;CIO len buf HI
1180 ICAX1 = $034A     ;CIO aux 1
1190 ICAX2 = $034B     ;CIO aux 2
1200 CIOV =   $E456    ;CIO entry pnt
1210 ;
1220 ;-------------------------------
1230 ; We have to set aside some space
1240 ; for various temporary variables
1250 ; All of that is done here.
1260 ;-------------------------------
1270 ;
1280 REPSET .DS 26    ;rplcment letters
1290 PMPTLN .DS 40    ;Save & rep line
1300 ROWTMP .DS 1
1310 UNDOBUF .DS 26   ;undo buffer
1320 REPLEN .DS 1     ;Length of reps
1330 ;
1340 ;-------------------------------
1350 ; Now we have some screen prompts
1360 ;-------------------------------
1370 ;
1380 REPPMPT .SBYTE "Replace>"
1390 WTHPMPT .SBYTE "With>"
1400 ENTPMPT .SBYTE "Press  START "
1410      .SBYTE " when done"
1420 CPZPMPT
1430      .SBYTE "Clear Puzzle Text?"
1440 CCGPMPT
1450      .SBYTE "Clear All Changes?"
1460 AYSPMPT .SBYTE "Are you sure"
1470      .SBYTE " (Y/N)"
1480 EDTPMPT
1490      .SBYTE "     Editing Puzzle"
1500      .SBYTE " Data          "
1510 QUTPMPT
1520      .SBYTE "Quit, Exit to DOS?"
1530 NEWPMPT
1540      .SBYTE "Start New Puzzle? "
1550 UNDPMPT .SBYTE "Last Change"
1560      .SBYTE " UNDONE."
1570 SCRN .BYTE "E:",$9B
1580 LFRT .BYTE "←→"
1590 ;
1600 ;--------------------------
1610 ; The first DLI starts here
1620 ;--------------------------
1630 ;
1640 COUNT1 .BYTE $FF
1650 COLOR1 .BYTE $80,$30,$02
1660 ;
1670 DLI1
1680      PHA           ;Interupt, Save A
1690      TXA           ;get X reg
1700      PHA           ;and save
1710      LDA VCOUNT    ;Get V line
1720      CMP #40       ;count and
1730      BCS CONT      ;reset count
1740      LDX #$FF      ;on top of
1750      STX COUNT1    ;screen
1760 CONT
1770      LDX COUNT1    ;Get color count
1780      INX           ;add one
1790      LDA COLOR1,X  ;get the color
1800      STA WSYNC     ;wait for horiz
1810      STA COLORBK   ;stuff color
1820      STX COUNT1    ;save new count
1830      PLA           ;Get saved X
1840      TAX           ;put in X
1850      PLA           ;Get saved A
1860      RTI           ;RETURN
1870 ;
1880 ;-------------------------------
1890 ; This section of code will open
1900 ; the screen editor for I/O
1910 ; to write characters to the
1920 ; screen.
1930 ;-------------------------------
1940 ;
1950 OPNKYSCR
1960      LDA $FEFE
1970      CMP #$6C
1980      BNE ISXL
1990      LDA #$FE
2000      STA $79
2010      STA $7A
2020 ISXL
2030      LDX #$20
2040      LDA #OPEN
2050      STA ICCMD,X
2060      LDA # <SCRN
2070      STA ICBAL,X
2080      LDA # >SCRN
2090      STA ICBAH,X
2100      LDA #12
2110      STA ICAX1,X
2120      LDA #0
2130      STA ICAX2,X
2140      JSR CIOV
2150      JMP SETMAIN
```

CIRCLE #107 ON READER SERVICE CARD.

CIRCLE #108 ON READER SERVICE CARD.

CIRCLE #109 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 _____

BITS & PIECES

Enough Is Enough

I lied. Just a little. I admit it! Last month I said you could turn your old 400/800 computer into a peripheral without any hardware changes. Well that is only 80% correct because while the computer goes untouched, you will need to build or modify cables (more on that later). Now, like I promised you, here is Atari Zucchini.

Zucchini is a hardware project.

Zucchini is a software project.

Zucchini allows connection of a slave Atari to a host Atari through the serial port.

Zucchini drives your printer directly through the joystick ports (remember them?).

Zucchini is a 1K machine-language program on a boot disk. This makes it modifiable to create any number of new peripherals and means it runs without DOS.

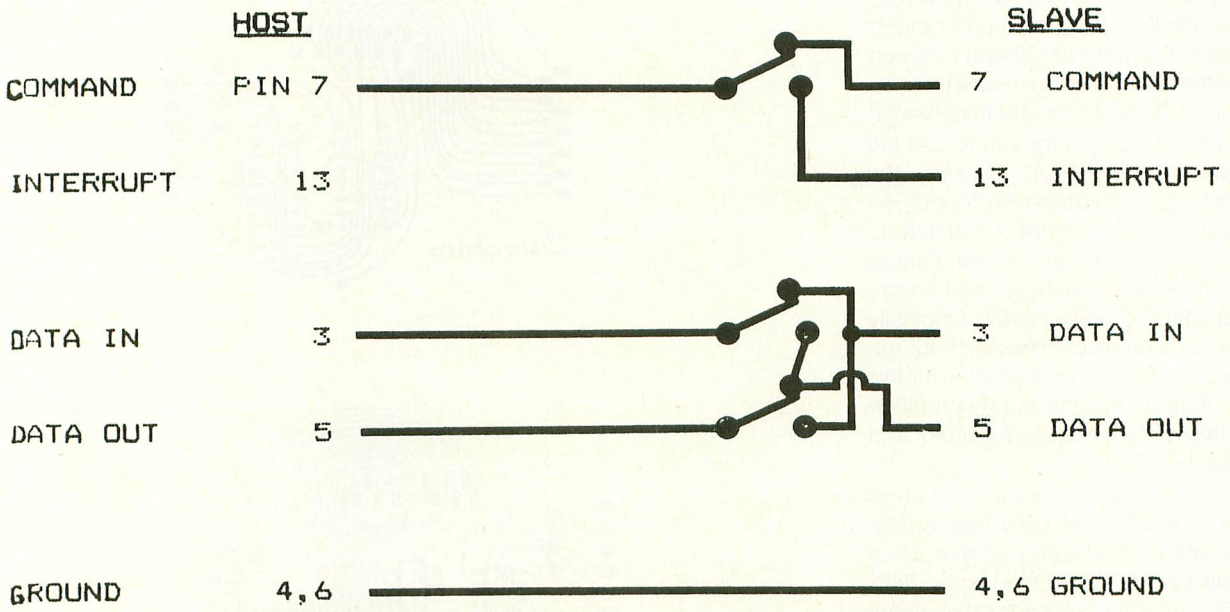
Zucchini allows you to off load text at high speed and run the printer at low speed via its buffer.

Zucchini allows you to print text as graphics to list your BASIC programs including all those unprintable ATASCII characters.

Now that you know a little bit about what Zucchini is, let's look at how to build it and how it works.

The Right Connections

Before we get into the cable modifications, let's look at the job we need the cables to do. In order for an Atari to act as a peripheral we need to make several switches. The Data-In and Data-Out lines need to be reversed, the COMMAND line from the host computer needs to be attached to the INTERRUPT pin on the slave's serial port, and



RADIO SHACK
#275-661

NOT CONNECTED:
1 CLOCK IN
2 CLOCK OUT
8 MOTOR
9 PROCEED
10 + 5 VOLTS
11 AUDIO INPUT
12 +12 VOLTS

Figure 1.

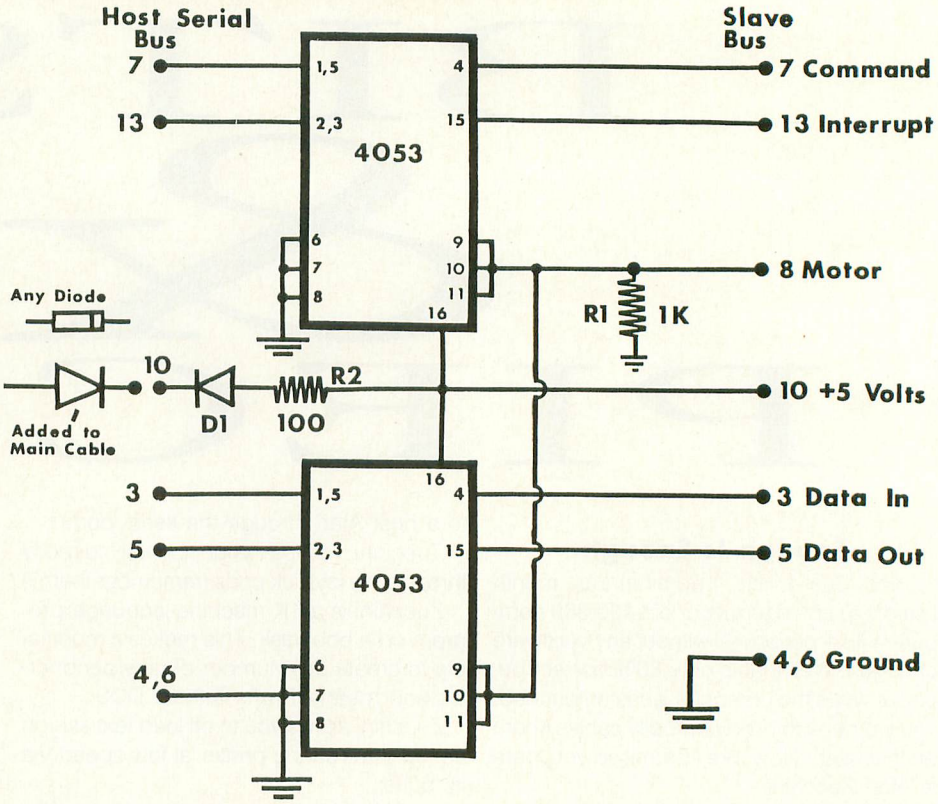
the Clock In and Clock Out completely disconnected. I/O is asynchronous, and so the clock lines are not needed. You can either cross the slave's COMMAND to the host INTERRUPT or you can leave it detached (see XL/XE addendum).

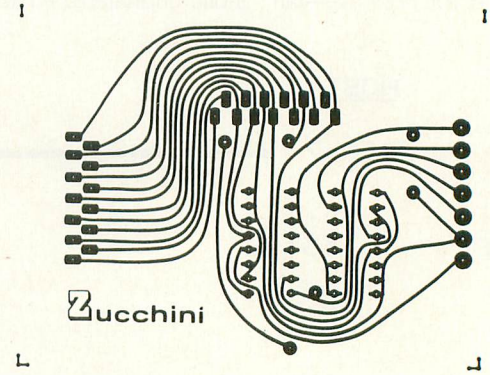
So that leaves us with three lines that need to be changed around. The problem is that you cannot simply exchange the pins in your cable because you need to boot load the Zucchini program before your extra computer can work as a slave. This requires the cable be "normal" to load Zucchini and "altered" to run it. What you need to do is add a multi-pole switch to the cable that connects your Zucchini 800 to your host computer. You can flip the switch one way to load and the other way to use as a peripheral (see Figure 1).

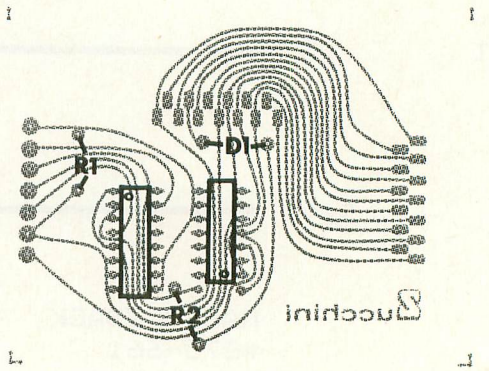
One other problem which you will incur: both host and slave have +5 volts available on Pin 10 of the serial plug. If both computers are connected together through these pins, then one can pull power from the other if it is turned off thereby overloading power supplies. If you simply cut this wire or disconnect the pin in your cable, you will not be able to boot load the program because the disk drive will not operate unless it senses +5 volts on Pin 10. So you also need to modify the cable that normally hooks to your host computer by cutting the lead from Pin 10. This blocks power from flowing between the two computers but it also means that you *must* boot up Zucchini first and that Zucchini must be powered up for you to use the disk drive with your host. Similarly the 800/400's have +12 volts on Pin 12, but you should disconnect this since it is unused.

If you are lazy like me, you will want to build an automatic switching mechanism. The 4053 CMOS IC suits the bill. Originally designed to be an electronic switch for audio signals, it seems to work just fine for this purpose. After all, there is not much difference between high-frequency audio and 19,200 baud.

This circuit connects the Zucchini serial port up in normal fashion, but will switch signals automatically to the slave position when the cassette control line is turned on by ZUCCHINISOFT after it boot loads. This circuit has a diode in the power lead to block power influx from the host. In order to prevent power egress, you need to insert another diode into the first cable from the host where you cut it before. I used the circuit board and added another 13-pin serial socket to allow additional peripherals to be added (such as the 410 cassette or my ALPHACOM 42 printer which do not have their own additional receptacles). I cannibalized a disk-





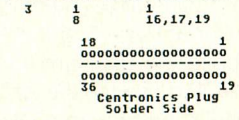


drive cable which I happened to have laying around, but you can build your own Zucchini interface using ribbon cable and new plugs. See the end of this article for sources.

Next you will need to build the printer cable. You need three joystick cables and a 36-pin male, Centronics printer plug. Solder the wires according to this diagram:

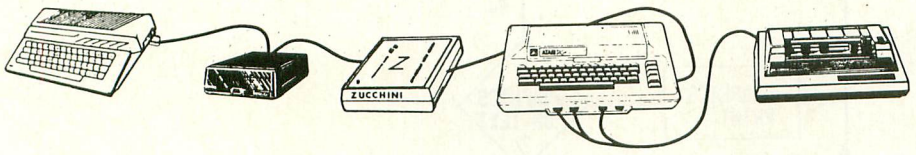
PRINTER CABLE

JOY-STICK	PIN #	CENTRONICS PLUG PIN #	FUNCTION
1	1	2	Data 0
	2	3	Data 1
	3	4	Data 2
	4	5	Data 3
	6	11	Busy
	8	16,17,19	Grounds & Returns
2	1	6	Data 4
	2	7	Data 5
	3	8	Data 6
	4	9	Data 7
	8	16,17,19	Grounds & Returns
3	1	1	Strobe
	8	16,17,19	Grounds & Returns



You can use IDC plugs and ribbon cables, but I found the joystick replacement cables as easy as any to use. When done, plug the 36-pin plug into your printer and the joystick cables into their proper outlets on your Zucchini. Run the following program to test the cable:

```
10 DIM WORD$(20):WORD$="ZUCC
HINI INTERFACE"
20 PORTA=54016:PORTB=54017:P
ACTL=54018:PBCTL=54019
30 P=PEEK(PACTL):POKE PACTL,P
-4:POKE PORTA,255:POKE PACTL,
P
40 P=PEEK(PBCTL):POKE PBCTL,P
-4:POKE PORTB,1:POKE PACTL,P
50 FOR S=1 TO LEN(WORD$):POKE
PORTA,ASC(WORD$(S))
60 IF TRIG(0)=1 THEN 60
70 POKE PORTB,0:POKE PORTB,1:
NEXT S
```



You should recognize how Lines 20-40 convert the joysticks to output. TRIG(0) is used as the "Busy" indicator for the printer and will read 0 when the printer is idle and 1 if busy or disconnected. If it works, then you are ready to go on to the programming.

The Program

If you subscribe to ANALOG on disk you can load Zucchini.BAS and skip on down a ways. But if you are not lucky enough to have the disk, then type in the BASIC listing and save it to disk as Zucchini.BAS *before running.* Now run the program. If the data is correct then the program will halt and give you a message to remove your program disk and insert a new, blank disk which will be formatted by your BASIC program. Then press return and the boot file will write to the disk. If you left your program disk in the drive you would have just lost everything!

If you want to understand the program better or modify it for your own purposes, then type in the source code and list to disk as ZUCCHINI.SRC *before assembling.* Now assemble and check for errors. When error-free then re-list to disk. At this point format a blank disk to hold your final assembly. Now for the final assembly, change the variable ORIGIN to $0700 and assemble again. Since the program assembles into DOS's space, you cannot save, list, or otherwise use DOS once you have done your final assembly. At the end of the assembled program is a routine to transfer the program to disk as a boot file, so go to BUG and run at the address of ENDPRO. If you want to modify your program then *all changes must be between* START *and* ENDPRO. You need to set ORIGIN about 2K above the end of your source code to allow space for assembly so as to not overwrite the source. You can delete all the commentary to create more free buffer space. When your modified program is debugged, LIST to disk and follow the above procedure to create the boot disk.

Eating Zucchini

To use your new interface, connect up your Zucchini 800 to the serial bus using your Zucchini interface and then connect the host computer with the modified regular cable.

Load the ZUCCHINISOFT boot disk into Drive 1 and power up your Zucchini 800. It takes about two seconds to load and the console speaker will beep once when it is ready, but will beep continuously if the printer is not online. Once your Zucchini is online with the printer running, remove the disk and proceed to boot up the host computer as usual. Now you can use your printer just as you normally would using LPRINT or PRINT #1. The big advantage is the speed with which the computer is freed up. Try this:

```
10 DIM WORD$(5200),AB$(26)
20 AB$="ABCDEFGHIJKLMNOPQRST
UVWXYZ"
30 FOR S=1 TO 5200 STEP26:WO
RD$(S)=AB$:NEXT S
40 OPEN #1,8,0,"P:":PRINT#1;
WORD$:CLOSE #1
```

It took one minute, 22 seconds to print WORD$ with my AXIOM AT-846 interface and my Gemini 10X, but with my Zucchini 800 it took only six seconds to transfer WORD$ to the interface and only one minute, eight seconds to print. It took one minute to send last month's article to the interface and about eight or nine minutes to print. This is quite a time saver. But of even more value is this little trick. When the Zucchini interface is idle, press ESCAPE once. Now load a BASIC program into your host and then type LIST "P:". Pretty neat, huh? In case you missed it, your program is listed verbatim to the printer in graphics, not text, in 38-column format, exactly as it appears on screen with inverse characters, graphics characters and so on. So what you say; you have several other programs to do this. Ah, but this one solves two problems. First is that it takes *forever* to print such a listing with a lister program while the Zucchini interface will take the program as *text* as fast as the host can send it. A typical program may take 30 to 90 seconds to dump but may take a half hour to print! Moreover, the carriage-return problem has finally been solved.

What is the carriage-return problem? ATARI uses ATASCII 155 (inverse ESCAPE) as the carriage return, but the printer and the rest of the world recognizes CHR$ (13) as a carriage return. So your interface must convert any bytes of value 155 to 13. When you go to print graphics, you will still convert any bit patterns of value 155 to 13. This plays havoc with your graphics representation of the inverse "A" character and requires you to alter the character set or to replace all 155s with some other number. My AT-846 interface has a jumper option to ignore the conversion, but then you don't get any carriage returns unless you add a CHR$ (13) at the end of everything you print or by adding your own printer handler. So the only way to avoid this problem is Zucchini. To return to standard text, press ESCAPE again and when the interface is idle the conversion will occur.

How much actual buffer space you have depends on your computer configuration. The OS uses memory up to Page 6 or 1536 bytes. Zucchini uses about 1200 bytes, so your buffer space starts at $0BB1. A stock ATARI 400 will have about 13.5K free buffer space and an 800 with 48K will actually have 46K buffer space or about 18 pages of text. I have only been able to fill up the buffer twice: once when I sent all three parts of this series at once (25 pages of text), and the other when printing a full-page poster from *print shop.* If you should be able to, the interface will merely cause the host computer to timeout for 28 seconds while the buffer empties a bit. When the retry occurs the host will then refill the buffer and timeout again. This continues until the host is done sending. You will also note that the printer slows

down a bit while data is being transferred. About the only thing that can go wrong is if the printer is offline and the buffer fills up. Then a real timeout can occur giving ERROR 138. RESET will empty memory and restart the program with console beep and all.

Zucchini Power

While all this is impressive enough it only touches the surface of possibilities. The real power of the Zucchini interface is that it is *programmable!* Examples. . .

There are many possible variations on the theme of printer interfaces that would allow some interesting possibilites. You still have three trigger lines and seven PORTB pins to play with. You can attach a second printer to PORTA and use TRIG1 for BUSY and bit 2 of PORTA for a strobe. Now you have two printers on the same interface. Both the XL and XE OSs support multiple "P:" devices which have device IDs of $40 plus the printer number -1. So "P4:" would have an ID of $43. You could have one printer interface that would respond to OPEN #1,8,0,"P2:" or could drive both a P: and P2: from the same interface. But why stop there. You could probably run four printers simultaneously from the same source. My XL supports up to P9: devices! Great if you do mailings!

You could have several keyboard selectable fonts.

Besides the standard print mode there are two other print modes supportable by the printer handler. OPEN #1,8,68,"P:" changes the AUX1 byte of the command frame from "N"(78) to "D"(68) and sends data in 20-byte frames. Originally this was for double-wide text. Similarly, OPEN #1,8,83,"P:" sends an "S"(83) for sideways (??) printing with 29-byte frames. You can use this for software selectable special effects by adding code to recognize this.

Your Zucchini can be programmed to convert ESCAPE or control code sequences from one format to another. Theoretically you could get your EPSON to respond just like an NEC printer or any other printer for that matter.

Other possibilities? How about a ramdisk? You can use Device ID of $37 for "D8:".

How about a slave terminal to your main computer? If you cross connect your COMMAND line from your slave to INTERRUPT on the main computer, your slave can signal the host it wants to send data. In fact it is possible to do networking or timesharing!

You can produce an 850 emulator with four serial ports from the joysticks. Have you ever seen an 850 with a built-in 46K buffer? Or perhaps a buffer between your modem which can run out of the joystick (already available commercially).

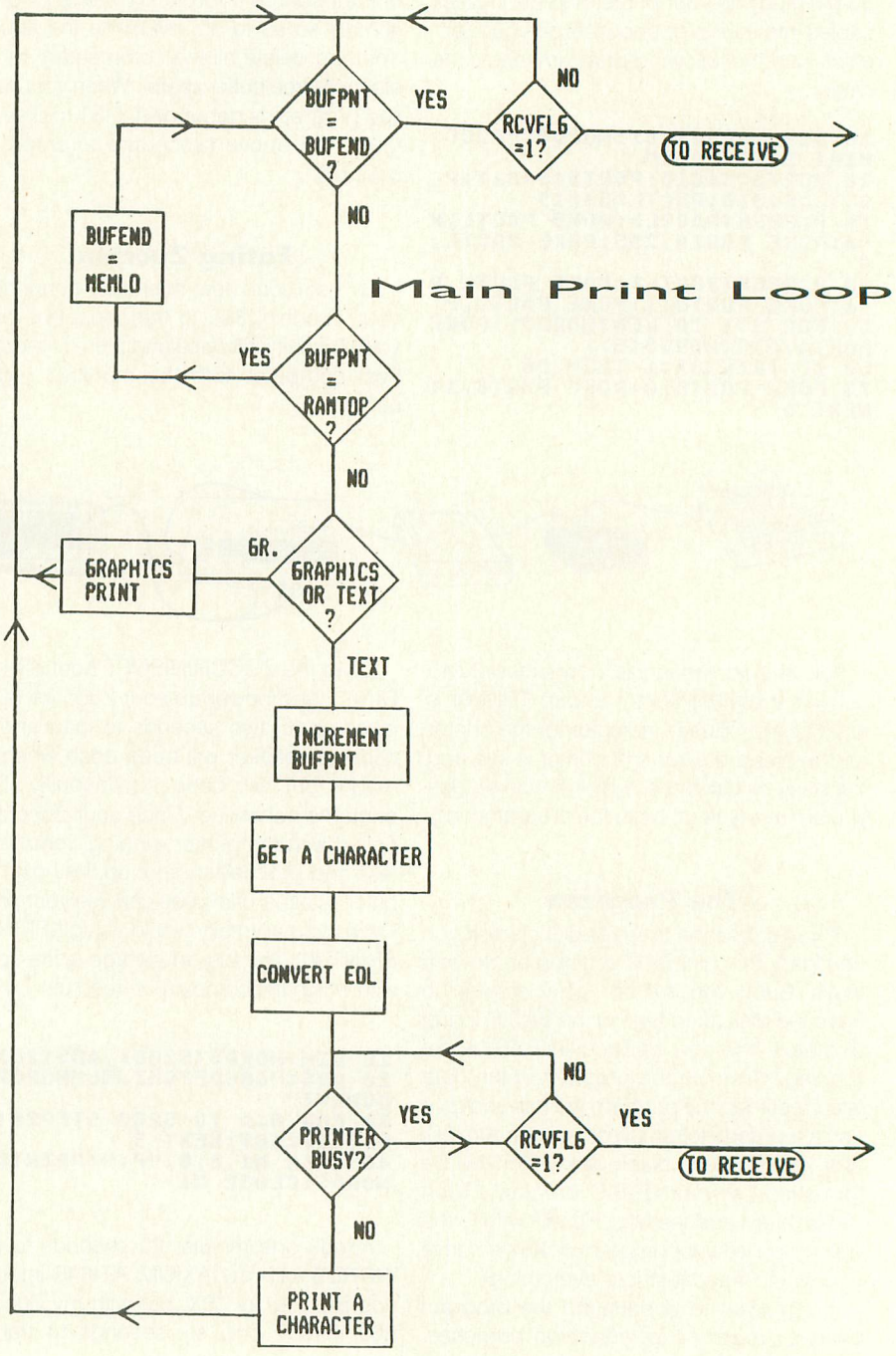
You can even create your own new kind of peripheral such as a large-scale security system with several "X:" devices all checking in with a central Atari computer.

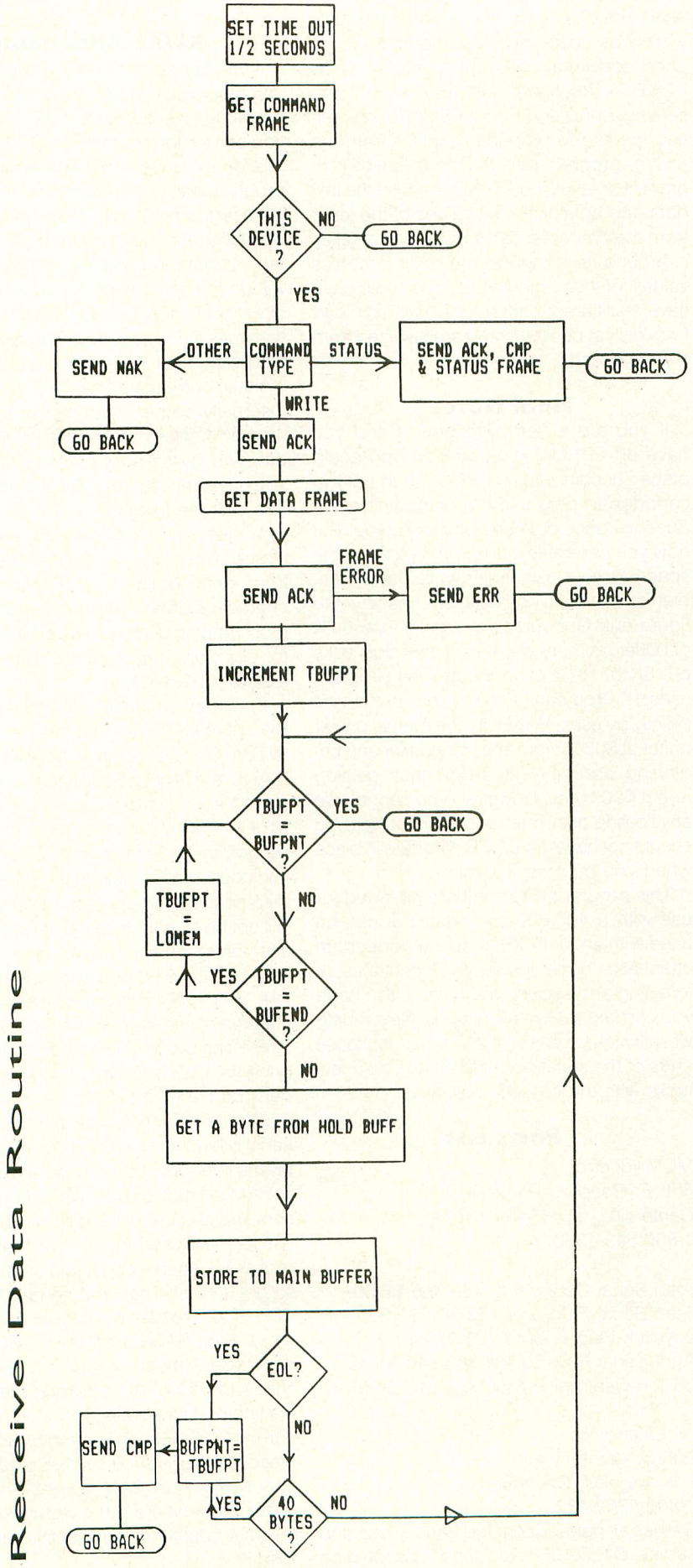
I could keep going on and on, but I am going to leave it up to you, the readers, to come up with the ideas. Please send them in. Let's create a Zucchini net.

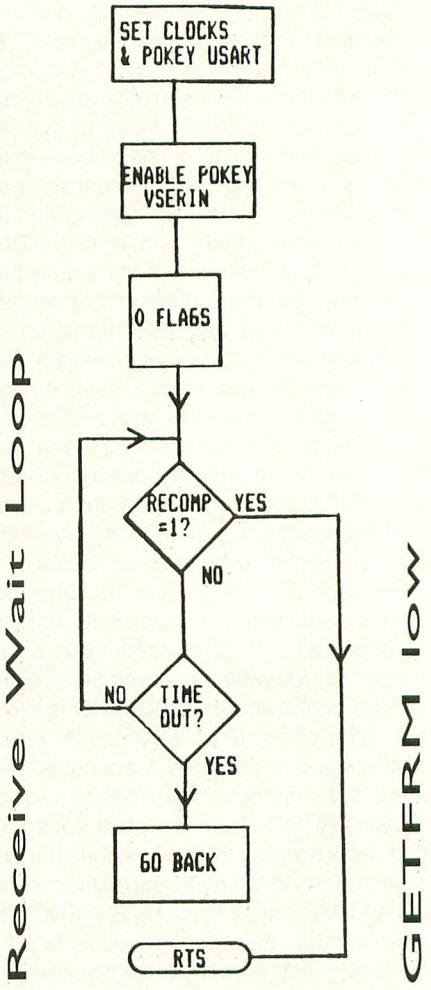
How It Works

If you don't want to know all the details of how Zucchini works, you can stop here, but if you have that insatiable desire to know how things work, or if you plan to modify the program then read on.

The first part of the program is all housekeeping and initialization. It sets up memory pointers and changes interrupt vectors to point to our own routines. ANTIC DMA is turned off, and we replace the keyboard handler with one of our own to detect the escape key. In addition we insert our own SIO interrupt handlers. After the initialization the printer is tested and the beep routines are used to signal readiness. RAM is partitioned with the program starting at $0700 or Page 7, the holding buffer at Page 6 and the remaining RAM above the program for the main buffer. RAMTOP is used as the top end pointer and LOMEM the bottom (LOMEM was moved during the boot to point to the end of the program).







The main printer loop has two pointers which control the flow of data. Refer to the Zucchini flow chart.

BUFEND points to the last byte loaded from the host computer and BUFPNT points to the current byte to be printed. So long as BUFPNT equals BUFEND the loop will idle. When new data is loaded by the receive routines, BUFEND is pushed up in RAM, and the printer loop begins transfer to the printer until the pointers are the same again. When the COMMAND line goes to 0 signaling a COMMAND frame, it causes an IRQ interrupt because of the connections we made in the cables and the new interrupt handlers. This interrupt sets the RCVFLG to 1. This flag will signal the main loop to break out and go to the RECEIV loop. There are two places in the main loop where this flag is tested: the idle loop at the beginning and the printer wait loop. If the flag is set at either point then the RECEIV loop is entered.

Receive Loop

The buffer size is set for four bytes and POKEY initialized for reception. An idle loop is entered and Zucchini waits until the RECeive COMplete flag is set indicating that

all four bytes and checksum have been received. Then the PA1 interrupt is altered to respond to the positive transition of COMMAND as it returns to Logic 1. Once the command frame has been received the program tests for the proper device ID number. If wrong, then you get dumped back to the main loop and there is no response to the host computer. If the ID is correct, then the program sends an ACK and tests for WRITE or STATUS. If it is a status command then a separate routine is entered that sends a CMP followed by the status frame of five bytes after which control returns to the main loop. If a WRITE was called for, then the program moves to the next phase. POKEY is then set up for a 40-byte data frame, and then you go into another holding loop until RECOMP is set again. If there are no errors in data reception, then an ACK is sent and the data is moved to the main buffer.

As each byte is moved, a temporary pointer is advanced until the whole 40 bytes is transferred, an EOL is encountered, or you run out of free buffer space. If the process can be completed then BUFEND is set to the value of the temporary pointer and a CMP byte is sent. If the main buffer gets filled, then the temporary pointer will be equal to BUFPNT at some point during the transfer process. If this occurs, then the program returns to the main loop without sending the CMP byte or updating BUFEND. The result is that the host computer does a timeout for 28 seconds, during which up to 2000 characters can be emptied from the buffer to make more space. Without updating BUFEND, no data was really transferred to the main buffer. If a pointer runs up to the top of RAM, it is "wrapped around" to the bottom of the buffer RAM. This means that every free byte in the main buffer is always available and as material is emptied from the buffer, more space is made available for use.

Besides the main-line program there are many interrupt routines including the VSERIN, VSEROR, VSEROC and the PA1 or INTERRUPT routines. Additionally, there is a routine to set a stage-one VBLANK timer to a half second for the timeout value so if the host computer blows up in the middle of data transfer, the program will not lock up but return to printing. You can see from this system that only a portion of time spent in data transfer is actually spent in receiving data. So long as both the host and the computer agree, you could change the host's print buffer and your Zucchini to expect 256 byte data frames or do burst I/O. This would greatly increase transfer speed.

The interrupt routines work essentially the same as the stock ones in the O.S., and you could save some RAM by using them. For some reason, though, I always got a checksum error when I used the stock-receive interrupt but the stock send routines work okay. The description of how these interrupts work was covered in last month's article. One additional interrupt program uses POKEY's keyboard interrupt vector and senses a pressing of the ESCAPE key setting the flag to produce graphics from the text for program listings. The graphics program itself takes the ATASCII code of the text data and obtains the bit values of the letter from the character set in ROM and shuffles it around to produce the 8-bit graphics values for the printer to use. You could have several different character sets which load from disk at boot time and are selected from the keyboard.

Final Notes

If you are a hardware genius and you have an EPROM programmer, you could place Zucchini into an EPROM and put it in cartridge to plug into the computer, since Zucchini does not need any language. But how you assemble a program into the same space as your assembler while the assembler is running is one problem I have yet to figure out! Of course you would lose 6-7K of buffer memory because a cartridge locks out 8K of RAM even though the program uses 1K. You can increase your buffer size slightly by using PRNBUF, the normal printer buffer at $03C0, for temporary buffer and beginning assembly at $0480 thus gaining about 600 bytes. However, you cannot use any floating point math or basic. But then you should not have the BASIC cartridge in place when you boot up Zucchini.

This program was originally intended for use with a 400/800 computer but can be used with an XL or XE: read the addendum attached. I hope you found Zucchini as interesting and exciting as I have. I also hope you can find many new recipes. Next month we will pick up some of the remaining loose ends of the serial port and show you a few tricks with the cassette player.

Parts List

MCM Electronics
858 E. Congress Park Dr.
Centerville, OH 45459-40721
1-800-543-4330

Atari Serial Cable, 6 ft. #83-365 $5.80
Atari Serial Plug only #83-360 $1.20
Joystick Plug & Cord #83-070 $2.05
Atari Serial P.C. Socket #83-140 $1.45
36 Pin Centronics male #83-310 $1.95

All Electronics
905 S. Vermont Ave.
Los Angeles, CA 90006
1-800-826-5432
Carries all parts except the Serial plugs and cables. D1, D2, IC1, IC2, and R1 should be available at any electronics store or TV shop.

XL/XE Addendum

If you happen to have a leftover XL or XE computer you can take advantage of the extra memory available to you. In the XL/XE you can switch out the O.S. ROM and have a large extra block of RAM available. Your program will have to work around the hardware registers though. This is okay because we really don't use any of the O.S. except the character set and the interrupts. The XE will do the same thing but also has its 64K of extra memory banks. Along with the advantages, you have several problems to contend with. First is the question of joysticks. In these computers PORTB is not available externally and is used internally to control memory bank selection. That leaves you with only two joysticks and no strobe. So since you have only eight output bits you must do one of three things:

1) Use seven bits for data and one for strobe. This is okay for text but prevents you from printing italics or special graphics characters depending on your printer. Also most graphic dump programs are designed for eight bits, not seven. Clearly this is not a desireable solution.

2) Rewire your Zucchini interface to free the slave's COMMAND line to use as a strobe. Once Zucchini is booted the COMMAND line is not needed and can be diverted to the printer cable and be used as the strobe. You will also need to change any programming relating to PBCTL; otherwise you could really muck up things. PBCTL will then control memory banks and so on. You will not be able to drive multiple printers with this configuration.

3) Use the 8-bit port adapter in ANALOG #44, July 1986. This device also has the advantage of allowing two printers to be used.

The second problem to overcome is that once the O.S. is turned off to reveal the underlying RAM, you have no interrupt handlers, and so none of the Zucchini software can work without crashing. You need to supply the interrupt vectors in the last eight bytes of RAM to point to your own interrupt processors and change all the global interrupt vectors for the IRQ interrupts. If you have a copy of the O.S. source code, you are home free, so get a copy. Then you need to set RAMTOP to somewhere below your routines and turn off the BASIC and MATH ROMs. You can bypass much of this by disabling the NMI and IRQ while accessing the RAM for printing or data transfer, then returning to the normal O.S. to process the interrupts. Using these techniques you can use a 130XE with about 120 K of RAM available for buffer space. Now where in the world can you get any other printer buffer with this capacity for $150?

BITS & PIECES

Listing 1: Basic

```
WN 0 REM ********************************
QC 1 REM * ZUCCHINI PRINTER INTERFACE *
WZ 2 REM *  by Lee S. Brilliant M.D.  *
YB 3 REM *   Converts a 400/800 to an *
ML 4 REM *    interface/buffer.       *
WS 5 REM ********************************
NL 6 REM
WI 10 DATA 0,10,0,7,6,7,169,177,141,231,2
   ,169,11,141,232,2,1305
NH 20 DATA 169,26,133,10,169,7,133,11,24,
   96,169,0,141,47,2,141,1278
FD 30 DATA 166,11,141,172,11,141,170,11,1
   41,2,211,141,3,211,169,255,1956
MA 40 DATA 141,0,211,169,52,141,2,211,169
   ,1,141,1,211,169,5,141,1765
IJ 50 DATA 3,211,162,0,169,120,141,4,2,16
   9,10,141,5,2,169,213,1521
YY 60 DATA 141,10,2,169,9,141,11,2,169,10
   4,141,14,2,169,10,141,1235
EL 70 DATA 15,2,169,41,141,12,2,169,10,14
   1,13,2,169,160,141,8,1195
VL 80 DATA 2,169,10,141,9,2,173,231,2,133
   ,203,141,156,11,173,232,1788
YM 90 DATA 2,133,204,141,157,11,169,40,14
   1,4,210,169,0,141,6,210,1738
CX 100 DATA 32,127,10,173,16,208,240,13,1
   73,172,11,240,246,169,160,32,2022
KO 110 DATA 195,10,76,144,7,169,255,32,19
   5,10,141,162,11,173,166,11,1757
DK 120 DATA 240,6,32,229,8,76,173,7,165,2
   03,205,156,11,208,21,165,1905
IJ 130 DATA 204,205,157,11,208,14,173,165
   ,11,205,164,11,240,223,141,164,2296
PB 140 DATA 11,76,173,7,165,204,197,106,1
   44,13,173,231,2,133,203,173,2011
NB 150 DATA 232,2,133,204,76,173,7,230,20
   3,208,2,230,204,173,164,11,2252
XR 160 DATA 240,3,76,34,8,160,0,177,203,2
   01,155,208,2,169,13,32,1681
VM 170 DATA 5,8,76,173,7,141,0,211,173,16
   ,208,240,11,173,166,11,1619
IV 180 DATA 240,246,32,229,8,76,8,8,206,1
   ,211,234,238,1,211,96,2045
YJ 190 DATA 234,96,160,0,140,163,11,177,2
   03,141,171,11,201,155,208,31,2102
DV 200 DATA 172,160,11,208,3,76,249,7,169
   ,0,162,8,32,5,8,202,1472
AE 210 DATA 208,250,238,160,11,173,160,11
   ,201,38,208,236,76,201,8,173,2352
LK 220 DATA 160,11,208,11,162,3,189,173,1
   1,32,5,8,202,16,247,173,1611
GA 230 DATA 171,11,16,8,41,127,141,171,11
   ,206,163,11,201,96,176,12,1562
JD 240 DATA 201,32,144,6,24,105,224,76,12
   4,8,105,64,133,201,169,0,1616
EY 250 DATA 133,202,162,3,6,201,38,202,20
   2,208,249,24,173,244,2,101,2150
AQ 260 DATA 202,133,202,169,0,141,171,11,
   169,128,141,155,11,162,7,160,1962
DW 270 DATA 7,177,201,24,45,155,11,240,1,
   56,110,171,11,136,16,241,1602
EZ 280 DATA 78,155,11,173,171,11,77,163,1
   1,32,5,8,202,16,224,238,1575
WP 290 DATA 160,11,173,160,11,201,38,208,
   10,169,0,141,160,11,169,13,1635
DV 300 DATA 32,5,8,76,173,7,104,170,104,1
   68,169,0,141,166,11,169,1503
EU 310 DATA 5,141,3,211,96,152,72,138,72,
   32,127,10,169,4,141,159,1532
LK 320 DATA 11,32,174,9,169,7,141,3,211,1
   69,0,141,166,11,173,166,1583
PE 330 DATA 11,208,15,173,172,11,240,246,
   76,214,8,172,170,11,192,143,2062
QP 340 DATA 240,196,173,0,6,201,64,240,3,
   76,214,8,173,1,6,201,1802
HN 350 DATA 87,240,23,201,83,208,6,32,21,
   11,76,214,8,160,78,32,1480
AW 360 DATA 230,10,76,169,9,160,65,76,230
   ,10,32,53,9,169,40,141,1479
AX 370 DATA 159,11,32,174,9,172,170,11,24
   0,14,160,69,32,230,10,76,1569
MD 380 DATA 169,9,32,230,10,76,214,8,32,5
   3,9,160,0,162,0,173,1337
KJ 390 DATA 156,11,133,205,173,157,11,133
   ,206,230,205,208,2,230,206,165,2431
LT 400 DATA 205,197,203,208,9,165,206,197
   ,204,208,3,76,214,8,165,206,2474
DG 410 DATA 197,106,144,13,173,231,2,133,
   205,173,232,2,133,206,76,111,2137
VE 420 DATA 9,189,0,6,232,145,205,201,155
   ,240,4,224,40,208,202,165,2225
GW 430 DATA 205,141,156,11,165,206,141,15
   7,11,160,67,76,82,9,169,0,1756
KV 440 DATA 141,158,11,141,162,11,141,161
   ,11,141,167,11,141,170,11,32,1610
QL 450 DATA 88,11,173,167,11,208,10,173,1
   72,11,240,246,169,138,141,170,2128
TK 460 DATA 11,32,136,11,96,152,72,173,15
   ,210,141,10,210,48,5,160,1482
JJ 470 DATA 140,140,170,11,41,32,208,5,16
   0,142,140,170,11,173,158,11,1712
UI 480 DATA 240,20,173,13,210,205,161,11,
   240,5,160,143,140,170,11,238,2140
IV 490 DATA 167,11,104,168,104,64,173,13,
   210,172,162,11,153,0,6,24,1542
JH 500 DATA 109,161,11,105,0,141,161,11,2
   00,140,162,11,204,159,11,48,1634
TN 510 DATA 225,169,1,141,158,11,76,2,10,
   152,72,238,162,11,172,162,1762
EK 520 DATA 11,204,159,11,144,17,173,168,
   11,240,31,165,16,9,8,133,1500
GI 530 DATA 16,141,14,210,76,86,10,185,0,
   6,141,13,210,24,109,161,1402
UX 540 DATA 11,105,0,141,161,11,104,168,1
   04,64,173,161,11,141,13,210,1578
KR 550 DATA 169,1,141,168,11,76,86,10,169
   ,1,141,169,11,165,16,41,1375
JT 560 DATA 247,141,14,210,133,16,104,64,
   169,1,141,166,11,104,64,169,1754
YQ 570 DATA 154,141,38,2,169,10,141,39,2,
   162,0,160,30,169,1,120,1338
HD 580 DATA 32,92,228,169,0,141,172,11,88
   ,96,169,1,141,172,11,96,1619
QQ 590 DATA 173,9,210,201,28,208,19,173,1
   65,11,240,6,206,165,11,76,1901
QZ 600 DATA 186,10,238,165,11,169,0,141,1
   60,11,104,64,202,208,253,136,2058
DH 610 DATA 208,250,96,141,159,11,141,162
   ,11,174,162,11,160,1,169,255,2111
NO 620 DATA 141,31,208,32,188,10,140,31,2
   08,200,174,162,11,32,188,10,1766
OF 630 DATA 206,159,11,208,228,96,152,72,
   169,0,141,169,11,141,159,11,1933
TW 640 DATA 141,162,11,32,123,11,160,2,14
   0,168,11,162,190,32,188,10,1543
XN 650 DATA 104,168,140,13,210,173,169,11
   ,208,8,173,172,11,240,246,76,2122
VN 660 DATA 204,9,76,209,9,160,65,32,230,
   10,160,67,32,230,10,169,1672
GI 670 DATA 128,172,170,11,240,2,169,129,
   141,0,6,173,2,6,141,1,1491
HJ 680 DATA 6,169,31,141,2,6,169,4,141,15
   9,11,169,0,141,3,6,1158
UC 690 DATA 141,162,11,141,169,11,141,168
   ,11,32,123,11,173,0,6,141,1441
RS 700 DATA 13,210,141,161,11,76,5,11,169
   ,19,141,50,2,141,15,210,1375
SR 710 DATA 141,10,210,160,224,132,16,140
   ,14,210,169,40,141,8,210,162,1987
LM 720 DATA 6,169,160,157,1,210,202,202,1
   6,249,96,160,208,169,35,141,2181
EC 730 DATA 50,2,141,15,210,76,101,11,169
   ,192,133,16,141,14,210,169,1650
VH 740 DATA 0,162,6,157,1,210,202,202,16,
```

```
   249,96,0,0,0,0,0,1301
XM 750 DATA 0,0,1,0,0,0,0,0,0,0,0,0,0,1,4
   8,75,125
BG 760 DATA 27,169,10,141,162,11,169,1,14
   1,1,3,169,87,141,2,3,1237
KU 770 DATA 169,0,141,4,3,169,7,141,5,3,1
   69,1,141,10,3,169,1135
UX 780 DATA 0,141,11,3,32,83,228,48,30,17
   3,4,3,24,105,128,141,1154
CA 790 DATA 4,3,173,5,3,105,0,141,5,3,238
   ,10,3,208,3,238,1142
NJ 800 DATA 11,3,206,162,11,208,221,104,9
   6,0,0,0,0,0,0,0,1022
UU 1000 ? "{CLR}":FOR LINE=10 TO 800 STEP 10:
   TOTAL=0:? "TESTING LINE ";LINE:FOR N=1
    TO 16:READ D:TOTAL=TOTAL+D:NEXT N
IB 1010 READ D:IF TOTAL=D THEN 1030
YM 1020 ? "{CLR}     ** ERROR IN LINE ";LINE
   ;" **":STOP
DD 1030 CKSUM=CKSUM+TOTAL:NEXT LINE:IF CK
   SUM<>134459 THEN ? "{CLR}  ** CHECSUM ER
   ROR **":END
KG 1040 ? "{CLR}  DATA IS CORRECT.  INSERT
    BLANK DISK AND PRESS {INV}RETURN{INV}.":? :? "{C
   {INV}AUTION, USE ONLY BLANK DISK!!!{INV}"
GG 1050 POKE 764,255
YS 1060 IF PEEK(764)<>12 THEN 1060
HC 1070 ? :? "     {INV}*** FORMATTING ***{INV}":XIO
    254,#1,0,0,"D:":? :? "     {INV}*** POKING
   ***{INV}"
YZ 1200 RESTORE :POKLOC=1792:FOR TIME=1 T
   O 80:FOR S=1 TO 16:READ D:POKE POKLOC,
   D:POKLOC=POKLOC+1
KN 1210 NEXT S:READ D:GOSUB 2000:NEXT TIM
   E:? :? "    {INV}*** WRITING ***{INV}":A=USR(299
   3):? :? "     *** DONE ***":END
AS 2000 IF PEEK(755)=2 THEN POKE 755,0:RE
   TURN
WR 2010 POKE 755,2:RETURN
```

Listing 2: Assembly

```
20        .TITLE      ZUCCHINI PRINTER
INTERFACE BUFFER CONVERSION
30 ;
40 ;*************************
50 ;* BY LEE BRILLIANT M.D. *
60 ;*************************
70 ;
80 ;
90 ;PROGRAM CONVERTS A 400/800
95 ;COMPUTER TO A PRINTER INTERFACE-
0100 ;BUFFER AND PROGRAM LISTER.
0110 ;PRESS ESC TO CHANGE TO GRAPHICS
0115 ;PRINTING OF TEXT.
0120 ;
0130 ;USES SERIAL PORT TO CONNECT WITH
0135 ;MAIN SYSTEM.
0140 ;USES JOYSTICK PORTS TO INTERFACE
0145 ;WITH PRINTER.
0150 ;
0160 ;
0170 ;*********************
0180 ;* RAM ASSIGNMENTS *
0190 ;*********************
0200 ;
0210 AUDC1 = $D201
0220 AUDCTL = $D208
0230 AUDF3 = $D204
0240 AUDF4 = $D206
0250 BUFF =  $0600
0260 BUFPNT = $CB
0270 CDTMA1 = $0226
0280 CHBAS = $02F4
0290 CHLOC = $C9
0300 CIOV =  $E456
0310 CONSOL = $D01F
0320 CRITIC = $42
0330 DOSVEC = $0A
0340 ICCOM = $0342
0350 IRQEN = $D20E
0360 KBCODE = $D209
0370 MEMLO = $02E7
0380 PACTL = $D302
0390 PBCTL = $D303
0400 POKMSK = $10
0410 PORTA = $D300
0420 PORTB = $D301
0430 RAMTOP = $6A
0440 SDMCTL = $022F
0450 SERIN = $D20D
0460 SEROUT = $D20D
0470 SETVBV = $E45C
0480 SKREST = $D20A
0490 SKCTL = $D20F
0500 SKSTAT = $D20F
0510 SSKCTL = $0232
0520 TBUFPT = $CD
0530 TRIG0 = $D010
0540 VINTER = $0204
0550 VKEYBD = $0208
0560 VSERIN = $020A
0570 VSEROC = $020E
0580 VSEROR = $020C
0590 ;
0600 ;
0610 ;**********
0620 ;* VALUES *
0630 ;**********
0640 ;
0650 ;
0660 ACK =   $41
0670 CHKERR = $8F
0680 CMPLET = $43
0690 EOL =   $9B
0700 ERR =   $45
0710 FRMERR = $8C
0720 LENGTH = $26
0730 NAK =   $4E
0740 OVRRUN = $8E
0750 TOUTER = $8A
0760 ;
0770 ORIGIN = $6600
0780 ;
0790 ;
0800 ;******************
0810 ;* BEGIN PROGRAM *
0820 ;******************
0830 ;
0840     *=  ORIGIN
0850 ;
0860     .BYTE 0      ;HEADER
0865 ; NUMBER OF SECTORS TO BOOT
0870     .BYTE ENDPRO-ORIGIN+127/128
0880     .WORD ORIGIN ;BOOT LOCATION
0890     .WORD RESET ;INIT WARMSTART
0900 ;
0910 RESET
0915     LDA #ENDPRO&255 ;RESRVE SPACE
0920     STA MEMLO    ;FOR PROGRAM
0930     LDA #ENDPRO/256
0940     STA MEMLO+1
0950     LDA #START&255 ;PLACE IN
0955     ;              RESET CHAIN
0960     STA DOSVEC
0970     LDA #START/256
0980     STA DOSVEC+1
0990     CLC
1000     RTS
1010 START LDA #0     ;INITIALIZE O.S.
1020     STA SDMCTL
1030     STA RCVFLG  ;ZERO ALL FLAGS
1040     STA TIMFLG
1050     STA STATUS
1060     STA PACTL   ;RESET PORTS
1070     STA PBCTL
1080     LDA #255    ;ALL PINS OUTPUT
1090     STA PORTA
1100     LDA #52     ;FIX OUTPUTS AND
1105     ;           SET INTERRUPT
1110     STA PACTL
1120     LDA #1
1130     STA PORTB   ;ONE PIN OUT FOR
1135     ;           STROBE
1140     LDA #5      ;FIX OUTPUTS AND
1145     ;           SET INTERRUPT
1150     STA PBCTL
1160     LDX #0
1170     LDA #COMINT&255 ;CHANGE
1175     ;           INTERRUPT VECTORS
1180     STA VINTER  ;FOR COMMND FRAME
1190     LDA #COMINT/256
```

```
1200      STA VINTER+1
1210      LDA #RCVINT&255
1220      STA VSERIN  ;SERIAL IN READY
1230      LDA #RCVINT/256
1240      STA VSERIN+1
1250      LDA #SNDINT&255
1260      STA VSEROC  ;SERIAL OUT
1265      ;           COMPLETE
1270      LDA #SNDINT/256
1280      STA VSEROC+1
1290      LDA #SNDFRM&255
1300      STA VSEROR  ;SERIAL OUT REQST
1310      LDA #SNDFRM/256
1320      STA VSEROR+1
1330      LDA #KEYBD&255
1340      STA VKEYBD  ;KEYBOARD
1345      ;           INTERRUPT VECTOR
1350      LDA #KEYBD/256
1360      STA VKEYBD+1
1370      LDA MEMLO   ;SET BUF POINTERS
1380      STA BUFPNT
1390      STA BUFEND
1400      LDA MEMLO+1
1410      STA BUFPNT+1
1420      STA BUFEND+1
1430      LDA #$28    ;SERIAL BAUD RATE
1440      STA AUDF3
1450      LDA #$00
1460      STA AUDF4
1470 PRNTON JSR SETVBX ;SET 1/2
1475      ;           SECOND TIMEOUT
1480 PR1  LDA TRIG0   ;IS PRINTER ON?
1490      BEQ BEEP    ;YES SO BRANCH
1500      LDA TIMFLG  ;WAIT FOR TIMEOUT
1510      BEQ PR1
1520      LDA #$A0    ;SET BEEP TONE
1530      JSR TONE
1540      JMP PRNTON  ;TRY AGAIN
1550 BEEP LDA #$FF    ;READY BEEPER
1560      JSR TONE
1570      STA COUNT
1580 ;
1590 ;
1600 ;-----------------------
1610 ; MAIN LOOP FOR PRINTER
1620 ;-----------------------
1630 ;
1640 MNLOOP LDA RCVFLG ;INCOMING DATA?
1650      BEQ TSTPNT
1660      JSR RECEIV
1670      JMP MNLOOP
1680 TSTPNT LDA BUFPNT ;ALL DATA SENT?
1690      CMP BUFEND  ;COMPARE END DATA
1695 ;           WITH CURRENT POINTER
1700      BNE TSTEND
1710      LDA BUFPNT+1
1720      CMP BUFEND+1
1730      BNE TSTEND
1740      LDA GRWANT  ;GRAPHIC OR TEXT?
1750      CMP GRFLG
1760      BEQ MNLOOP
1770      STA GRFLG   ;YES, SET FLAG
1780      JMP MNLOOP  ;NO, GO BACK
1790 TSTEND LDA BUFPNT+1 ;END OF RAM?
1800      CMP RAMTOP
1810      BCC INCPNT  ;OK
1820      LDA MEMLO   ;MOVE POINTERS
1825      ;           TO START OF RAM
1830      STA BUFPNT
1840      LDA MEMLO+1
1850      STA BUFPNT+1
1860      JMP MNLOOP  ;LOOP BACK
1870 INCPNT INC BUFPNT ;INC POINTER
1880      BNE GRTST
1890      INC BUFPNT+1
1900 GRTST LDA GRFLG ;GRAPHICS?
1910      BEQ GETCH   ;NO, PRINT TEXT
1920      JMP GRPRNT  ;YES, GRAPHICS
1930 GETCH LDY #0     ;GET A CHARACTER
1940      LDA (BUFPNT),Y
1950 PREOL CMP #EOL   ;CONVERT ATASCII
1955      ;           EOL TO PROPER EOL
1960      BNE PRNTC1
1970      LDA #$0D
1980 PRNTC1 JSR PRINT ;PRINT & RETURN
1990      JMP MNLOOP
2000 PRINT STA PORTA ;BITS TO PRINTER
2010 WTBUSY LDA TRIG0 ;PRINTER BUSY?
2020      BEQ PRNTCH  ;NO, BRANCH
2030      LDA RCVFLG  ;YES, TEST FOR
2035      ;           INCOMING DATA
2040      BEQ WTBUSY
2050      JSR RECEIV  ;GO RECEIVE
2060      JMP WTBUSY
2070 PRNTCH DEC PORTB ;SET STROBE
2080      NOP         ;WAIT FOR PRINTER
2090      INC PORTB   ;RESET STROBE
2100      RTS
2110 DELAY NOP        ;DUMMY DELAY
2120      RTS
2130 GRPRNT LDY #0    ;PRINT TEXT
2135      ;           AS GRAPHICS
2140      STY INVFLG
2150      LDA (BUFPNT),Y ;GET A CHAR
2160      STA TEMP
2170      CMP #EOL    ;EOL?
2180      BNE TSTCNT  ;NO BRCH TO PRINT
2190      LDY CCOUNT  ;YES-LINE FULL?
2200      BNE LINFIL  ;NO SO FILL
2210      JMP PREOL   ;YES SO PRINT EOL
2220 LINFIL LDA #0    ;FILL OUT PRINTER
2225      ;           LINE WITH 0s
2230      LDX #8
2240 FILOOP JSR PRINT ;8 0'S PER CHAR
2250      DEX
2260      BNE FILOOP
2270      INC CCOUNT
2280      LDA CCOUNT
2290      CMP #LENGTH ;ALL LINE SENT?
2300      BNE LINFIL
2310      JMP EXIT
2320 TSTCNT LDA CCOUNT ;NEW LINE?
2330      BNE CONVRT  ;NO SO BRANCH
2340      LDX #3
2350 SNDCOD LDA CODE,X ;SEND ESCPE SEQ
2360      JSR PRINT
2370      DEX
2380      BPL SNDCOD
2390 CONVRT LDA TEMP ;ASCII TO ATASCII
2400      BPL CONVR1
2410      AND #$7F
2420      STA TEMP
2430      DEC INVFLG
2440 CONVR1 CMP #96
2450      BCS X8
2460      CMP #32
2470      BCC ADD64
2480      CLC
2490      ADC #224
2500      JMP X8
2510 ADD64 ADC #64
2520 X8   STA CHLOC   ;MULTIPLY X8 TO
2525      ;           GET CHSET OFFSET
2530      LDA #0
2540      STA CHLOC+1
2550      LDX #3
2560 ROT  ASL CHLOC
2570      ROL CHLOC+1
2580      DEX
2590      BNE ROT
2600      CLC
2610      LDA CHBAS   ;ADD TO CHBASE TO
2615      ;           FIND LOC. IN RAM
2620      ADC CHLOC+1
2630      STA CHLOC+1
2640 SHIFT LDA #0     ;CREATE NEW BYTE
2645      ;           FROM BITS OF
2650      STA TEMP    ;EACH BYTE IN CHR
2660      LDA #$80
2670      STA BITMSK  ;SELECT WHICH BIT
2680      LDX #7
2690 SLOOP1 LDY #7
2700 SLOOP2 LDA (CHLOC),Y ;SAME BIT IN
2705      ;           EACH BYTE
2710      CLC
2720      AND BITMSK
2730      BEQ NXTBIT
2740      SEC
2750 NXTBIT ROR TEMP
2760      DEY
2770      BPL SLOOP2
2780      LSR BITMSK  ;NEXT BIT
2790      LDA TEMP
2800      EOR INVFLG  ;INVERSE CHAR?
```

```
2810      JSR PRINT
2820      DEX
2830      BPL SLOOP1
2840      INC CCOUNT
2850      LDA CCOUNT  ;ALL CHARS SENT?
2860      CMP #LENGTH
2870      BNE EXIT2
2880 EXIT LDA #0      ;RESET CCOUNT
2890      STA CCOUNT
2900      LDA #$0D    ;SEND EOL
2910      JSR PRINT
2920 EXIT2 JMP MNLOOP
2930 ;
2940 ;
2950 ;**********
2960 ;* RETURN *
2970 ;**********
2980 ;
2990 RETURN PLA       ;RESTORE REGS
3000      TAX
3010      PLA
3020      TAY
3030      LDA #0
3040      STA RCVFLG  ;ZERO FLAG
3050      LDA #5
3060      STA PBCTL   ;RESTORE VINTER
3070      RTS
3080 ;
3090 ;
3100 ;********************
3110 ;* RECEIVE ROUTINE *
3120 ;********************
3130 ;
3140 ;
3150 RECEIV TYA       ;RECEIVE A DATA
3155      ;           FRAME
3160      PHA         ;SAVE REGISTERS
3170      TXA
3180      PHA
3190      JSR SETVBX  ;SET TIMEOUT
3200      LDA #4
3210      STA BUFSIZ  ;# OF BYTES IN
3215      ;           IN COMMAND FRAME
3220      JSR GETFRM
3230 CNGINT LDA #7    ;CHANGE VINTER TO
3235      ;           RESPOND
3240      STA PBCTL   ;TO + TRANSITION
3250      LDA #0
3260      STA RCVFLG
3270 INTWAT LDA RCVFLG ;WAIT FOR END
3275 ;     OF  COMMAND FRAME INTERRUPT
3280      BNE TSTDEV  ;YES SO BRANCH
3290      LDA TIMFLG  ;TIMEOUT?
3300      BEQ INTWAT  ;NO-KEEP WAITING
3310      JMP RETURN  ;YES-RETURN
3320      LDY STATUS  ;OPERATION OK?
3330      CPY #CHKERR ;YES SO BRANCH
3340      BEQ RETURN  ;ERROR SO GO BACK
3350 TSTDEV LDA BUFF  ;FIRST BYTE IS
3355      ;           DEVICE ID
3360      CMP #$40    ;RIGHT DEVICE ID?
3370      BEQ THISDV  ;YES SO BRANCH
3380      JMP RETURN  ;NO SO RETURN
3390 THISDV LDA BUFF+1 ;CHECK COMMAND
3400      CMP #$57    ;WRITE?
3410      BEQ FRMOK   ;YES SO GO ON
3420      CMP #$53    ;STATUS?
3430      BNE WNGCOM  ;NO-WRONG COMMAND
3440      JSR GDSTAT  ;SEND STAT FRAME
3450      JMP RETURN  ;BACK TO MNLOOP
3460 WNGCOM LDY #NAK  ;WRONG COMMAND
3470      JSR SDSTAT
3480      JMP COMPLT  ;SEND A COMPLETE
3485      ;           AND RETURN
3490 SNDACK LDY #ACK  ;SEND AN ACK
3500      JMP SDSTAT
3510 FRMOK JSR SNDACK ;COMMND FRAME OK
3520      LDA #40     ;DATA BUF SIZE=40
3530      STA BUFSIZ
3540      JSR GETFRM  ;GO RECEIVE FRAME
3550      LDY STATUS
3560      BEQ MOVBUF  ;FRAME OK-BRANCH
3570      LDY #ERR
3580      JSR SDSTAT
3590      JMP COMPLT
3600 SNDSTS JSR SDSTAT ;NO SO SEND
3605      ;           ERROR AND RETURN
3610      JMP RETURN
3620 MOVBUF JSR SNDACK ;ACK FRAME
3630      LDY #0
3640      LDX #0
3650      LDA BUFEND  ;SET TEMP POINTER
3655 ;           TO END OF MAIN BUFFER
3660      STA TBUFPT
3670      LDA BUFEND+1
3680      STA TBUFPT+1
3690 INBFPT INC TBUFPT
3700      BNE MOVBF1
3710      INC TBUFPT+1
3715 MOVBF1
3720      LDA TBUFPT  ;MAIN BUFF FULL?
3730      CMP BUFPNT
3740      BNE MOVBYT  ;NO SO BRANCH
3750      LDA TBUFPT+1
3760      CMP BUFPNT+1
3770      BNE MOVBYT
3780      JMP RETURN  ;MAIN BUFFER
3785      ;           FULL SO RETURN
3790 ;RETURN WITHOUT CMP SENT CAUSES
3795 ;HOST TO TIMEOUT FOR UP TO 56
3796 ;SECONDS.
3800 ;THIS GIVES TIME FOR BUFFER TO
3805 ;EMPTY SOME.
3810 ;CAN BE CHANGED BY ALTERING
3815 ;COMMAND RESPONSE FRAME BYTE 3
3820 MOVBYT LDA TBUFPT+1 ;END OF RAM?
3830      CMP RAMTOP
3840      BCC GETBYT  ;NO SO BRANCH
3850      LDA MEMLO   ;RESET POINTER
3855 ;           TO START OF BUFFER
3860      STA TBUFPT
3870      LDA MEMLO+1
3880      STA TBUFPT+1
3890      JMP MOVBF1
3900 GETBYT LDA BUFF,X ;MOVE A BYTE
3905      ;           FROM TEMP
3910      INX
3920 STOBYT STA (TBUFPT),Y
3930      CMP #EOL    ;CARRIAGE RETURN?
3940      BEQ ENDMOV  ;YES-STOP DATA
3950      CPX #40     ;NO THEN WHOLE
3955      ;           BUFF MOVED?
3960      BNE INBFPT  ;NO-GET NEXT BYTE
3970 ENDMOV LDA TBUFPT ;YES SO UPDATE
3975      ;           BUFFER POINTERS
3980      STA BUFEND
3990      LDA TBUFPT+1
4000      STA BUFEND+1
4010 COMPLT LDY #CMPLET ;SEND COMPLETE
4015      ;           BYTE, AND RETURN
4020      JMP SNDSTS
4030 ;-------------
4040 ; GET A FRAME
4050 ;-------------
4060 ;
4070 GETFRM LDA #0    ;GET A DATA FRAME
4080      STA BUFRFL  ;ZERO VARIABLES
4090      STA COUNT
4100      STA CHKSUM
4110      STA RECOMP
4120      STA STATUS
4130      JSR RECVEN  ;ENABLE RECEIVE
4140 CKTIME LDA RECOMP ;RECV COMPLETE?
4150      BNE GOBACK  ;YES, RETURN
4160      LDA TIMFLG  ;TIMEOUT?
4170      BEQ CKTIME  ;NO SO WAIT AGAIN
4180 TIMOUT LDA #TOUTER ;TIMEOUT
4190      STA STATUS
4200 GOBACK JSR DISABLE
4210      RTS
4220 ;
4230 ;
4240 ;--------------------
4250 ; INTERRUPT ROUTINES
4260 ;--------------------
4270 ;RECEIVE INTERRUPT DRIVEN
4275 ; BY VSERIN
4280 ;
4290 RCVINT TYA       ;EACH TIME THERE
4295      ;           IS A BYTE READY
4300      PHA         ;THIS ROUTINE
4305      ;           EXECUTES.
4310      LDA SKSTAT  ;CHECK FOR SERIAL
4315      ;           ERRORS
```

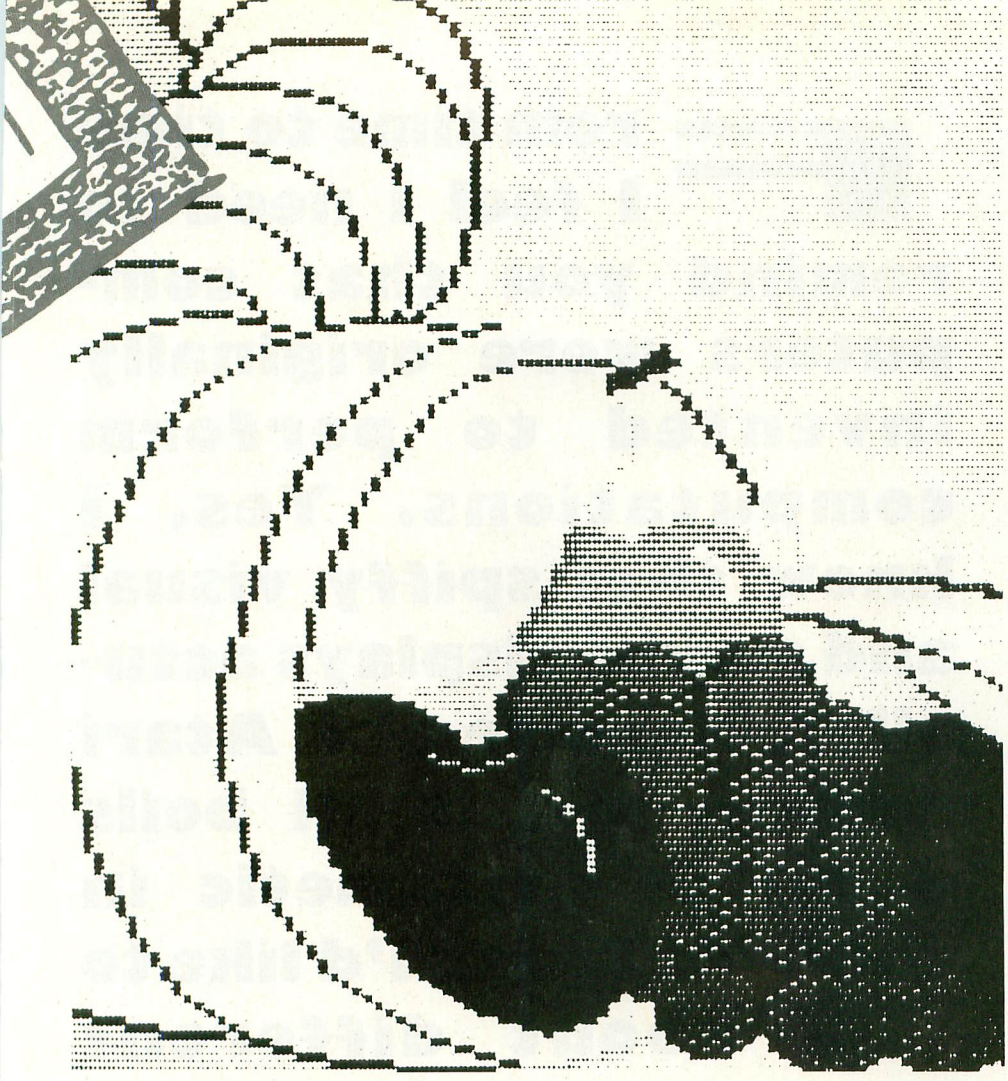
```
4320      STA SKREST  ;RESET ERR REGS
4330      BMI NTFRM
4340      LDY #FRMERR
4350      STY STATUS
4360 NTFRM AND #$20
4370      BNE NTOVRN  ;OVERRUN ERR?
4380      LDY #OVRRUN
4390      STY STATUS
4400 NTOVRN LDA BUFRFL
4410      BEQ NOTDON  ;TEMP BUF FULL?
4420      LDA SERIN   ;CHECKSUM OK?
4430      CMP CHKSUM
4440      BEQ SRETRN
4450      LDY #CHKERR
4460      STY STATUS
4470 SRETRN INC RECOMP ;SET RECEIVE
4475      ;           COMPLETE FLAG
4480 INTDON PLA       ;RESTORE Y
4490      TAY
4500      PLA
4510      RTI         ;RETURN FROM
4515      ;           INTERRUPT
4520 NOTDON LDA SERIN ;GET A BYTE
4530      LDY COUNT
4540      STA BUFF,Y  ;PUT IN TEMP BUF
4550      CLC
4560      ADC CHKSUM  ;TOTAL CHECKSUM
4570      ADC #0
4580      STA CHKSUM
4590      INY
4600      STY COUNT   ;INC BUFF COUNTER
4610      CPY BUFSIZ
4620      BMI INTDON  ;BUFF FULL?
4630      LDA #1      ;YES SET FLAG
4640      STA BUFRFL
4650      JMP INTDON
4660 ;
4670 ;
4680 ;SERIAL OUTPUT DATA REQUEST
4690 ;DRIVEN BY VSEROR
4695 ;
4700 SNDFRM TYA       ;SAVE Y
4710      PHA
4720      INC COUNT   ;ALL DATA OUT?
4730      LDY COUNT
4740      CPY BUFSIZ
4750      BCC ENDSND  ;NO SO SEND
4760      LDA CHKSNT  ;CHECKSUM SENT?
4770      BEQ SNDCHK  ;NO SO SEND
4780      LDA POKMSK  ;YES SO CHANGE
4785      ;           INTERRUPTS
4790      ORA #$08
4800      STA POKMSK
4810      STA IRQEN
4820      JMP IRETRN
4830 ENDSND LDA BUFF,Y ;SEND A BYTE
4840      STA SEROUT
4850      CLC
4860      ADC CHKSUM  ;ADD TO CHECKSUM
4870      ADC #0
4880      STA CHKSUM
4890 IRETRN PLA       ;RETURN FROM
4895      ;           INTERRUPT
4900      TAY
4910      PLA
4920      RTI
4930 SNDCHK LDA CHKSUM ;SEND CHECKSUM
4940      STA SEROUT
4950      LDA #1      ;AND SET FLAG
4960      STA CHKSNT
4970      JMP IRETRN
4980 ;
4990 ;SERIAL OUTPUT INTERRUPT
5000 ;DRIVEN BY VSEROC
5005 ;
5010 SNDINT LDA #1    ;SETS FLAG AFTER
5015      ;           CHECKSUM SENT
5020      STA SNDFLG
5030      LDA POKMSK  ;RESTORE INTRRPTS
5040      AND #$F7
5050      STA IRQEN
5060      STA POKMSK
5070      PLA
5080      RTI
5090 ;
5100 ;
5110 ;COMMAND FRAME INTERRUPT
5120 ;DRIVEN BY VINTER
5125 ;
5130 COMINT LDA #1    ;SET FLAG WHEN
5135 ;       WHEN COMMAND FRAME SET
5140      STA RCVFLG
5150      PLA
5160      RTI
5170 ;
5180 ;
5190 ;SET TIMEOUT INTERRUPT
5200 ;
5205 SETVBX
5210      LDA #STIMOT&255 ;SET POINTER
5215      ;           TO ROUTINE
5220      STA CDTMA1
5230      LDA #STIMOT/256
5240      STA CDTMA1+1
5250      LDX #0      ;SET VB INTERRUPT
5260      LDY #30     ;1/2 SECOND WAIT
5270      LDA #1
5280      SEI
5290      JSR SETVBV
5300      LDA #0
5310      STA TIMFLG  ;ZERO TIMEOUT FLG
5320      CLI
5330      RTS
5340 ;
5350 STIMOT LDA #1    ;SETS FLAG AFTER
5355      ;           TIMEOUT
5360      STA TIMFLG
5370      RTS
5390 ;
5400 ;NEW KEYBOARD INTERRUPT HANDLER
5410 ;SETS FLAG ONLY IF ESC PRESSED
5415 ;
5420 KEYBD LDA KBCODE ;GET ICODE
5425      ;           FOR KEYPRESS
5430      CMP #28     ; ESCAPE?
5440      BNE KRETRN  ;NO, RETURN
5460      BEQ WANTGR
5470      DEC GRWANT
5480      JMP KRETRN
5490 WANTGR INC GRWANT
5500      LDA #0
5510      STA CCOUNT  ;CHAR COUNT=0
5520 KRETRN PLA
5530      RTI
5540 ;
5550 ;TIMER
5560 ;
5565 CONTDN
5570 CONTDN DEX       ;SHORT INTERVAL
5575      ;           TIMER
5580      BNE CONTDN
5590      DEY
5600      BNE CONTDN
5610      RTS
5620 ;
5630 ;
5640 ;SPEAKER TONE ROUTINE
5650 TONE STA BUFSIZ
5660      STA COUNT
5670 TONE1 LDX COUNT ;SET VALUES
5680      LDY #1      ;FOR TIMER
5690      LDA #255
5700      STA CONSOL  ;MOVE SPEAKER
5710      JSR CONTDN  ;TIME DELAY
5720      STY CONSOL  ;MOVE OTHER WAY
5730      INY
5740      LDX COUNT   ;DELAY
5750      JSR CONTDN
5770      BNE TONE1   ;KEEP GOING
5780      RTS
5790 ;
5800 ;
5810 ;**************************
5830 ;**************************
5840 ;
5850 SDSTAT TYA       ;SAVE Y
5860      PHA
5880      STA SNDFLG
5890      STA BUFSIZ
5900      STA COUNT
5910      JSR SENDEN  ;ENABLE POKEY
5920      LDY #2      ;1 MILLISECOND
5925      ;           DELAY
5930      STY CHKSNT
```

8-Bit News

Ever try to use your XL or XE to manipulate a spreadsheet or type a document with wide margins? The 40-column screen of your 130 XE is fine for utilities, games and some light business programs, but it is limiting when you try to use it in an office setting. Atari engineers realized this years ago, when Warner still owned Atari home computers. They began design of an add-on to your XL or XE that would double the number of columns of your screen. After four years, the 80-column board is finally shipping, and reactions from the general public seem to be fairly good, if not a little confused.

The 80-column display unit (Model XEP80, $79.95) plugs into one of your joystick ports to communicate with your XE or XL. A jack on the rear of the box provides composite video output that can be connected to a standard monitor or TV with RF Modulator. According to Atari, several options were explored in which to connect the XEP80 to the Atari CPU. The joystick option won out.

At the past Consumer Electronics Shows, Atari has shown the XEP80 running with a new version of AtariWriter Plus. The new AtariWriter 80 gives you the same functions as the original, but works only with the XEP80 in operation. Atari first announced AtariWriter 80 two years ago, and has steadily pushed back the release date. According to William Robinson, the author of AtariWriter Plus, the original AtariWriter Plus depends heavily on quickly rewriting the XE's screen while you edit a document. The XEP80 must communicate over the joystick ports every time a character is to be plotted on the 80-column screen. This takes a lot of time, so the original AtariWriter Plus program



has to be rewritten to communicate quickly with the XEP80 unit.

Atari has also shown copies of a special version of **Silent Butler**, a general office-mate program which includes scheduling and time-management utilities. The new version makes use of the XEP80's graphic abilities.

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086

Add Some Speed to Your XE

Alpha Systems is shipping the BASIC **TurboCharger**. Since **Atari Basic** can run machine-language programs, J. Bader came up with the novel approach of putting together a library of machine-language programs that can be incorporated into your BASIC programs. The result is machine-language speed for commonly used functions such as memory movers, screen data manipulators, data searches and sorts, and screen animations.

Machine-language routines to handle screen player/missile movement and other animation are included. High-speed disk load and save routines allow you to save **Micropainter** and character sets quickly. Some bit-blit operations are included, which are handy if you are writing programs with screen painting functions.

The **TurboCharger** comes with a 128-page manual and diskette for only $24.95. The diskette contains many short machine-language routines that are completely documented in the manual.

Alpha Systems
1012 Skyland Drive
Macedonia, OH 44056

XE Game

The XE Game system was released last year and has been well received among the stores that sell video-game machines such as the Nintendo and Sega Systems. Atari is releasing even more games for the XE game system. Among the new products are some 8-bit reworkings of classic arcade games. Cartridges of **Food Fight**, **Crossbow, Mario Brothers, Comando** and **Crystal Castles** are all slated for release this summer at a $19.95 list price. Original titles are also being released. **Into the Eagles Nest** is an arcade-style game from Pandora Software. **Airball**, **Dark Chamber** (a D&D-style game) and an original 8-bit version of **Mean 18** (a popular golfing game) are slated for summer release.

The XE game system has been selling fairly well into a market dominated by Nintendo and Sega. The XE Game system has a list price of $149.95, which is high compared to its competition. Some of the success is attributed to the ability to use the XE Game system as a home computer by attaching an available keyboard.

Atari Title, Just Call Me Sir

Datasoft is shipping **Video Title Shop**, a graphic utility program that lets you create your own video effects that you can use for your home-videotape movies. This is a 'must have' program if you are one of the millions of people that have purchased video Camcorders (those little hand-held portable videotape camera and recorders).

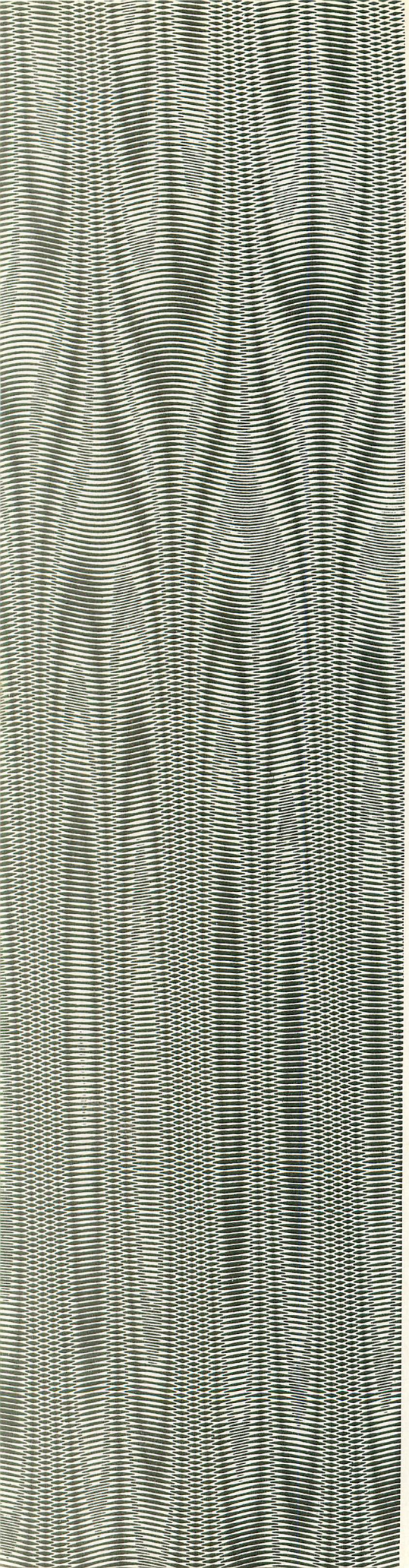
Video Title Shop comes with a copy of **Micropainter Plus**, the original painting program for the 400/800. **Micropainter** is still one of the easiest and most powerful graphics-painting utilities for the XE and XL. Using **Micropainter**, you can draw a screen graphic and then use the other utility programs to superimpose animated text and icons moving over the screen image. The result is semiprofessional-looking animated video titles.

The program carries a $29.95 list price and includes a fairly comprehensive manual that describes most of the program's features. A second diskette is included that contains a number of interesting paintings that can be modified for your own use.

Hamburger/BASIC Helper

A program that adds extra commands to your Atari BASIC is now available from Ronald Hathaway. **Enhancements to BASIC II** adds some handy functions to BASIC. String searches through your programs, multiple-line deletions, renumbering, automatic renumbering, program tracing and variable name changes are now supported. The package comes with a small 30-page manual and reference card for only $24.95. The program works with Atari BASIC revisions A, B and C.

Hathaway Electronics
PO Box 168
Rices Landing, PA 15357



From time to time I feel I need to remind you that computers were originally invented to perform computations. Yes, I know that spiffy visual and sound displays actually comprise the Atari appeal, but it all boils down to arithmetic in the end. Today I'd like to talk about different ways to store numbers in the computer, and present some methods for interconverting characters and numbers. A consequence of this discussion is that soon we'll learn how to keep track of scores in such exciting games as "Attack of the Suicidal Road-Racing Aliens."

Boot Camp

by Karl Wiegers

Storing Numbers

So far, Boot Camp has focused on the most basic method for storing numbers in the computer, as binary integers. As you know, each byte of RAM can contain decimal value from 0 through 255 (hex $FF), based on the pattern of ones and zeros in the eight bits which make up the byte. If both positive and negative numbers must be accommodated, the most significant bit (bit 7) is reserved as a sign bit. If bit 7 is set, the number is negative; if cleared, the number is positive. This method leaves only seven data bits, so signed numbers ranging from -128 through +127 can be represented in this way.

Often we must deal with numbers larger than 255. We've used two adjacent bytes in RAM for this purpose, giving us 16 bits of unsigned data, or 15 bits for signed numbers. With 16 bits we can represent decimal numbers ranging from 0 through 65535 ($00 through $FFFF), or signed numbers from -32768 through +32767.

This is all very fine, but it doesn't cover all our needs. Many numbers encountered in real life have fractional (decimal) parts, such as 7345.022. Obviously, the integer representation fails here. A more elaborate method for storing these so-called floating point numbers is used in the Atari, wherein each number occupies six bytes of RAM, regardless of its magnitude. Floating point storage uses a numeric representation called "binary-coded decimal" or BCD, which we'll discuss more next month. The *Boot Camp* column in issue 43 covered floating point numbers and computations in grim detail.

Another problem arises when we wish to write a program in which the user enters a number that's used in subsequent calculations, or when a calculated number needs to be output to the screen or printer. A number like 7239 is stored internally in only two bytes, with the hex value $1C47. But to print "7239" on the screen requires four characters. To further complicate the issue, to make the character "7" appear we actually have to print the ASCII character code 55 ($37).

(Things are even worse than they appear. The character with ASCII code 5 actually is stored internally in the Atari as character code 23. We won't worry about this today.)

So, if we know that we want to print some numbers, we may want to choose another method for storing them internally, rather than using the standard two-byte integer. One possibility is to reserve one byte for each digit in our number. For the example of "7239," we would use four bytes. But what to put in each byte? We could, of course, simply store "7" in the first byte, "2" in the second, and so on. But we still couldn't print the number out this way. If we output an ASCII code of "7" to the screen, we get the same graphics symbol as you obtain by typing a control-G on the Atari keyboard (a diagonal slash). And you can't even print an ASCII code 7 on a printer. Sending an ASCII 7 to an Epson printer makes the printer's bell ring!

Here's another option. Rather than storing "7" in the first byte, store the ASCII code for "7." Table 1 lists the ASCII codes and bit patterns for the digits 0-9. Note that each digit has an ASCII code equal to the digit value plus $30. Hence, if we printed a byte containing $37, a 7 would indeed appear on the screen or printer.

There are a few problems associated with storing numbers in ASCII form. First, this requires more RAM than does the binary integer form. Also, you can't use the normal addition and subtraction operations, since they are designed to

work with binary numbers.

One good solution to the problem is to go ahead and store numbers in two-byte integer format, and simply convert them to an ASCII string before printing. For input, we must convert the ASCII string typed by the user into its binary numeric representation. The Atari operating system contains built-in routines to convert ASCII strings into their floating point form and back again. Unfortunately, no such routines exist to interconvert integers and ASCII strings. Today I'll present some macros and subroutines to perform all the necessary conversions.

Interconverting ASCII and Binary

Today's example program lets you enter a number containing 1-5 digits at the keyboard. This number is checked to make sure it's valid and then is converted to a two-byte binary integer. Then, the value 25 is added to the integer, and the result is converted to ASCII format and printed on the screen. Let's dive in.

Listing 1 contains three macros (MAC/65 format) that should be appended to your by-now-enormous MACRO.LIB file, using the line numbers shown. These macros use some bytes for work space, which I've defined in the equates in Lines 7380-7400. ASCII is the address where the ASCII string being converted is stored, and NUM is the address where the binary integer value for the number resides. Six bytes are reserved for ASCII, five for digits (the maximum value that works correctly is 65535) and one for an end-of-line character, $9B. The input routine uses the EOL character to know when to stop converting digits, and the output routine adds an EOL so the result can be printed on the screen. COUNTER is just a one-byte work variable.

The first macro, ASC2INT, converts a numeric ASCII string into a two-byte binary integer. Parameter 1 is the address of the string to be converted (for example, an input buffer address), and parameter 2 is the address where the integer should be placed after conversion. This macro calls two subroutines that do most of the work, VALIDASC and ASC2INT (you can give a macro and a subroutine the same name). These and some other subroutines are found in Listing 2, which should be appended to your SUBS.LIB file using the line numbers shown.

The second macro, INT2ASC, converts a binary integer into a printable ASCII string. Parameter 1 is the address of the integer to convert, and parameter 2 is the address where the ASCII string should be placed. As you might expect, this macro calls subroutine INT2ASC, which is also found in Listing 2.

The ASCII string produced by the INT2ASC macro might not require all five characters reserved for it. For example, converting the number 43 to ASCII requires only two bytes for the character string. These digits are right-justified in the five-character ASCII string produced, so the result produced from INT2ASC would have the form 00043.

The LDGZERO macro (Lines 8110-8360 of Listing 1) can be used to convert any leading (that is, on the left) zeros into blanks for printing purposes. However, this macro does not left-justify the result in the five-character field, so if you printed the output ASCII string, you really would print three blanks in front of the 43. LDGZERO doesn't call any subroutines. It takes two parameters. The first is the address of the string to be processed, and the second is the maximum number of digits to examine for leading zeros.

Now let's walk through a sample program and see how these conversion macros and subroutines do their stuff.

ASCII to Integer

Please type in Listing 3, today's sample program. Note the .INCLUDE directives in lines 160 and 650. If your MACRO.LIB and SUBS.LIB files are not on a RAM disk, change the drive designation from D8: to the correct drive number.

Almost every line in this example program is a macro call. This makes the source code much shorter and easier to understand than if we had to expand each procedure into its individual instructions. Also, notice my approach of using a macro in combination with one or more subroutines. The macro sets up the specifics of the particular operation, by virtue of addresses or values passed as parameters. I place the common details of the procedure into a subroutine wherever possible, using reserved pieces of RAM as general work variables. This method makes the resulting object code shorter and yet keeps the source code compact; a satisfactory compromise from my point of view.

Line 380 of Listing 3 makes sure we are in binary mode for arithmetic operations (more about this next month), and Line 390 clears the display screen. Line 400 prints a message prompting you to enter a number containing from one to five decimal digits. Lines 410-420 store your response at address ENTRY, a block of six bytes reserved in Line 540. Line 430 invokes the macro to convert this ASCII string to a two-byte binary integer stored at address INTEGER (defined in Line 550). If the carry flag is set upon completing the macro execution, we know an error has taken place, so Line 440 simply branches to the end of the program.

If we've ended up with a valid number at INTEGER, Line 450 adds 25 to that number. There's nothing magical about this; it's just a way to change the number you entered before I print it out again. Line 460 then converts that sum into an ASCII string at address ENTRY. Line 470 uses the LDGZERO macro to translate any leading zeros to blanks. You might try commenting Line 470 out and seeing what you get. Finally, Lines 480-520 print the resulting ASCII string on the screen and wait for you to press RESET. As usual, you can run this program from address $5000.

Let's look at the ASCII to integer conversion in more detail. The first step is to make sure the user has entered a valid string of ASCII digits. Lines 7560-7640 in the ASC2INT macro definition in Listing 1 handle this chore. The loop simply looks through all the characters stored at the input buffer address (passed as parameter 1) until it finds an end-of-line character. Line 7600 stores each character in the appropriate position in the work variable called ASCII as the checking takes place. The subroutine VALIDASC is called to make sure the characters are all legitimate.

I apologize for bouncing you around

the listings, but now we need to examine subroutine VALIDASC, starting at Line 2980 in Listing 2. Lines 3100-3160 pluck one character at a time out of the ASCII string and check for an EOL. If the first character found is an EOL, then the user just pressed RETURN without entering anything, so Line 3160 branches to an error routine at label INVALID (Line 3280). An error message is printed and the carry flag is set to indicate to the calling macro that an error took place.

The CHKASC routine beginning at Line 3190 tests whether each character has an ASCII value greater than $30 (decimal 0) and smaller than $3A (":," the first character past decimal 9). If not, control again branches to the INVALID routine. If the digit is okay, Lines 3240-3270 strip off the four high-order bits (thereby changing a $37 into a 7, for example), store the result back into the correct position in the ASCII string, and go get the next character.

This procedure underscores my contention that the largest portions of most good computer programs are devoted to input/output routines and error checking. If we knew our users would make only valid entries, our programs could be much shorter. Never make such a shaky assumption, though!

Okay, now the string at address ASCII consists only of valid digits, from one to five of them. The next step is to convert these digits into a binary number. The ASC2INT subroutine (Lines 3390-3700 of Listing 2) does the trick.

Let's contemplate the philosophy of number representation once again. A decimal number like 7239 actually means to multiply 1000 by 7, multiply 100 by 2, multiply 10 by 3, multiply 1 by 9, and add all these products together. To transform a bunch of characters from the ASCII string "7239" into the binary equivalent, we must perform precisely these same operations. The ASC2INT subroutine does the work, with the help of another subroutine called MULT10 (Lines 3740-4020 of Listing 2). The MULT10 subroutine actually carries out the power of ten multiplications.

We begin with the most significant digit in the string to be converted. In the case of "7239," this digit is a 7. Load the 7 into a byte and multiply by 10. This gives 70. Add the next digit in, yielding 72.

Multiply this result by 10 to get 720 and add in the next digit, giving 723. Multiply this result by 10 to get 7230 and add in the final digit, to wind up with 7239. Of course, this answer doesn't look like 7239 in its binary representation. In binary it will look like 0001110001000111, and in hexadecimal it will be $1C47. There's one final twist. The Atari stores two-byte integers in low-byte/high-byte format, so decimal 7239 is represented in two adjacent bytes of RAM in the Atari as hexadecimal 471C. And you thought this stuff was going to be simple!

Lines 3480-3510 of Listing 2 store a zero in the high-byte of our destination integer at address NUM and load the first (most significant) ASCII byte into the low-byte of NUM. If there's only one ASCII character, our conversion is complete; Lines 3520-3550 check for this condition. If the second character is indeed the EOL, Lines 3560-3570 clear the carry flag (our signal to the calling macro that all is well) and return. Otherwise, we go on to the NEXTDIGIT label to continue processing.

The first step is to multiply this leftmost digit by 10. Subroutine MULT10 (Lines 3740-4020 of Listing 2) takes care of this for us. But how do we multiply using the 6502 processor? We've learned how to add and subtract using the ADC and SBC instructions. However, the 6502 contains no intrinsic multiplication or division instructions. You may recall that performing an ASL or Accumulator Shift Left operation is the same as multiplying the contents of a byte by two, and a LSR or Logical Shift Right operation divides the contents of a byte by two. Now we need to extend these concepts to handle a two-byte number and combine shift and add operations to perform integer multiplication.

Remember that multiplication is really just a bunch of sequential additions. The 6502 gives us an easy way to multiply by 2. To multiply some number by 10 we could multiply it by 2; multiply by 2 again (net result is multiply by 4); add the original number back to the result (net result is multiply by 5); and multiply by 2 once again, to give a net result of multiplying by 10. This is precisely what happens in subroutine MULT10.

One more point and then we'll look at the code. Suppose our original number is decimal 150, stored in a single byte as $96. If we multiply that by 2 we get 300 in decimal terms ($012D), but the maximum value that fits in a single byte is 255. Whatever shall we do? When an overflow like this takes place, the carry flag in the processor status register is set, and the original byte contains the value of 300 minus the maximum 255, or 45 ($2D). This carry value must be added to the high-byte of our two-byte number, which also underwent a left shift operation during the multiply by 2 step. Fortunately, the 6502's instruction set contains an instruction to handle all these details, the ROL or Rotate Left instruction.

Each bit shifts to the next higher order position (i.e., to the left). The carry flag shifts into bit 0, and bit 7 shifts into the carry flag. If the carry flag is cleared, ROL is the same as an ASL, simply multiplying the byte's contents by 2. But if the carry is set, the ROL effectively multiplies by 2 and adds 1 to the original byte contents. Hence, a two-byte number can be multiplied by 2 simply by performing an ASL on the low-byte, followed by an ROL on the high-byte to account for the carry flag. I can't believe you didn't think of this solution immediately. (Wiegers' First Law of Computing: Almost nothing you can do with a computer is difficult. Wiegers' Second Law of Computing: Almost nothing you can do with a computer is obvious.)

In sum (pun intended), to multiply a two-byte binary integer by 2, you can simply perform an ASL operation on the low-byte, followed by an ROL operation on the high-byte.

As promised, you may now look at the MULT10 subroutine in Listing 2. Lines 3820-3830 store the high-byte of the original number on the stack so we can grab it for the necessary addition. Line 3840 places the original low-byte into the accumulator. Lines 3850-3860 multiply the original number by 2, and Lines 3870-3880 do it again. Lines 3890-3930 add in the original number, so now we've effectively multiplied it by 5. (Notice that all intermediate results

are stored back in the original location at NUM and NUM+1.) Lines 3940-3950 complete the multiplication by 10. Lines 3960-4010 add in the next digit, as we discussed earlier.

The loop in Lines 3580-3640 of Listing 2 (subroutine ASC2INT) continue this monkey business until an EOL character is reached in the ASCII string, at which point the carry flag is cleared to indicate success and control returns to the calling ASC2INT macro.

We're now back at Line 7660 of Listing 1, in the middle of the ASC2INT macro. If the carry flag is set, there was a problem with the conversion, and an appropriate error message (which lives at Lines 3680-3700 of Listing 2) is printed. Otherwise, the binary result in address NUM is moved to the location specified in the second parameter in the ASC2INT call (Lines 7670-7700), and we're all done.

Integer to ASCII

Whew! We finally got the simple number you entered stored in binary form. Now let's see how to go the other way. Our sample program adds 25 to whatever number you enter, just to change it. The INT2ASC macro converts the number whose address is supplied in parameter 1 to a character string stored at the address specified in parameter 2. The INT2ASC macro is in Lines 7820-8070 of Listing 1. Lines 7940-7970 just copy the number to be transformed to our work space at address NUM.

Subroutine INT2ASC does all the work, creating a five-digit ASCII string of printable characters at address ASCII. Lines 7990-8050 copy this string, up through the EOL character, to the desired destination address in parameter 2. Subroutine INT2ASC is in Lines 4060-4590 of Listing 2. As with ASC2INT, this procedure is based on the fact that the position of a digit in a decimal number indicates the number of times a particular power of 10 must be added to zero to obtain that number. Algorithmically, it's easier to work backwards, performing multiple subtractions. You keep subtracting a particular power of 10 (10, 100, 1000 or 10000) from the integer in question until you obtain a negative result. The number of subtractions you can do before going negative is equal to the value of the digit in a specific column (tens, hundreds, thousands, or ten thousands).

Here's an illustration. Begin with the familiar integer 7239. Let's set a counter equal to 0. How many times can you subtract 10000 from 7239 before you get a negative result? The answer is 0. Hence, the first of our five output digits (the ten thousands column) is 0. Next, how many times can you subtract 1000 from 7239 before obtaining a negative result? Seven, of course. Increment the counter for each successful subtraction. If your counter reaches 8 (representing 7239 minus 8000), the subtraction result is negative, and you know you've gone a digit too far. Add 1000 back in to get back to a positive number (7239-7000 = 239), and use the counter's value of 7 for the second digit in the output ASCII string.

Continue this procedure until all powers of ten from 10000 to 10 have been done, and the remainder (the units column) is the fifth and final digit in the ASCII number. This is awkward to describe in words, but it actually makes some sense.

We'll have to set bits 4 and 5 (ORA –$30) in our counter for the number of successful subtractions to convert it to the ASCII representation. If you walk through the commented INT2ASC subroutine you should understand this technique better. As you can see, it's a pretty cumbersome way to turn a two-byte binary integer into a five-character ASCII string, but it's just about the only way to do it. Lines 4500-4520 of the subroutine add an EOL to the end of the string so it can be printed properly using the PRINT macro, as done in the sample program of Listing 3.

Zero Zapper

The INT2ASC conversion routine produces a five character ASCII string, plus an EOL character. If the integer being converted is smaller than 10000 decimal, the first ASCII digit will be a zero. The number of leading zeros equals five minus the number of decimal digits in the number being converted. Often, you wish to print a number with just significant digits shown, that is, without any leading zeros appearing. The LDGZERO macro, Lines 8110-8360 of Listing 1, replaces leading zeros with spaces.

LDGZERO requires two parameters, the address of the string to be processed and the number of bytes to process before quitting. If a non-zero character (ASCII values $31—$39) is encountered, the routine terminates. The entire logic of this macro consists of looping through the bytes in the ASCII string replacing characters with ASCII code $30 (zero) with ASCII code $20 (blank or space character), until an end condition is satisfied.

As you see when you run the program in Listing 3, leading blanks do "print," effectively shifting the significant digits to the right on the screen. You might want to write a macro or subroutine (or combination) to left-justify a string by simply removing leading zeros, rather than translating them into blanks. That's not a hard exercise to do. While you're at it, why not write a routine to right-justify a string in a field of some specified length? Don't forget error checking. What would happen if you tried to right-justify a string of 11 characters in a field only 8 characters long? Oops.

Decimal Pointers

I alluded to another numeric data storage format, binary-coded decimal. Next month we'll take a close look at BCD and see some routines for converting ASCII strings to BCD and vice-versa.

Table 1.
ASCII / Character / Binary Equivalents.

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

Boot Camp

Listing 1: Assembly

```
7370 ;
7380 ASCII = $0690
7390 NUM =   $0696
7400 COUNTER = $0698
7410 ;
7420 ;*********************************
7430 ;
7440 ;ASC2INT macro
7450 ;
7460 ;Usage:  ASC2INT chars,number
7470 ;
7480 ;'chars' is address of ASCII
7490 ; string to convert,ending w/ EOL
7500 ;'number' is address of integer
7510 ;
7520     .MACRO ASC2INT
7530       .IF %0<>2
7540       .ERROR "Error in ASC2INT"
7550       .ELSE
7560       LDX #255
7570 @ASCLOOP
7580       INX
7590       LDA %1,X
7600       STA ASCII,X
7610       CMP #EOL
7620       BNE @ASCLOOP
7630       JSR VALIDASC
7640       BCS @DONE
7650       JSR ASC2INT
7660       BCS @ASCERROR
7670       LDA NUM
7680       STA %2
7690       LDA NUM+1
7700       STA %2+1
7710       CLC
7720       BCC @DONE
7730 @ASCERROR
7740        PRINT  CONVERTMSG
7750       SEC
7760 @DONE
7770       .ENDIF
7780     .ENDM
7790 ;
7800 ;*********************************
7810 ;
7820 ;INT2ASC macro
7830 ;
7840 ;Usage:  INT2ASC number,chars
7850 ;
7860 ;'number' is address of integer
7870 ;'chars' is address of resulting
7880 ; ASCII string, ending with EOL
7890 ;
7900     .MACRO INT2ASC
7910       .IF %0<>2
7920       .ERROR "Error in INT2ASC"
7930       .ELSE
7940       LDA %1
7950       STA NUM
7960       LDA %1+1
7970       STA NUM+1
7980       JSR INT2ASC
7990       LDX #255
8000 @INTLOOP
8010       INX
8020       LDA ASCII,X
8030       STA %2,X
8040       CMP #EOL
8050       BNE @INTLOOP
8060       .ENDIF
8070     .ENDM
8080 ;
8090 ;*********************************
8100 ;
8110 ;LDGZERO macro
8120 ;
8130 ;Usage:  LDGZERO address,bytes
8140 ;
8150 ;'address' is beginning of ASCII
8160 ; string of digits
8170 ;'bytes' is max number of digits
8180 ; to check for leading zeros
8190 ;
8200     .MACRO LDGZERO
8210       .IF %0<>2
8220       .ERROR "Error in LDGZERO"
8230       .ELSE
8240       LDX #255
8250 @SUPZERO
8260       INX
8270       LDA %1,X
8280       CMP #$30
8290       BNE @LZDONE
8300       LDA #$20
8310       STA %1,X
8320       CPX #%2
8330       BNE @SUPZERO
8340 @LZDONE
8350       .ENDIF
8360     .ENDM
```

Listing 2: Assembly

```
2950 ;
2960 ;*********************************
2970 ;
2980 ;subroutine VALIDASC
2990 ;called by ASC2INT,ASC2BCD macros
3000 ;
3010 ;makes sure all characters in
3020 ;string beginning at address
3030 ;ASCII are valid ASCII codes for
3040 ;numeric digits; looks until it
3050 ;hits an EOL; error message is
3060 ;printed and carry flag is set
3070 ;if an invalid char. is found
3080 ;
3090 VALIDASC
3100     LDX #0
3110 LOOPASC
3120     LDA ASCII,X ;get a char
3130     CMP #EOL    ;EOL?
3140     BNE CHKASC  ;no,go check it
3150     CPX #0      ;yes, 1st char?
3160     BEQ INVALID ;yes,null entry
3170     CLC         ;no, all done
3180     RTS         ;go back
3190 CHKASC
3200     CMP #$30    ;less than 0?
3210     BCC INVALID ;yes, no good
3220     CMP #$3A    ;greater than 9?
3230     BCS INVALID ;yes, no good
3240     AND #$0F    ;clear 4 hi bits
3250     STA ASCII,X ;save 4 lo bits
3260     INX         ;ready for next
3270     BCC LOOPASC ;char
3280 INVALID
3290      PRINT  ASCERRMSG
3300     SEC         ;set carry to
3310     RTS         ;show an error
3320 ;
3330 ASCERRMSG
3340     .BYTE "Non-numeric "
3350     .BYTE "character found",EOL
3360 ;
3370 ;*********************************
3380 ;
3390 ;subroutine ASC2INT
3400 ;called by ASC2INT macro
3410 ;
3420 ;converts string of ASCII digits
3430 ;at address ASCII to a 2-byte
```

```
3440 ;binary integer at address NUM;
3450 ;carry flag set if error
3460 ;
3470 ASC2INT
3480     LDA #0       ;zero hi byte
3490     STA NUM+1
3500     LDA ASCII    ;get first char
3510     STA NUM
3520     LDX #1       ;next char EOL?
3530     LDA ASCII,X  ;next char EOL?
3540     CMP #EOL
3550     BNE NEXTDIGIT ;no, go on
3560     CLC          ;yes all done
3570     RTS
3580 NEXTDIGIT
3590     JSR MULT10   ;multiply by 10
3600     BCS ABORT    ;carry set? error
3610     INX
3620     LDA ASCII,X ;next char EOL?
3630     CMP #EOL
3640     BNE NEXTDIGIT ;no, go on
3650     CLC
3660 ABORT
3670     RTS          ;exit
3680 CONVERTMSG
3690     .BYTE "ASCII conversion"
3700     .BYTE " error...",EOL
3710 ;
3720 ;*******************************
3730 ;
3740 ;subroutine MULT10
3750 ;called by subroutine ASC2INT
3760 ;
3770 ;multiplies binary integer at
3780 ;address NUM and NUM+1 by 10
3790 ;and adds the next digit in
3800 ;
3810 MULT10
3820     LDA NUM+1    ;save high byte
3830     PHA
3840     LDA NUM      ;get low byte
3850     ASL NUM      ;multiply x 2
3860     ROL NUM+1
3870     ASL NUM      ;times 2 again
3880     ROL NUM+1
3890     ADC NUM      ;add to self to
3900     STA NUM      ;effectively
3910     PLA          ;multiply x 5
3920     ADC NUM+1
3930     STA NUM+1
3940     ASL NUM      ;times 2 again,
3950     ROL NUM+1    ;total is now x10
3960     LDA ASCII,X ;add in next char
3970     ADC NUM
3980     STA NUM
3990     LDA #0       ;adding 0 to high
4000     ADC NUM+1    ;byte just pulls
4010     STA NUM+1    ;in carry value
4020     RTS
4030 ;
4040 ;*******************************
4050 ;
4060 ;subroutine INT2ASC
4070 ;called by INT2ASC macro
4080 ;
4090 ;converts a 2-byte binary integer
4100 ;at address NUM to a string of
4110 ;ASCII digits at address ASCII
4120 ;
4130 INT2ASC
4140     LDY #0       ;pointer to table
4150     STY COUNTER ;of powers of 10
4160 NEXTDIGIT2
4170     LDX #0       ;digit counter
4180 SUBLOOP
4190     LDA NUM      ;get low byte
4200     SEC          ;subtract lo byte
4210     SBC DECTABLE,Y ;of current
4220     STA NUM      ;power of 10
4230     LDA NUM+1    ;now subtract hi
4240     INY          ;byte of current
4250     SBC DECTABLE,Y ;power of 10
4260     BCC ADDITBACK ;if neg,restore
4270     STA NUM+1    ;save hi byte
4280     INX          ;digit counter
4290     DEY          ;point to lo-byte
4300     CLC          ;of current power
4310     BCC SUBLOOP ;of 10 again
4320 ADDITBACK
4330     DEY          ;point to lo byte
4340     LDA NUM      ;add lo byte of
4350     ADC DECTABLE,Y ;power of 10
4360     STA NUM      ;back in
4370     TXA          ;convert digit
4380     ORA #$30     ;counter to ASCII
4390     LDX COUNTER ;and store at
4400     STA ASCII,X ;next position
4410     INC COUNTER
4420     INY          ;point to next
4430     INY          ;power of 10
4440     CPY #8       ;at end of table?
4450     BCC NEXTDIGIT2 ;no, go on
4460     LDA NUM      ;get units column
4470     ORA #$30     ;convert to ASCII
4480     LDX COUNTER ;store it
4490     STA ASCII,X
4500     INX
4510     LDA #EOL     ;add an EOL in
4520     STA ASCII,X ;next position
4530     RTS          ;all done
4540 ;
4550 DECTABLE
4560     .WORD 10000
4570     .WORD 1000
4580     .WORD 100
4590     .WORD 10
```

Listing 3: Assembly

```
0100 ;Example 1. Interconverting ASCIi
0110 ;strings and 2-byte integers
0120 ;
0130 ;by Karl E. Wiegers
0140 ;
0150     .OPT NO LIST,OBJ
0160     .INCLUDE #D8:MACRO.LIB
0170 ;
0180 ;-------------------------------
0190 ;     PROGRAM STARTS HERE
0200 ;
0210 ;You'll be prompted to enter a
0220 ;number with 1-5 digits.  This
0230 ;is stored at address ENTRY.
0240 ;The binary integer produced
0250 ;is stored at address INTEGER.
0260 ;If the number is too large,
0270 ;missing (null entry) or has non-
0280 ;digits in it, you'll get an
0290 ;error message.  25 will be added
0300 ;to the value you entered, and
0310 ;the result will be converted to
0320 ;ASCII and printed on the screen
0330 ;
0340 ;-------------------------------
0350 ;
0360     *=  $5000
0370 ;
0380     CLD
0390     JSR CLS
0400      PRINT  PROMPT
0410      POSITION  5,5
0420      INPUT  0,ENTRY
0430      ASC2INT  ENTRY,INTEGER
0440     BCS END
0450      ADD  INTEGER,25
0460      INT2ASC  INTEGER,ENTRY
0470      LDGZERO  ENTRY,5
0480      POSITION  2,8
0490      PRINT  AFTER
0500      POSITION  5,10
0510      PRINT  ENTRY
0520 END JMP END
0530 ;
0540 ENTRY .DS 6
0550 INTEGER .DS 2
0560 ;
0570 PROMPT
0580     .BYTE "Enter a number "
0590     .BYTE "with 1-5 digits:",EOL
0600 AFTER
0610     .BYTE "After adding 25:",EOL
0620 ;
0630 ;-------------------------------
0640 ;
0650     .INCLUDE #D8:SUBS.LIB
```

DATABASE DELPHI

by Michael A. Banks

The Forum

In the preceding installment of this column, we took a look at the electronic mail feature offered by ANALOG's Atari SIG on Delphi. This time, we'll focus on another mode of communications for ATARI SIG members: the Forum, the Atari user's information central.

I've mentioned the Forum before, but I know that a number of you aren't using it, either because you don't know what it is, or because you find it confusing. If you haven't used the Forum, you're missing a lot, believe me! Current hot topics include GEM and MS-DOS compatibility, books and bombs, trackballs and memory upgrades, along with conversations on many other fascinating topics—technical and nontechnical. And if you're looking for an answer to a technical problem, the Atari SIG Forum is the place to go; post a question or comment on virtually any Atari topic, and you'll usually have several answers within two or three days. The Atari SIG Forum is a living database of information on Atari—take advantage of it!

For those of you who are new to using the Forum, what follows is a Forum mini-manual. (You regular Forum users will find this interesting, too; the Forum boasts at least one recently added feature of which you may not be aware.)

What is the Forum?

Simply put, the Forum is a public-message system, similar to public bulletin boards you may have used on your local BBSs, but far more powerful. One way to visualize the Forum is to imagine a series of rooms with chalkboards. Each room is assigned a different topic (General Interest, Toolbox for the ST, etc.—the same basic topics as are assigned the Atari SIG's databases). Within each room there are any number of chalkboards, each with its own subject; on the boards are written messages on that subject, and their replies. The topics, subjects, and messages within the Forum fit this analogy well. The net effect is a series of ongoing, open conversations, any of which you may read and/or participate in.

For your convenience, messages and their replies are organized into groups called "threads." A thread begins with an original message, and continues through each response to that message (including responses to the responses). As you'll learn, you can make the system display all the messages in a thread in a specified sequence, if you wish.

Getting There

To get to the Forum, type FORUM at the main SIG menu's ANALOG> prompt. You'll see the Forum banner and prompt:

```
The Forum
The Voice of the Atari User
Welcome to the ATARI Forum.

Forum contains messages 17000 through 35651.
Highest message you've read is 35646.
    FORUM> Reply, Add, Read,
    "?" or Exit>
```

The commands you'll use most frequently are listed with the prompt, and are pretty much self-explanatory.

The Forum Menu and Commands

To see all available options at the Forum prompt, type ?. The full Forum menu will be displayed:

FORUM Menu:

```
ADD New Message (Thread)
FORWARD
Message by Mail
REPLY
To Current Message
DELETE
Message
READ
Message(s)
EDIT
a Posted Message
```

```
FOLLOW
Thread
NEXT
Message
BACK
to Previous Message
TOPICS
(Set/Show)
DIRECTORY
of Messages
HIGH
Message (Set/Show)
MAIL
HELP
TAG
Interesting Message
EXIT
FILE
Message into Workspace
FORUM> Reply, Add, Read,
"?" or Exit>
```

(If you are using a 40-column display, these commands will be displayed in that format, and will take more than one screen to display).

The commands are self-explanatory, (but I'll explain the more important of them in the following paragraphs). For help with any command, type HELP followed by the name of the command (Example: HELP DIRECTORY).

The Forum Directory (DIR)

The Forum has an extremely sophisticated directory system that allows you to quickly locate messages by date, subject, addressee or message number. You can also locate messages posted by a certain Delphi member. These criteria can be used individually, or combined in any fashion you wish, in the form of qualifiers used with the DIRECTORY (DIR) command.

Here's an example of how this works. If you were interested only in reading Forum messages on the subject of Omnires, you would enter the Forum and type DIR SUBJ Omnires. The system would display a list of all messages on that subject. If you wanted to get more precise, you could type DIR SUBJ Omnires FROM GBA to see a directory of all messages from membername GBA on the subject of Omnires monitors.

Reading Forum Messages (READ)

You can use the same criteria and qualifiers used with the Forum's DIRECTORY command (as detailed in the paragraphs immediately preceding) to read messages. You can also read messages by typing the number of a message (handy after you've used DIR), or simply press <RETURN> to read the next unread (new to you) message.

If Forum messages addressed to you are waiting, you are notified automatically when you enter the Atari SIG. You are notified again of waiting messages when you enter the Forum area. Delphi automatically keeps track of whether you've read waiting messages, and displays them first if you press RETURN when you first enter Forum. If you read other messages before reading those addressed to you, you can display all the messages waiting for you by typing READ WAITING.

You can read only the messages in a thread if you wish—in forward or backward order—using special commands.

Beginning a New Thread/ Adding a Message (ADD)

ADD opens up a new message. When you type ADD, you are prompted for an addressee for the message (type ALL if the message is to everyone), a topic for the message (one of the existing topics), and the message's subject (whatever you want it to be). After you enter this header information, simply type in your message, and press CONTROL-Z when finished; the message will be automatically posted. If it is addressed to a specific individual, he will be notified when he enters the SIG that a Forum message is waiting for him. (If your message receives replies, they will be grouped together as a thread.)

Replying to a Message (REPLY)

If you wish to reply to a message, type REPLY at the Forum prompt after reading the message. You will be prompted to enter the addressee of the message; press <RETURN> and the message will be addressed to the person to whom you are replying, and the Topic and Subject headers will be filled in automatically. Type your message and press CONTROL-Z to post it when you're finished.

Setting Topics (TOPICS)

If you're overwhelmed by the number of messages in the Forum, and know you won't be interested in all the threads, you can, of course, use DIR and READ qualifiers to select specific messages. You can also limit the number of messages you'll see by selecting or de-selecting topics. For example, if you use an Atari 8-bit machine, you might not want to see messages concerning the ST; you can eliminate those messages from being displayed to you using the Forum's SET TOPICS command. Simply type TOPICS, and follow the prompts to select or de-select the appropriate topics. (The topics available in the ATARI SIG are General Interests, Games & Entertainment, Telecommunications, Utilities, Toolbox for the ST, Sight & Sound, Education, Entertainment on the ST, Reviews & News, ST Programs, Koala Pictures, Art on the ST, Current Issue, Home Use and Applications for the ST.)

Other Forum Commands

Additional commands allow you to reply to a member who posted a particular message by private E-mail (type REPLY MAIL), to send copies of messages to other Delphi members by E-mail (type FORWARD) and to copy Forum messages to files in your personal Workspace (type FILE). You would type these commands immediately after reading the message.

Weekly Conferences

Don't forget that SIG Atari hosts a real-time conference each Tuesday at 10 p.m. EST. You'll find the conferences an excellent venue for sharing information about Atari computers, getting answers to questions and participating in friendly discussions of all types.

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

Panak Strikes

by Steve Panak

I thought we'd do something a little different this month. Since this time of year is generally slow as far as new software is concerned, and because a lot of you might be relatively new to the Atari scene, I'll take the next couple of minutes reviewing the current state of entertainment software as a whole, one genre at a time, rather than one game at a time. And we'll start, fittingly enough, where Atari began with arcade games.

Unfortunately, once I began to undertake this monumental task, I discovered that not only were there dozens of games out there fitting any given classification, there were also dozens of games, good games, that defied classification. Games that struggled to bridge the gap between two or three genres. And these games tended to be the best of all, which explains why you've been hearing so much about them over the years.

The Atari 800 was introduced in the early '80s. My first machine, which at 48K was illiterate when compared to my ST, cost me $700. The old 810 drive set me back another $500. And one of the first games I crammed into that drive was Night Mission Pinball, from Sublogic. But what is truly amazing is the fact that in the five-plus years since I bought this game (an eon in the high-tech computer biz) Night Mission still survives as one of the best arcade games available. This pinball simulation is fast and furious, and while it lacks the screen-editing features offered by some of its competitors, I've found that I only rarely use the "construction set" portions of any program. What Night Mission does allow is modification of such play parameters as field incline, elasticity, and ball speed. Up to four can compete head to head, with the

Panak Strikes

high score saved to disk, providing a continual impetus to pump in just one more quarter. This game is so real that you can, and will, tilt in your attempts to nudge a few more out of your last ball.

Another game that's been around since day one (or thereabouts) is Boulder Dash. This Pac Man/Donkey Kong style game features fast action and graphics that push the 8-bits to their limit. While the latest incarnation features a construction-set feature, steer toward Super Boulder Dash for the finest set of challenging screens available in or outside the arcade. The object is to dig through the earth in search of diamonds, while avoiding a number of dangers, most notably falling rocks. Each maze is a puzzle requiring thought, strategy and, of course, great reflexes to survive. Multiple levels of play keep this thriller interesting for months to come. A very similar game, Bounty Bob Strikes Back, is, despite its average graphics, deserving of an honorable mention. Like Boulder Dash, it has a seemingly infinite number of screens to navigate, all crammed into a single high-speed cartridge.

If your tastes run more toward outer space encounters, Star Raiders II should satisfy your lust for violence. In this game the evil Zylon empire has invaded your star system, establishing bases which pump out wave after wave of deadly fighter ships. Your mission, should you choose to accept it, is to stamp out this menace. You do this by traveling throughout the system, destroying fighters, protecting your refueling bases, and searching out and eradicating Zylon strongholds. Detailed graphics and fast action are the hallmarks of great arcade games, and Star Raiders II does not disappoint. Your cockpit contains all the dials and readouts you'll need to keep abreast of game developments, and its level of sophistication approaches that of a simulation, making for a realistic and entertaining campaign. Just don't forget your space suit.

While it's not always true that two can have as much or more fun than one, there are a number of programs out there that excel at placing good friends at one another's throats. Indeed, the most enjoyable of the two-player games are those that require opposition rather than cooperation—competition seems to be an inbred quality of mankind. I've found the best two-player games to consistently be the Spy vs. Spy series. Inspired by the popular characters in *MAD* Magazine, the three games which comprise this trilogy place the two familiar agents in differing locales with one common goal, to be the first to collect and assemble a device, usually a bomb. Rising above a mere search, the game allows the players to slow each other's progress by setting deadly booby traps and engaging in hand-to-hand combat. The graphics are distinct and imaginative, with your spy laughing diabolically as your opponent is incinerated by an expertly placed napalm bomb. And the split-screen layout, allowing simultaneous play when combined with the complex, yet easy to learn set of joystick controls, makes the game effortless to play.

If you prefer a battle of wits, Archon is an Electronic Arts original that combines the strategy of chess with the dexterity of an arcade wrist-buster. It was four years ago that I first played this game, and it still survives as one of the best. Played on a checkered board, each man has a different attack mode or power, with some pieces being stronger than others. Numerous battle modes are used, from clubs to projectiles to magic. Your men are more powerful on their own color squares—but don't get lazy, as even the board colors are subject to change. As you fight, on-screen life lines keep you apprised of your progress (or demise). Archon and its sequel (Archon II) are classics which, like chess, survive the test of time.

If a traditional one-on-one confrontation is more to your liking, then a couple of packages may satisfy you. World Championship Karate from Epyx outfits you in traditional oriental garb and pits two Kung Fu warriors against each other in a fight to the death. Complex joystick commands unleash a battery of offensive and defensive moves, from simple kicks

This game is so real that you can, and will, tilt in your attempts to nudge *a few more out of your last ball.*

and punches to elaborate leaps and spins. Intricate and sophisticated graphics keep this game interesting, and I still love the way your computer opponent turns and looks at you mockingly when you are slow to attack. For those that need something to fight for, Datasoft's Karateka pits you against a succession of Karate experts of increasing strength as you attempt to rescue a princess held hostage in a mountain fortress.

If traditional sparring is more your game, Accolade's Fight Night straps on the gloves and lets you go ten rounds against some of the toughest pugilists you'll ever set your sore eyes on. Loosely patterned after the arcade version, in this game you square off against six boxers, from the lightweight "Dipstick" to the super-heavyweight "Bronx Bomber." One or two compete, and a construction set allows you to design and build your own opponents. While the difficult command structure and slow execution makes Fight Night inferior to either of the Karate games, it is still entertaining. Another game that is great for two players is Trailblazer. This sleeper is so different from anything else on the market that it was destined to be a winner even if it didn't have superb, fast graphics. In this variation of the race game, Mindscape's masterpiece puts each player in control of a checkered ball, with the object being to complete a racetrack suspended in space in the least amount of time. This simple concept is supplemented by a number of special areas on the track which speed you up, slow you down, and bounce you over bottomless pits. The split-screen design is ingenious, allowing each competitor complete control over his ball and providing first-person perspective of the action. You can almost feel your opponent's ball shooting over your head as he passes you.

A number of packages let you take to the air. Probably the most realistic, and most difficult, is Flight Simulator from Sublogic. This complex program mimics perfectly the flight characteristics of a single-engine aircraft, the Cessna 182. So perfectly, in fact, that without flight training you're likely to spend the majority of your time power diving into the Earth—assuming you can take off. And despite the complexity, the program executes remarkably fast. Once you've earned your wings, embellishments include a World War I dogfight battle game, and a number of optical scenery disks covering the entire United States, as well as a number of foreign countries. This is truly a premium program.

If you'd rather forego some of this realism and simply dogfight, a couple of packages will appease you. Ace of Aces from Accolade puts you over Europe during WWII (the big one), where you attempt missions requiring you to fight air to air, air to ground and air to sea, with the Nazis as formidable opponents. Mindscape's Infiltrator sticks you behind the stick of an ultra-sophisticated helicopter, a la Blue Thunder. Both of these programs are complex simulations which push your machine to its limits. And while I thought their attempts to simulate every aspect of their respective missions made the games a little long-winded, the main portions of each are engaging.

Of course, I could go on and on about the rest of the arcade games out there. There are tons of cartridge-based games, many based on arcade blockbusters, others based on movies and cartoon characters. I leave it up to you to peruse though these.

Now, having looked at the entire spectrum of the arcade genre, let's step back a moment and look at one of the latest entrants into this fraternity.

Saracen
by Datasoft Electronics Arts
1820 Gateway Drive
San Mateo, CA 94404
48K Disk $19.95

Saracen, the latest arcade game from Datasoft, is billed as an action adventure in the Middle Ages. When I first saw the game, opened it and read the slight manual, I started to become concerned. Here I sat, discussing some of the best arcade games I'd ever played, and then I opened this. To say that I was prepared

Panak Strikes

for the worst would be a gross understatement. But my tense trepidation turned to relief when I discovered that Saracen was nowhere near as bad as I thought it would be. Actually, it was pretty good.

Of course, not original by any stretch of the imagination. When the first of the 100 levels appears on the screen, Boulder Dash (*see previous page*) immediately comes to mind. Each level is basically a maze, through which you navigate in search of the Saracen Chief hidden within. A storyline as thin as the paper it's printed on casts you in the role of Ilan the Crusader, who, like any other red-blooded young Christian adventurer, is on a sacred mission to search out and expel the Infidel Saracens from the Holy Land. Play opens with the gallant warrior trapped within a Saracen fortress, longbow in hand, ready for action.

Actually, it's not quite as heroic as this may sound. The screen is a maze, a rather complex and confusing maze, filled with a number of items to keep play interesting. There are arrows which you can pick up and shoot at the soldiers who dog your every move; one-way doors, bombs and cannonballs which slow your progress; and, of course, the Saracen Chief, who stands between you and the next level, and thus must be destroyed. Grab the grenade, place it next to the chief, and shoot it with an arrow, blowing him away. Then move on to the next level and do it again. The graphics are good enough, the speed fast enough that you'll want to attempt each of the 100 levels.

This simple formula is complicated by few options. You can choose to get involved via your joystick or the keyboard, and a pause key allows the obsessed a chance to eat. For the impatient, and perhaps inept challengers, you are permitted to play any of the levels independently—so you are guaranteed a visit to each dungeon of doom. And you'll probably want to drop in for at least a minute on each and every one.

The Eternal Dagger
by Paul Murray and Victor Penman
SS1
1046 N. Rengstorff Avenue
Mountain View, CA 94043
8K Disk $39.95

Alternate Reality: The Dungeon
by Ken Jordan and Dan Pinal
Datasoft
19808 Nordhoff Place
Chatsworth, CA 91311
48K Disk $39.95

After months of nothing, we fantasy lovers get not one, but two new universes. Well, not entirely new, but at least new passes to a pair of worlds that many of us have grown (or groan, depending on whether you were able to finish either of these) to love. Each is a sequel to a previously issued game, and each plays pretty much like its respective predecessor. And while there's nothing spectacular to distinguish either of these, they are nonetheless fine additions to the pool of adventure games on the market. We'll start with The Eternal Dagger.

The cover art of the second Wizard's Crown adventure depicts brave warriors stepping through a transport portal to battle against an evil wizard who awaits them, energy pouring forth from his hands. And while it is unlikely you will actually imagine yourself in this position while playing the game, if you take the time to boot up, you'll likely find yourself not disappointed.

First you form a group of up to eight brave souls of varying skills and strengths. If you've played Wizard's Crown, you can at this point transfer in your favorite characters. If not, you can use the eight stock characters, or create your own. Each has familiar D&D attributes, and each can have weapons, shields and armor. Depending on his particular profession, characters also have special powers, such as magic, healing and thieving skills.

Once you begin play, the screen fills with a map perspective, on which your party moves, searches, battles and dies.

Actually, it's not quite *as heroic as this may sound.*

The sophisticated program supports a mind-boggling level of detail and allows both quick and tactical battle modes. The former displays only the outcome of a given engagement, while the latter allows full control of the altercation, from character position to prayer (when things get really desperate). Graphics are of the standard SSI format, the many command menus and prompts easily readable.

Probably the very worst thing about this game is the agony the setup routines inflict upon the helpless owner. First, format four diskettes (assuming you can find four blank diskettes). These are then inserted into your drive in a 20-minute ordeal requiring you to precisely follow the prompts to avoid starting this hell all over again. I hope you've got the time and patience to sit still to do it.

Those who have played the first installment of Alternate Reality will no doubt immediately recognize the familiar portal adorning the cover of the packaging of Alternate Reality: The Dungeon. In fact, some might even find it hard to discern this game from the original. And, like Eternal Dagger, this similarity continues throughout play.

In Alternate Reality, the scenario is not much different. You have been captured from your dreary life by an alien ship, which blasts off to your Alternate Reality, where you are an adventurer trapped in a strange, primitive world. This time, though, you find yourself in a dungeon rather than a city. After jumping through the transit window, a portal above which spins numbered dials representing the values of your various attributes—the values freeze at entry to determine your skills. Or you can choose to import a character from Alternate Reality: The City. Either way, from then on it's a first-person perspective as you move through the dungeon.

A small window in the center of the screen is your eyes. At the top and bottom of the screen are readouts providing game status, displaying prompts and action menus. Each encounter requires some sort of reaction, such as fight, flee or greet. But while you'll never know exactly what hides behind any of the many doors you'll encounter in your exploration, you can be sure that it will take a lengthy disk swap and access to gain entry. Such is a drawback of 48K. It will take you some time to get through each of the three disks which contain the game, and be warned that aggression is not always the best course of action. The ultimate goal is, as you might expect, to return to Earth and/or obtain revenge against your captors.

Even though each game has its own set of superb documentation, I have to award this battle to the The Dungeon. Mind you, this is an aesthetic call, not one based on information content. This is because each manual was filled with everything you might want (or have) to know about the game. Each had full descriptions of creatures and locations, as well as tons of hints and strategies. And actually, Dagger's was a little more colorul. I think it's the map that gave it to Dungeon.

Overall, I think Alternate Reality: The Dungeon was a little better. What probably tipped the scales was the graphic orientation of Reality—I prefer the first-person perspective over a map-oriented game. The Eternal Dagger was much more complex, and hence more difficult to play, but in the end simpler to complete. This is because Alternate Reality is brutal and unforgiving. One mistake, one underestimation of his motives, and you're dead. The hardest lesson you'll learn is the loss of two hours of progress because you were too lazy (or too weary) to save your position. But if you avoid this disaster, both games will provide hours of great adverturing.

When next we meet, I'll move on to simulations, including the latest civil war epic from Strategic Simulations. And stay tuned for a rundown of fantasy games as well. All things considered, it's going to be a great year for Atari gamers, so don't get left out. For now, I'm going back to Saracen. You see, there's that cannonball haunting me on Level 56. Or was it 65? I'll take a look and let you know.

Review

▽ ▽ ▽

The Newsroom
Springboard Software, Inc.
7808 Creedridge Circle
Minneapolis, Minnesota 55435
48K
Disk $49.94
Reviewed by Clayton Walnum

The Newsroom is, I believe, Springboard Software's first entry into the Atari 8-bit software market (at least, there's nothing else listed for the 8-bits in their catalog, and I don't recall seeing anything in the past), and if this product is typical of the rest of their software, I have only one request for all of you: Support this company! This program is a top-notch effort, from the software right through to the manual, and it would be terrific to see more Atari-compatible products from these people.

What is The Newsroom? Basically, it's a stripped-down desktop publishing system that allows you to create newsletters, brochures, forms and other simple publications. It doesn't have anywhere near the power of such desktop publishing programs as Publishing Partner for the ST, but that's not to its detriment. In fact, its simplicity is actually a good part of its charm. This program is almost as easy to use as Broderbund's famous Print Shop. You'll find yourself printing out your first newsletter after only a couple of hours with The Newsroom (and most of that time will be spent designing the newsletter rather than anxiously scrambling through the manual).

During the creation of your publication you'll visit each of The Newsroom's five "departments": Banner, Photo Lab, Copy Desk, Layout and Press. Access to these departments is attained through the use of a graphic menu where you use either the joystick or the keyboard to choose which department to visit. (All sections of the program can be driven from both the joystick or the keyboard.) In this way dividing the different functions necessary to create your publication makes the entire process about as transparent to the user as possible.

One reason The Newsroom is so easy to use is that the structure of each page is required to fit into one of two categories: one made up of a "banner" and six "panels," or one made up of just eight panels. (On 14-inch paper you can get an extra two panels into each of the layouts.) A banner is your newsletter's masthead or header, and the panels are the partitions into which each page is separated. If you took a sheet of paper, drew a line lengthwise down the middle, then drew three horizontal lines to divide the paper into six equal-sized rectangles, you'd see the way The Newsroom lays out its panels.

The first step in putting together your publication is a visit to the Banner department. The banner can be created using clip art (over 600 individual graphics are included with the package; also extra clip art packages may be purchased, allowing The Newsroom user to add 2,000 more graphics to his library), text (with a choice of several fonts and sizes) and a "Graphics Tool" section that allows you to do everything from drawing simple lines to laying down circles and squares and filling shapes with one of the ten available fill patterns. Though the banner is restricted to a preset size, you have all the tools necessary to create just about any graphics you want.

Once the banner is complete, you'll want to start putting together each of your panels. A panel is usually made up of a "photo" and some text, so you'll probably want to visit both the Photo Lab (if you want a graphic) and the Copy Desk for each of the panels of your page.

The Photo Lab allows you to put together a photo (the name The Newsroom uses for a rectangle containing a graphic and some text, the text usually used as a caption). A photo is actually very similar to a banner—only the size and shape are different. As a matter of fact, the Photo Lab offers the same functions—clip art, graphics' tools and text—as the banner department, and the process for creating a photo is virtually identical, the only real difference being the addition of a "camera" function that allows you to define the area of the screen that will become your photo. Photos can be any size equal to or smaller than the size of a panel.

When your photo is complete, it's time to move to the Copy Desk, where you will place the photo in the panel and enter the panel's text. The text editor supplied is actually a simple word processor that even allows some block functions, such as deleting or moving blocks of text. You may enter your text in two different character sizes and choose from three different fonts. The fonts included are the same as used in the Banner department: serif, sans serif and old English. Using the large character size lets you enter headlines, while the smaller text sizes are used for the body of the text.

As you enter your text, it automatically "flows" around the photo, relieving you of the agonizing chore of formatting the text to fit the remaining space. Amazingly enough, you can actually move the photo after the text has been entered, and the text will refit itself around the photo's new position.

To finish your newsletter, you'll need to create at least six panels as described above (eight if you've not used a banner) and get them all saved to disk. Then you need to pop into the Layout department to tell The Newsroom in what positions you want the panels placed.

Finally, it's off to the Press to print out your creation. The only thing you really need to do here is make sure the program is set for your printer. Since over 50 printers are supported, the chances are good that something on this list will work for you.

One word of warning: The Newsroom doesn't seem to be compatible with the Atari 850 interface, although from talking to Springboard Software's representatives, I get the impression that they're planning to correct this oversight. So make sure that The Newsroom is compatible with your printer and interface before you

buy. This is doubly important, since nowhere on the box or in the documentation does it tell you what printers are actually supported.

And speaking of the documentation, I have to say that The Newsroom's manual is one of the best I've ever read. It's laid out in a logical and readable manner and is well written. It includes not only a complete instruction and reference section, but a full tutorial that'll lead you through the designing of a newsletter: from the creation of the banner to the actual printing. It's a rare treat indeed to come across a truly professional piece of documentation these days.

Though $49.95 is a fairly high price for a piece of 8-bit software, I think that those who take the plunge will be gratified by their purchase. The Newsroom is a real class act and will be a welcome addition to most anyone's software library. Though it's not appropriate for serious desktop publishing, it's a program that the entire family will enjoy and come back to again and again. I sincerely hope that we'll be seeing more of Springboard Software in the future.

The author would like to thank 20th Century Video in South Windsor, Connecticut, for supplying some of the hardware needed to evaluate this product.

Review

221 B. Baker St.
by Sculptured Software, Inc.
Datasoft
19808 Nordhoff Place
Chatsworth, CA 91311
Low resolution $39.95
Reviewed by Steve Panak

One of the favorite forms of entertainment is the mystery. Whether it be an Agatha Christie whodunit or a riotous Pink Panther comedy, the written word or the silver screen, people just can't seem to get enough of it. Unfortunately for mystery and computer lovers, software has, for the most part, failed to enrich their lives with death and despair. Until now.

221 B. Baker Street pits software sleuths against one another and themselves in a race to solve a murder using the least number of clues. After booting up the autoloading disk, up to four players first choose a game persona. You could be Holmes or Watson, Irene Adler or Inspector Lestrade. You then decide whether the program will refer to you by your own name or that of your character, and whether you wish clues to be given in code. If you choose coded clues, the hints are scrambled so the other players can't use them, and won't have to avert their eyes from the screen as your clues are displayed. This feature can also be used to add another dimension to the game, that of cryptography—trying to decipher the other players' clues to gain an advantage. The option to change your code midstream makes decryption all the more difficult. After setting all the preliminary options, you select one of the 30 cases and start pounding the streets for clues.

The highly detailed screen, depicting streets and buildings, scrolls smoothly, displaying only a portion of the city at any one time. As each player begins his turn, the image of his character fills the bottom of the screen, along with an inventory of the items he possesses. Pressing the space bar to roll the die, you move your man up to the maximum allowed spaces. One complaint I had here was the fact that, while you can retrace your steps if you make a wrong turn, you cannot do so if you've moved your total allotment of spaces. This was occasionally annoying and could have been easily remedied.

After a couple of rolls, you'll reach one of the many buildings in the town, where you'll receive a clue. This pattern is repeated until you amass what you consider to be enough clues, whereupon you hightail it back to Baker Street to give the solution to the crime. You solve the crime by answering a series of multiple-choice questions, and then you receive an explanation of the motive behind the murder. The first to solve the crime wins, and a ranking is assigned depending on the number of clues it took to reach the solution.

To spice up play, the board is covered with a number of special items and buildings. Secret tunnels and a carriage service speed you about town. You must get to Scotland Yard and receive a badge before you are allowed to offer your solution. These badges can also be used to lock a location, thwarting the access of other players to valuable clues. Keys are required to unlock locations, and can be obtained at the locksmith's (naturally).

Documentation is quite extensive. A manual contains full instruction on the operation of the program, as well as playing tips and tables to decipher the codes. These keys are placed in the center of the book for easy copying. A separate casebook contains short scenarios setting up each of the 30 cases, which must be read before attempting to solve each crime. Fortunately, all the written materials are intelligently executed and engaging, and the manual and the separate machine specific reference card are spiced up with quotes from the works of Sherlock Holmes. A pad of worksheets, on which players may make notes on their journeys, is thoughtfully provided.

The program itself is enjoyable to play, although it truly shines when played by more than one person. It most closely resembles, as one might guess, the board game Clue, except that the solution of these crimes requires a different style of deduction. However, for adults at least, the solutions should be no problem, as the clues leading you to the murderer, the motive and the weapon are quite simple. For example, the clue indicating that Lord Longsworth was the killer was the phrase "the opposite of short." Still, the chase throughout the town is good fun, and some bit of strategy is required, to assure efficient use of moves. The 30 cases keep the game interesting for some time, although, as you might expect, the basic framework of each game is identical. For those that really love the game, additional data disks are promised.

Overall, I found 221 B. Baker Street to be fun to play, but not very challenging. It is probably best suited for younger players, probably preteens, and is best played in a group setting rather than alone. Still, it should come as no mystery that budding armchair detectives will love it.

Shuffleboard
by George Breen
Shelbourne Software Systems, Inc.
7221 Rising Sun Ave., Suite 191
Philadelphia, PA 19111
ST Disk $29.95
Reviewed by Steve Panak

The vast number of activities that programmers attempt to force their computers to simulate sometimes amazes even me. We have war simulations, sport simulations, flight and driving simulations. And usually, something is lost in the translation. Especially as far as control of the game goes, these computers just can't cut it. For example, how do you simulate a golf swing? Most of the programs currently available require you to tap a mouse button, or the space bar, to start your swing and again to end it. Using this adequate, although crude approximation, golf becomes a game of timing. But when they tried to simulate gymnastics—well, I wasn't impressed.

Although ST Shuffleboard continues this trend, it stands apart from the other simulations due mainly to the manner in which you control the sliding of your weight. I should change that to the way you slide your weight. I say this because this program more closely simulates the original product than any game I've ever seen. It's unfortunate that shuffleboard is usually considered a boring game for the elderly. But this is not the shipdeck shuffleboard, in which you use a stick to propel your weights down the board, but the table variety, in which you slide your weights with your hands. This latter variation is often popular on college campuses.

The screen display offers two simultaneous points of view. The majority of the monitor screen is occupied by a three-dimensional representation of the table as it stretches out in front of you. Suspended above it are two scoring tallies and an indicator showing who is up. The right border of the screen holds an overhead view of the scoring portion of the playfield. Your weights move in both of these displays simultaneously, appearing on the right display as your weight reaches the end of the board. And, like its real-life counterpart, this game has a number of options and variations.

You can choose to play on one of two tables. The longboard contains three scoring areas at its far end, while the cushion board contains the familiar triangular scoring area and allows banking off the sides. The longboard supports three game variations, the cushion table two. Of the five games possible, most will find one to suit his tastes, although I would have liked to have seen more control over variables, such as being allowed to create a longboard table with cushions. Such a design would allow the user much more flexibility.

You can choose to play against a human or computer opponent. You may also allow the computer to play against itself. The human player may customize the relationship between the speed of the mouse and the speed of the weight, as well as set the computer opponent's skill anywhere from moron to professional. Table options include the positioning of the foul line, the amount of cornstarch on the playfield (which determines the friction, and hence the speed, of the weight), and the placement of the scoring areas on the longboard model. You can also choose to play to one of four possible winning scores, but, in another show of inflexibility, you cannot choose your own target score.

When it comes time to slide the weight, the game truly shines. Holding down the left mouse button, using the mouse you move the weight around on the table—left, right, back and forth. Push the mouse forward and release the button to send the weight sliding down the table. The ALT key locks the horizontal motion of the weight, making aiming easier, while CONTROL and LEFT SHIFT put left and right spin on the weight. One note here is that due to the typical mouse placement—that is, to the right of the keyboard—lefties might have trouble playing this game.

The manual is nearly unnecessary. Good program design allows the newcomer to quickly learn the game without any documentation. When you use it, you will find it simple to understand, with numerous screen-dump illustrations. In addition, there is a booklet (very dated, circa 1950s) put out by the American Shuffleboard Company fully describing the variations and rules of shuffleboard along with a scoring sheet. Given all the considerations, the scales tip slightly in favor of this one. The price is reasonable, and the only real flaw is the program's inflexibility. Shuffle on down to your dealer and get ST Shuffleboard. You won't be disappointed.

Review

▽ ▽ ▽

Bridge 5.0
by Arthur M. Walsh
Artworx Software Company, Inc.
1844 Penfield Road
Penfield, NY 14526
Medium Resolution $29.95
Reviewed by Steve Panak

Most of my favorite games are thinking games: Chess, Go and any of a score of others that require some thought and strategy to succeed. Don't get me wrong.I like the wrist-busters too. It's just that thinking games really fascinate me, probably due to the fact that they outperform pathetic humans at their own games. And while it may not be the greatest player in the world, Bridge 5.0 proves to be more than a match for most of us.

Artworx has completely refurbished their bridge simulation, supplying it in compiled BASIC to speed up the program execution, as well as incorporating numerous suggestions from bridge experts. The resultant game, while still possessing a few rough edges, offers newcomers and pros alike a challenging game. Play follows the normal pattern of bridge, allowing you to first bid, then play a given hand. Just as the documentation refers you to other works to learn the rules of this complex game, so will I refer you to others for instruction. Probably the best book on the game is *Goren's Bridge Complete*. Another option might be *Artworx's Compubridge* tutorial, which allows interaction with a computer teacher. Either will get the novice started on the right track.

Upon booting the program you are presented with a list of options. Click on "PLAY" to set up a randomly generated hand. Or you can choose to be dealt a hand with at least 13, 17 or 22 points. These would be considered to be stronger hands. More advanced players might choose to play a previously saved hand, or to set up the cards into four hands of their own design. This final option allows you to study the bridge problems which so often appear in major newspapers.

Using the mouse, you first select your bid, then play your hand. Tossing in a card is as easy as clicking it on with the left button, while the right one brings up a mini menu with options terminating the hand. After each hand you can choose to display all the cards, and you may play the hand over if you so desire. The screen displays are just what you would expect from an ST in medium resolution. The playing cards are easily discernible, even when all 52 cover the screen. And the playing field is designed nicely, with the only distractions from the cards being unobtrusive selection grids. A status line at the top gives the current tricks awarded to each team, as well as the bid and the next hand to be played. Still, there were a couple of problems, mostly in program execution.

When you set up a hand, the program wastes your time by making you click on each card to go into the final hand, rather than simply using these remaining 13 cards. When displaying the hands, the trump suit is not placed on the left, as is customary. Also annoying is a small "hot spot" on the cards which often forces you to click all about on a card in order to select it. Finally, a bug exists in the version I tested which caused the program to renege, playing trump improperly in certain instances. This latter flaw has since been corrected (in program files dated later than August '87), and updates are available from Artworx at no cost.

As for how well the program plays the game, it can keep up with most of the population, although experts might find it a little lacking. Its algorithms underbid the hands, ensuring that the bid is usually made. It follows the Standard American (nearly identical to Goren) bidding system and, if it should mean anything to you, it follows the Blackwood and Stayman conventions. It plays its cards rapidly, and is very addictive, especially if you enjoy the game. Documentation is slight: a single, folded sheet of thick paper holding the three pages of instructions, which are applicable to seven different machines. It takes careful reading to determine just what has to be done to get the ST version to work.

Overall, despite its rough edges, I must recommend Bridge 5.0 to anyone interested in playing and studying this complex game. While it could have offered a little more in the way of options, and been a little more attentive to ease of use, its price/performance ratio still makes it a very good buy.

Attention Programmers!

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

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

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

End User

by Arthur Leyenberger

It's been about a year since Atari last showed the elusive Atari PC Clone at the summer Consumer Electronics Show in Chicago in June 1987. I have had some pretty strong opinions about that product since it first debuted and continue to feel strongly about it.

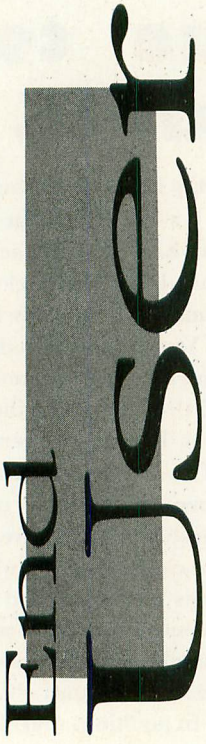
The Atari PC was first shown at the January 1987 CES and billed as the first PC clone to have EGA (Enhanced Graphics Adapter—an IBM PC graphics standard) built in. For $699, you would get a fast processor (faster than the standard IBM PC anyway), standard serial and parallel ports, monochrome screen and no slots. The specs were still the same in June but the delivery date was pushed back to the fall.

I doubted at the time that the Atari PC would ever hit the streets. For one thing, the PC clone market has become a commodity market where price is the most important aspect. For another thing, there are already several big players involved in the clone game such as Tandy, PC's Limited, Epson, Leading Edge and a host of others. In fact, even Hyundai, the Korean manufacturer of inexpensive cars, is selling PC clone computers in the United States under the name Blue Chip.

You can go out today and buy the same box that Atari promised to sell in the fall of 1987 for about the same price or even less. The big difference, aside from many of the other companies already having a corporate-familiar name, is that all of these machines have slots for adding extra circuit boards, whereas the Atari PC does not. Extended memory, enhanced memory, hard-disk cards, more serial and parallel ports, clocks, etc., cannot be added to the Atari PC.

In addition, Atari faces an uphill battle

It's been 30 years, give or take a few, since I last saw a copy of this magazine, and I was surprised to read about articles and see projects that I could have sworn I read about as a youth.



to get corporate purchasing agents or DP managers to buy a PC that has the Atari name on it. It seems unlikely that the corporate purchaser of computers is going to even consider a non-standard computer (or hundreds of computers) from a game company, let alone buying it from a toy store. Although the Atari PC was shown several times publicly, it seems almost certain now that it is a doomed product.

At the time, I asked Atari's new marketing wiz, Jerry Brown (no, not that Jerry Brown), about the lack of slots on the Atari PC. He emphasized that it was the only PC clone that had a built-in EGA, in addition to its CGA and monochrome graphics output. I noted that it would take another $300 to add a color monitor, bringing the price of a color Atari PC up to $1000, and suggested that, to the purchasers of low-end PC clones, slots may be more important than EGA. Mr. Brown responded with something like, "So they won't buy our machine."

Considering the facts, it looks like the Atari PC is a non-product and Atari's marketing attitude towards it and their potential buyers is a non-attitude as well. Atari's emphasis on their video-game products in the last six months further dooms the Atari PC. I guess this is just one more episode in the never-ending story of future Atari products that never were.

Some ST Info

Not long ago I found myself sitting in a doctor's office waiting for my appointment. As I waited for my turn in the overbooked queue, I began to get bored, so I looked around for something to read. I spyed a copy of *Highlights for Children* and picked it up to look through it. It's been 30 years, give or a take a few, since I last saw a copy of this magazine and I was surprised to read about articles and see projects that I could have sworn I read about as a youth.

Then it dawned on me. Of course they were the same or similar projects. The kids reading the magazine today were obviously not around three decades ago and therefore to them, this stuff was new. The same is true for readers of ANALOG. Newcomers to computers as well as to the magazine are reading about the ST for the first time. Because of this, I'd like to mention a few things that would have helped me when I was a first-time ST user. If Heloise used an ST, this month's column would be called "ST Hints From Heloise."

One of the first potentially confusing aspects of using the ST are the various program types available for the machine. There are four types of programs on the ST and it is useful to have a brief understanding of each. Each of the programs has a different name extension (a maximum of three letters after the period in the program name).

A GEM (Graphics Environment Manager) application program uses the GEM interface (windows, drop-down menus, dialog boxes, etc.) and both enters and exits from the GEM Desktop. It has a ".PRG" at the end of the program name. A non-GEM program is one that does not necessarily use the GEM interface or built-in GEM functions. They may use the GEM routines but always provide their own user interface. Their extension is ".TOS" (The Operating System).

A special type of "TOS" program requires one or several arguments or additional pieces of information that are supplied when the program is run. When these programs are run from the desktop, a dialog box appears to let you enter the list of arguments. After the argument(s) is entered, you press return and the program runs. There are several "command processors" available for the ST and these programs allow you to enter commands much like you do in MS-DOS or CP/M, that is directly from the keyboard. If you were using a command processor, you would run this type of program by typing its name followed by the list of arguments. Programs that use a list of arguments have a ".TTP" name extension which stands for TOS Takes Parameters.

There is a final type of program that can be run on the ST which is slightly different then the ones mentioned above. This program type is called a "Desktop Accessory" because once run, it is always available to you much like a stapler, pencil holder or calculator is—to use the desktop metaphor. When a Desktop Accessory is run it is loaded into memory and takes up a portion of your ST's random access memory (RAM). The accessory, which is typically a small program,

is available from any GEM application program from the "Desk" drop-down menu. This is one of the many built-in features of GEM. Desktop Accessory programs have a ".ACC" name extension and have to be programmed to specifically be an accessory. Any other program will not function as an accessory, even if you were to change the extension to ".ACC."

There are a number of ways for you to get the most out of using the GEM Desktop. One of the simplest tricks is to rename the disk icons (small pictures on the Desktop). For example, if you have two disk drives, stacked one above the other (on your desk), it may be easier for you to refer to these drives as the "top disk" and "bottom disk." To do this, click once on the drive icon and then choose "install disk drive" from the "Option" menu. Type in the new name in the name field and click on "install." That's all there is to it and your new name will remain in effect until the computer is turned off. To save the name permanently, you will have to save the Desktop (see below).

Another method you can use to get the most out of using the Desktop is to first organize it the way you want, and then save the Desktop so that each and every time you use your ST, the Desktop will look just as you left it. What will be saved with the Desktop? Icon names and positions, screen resolution, number of displayed windows, their size and position. From the "Option" menu there is a choice labeled "Save Desktop." Clicking on this option creates (or overwrites a previous) a file called DESKTOP.INF. Whenever the ST is first turned on, it checks to see if this file is present and if it is, loads the Desktop exactly as you had saved it.

If for some reason your mouse is incapacitated, missing, on strike or otherwise unavailable, you can still maneuver around the desktop via the ST keyboard. This information is buried in the ST user manual but it is really quite straightforward. To move the screen cursor around you hold down the Alternate key and press any of the four arrow keys for direction. If you want a finer movement of the cursor, hold down both the Alternate and Shift keys and press any of the four arrow keys. To give a left mouse button click, press the Alternate and Insert keys. The Alternate and Clr/Home keys pressed together act like a right button click. After a little practice it begins to feel natural although not as much fun as driving the little furry guy around your desk.

Unlike some computers such as the Macintosh which not only keep track of what disks are in the drives but also control when they can be removed, the ST allows you to insert and remove a disk at any time. However, if you have an open window on a particular disk drive, and then replace the original disk with another, the screen still displays the contents of the original disk. One way of updating the displayed directory is to close the window and then open it again. That's cumbersome, time consuming and no fun. A better way is to simply use the Escape (Esc) key on the keyboard. Pressing Escape causes the ST to update the contents of the currently open window. By the way, the escape key can also be used to erase text fields in GEM dialog boxes. For example, to enter a new time in the control panel, press Escape to wipe out the field and move the cursor to the beginning and then type the new time.

When I first started using Unix many years ago, there was one concept that I didn't fully understand or appreciate. It wasn't until I started using the system on a regular basis that I began to realize the importance of folders (called directories in the MS-DOS and Unix world). The best way I can now explain their use is to ask you to imagine many files of different types. For example, some files are text files used with your word processor, other files are used with a spreadsheet, other files are DEGAS graphic files, and on and on. It doesn't take more than a screenful or two of files to make the task of finding any one specific file difficult. Here is where folders become important.

Instead of having to look high and low for a particular file in one directory listing or window, folders let you categorize your files for easier access as well as potentially faster operation. To create a new folder, select the File Menu on the desktop and provide a name when the dialog box appears. Remember that you cannot rename the folder later on so choose names that describe its intended purpose. For example, I have such folders as "words," "graphics" and "games." You can even have folders inside of folders. Within "words," I have a folder called "1st_Word," "ST_Writer" and "Regent" to hold the programs and files of three different word processors. Note that there can be no blank spaces in a folder name so you need to use an underscore character.

If you want to see the contents of a folder just click twice on the folder name or icon. Files can be copied to a folder name or icon from another window so there is no need to open the folder first. Finally, when a window is open showing the contents of a specific folder, the PATH or folder order is displayed at the top of the window and the number of bytes for that folder only is also displayed.

The last tip allows you to run your application programs a little faster. If you wanted to use your word processor, you would double click on the program and then from within the program select an existing text file to work on. By installing an application with its document type you can simply double click on the file you want to use and the program associated with it will automatically run. Here's how to do it:

First of all, you need to be consistent with the name extension of your similar files. In this example I am using 1st Word so my name extensions are ".DOC." From the Desktop, click once on the application name, 1st Word. Then go to the Option Menu and select "Install Application." When the dialog box appears, type the three letters associated with the application—"DOC" in this case. Then click on "OK" and save the Desktop to make your selections permanent. From now on, all you need to do is double click on any file with a ".DOC" extension name and 1st Word will automatically run and load the document file you selected. Be sure that the application program and its associated files all reside on the same disk for this technique to work.

Knowing how to use your ST computer more effectively means that you will get the most out of computing. And getting the most out of computing is something we are all interested in.

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

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

SAVE TIME AND MONEY SUBSCRIBE TO ANALOG

SAVE $14 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ANALOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS. GET THE MOST OUT OF YOUR COMPUTER.

SUBSCRIBE TO ANALOG TODAY

☐ 1 YEAR @ $28 — SAVE $14! MCGYY
FOREIGN — ADD $7 PER YEAR

☐ 1 YEAR WITH DISK @ $105 DCGYY
FOREIGN — ADD $15 PER YEAR

☐ PAYMENT ENCLOSED ☐ BILL ME

CHARGE MY: ☐ VISA ☐ MC #_____

EXPIRATION DATE SIGNATURE

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16927, N. Hollywood, CA 91615. Offer expires August 31, 1988. Your first issue will arrive in 6 TO 8 weeks.

WATCH FOR IT!

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

SAVE TIME AND MONEY SUBSCRIBE TO ST-LOG

SAVE $14 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ST-LOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS! GET THE MOST OUT OF YOUR COMPUTER

SUBSCRIBE TO ST-LOG TODAY!

☐ 1 YEAR @ $28 — SAVE $14! MCGYW
FOREIGN — ADD $7 PER YEAR

☐ 1 YEAR WITH DISK @ $105 DCGYW
FOREIGN — ADD $15 PER YEAR

☐ PAYMENT ENCLOSED ☐ BILL ME

CHARGE MY: ☐ VISA ☐ MC #_____

EXPIRATION DATE SIGNATURE

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME

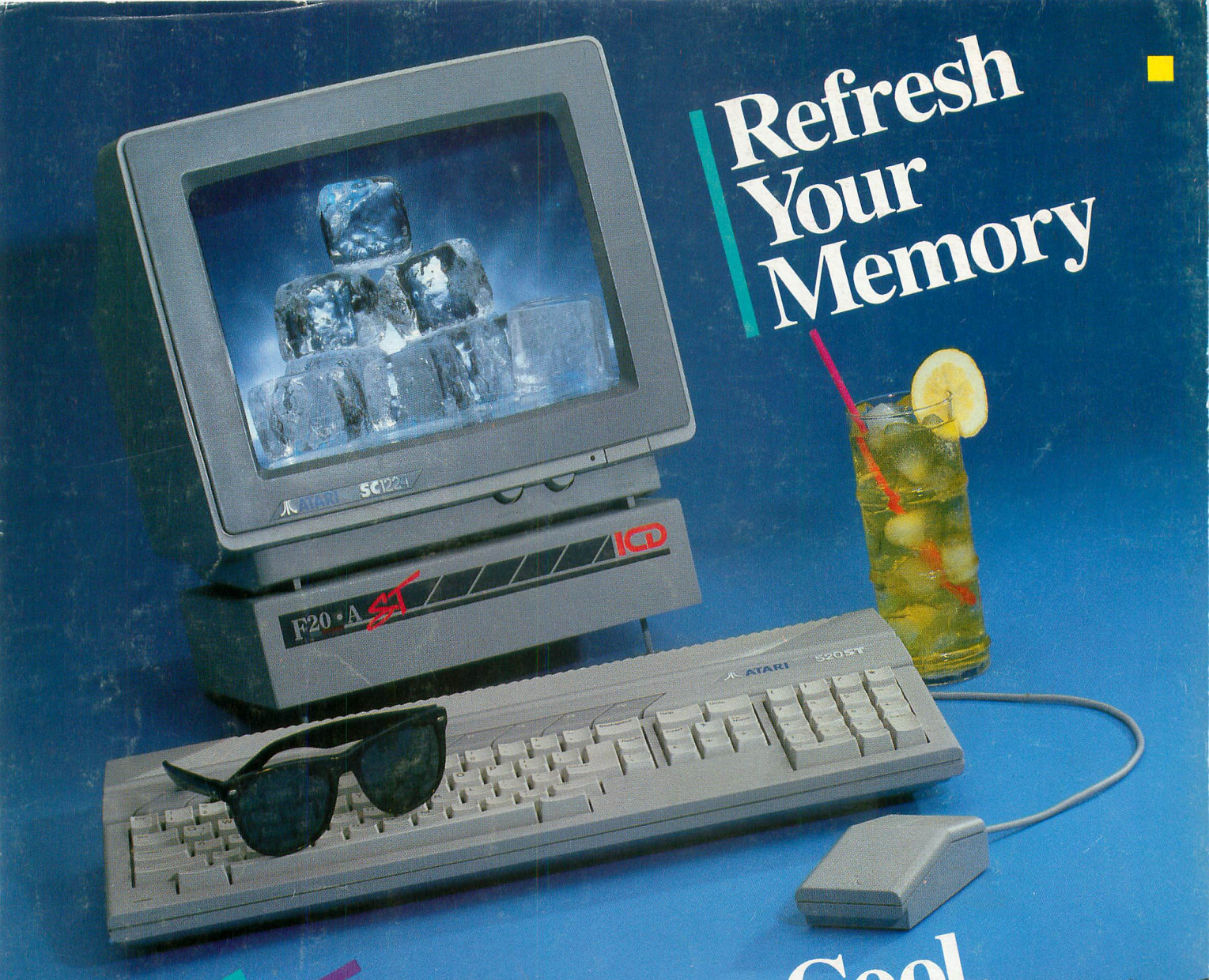
ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires August 31, 1988. Your first issue will arrive in 6 to 8 weeks.

WATCH FOR IT!

Refresh Your Memory

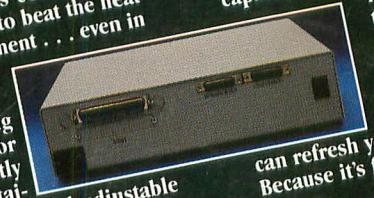


And Keep Your Cool.

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.

1220 Rock Street
Rockford, IL 61101-1437
(815)968-2228
MODEM: (815)968-2229
FAX: (815)968-6888