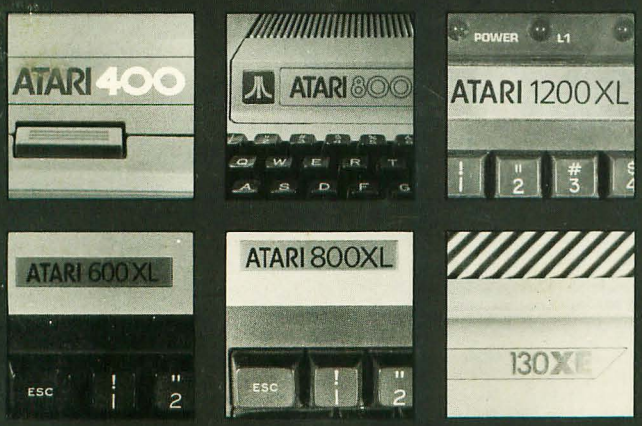
An Atari 8-bit Extra

from

ANALOG COMPUTING

1987
$8.95



THE ATARI WORKFORCE

PAPERCLIP

"The # 1 Best Selling Word Processing Package"
– BILLBOARD'S computer software chart

"... by far the best word processor ever available for the Atari" – ANTIC

- Editing features include Block Move, Copy and Delete, Global Search and Replace
- Enter repetitive words, sentences, or paragraphs instantly with Macro Command
- Edit two files simultaneously and transfer text between documents using Dual Text Windows
- Automatic Page Numbering, Table of Contents, Headers and Footers
- Editing screen extends up to 130 columns wide and scrolls in any direction
- Print Preview displays formatted text exactly as it will be printed
- Automatically saves files as you write

NEW! SPELL PACK FOR THE 130XE WITH A 36,000 WORD DICTIONARY WITH ON SCREEN WORD SEARCH.

B/GRAPH

"Graph-generating and statistical analysis ... we recommend B/Graph! – INFOWORLD

- graph up to three factors with 100 data points each
- choose pie charts, line and area graphs, 2 and 3 dimensional bar charts and more
- convert instantly between graph types without re-entering data
- full screen editor, multiple grid and graph scaling, automatic labelling, overlays, "slide show" capability
- statistical analysis functions include standard deviation, variance, Chi-square, regressions, plotting and many more
- reads and writes to VisiCalc DIF – use VisiCalc files with B/Graph and vice-versa
- compatibile with most popular printers, printer cards, interfaces

HOMEPAK

"... inexpensive, powerful, integrated software. As such, HomePak is the winner of InfoWorld's Best Buy Award."
– INFOWORLD MAGAZINE

"...quite simply, the best ... the highest rating possible." – ANALOG COMPUTING

Three easy-to-use programs on one disk:

1. HOMETERM TELECOMMUNICATIONS

- Puts you in touch with bulletin boards, public databases and on-line services
- Powerful user-defined Macro facility – log on to your favorite service or bulletin board with just one command
- Store up to 10 macros per document
- X-Modem protocol, the virtual on-line standard
- Download files of any virtually any length
- Flexible data handling – save incoming text to disk, edit it, print it

2. HOMETEXT WORD-PROCESSOR

- Over 20 full-screen editing and formatting features: move & copy, word-wrap, justification, automatic paging and many more
- What You See Is What You Get (WYSIWYG) – screen is an exact representation of the printed page
- Supports most major printer functions including boldface, underlining and extended characters

3. HOMEFIND DATABASE MANAGER

- Natural English-language data entry/retrieval system for simplified electronic filing
- Includes the key search/sort functions, flexible queries, easy output commands and sophisticated Report Composer

All three HomePak programs reside in memory together – it's easy to transfer data between them and perform integrated tasks.

AND FOR THE ATARI ST.

AND COMING SOON: PAPERCLIP ELITE, CONSULTANT, B/ GRAPH ELITE, BTS, THE SPREADSHEET, AND MORE!

BATTERIES INCLUDED

BATTERIES INCLUDED, an ITM company, 30 Mural Street, Richmond Hill, Ontario, Canada, L4B 1B5 (416)881-9941, Customer Information (416)881-9816. If you can't find this product at your local retailer, you may order it direct from us at the full suggested list price plus $5.00 for postage and handling. For product orders please call 1-800-387-5707 (U.S. only). For most Batteries Included products you can always have the latest version of your program by returning the original disk and $10.00. Write to us for our full color catalog of products for the APPLE, APPLE MACINTOSH, ATARI, ATARI ST, COMMODORE, COMMODORE AMIGA, AND IBM SYSTEMS.

Some features may vary with computer system used.

*AS COMPILED FROM NATIONAL RETAIL SALES REPORTS FOR WEEK ENDING JANUARY 5, 1985

Integral Solutions

An Atari 8-bit Extra

from

The publishers of

1987

ANALOG Computing Worcester, Massachusetts

This volume, from the publishers of **ANALOG Computing**, is dedicated to 8-bit Atari users everywhere, and to the readers who have contributed so much to our success and to the Atari Adventure.

Printed in the United States of America.
ISBN #0-914177-01-X

CONTENTS

M/L Editor

For use in machine language entry

by Clayton Walnum

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Listing 1.
BASIC listing.

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

4

APPLICATIONS

Hi-Score Display

Finally, an easy way to keep track of your best scores.

by Kevin Peck

I play a lot of games on my Atari. I used to keep a sheet of paper at the computer desk, to jot down my high scores. When things got crowded, I would write the current high scores on a new paper and keep that one, until it too became nearly impossible to read. Not anymore. I stopped playing games long enough to write a custom database for my high scores, one that will print out a clean list any time I wish. It's a lot easier to read now.

To use **Hi-Score Display**, you'll need to type in Listing 1, then check it with the **BASIC Editor II** (see **ANALOG Computing** issue 47). Listing 1 will create four strings containing the machine language routines used in the program. Save the program to disk before running it, because it will erase itself from memory, leaving the newly created lines. These will be the only lines in memory. Enter **BASIC Editor II** into memory and type in Listing 2. After you've finished, you'll have the complete **Hi-Score** program. Save it to disk at this time.

You'll need nineteen free sectors in single density, or ten in double density, to run the program. These sectors are necessary for the actual game data. The size of the data file, GAME.DAT, will never change. It's set up to hold a maximum of forty-two games, with three scores per game. You'll never have to worry about booting the program without enough disk space to add new scores.

When you first run the program, it will create the blank file GAME.DAT on the disk. This will take a few moments, and will only occur the first time you run the program. After the data file is created, the main menu will appear.

Next to the words *OPEN* and *USED* are two numbers. The number beside *OPEN* will be *42*, and the number next to *USED* will be *0*. This means that no scores have been entered; all forty-two are unused. The number next to *OPEN* will decrease as you add games to the list; the number next to *USED* will increase. These two numbers added together will always equal *42*, the maximum number of games per disk that the program can handle.

At this point, you're presented with six options. Right now, we have no games in our list, so we need to add some. We press the *1* key, for "Add New Games and Scores."

After pressing *1*, you'll see the score entry screen. You'll be asked to enter the program name, which may be up to fourteen characters long. If you accidentally pressed *1* while in the main menu, and you really don't want to add any games to the list, then press RETURN to get back to the main menu.

Back to adding games and scores. . .enter the program name, then press RETURN. You may use any characters you want, but the name must fit between the two arrows above your typing area. If you try to type beyond the fourteen-character limit, the program will ignore all extras. Of course, the name can be less than fourteen characters.

After typing the program name, you'll be asked for the score. You're allowed three scores per game, and you may enter them in any order. The program will sort them after you've entered all three. Scores may be up to six digits long, which allows for scores in the hundred-thousands. I know of few games that go into millions of points, so this should be more than adequate. The program will allow no more than six numbers for this entry, ignoring non-

numeric characters and commas, which it places automatically.

The next data item is the game level at which you obtained the score. Not all games have levels, so you may just press RETURN to leave this field blank. You're allowed two characters for the level, and only numbers are allowed.

Next, you'll be asked to enter the name of the person who attained this score. The name is limited to five characters, but they may be whatever you wish. Five characters allows for two initials, the ampersand (&) and two more initials, for those times when two players cooperated to get the score (five also happens to be the length of my first name). You don't need to enter all five characters. You may want to stick to three, as most arcade games do.

You must enter at least one score per game. **Hi-Score** will then prompt you to enter the second score. If you don't wish to enter a second, press RETURN. If you do enter the second score, you'll be prompted for the third.

Now that you've entered all the data for this game, you have three choices. You may press the *O* key if all information is okay. You may press *A* to abort this game. When you hit *A*, you'll be asked if you're sure you want to abort. If you press *Y*, this game won't be added to the list, and you'll be asked if you have other games to add.

If you decide some of the information is incorrect, press the *C* key, to correct the errors. The numbers *1-4* will appear on the right edge of the screen, indicating various pieces of information. If you don't wish to correct any of the information, press the *0* key; otherwise, press the number *(1-4)* that corresponds to the area you want to correct. If you do correct one of the scores, you'll have to enter all three pieces of data for that score. Press the *0* when finished, and the scores will be sorted.

After pressing either the *A* or the *O*, you'll be asked if you have more entries. If you do, press the *Y*, and the screen will clear for your next entry. If you're done entering games, press the *N* key. All games will be sorted by game name, then saved to disk. When this is done, you'll return to the main menu.

Now that we have some games in our data file, we can explore some of the other options on the main menu. Let's go through the rest of the options, in the order they appear on the screen.

Option 2 allows you to update the scores of any games on file. After pressing *2*, you'll be presented with the game selection screen. All games currently on file will be listed to the screen. If you have more than twenty-one games, they'll appear in two columns. Valid keystrokes are shown at the bottom of the screen. Press the *X* key if you wish to return to the main menu without making any changes.

To select a game to update, move the arrow to the proper name by pressing the arrow keys *without* holding down the CTRL key.

When you've selected the game you wish to update, press RETURN. The game selected will appear at the bottom of the screen, and you'll be asked to verify your choice. Remember, you may press *X* at any time during the selection process, to abort the operation and return to the main menu.

Once you've verified your choice, a new screen will appear, showing the game's current scores. You'll be asked for the new score. If you decide not to update the scores after all, press RETURN. The new score doesn't have to be a new high score, but it must be greater than the third score on the list. If there is no third score, any score will be accepted.

After entering the new score, the **Hi-Score** program will check to be sure it's eligible. If not, you'll be asked if you want to re-enter the score or abort the update process. Press the letter of your choice.

If the score is valid, you must enter the level and the name of the person who obtained the score.

After entering the information, you'll have three options: *O* for okay, *R* for re-enter and *A* for abort. If you abort, you'll be asked to confirm with a *Y* or an *N*.

When the new score is correct, press *O* for okay. **Hi-Score** will ask if you have more scores to update. If you do, press *Y*. If not, press *N*. The new information will be written to disk, then you'll return to the main menu.

Option 3 allows you to delete a game from the list. After selecting the game to delete, you'll be shown the scores on the screen. Type the word *DELETE* at the prompt. Any other entry, including a RETURN alone, will abort the deletion process. If you delete the game, you'll be asked if you have more to delete. If not, the disk file will be updated, and you'll return to the main menu.

Option 4 on the main menu allows you to view your scores, six games at a time. The game names will be in inverse video, to set them apart from the scores. There are three valid keystrokes at this point. They are: *M* for menu, *P* for previous screen and *N* for next screen.

If you're viewing the first screen of data and have more than six games on file, the *N* will appear on-screen, informing you that there's more data in the file. You may press the *N* key to view the next screen of scores.

The *P* key option will never appear on the first screen of data—there's no previous screen. It will appear on the second screen of data and beyond. The *N* will disappear on the last screen. Press the *M* key at any time to return to the main menu.

Option 5 on the main menu presents you with a new menu of four options. All options from this menu will send output to the printer.

The program will ask if you're using a 40- or 80-column printer. I added this option for ease of use with the Atari 1020 plotter. Most users will press *E*, to select 80-column print. All printed reports will fit on a single 8½×11-inch sheet of paper.

The next screen will show you the function selected and ask you to check the printer. To cancel the option, hit *A* for abort. If you're ready to print, press *P*. After the print is complete, you'll be returned to the print menu. You may choose another print function, or return to the main menu, where the final option is "Exit Program."

Technical notes.

I wrote four machine language routines for use in **Hi-Score Display**. One changes a string of characters to inverse video. One fills lines on the screen with a chosen

character. The third pulls game names from the main data string for fast display on the select game screen. The final routine is a general-purpose, multi-key sort program. I used the CIO routines presented in **ANALOG Computing**'s issue 13 for the high-speed disk reads and writes. I also wrote a custom input routine for use throughout this program.

The only place the screen colors are altered is in Line 10, so you may use any colors you like by changing the POKE values. Since **Hi-Score** has custom input routines, there's no keyclick on any of the inputs. If this bothers you, you'll have to add some SOUND statements to the input routines in Lines 20-200.

I've been using **Hi-Score** for over a year, making improvement as I went along. I hope you'll enjoy it.

Kevin Peck is currently in studying Computer Science. He's been working on Ataris for four years, and is in the process of reading every book on Atari machine language he can get his hands on, in the hope of writing an all-machine-language game.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
QI 10 GRAPHICS 0:POKE 82,2:POKE 710,145
SP 20 ? :? :? "NEW":? :?
MB 30 ? "3730 LF$=";CHR$(34);
BC 40 FOR I=1 TO 59:READ A:? CHR$(27);CHR
   $(A);:NEXT I:? CHR$(34)
US 50 ? "3740 SP$=";CHR$(34);
XJ 60 FOR I=1 TO 80:READ A:? CHR$(27);CHR
   $(A);:NEXT I:? CHR$(34)
FS 70 ? "3750 SP$(81)=";CHR$(34);
IL 80 FOR I=81 TO 105:READ A:? CHR$(27);C
   HR$(A);:NEXT I:? CHR$(34)
YC 90 ? "3760 RV$=";CHR$(34);
CA 100 FOR I=1 TO 21:READ A:? CHR$(27);CH
   R$(A);:NEXT I:? CHR$(34)
OF 110 ? "3770 MKS$=";CHR$(34);
FS 115 FOR I=1 TO 80:READ A:? CHR$(27);CH
   R$(A);:NEXT I:? CHR$(34)
NO 120 ? "3780 MKS$(81)=";CHR$(34);
US 130 FOR I=81 TO 160:READ A:? CHR$(27);
   CHR$(A);:NEXT I:? CHR$(34)
ZI 140 ? "3790 MKS$(161)=";CHR$(34);
IF 150 FOR I=161 TO 192:READ A:? CHR$(27)
   ;CHR$(A);:NEXT I:? CHR$(34)
VB 160 ? "POKE 842,12:GR.0:L."
MS 170 POSITION 0,0:POKE 842,13:STOP
NA 2000 DATA 104,104,101,89,133,207,24,10
   4,101,88,133,206,144,2,230,207
TJ 2010 DATA 104,104,170,104,104,133,203,
   104,104,133,204,104,104,133,205,160
DV 2020 DATA 0,165,205,145,206,200,196,20
   4,208,249,202,208,1,96,24,165
RC 2030 DATA 206,101,203,133,206,144,232,
   230,207,208,228
LN 2040 REM * 59 BYTES
PZ 3000 DATA 104,104,133,206,104,133,205,
   104,104,133,208,104,101,89,133,204
IZ 3010 DATA 24,104,101,88,133,203,144,2,
   230,204,104,104,133,207,104,104
KS 3020 DATA 133,209,104,104,170,160,0,24
   ,177,205,201,244,176,24,201,160
AF 3030 DATA 176,17,201,128,176,8,201,96,
   176,12,201,32,176,5,24,105
HH 3040 DATA 64,144,3,56,233,32,145,203,2
   00,196,209,208,218,202,208,1
XI 3050 DATA 96,24,165,205,101,208,133,20
   5,144,2,230,206,24,165,203,101
LE 3060 DATA 207,133,203,144,192,230,204,
   208,188
YJ 3070 REM * 105 BYTES
ZX 4000 DATA 104,104,133,213,104,133,212,
   160,0,177,212,9,128,145,212,200
RR 4010 DATA 192,16,208,245,96
FZ 4020 REM * 21 BYTES
ZQ 5000 DATA 216,104,104,133,206,104,133,
   205,104,133,215,104,133,214,104,104
WL 5010 DATA 133,203,104,104,133,207,24,1
   01,203,133,216,104,104,133,208,104
DP 5020 DATA 104,133,224,24,101,208,133,2
   09,104,104,133,204,104,104,133,225
HT 5030 DATA 165,215,133,1,56,165,214,229
   ,204,133,0,176,2,198,1,24
HT 5040 DATA 165,206,133,213,165,205,101,
   204,133,212,144,2,230,213,164,207
IV 5050 DATA 177,205,209,212,240,4,144,53
   ,176,28,200,196,216,208,241,165
DX 5060 DATA 208,240,46,164,224,177,205,2
   09,212,240,4,144,32,176,7,200
IY 5070 DATA 196,209,208,241,240,27,165,2
   25,208,23,160,0,177,205,72,177
LY 5080 DATA 212,145,205,104,145,212,200,
   196,204,208,241,240,4,165,225,208
PT 5090 DATA 233,24,165,212,101,204,133,2
   12,165,213,105,0,133,213,197,215
ZK 5100 DATA 208,172,165,212,197,214,208,
   166,24,165,205,101,204,133,205,165
VX 5110 DATA 206,105,0,133,206,197,1,208,
   134,165,205,197,0,208,128,96
AV 5120 REM * 192 BYTES
```

●

Listing 2.
BASIC listing.

```
MU 10 GOSUB 3700:POKE 709,C0:POKE 710,156
   :POKE 82,C2:POKE C752,C1:POKE 712,144:
   GOTO 340
JB 20 PP=C1:A$=" ":A$(14)=" ":A$(C2)=A$:?
   ">_←";
AF 30 POKE 702,C64:POKE 694,C0:GET #C1,A:
   IF A>90 AND A<>126 AND A<>155 THEN 30
BT 40 IF A<32 OR (A=32 AND PP=C1) THEN 30
CO 50 IF A=155 THEN ? " ":A$=A$(C1,PP):RE
   TURN
IW 60 IF A=126 AND PP<>C1 THEN ? " ←←_←";
   :PP=PP-C1:A$(PP,PP)=" ":GOTO 30
RB 70 PRINT CHR$(A);"_←";:A$(PP,PP)=CHR$(
   A):PP=PP+C1:IF PP>L THEN 90
SH 80 GOTO 30
OV 90 GET #C1,A:IF A<>126 AND A<>155 THEN
   90
NR 100 IF A=126 THEN 60
KV 110 A$=A$(C1,L):RETURN
PU 120 PP=C1:N$="        ":? ">_←";:D$="1":
   D=VAL(D$)
YG 130 GET #C1,A:IF A=155 THEN N$=N$(C1,P
   P):? " ":RETURN
FQ 140 IF A=126 AND PP<>C1 THEN ? " ←←_←"
   ;:PP=PP-C1:N$(PP,PP)=" ":GOTO 130
OC 150 IF A<48 OR A>57 THEN POKE 694,C0:P
   OKE 702,C64:GOTO 130
```

```
5B 160 PRINT CHR$(A);"_←";:N$(PP,PP)=CHR$
   (A):PP=PP+C1:IF PP>L THEN 180
NB 170 GOTO 130
DJ 180 GET #C1,A:IF A<>126 AND A<>155 THE
   N 180
AO 190 IF A=126 THEN 140
YY 200 RETURN
YP 210 IO=16*IO:IOCB=832+IO:POKE IOCB+2,1
   1:ADRHI=INT(ADDRESS/256):ADRLO=ADDRESS
   -ADRHI*256
NZ 220 POKE IOCB+4,ADRLO:POKE IOCB+5,ADRH
   I:HI=INT(BYTES/256):LO=BYTES-256*HI
GV 230 POKE IOCB+8,LO:POKE IOCB+9,HI:I=US
   R(ADR("hhh[redacted]LV[redacted]"),IO):CLOSE #IO/16:RETU
   RN
TA 240 IO=16*IO:TRAP 270:IOCB=832+IO:POKE
    IOCB+2,7:ADRHI=INT(ADDRESS/256):ADRLO
   =ADDRESS-ADRHI*256
OF 250 POKE IOCB+4,ADRLO:POKE IOCB+5,ADRH
   I:HI=INT(BYTES/256):LO=BYTES-256*HI
WQ 260 POKE IOCB+8,LO:POKE IOCB+9,HI:I=US
   R(ADR("hhh[redacted]LV[redacted]"),IO)
TQ 270 CLOSE #IO/16:RETURN
YB 280 N$=N$(C1,PP-C1):TEMP$=N$
YR 290 IF LEN(N$)>C3 THEN TEMP$=N$(C1,LEN
   (N$)-C3):TEMP$(LEN(TEMP$)+1)=",":TEMP$
   (LEN(TEMP$)+C1)=N$(LEN(N$)-C2,LEN(N$))
YZ 300 RETURN
PC 310 ZZ=USR(LF,C40*YP+XP,LC,C40,C40,FB)
   :RETURN
EL 320 ZZ=USR(LF,C40*YP+XP,LC,C40,BL,FB):
   RETURN
WY 330 ZZ=USR(SP,TP,C56,YP*C40+XP,C40,14,
   21):RETURN
XN 340 FILE$="D:GAME.DAT"
YS 350 SCRS$=" ":SCRS$(2352)=" ":SCRS$(C2
   )=SCRS$:IO=2:BYTES=2352:ADDRESS=ADR(SC
   RS$)
ZI 360 ? "↰↓↓↓Reading Data File....":OPEN
    #C1,C4,C0,"K:"
AB 370 TRAP 380:OPEN #C2,C4,C0,FILE$:GOSU
   B 240:CLOSE #C2:TRAP 40000:GOTO 400
EH 380 ? "↓↓↓Data File does not exist.":?
    "↓Creating new Data File."
BI 390 CLOSE #C2:TRAP 40000:OPEN #C2,C8,C
   0,FILE$:GOSUB 210:CLOSE #C2
IM 400 FOR I=C1 TO 2352 STEP C56:IF SCRS$
   (I,I)=" " THEN NXGM=I:POP :GOTO 420
SI 410 NEXT I:NXGM=I
YU 420 NXGM=INT(NXGM/C56)
LV 430 ? "↰_____ HIGH SCORES ___
   _____":? "|   USED |
   |    OPEN |":TRAP 3610
ZW 440 ? "_____ MAIN MENU ___
   _____":POSITION C3+(NXGM<C10),1:? NXG
   M:POSITION 31+(42-NXGM<10),1:? 42-NXGM
JW 450 ? "↓↓↓↓  1 Add New Games and Sc
   ores↓":? "  2 Update Scores↓":POKE
   16,112:POKE 53774,112
BN 460 ? "  3 Delete Game from File↓":
   ? "  4 View Scores on Screen↓":? "
    5 Print File Menu↓"
OC 470 ? "  6 Exit Program":POSITION 2
   ,20:? "Enter number of your choice >[redacted]"
   ;
DT 480 POKE 694,C0:GET #C1,A:IF A<48 OR A
   >54 THEN 480
AW 490 ON A-48 GOTO 510,2200,2770,1540,30
   30,500
KN 500 ? "↰Remove disk and store in a saf
   e place.":POKE C752,C0:POKE 16,192:POK
   E 53774,247:CLR :END
FI 510 IF NXGM<42 THEN 580
GC 520 ? "↰↓↓Unable to add any more Games
    to list. There are no open areas.↓"
YM 530 ? "You have two options open to yo
   u now. 1. Copy the Main Program onto a
   nother disk and start a new ";
JH 540 ? "list or 2. Deletesome of the Ga
   mes that you know longerplay or care a
   bout from this list to ";
TD 550 ? " free up space.↓↓↓↓":? "Press [redacted]
   C [redacted] to return to Main Menu."
HR 560 IF PEEK(C764)<>18 THEN 560
RL 570 POKE C764,C255:GOTO 430
DS 580 ? "↰  ADD NEW GAMES TO LIST
   ":POSITION 13,5:? " PROGRAM NAME ":A(C
   1)=C0:A(C2)=C0:A(C3)=C0:TLEVEL$=" "
DF 590 POSITION 6,9:? " SCORE  LEVEL
    NAME ":SLEVEL$=" ":TNAME$=" ":SNAM
   E$=" ":TSCORE$=" ":SSCORE$=" "
ND 600 YP=18:XP=0:FB=0:LC=6:GOSUB 310:POS
   ITION 15,18:? "Ę→          Ę←"
KV 605 POSITION 2,22:? "Press RETURN only
    to Exit"
NI 610 POSITION 2,19:? "Program Name ";:L
   =14:GOSUB 20:GAME$=A$:IF A$=" " AND UP
   FLAG=0 THEN 1240
NA 620 IF A$=" " THEN POP :GOTO 1240
XM 630 POSITION C20-(LEN(A$)/C2),C7:? A$
FV 640 FOR I=1 TO NXGM*C56 STEP C56:IF A$
   =SCRS$(I,I+LEN(A$)-C1) THEN POP :GOTO
   660
HX 650 NEXT I:GOTO 730
ZM 660 GOSUB 310:POSITION 0,16:? "This ga
   me exists on file.":? "If you wish to
   update the ";
CN 670 ? "scores, use option 2 from the m
   ain Menu.↓":? "Press [redacted] A [redacted] to Abort":? "
   ↓      [redacted] R [redacted] to Re-enter with new name."
ER 680 IF PEEK(C764)<>63 AND PEEK(C764)<>
   40 THEN 680
LL 690 IF PEEK(C764)=63 AND UPFLAG=C1 THE
   N POP
PD 700 IF PEEK(C764)=63 AND DOWRITE=C0 TH
   EN POKE C764,C255:GOTO 430
QJ 710 IF PEEK(C764)=63 THEN POKE C764,C2
   55:GOTO 1300
QC 720 POKE C764,C255:GOTO 510
PK 730 IF UPFLAG=C1 THEN RETURN
GC 740 GOSUB 310:POSITION 2,22:? "Enter N
   umbers only. No commas."
VD 750 POSITION 14,18:? "Ę→      Ę←":POSI
   TION 2,19:? "Enter Score ";:L=6:GOSUB
   120:SCORE$=N$:IF N$=" " THEN 750
NH 760 A(1)=VAL(N$):GOSUB 280:SCORE$=TEMP
   $
AI 770 POSITION 13-LEN(SCORE$),11:? SCORE
   $
NF 780 GOSUB 310:POSITION 2,22:? "Press R
   ETURN only if none.":POSITION 14,18:?
   "Ę→  Ę←"
BT 790 POSITION 2,19:? "Enter Level ";:L=
   2:GOSUB 120:LEVEL$=N$:IF N$=" " THEN L
   EVEL$="  "
RD 800 POSITION 21-LEN(LEVEL$),11:? LEVEL
   $
WY 810 GOSUB 310:POSITION 13,18:? "Ę→
    Ę←":POSITION 2,19:? "Enter Name ";:L=
   5:GOSUB 20:IF A$=" " THEN 810
HA 820 NAME$=A$:POSITION 27,11:? NAME$:IF
    UPFLAG=1 THEN RETURN
GM 830 GOSUB 310:POSITION 2,22:? "Optiona
   l Information":? "RETURN only to leave
    blank";
UD 840 POSITION 14,18:? "Ę→      Ę←":POSI
   TION 2,19:? "Enter Score ";:L=6:GOSUB
   120:SSCORE$=N$
SX 850 IF N$=" " THEN SLEVEL$=" ":SNAME$=
   " ":A(C2)=C0:IF UPFLAG=C1 THEN RETURN
NQ 860 IF N$=" " THEN 1070
CB 870 A(C2)=VAL(N$):GOSUB 280:SSCORE$=TE
   MP$
IN 880 POSITION 13-LEN(SSCORE$),13:? SSCO
   RE$
KD 890 GOSUB 310:POSITION 14,18:? "Ę→  Ę←
```

```
   ":POSITION 2,23:? "RETURN only to leav
   e blank";
SN 900 POSITION 2,19:? "Enter Level ";:L=
   2:GOSUB 120:SLEVEL$=N$:IF N$=" " THEN
   SLEVEL$="  "
YW 910 POSITION 21-LEN(SLEVEL$),13:? SLEV
   EL$
VH 920 GOSUB 310:POSITION 13,18:? "↳
    ↳←":POSITION 2,19:? "Enter Name ";:L=
   5:GOSUB 20
UV 930 IF A$=" " THEN 920
CE 940 SNAME$=A$:POSITION 27,13:? SNAME$:
   IF UPFLAG=1 THEN RETURN
GR 950 GOSUB 310:POSITION 2,22:? "Optiona
   l Information":? "RETURN only to leave
    blank";
XL 960 POSITION 14,18:? "↳       ↳←":POSI
   TION 2,19:? "Enter Score ";:L=6:GOSUB
   120:TSCORE$=N$
WW 970 IF N$=" " THEN TLEVEL$=" ":TNAME$=
   " ":A(C3)=C0:IF UPFLAG=C1 THEN RETURN
NV 980 IF N$=" " THEN 1070
DR 990 A(C3)=VAL(N$):GOSUB 280:TSCORE$=TE
   MP$
JL 1000 POSITION 13-LEN(TSCORE$),15:? TSC
   ORE$
TU 1010 GOSUB 310:POSITION 14,18:? "↳  ↳
   ←":POSITION 2,23:? "RETURN only to lea
   ve blank";
EV 1020 POSITION 2,19:? "Enter Level ";:L
   =2:GOSUB 120:TLEVEL$=N$:IF N$=" " THEN
    TLEVEL$="  "
AH 1030 POSITION 21-LEN(TLEVEL$),15:? TLE
   VEL$
OL 1040 GOSUB 310:POSITION 13,18:? "↳
     ↳←":POSITION 2,19:? "Enter Name ";:L
   =5:GOSUB 20
CB 1050 IF A$=" " THEN 1040
CH 1060 TNAME$=A$:POSITION 27,15:? TNAME$
   :IF UPFLAG=1 THEN RETURN
VP 1070 GOSUB 310:IF A(C1)>A(C2) AND A(C2
   )>A(C3) THEN 1210
YL 1080 IF A(C1)>A(C3) THEN 1110
GT 1090 T=A(C1):A(C1)=A(C3):A(C3)=T:TEMP$
   =TNAME$:TNAME$=NAME$:NAME$=TEMP$
PD 1100 TEMP$=SCORE$:SCORE$=TSCORE$:TSCOR
   E$=TEMP$:TEMP$=LEVEL$:LEVEL$=TLEVEL$:T
   LEVEL$=TEMP$
AG 1110 IF A(C1)>A(C2) THEN 1140
AR 1120 T=A(C1):A(C1)=A(C2):A(C2)=T:TEMP$
   =SNAME$:SNAME$=NAME$:NAME$=TEMP$
WX 1130 TEMP$=SLEVEL$:SLEVEL$=LEVEL$:LEVE
   L$=SLEVEL$:TEMP$=SSCORE$:SSCORE$=SCORE
   $:SCORE$=TEMP$
EZ 1140 IF A(C2)>A(C3) THEN 1170
ER 1150 T=A(C2):A(C2)=A(C3):A(C3)=T:TEMP$
   =SNAME$:SNAME$=TNAME$:TNAME$=TEMP$
QR 1160 TEMP$=SLEVEL$:SLEVEL$=TLEVEL$:TLE
   VEL$=TEMP$:TEMP$=SSCORE$:SSCORE$=TSCOR
   E$:TSCORE$=TEMP$
UM 1170 YP=11:GOSUB 310:YP=18:POSITION 13
   -LEN(SCORE$),11:? SCORE$:POSITION 21-L
   EN(LEVEL$),11:? LEVEL$
MN 1180 POSITION 27,11:? NAME$:POSITION 1
   3-LEN(SSCORE$),13:? SSCORE$:POSITION 2
   1-LEN(SLEVEL$),13:? SLEVEL$
ZT 1190 POSITION 27,13:? SNAME$:POSITION
   13-LEN(TSCORE$),15:? TSCORE$:POSITION
   21-LEN(TLEVEL$),15:? TLEVEL$
SE 1200 POSITION 27,15:? TNAME$
RU 1210 POSITION 2,19:? "O all Ok    C
   Correct Errors":? :? "A Abort"
YM 1220 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C65 AND A<>79 AND A<>67 THEN
    1220
CF 1230 IF A<>C65 THEN 1290
YQ 1240 GOSUB 310:POSITION 2,18:? "Ready
   to ABORT":? :? "Are you sure (Yes/No)
   ? ";
AL 1250 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C89 AND A<>C78 THEN 1250
OG 1260 IF A=C78 THEN YP=18:XP=0:LC=6:FB=
   0:GOSUB 310:GOTO 1210
FU 1270 IF DOWRITE=C0 THEN 430
BQ 1280 POSITION 2,18:? "This entry Abort
   ed.":GOTO 1400
WX 1290 IF A<>79 THEN 1430
JL 1300 YP=19:LC=5:GOSUB 310:YP=18:LC=6:P
   OSITION 2,20:? "  Any more entries (Ye
   s/No)? ";
RV 1310 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C89 AND A<>C78 THEN 1310
NQ 1320 I=NXGM*C56+C1:SCR5$(I,I+13)=GAME$
   :SCR5$(I+21-LEN(SCORE$),I+20)=SCORE$:S
   CR5$(I+23-LEN(LEVEL$),I+22)=LEVEL$
VV 1330 SCR5$(I+23,I+27)=NAME$:SCR5$(I+35
   -LEN(SSCORE$),I+34)=SSCORE$:SCR5$(I+37
   -LEN(SLEVEL$),I+36)=SLEVEL$
MA 1340 SCR5$(I+37,I+41)=SNAME$
PD 1350 SCR5$(I+49-LEN(TSCORE$),I+48)=TSC
   ORE$:SCR5$(I+51-LEN(TLEVEL$),I+50)=TLE
   VEL$:SCR5$(I+51,I+55)=TNAME$
DZ 1360 NXGM=NXGM+C1:DOWRITE=C1
ZQ 1370 IF A<>C89 THEN 1400
BW 1380 IF NXGM<42 THEN 510
GT 1390 YP=19:LC=5:GOSUB 310:YP=18:LC=6:P
   OSITION 2,17:? "All 42 spaces filled."
KP 1400 POSITION 2,19:? "Saving all Chang
   es to Disk.....":? "YY":DOWRITE=0
XA 1410 ZZ=USR(MK5,ADR(SCR5$),ADR(SCR5$)+
   NXGM*56,14,0,0,0,56,0)
GB 1420 OPEN #2,8,0,FILE$:BYTE5=2352:ADDR
   ESS=ADR(SCR5$):IO=2:GOSUB 210:TRAP 400
   00:GOTO 430
EQ 1430 GOSUB 310:POSITION 2,18:? "Which
   to Correct ? ":? :? "Press 0 if all
   are correct.";
HF 1440 POSITION 33,7:? "↳← 1":POSITION 3
   3,11:? "↳← 2":POSITION 33,13:? "↳← 3":
   POSITION 33,15:? "↳← 4"
MV 1450 POKE C694,C0:GET #C1,A:IF A<48 OR
    A>52 THEN 1450
JY 1460 A=A-48:IF A=C0 THEN 1070
XE 1470 FB=0:YP=18:FB=0:GOSUB 310
MR 1480 UPFLAG=C1:XP=0:LC=1:FB=0:ON A GOS
   UB 1500,1510,1520,1530
XW 1490 UPFLAG=C0:GOTO 1430
IC 1500 YP=7:GOSUB 310:YP=18:LC=6:GOTO 60
   0
AX 1510 YP=11:GOSUB 310:YP=18:LC=6:GOTO 7
   40
BT 1520 YP=13:GOSUB 310:YP=18:LC=6:GOTO 8
   30
HC 1530 YP=15:GOSUB 310:YP=18:LC=6:GOTO 9
   50
OC 1540 ? "↳":POSITION 10,0:? "LIST OF A
   LL SCORES":IF NXGM<>C0 THEN 1580
KB 1550 ? "↓↓No games on file to see.":?
   "↓Press C to Continue"
GB 1560 IF PEEK(C764)<>18 THEN 1560
BU 1570 POKE C764,C255:GOTO 430
VX 1580 NX=C6:PR=C0
JL 1590 N=C0:POSITION 16,22:? "Menu";:I
   F NX<NXGM THEN POSITION 2,22:? "Next
   ":N=C1
IO 1600 PS=C0:IF PR>C0 THEN POSITION 29,2
   2:? "Previous":PS=C1
JB 1610 IF NX>NXGM THEN NX=NXGM
YN 1620 X=C2:Y=C3:FOR I=PR TO NX-C1:P=I*5
   6+C1
MM 1630 TEMP$="                    ":GAME$=SC
   R5$(P,P+13):FOR Z=14 TO C1 STEP -C1
ZG 1640 IF GAME$(Z,Z)<>" " THEN POP :GOTO
    1660
MV 1650 NEXT Z
KX 1660 Q=C9-INT(Z/C2):TEMP$(Q,Q+Z-C1)=GA
```

```
   ME$:U=USR(RV,ADR(TEMP$))
A5 1670 POSITION X,Y:? TEMP$:POSITION X,Y
   +2:? SCR5$(P+14,P+20);" ";SCR5$(P+21,P
   +22);" ";SCR5$(P+23,P+27)
ZJ 1680 POSITION X,Y+3:? SCR5$(P+28,P+34)
   ;" ";SCR5$(P+35,P+36);" ";SCR5$(P+37,P
   +41)
HD 1690 POSITION X,Y+4:? SCR5$(P+42,P+48)
   ;" ";SCR5$(P+49,P+50);" ";SCR5$(P+51,P
   +55)
VR 1700 IF X=C2 THEN X=21:GOTO 1720
OF 1710 Y=Y+C6:X=C2
FJ 1720 NEXT I
IR 1730 POKE C702,C64:POKE C694,C0:GET #C
   1,A
RK 1740 IF N=C0 AND A=C78 THEN 1730
OV 1750 IF P5=C0 AND A=80 THEN 1730
GL 1760 IF A=77 THEN 430
GE 1770 IF N=C1 AND A=C78 THEN PR=NX:NX=P
   R+C6:GOTO 1790
AQ 1780 IF P5=C1 AND A=80 THEN NX=PR:PR=P
   R-C6:GOTO 1790
YD 1790 ? "K":POSITION 10,0:? "  LIST OF A
   LL SCORES ":GOTO 1590
XH 1800 ? "K":POKE C752,C1:POSITION 0,0:?
   "          Select Game to ";TEMP$;"
   "
SO 1810 XP=C0:YP=C0:LC=23:FB=C128:BL=C1:G
   OSUB 320:XP=39:GOSUB 320
TV 1820 IF NXGM<>C0 THEN 1870
HG 1830 ? "↓↓No Games on file. Use Option
    1 on the Main Menu to add games and s
   cores to  file."
WQ 1840 ? "↓↓Press  C  to Continue"
CW 1850 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>67 THEN 1850
AK 1860 GN=C0:RETURN
GN 1870 XP=C4:YP=C1:TP=ADR(SCR5$):GOSUB 3
   30
WU 1880 XP=22:TP=ADR(SCR5$)+1176:GOSUB 33
   0
BS 1890 XP=0:YP=22:LC=2:GOSUB 310:POSITIO
   N 8,22:? "Arrow Keys Move Pointer"
WN 1900 POSITION 6,23:? "RETURN  - Select
   S   X  - Exit";:X=C2:Y=C1:DX=C2:DY=C1
IB 1910 POSITION X,Y:? "=>"
HW 1920 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A=88 THEN GN=C0:RETURN
RI 1930 IF A=155 THEN 2090
JY 1940 IF A=45 THEN DY=Y-C1:GOTO 2020
LF 1950 IF A=61 THEN DY=Y+C1:GOTO 2050
CX 1960 IF A=42 THEN DX=C20:GOTO 1990
NO 1970 IF A=43 THEN DX=C2:GOTO 1990
TI 1980 GOTO 1920
CZ 1990 IF DX=C20 AND NXGM<22 THEN DX=C2
RY 2000 IF DX=C20 AND Y>NXGM-C20 THEN DY=
   NXGM-21
RD 2010 GOTO 2080
QS 2020 IF DY<C1 AND X=C20 THEN DY=NXGM-2
   1
QR 2030 IF DY<C1 AND X=C2 AND NXGM>21 THE
   N DY=21
PH 2040 IF DY<C1 AND X=C2 THEN DY=NXGM
NF 2050 IF DY>21 THEN DY=C1
AI 2060 IF DY>NXGM THEN DY=C1
QI 2070 IF X=C20 AND DY>NXGM-21 THEN DY=C
   1
HJ 2080 POSITION X,Y:? "  ":X=DX:Y=DY:GOT
   O 1910
BQ 2090 GN=Y:IF X=C20 THEN GN=GN+21
ER 2100 TEMP$=SCR5$((GN-C1)*C56+C1,(GN-C1
   )*C56+14):TEMP$(15)="  ":U=USR(RV,ADR(
   TEMP$))
TO 2110 XP=0:YP=22:LC=2:GOSUB 310:POSITIO
   N 3,22:? "   Selected ";TEMP$;
QP 2120 POSITION 0,23:? "   Is This Corre
   ct (Yes/No) ?";:POSITION 10,10:?
VN 2130 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C78 AND A<>C89 THEN 2130
PS 2140 IF A=C89 THEN RETURN
SC 2150 YP=22:LC=2:XP=0:FB=128:GOSUB 310:
   POSITION 0,23:? "  S  Select Differen
   t Game   A  Abort ";
CZ 2160 POSITION 10,10:?
ME 2170 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C65 AND A<>83 THEN 2170
KP 2180 IF A=C65 THEN GN=C0:RETURN
UC 2190 POSITION X,Y:? "  ":GOTO 1890
MP 2200 TEMP$="Update ":GOSUB 1800:IF GN=
   C0 AND DOWRITE=C0 THEN 430
DC 2210 IF GN=C0 THEN 1400
OE 2220 GN=(GN-C1)*C56+C1
LT 2230 GAME$=SCR5$(GN,GN+13):SCORE$=SCR5
   $(GN+14,GN+20):LEVEL$=SCR5$(GN+21,GN+2
   2):NAME$=SCR5$(GN+23,GN+27)
DE 2240 SSCORE$=SCR5$(GN+28,GN+34):SLEVEL
   $=SCR5$(GN+35,GN+36):SNAME$=SCR5$(GN+3
   7,GN+41):TSCORE$=SCR5$(GN+42,GN+48)
EE 2250 TLEVEL$=SCR5$(GN+49,GN+50):TNAME$
   =SCR5$(GN+51,GN+55):TEMP$=SCORE$(1,3):
   TEMP$(4)=SCORE$(5,7):A(1)=VAL(TEMP$)
ZG 2260 TEMP$=SSCORE$(1,3):TEMP$(4)=SSCOR
   E$(5,7):IF TEMP$="      " THEN TEMP$="
   0"
AX 2270 A(C2)=VAL(TEMP$):TEMP$=TSCORE$(C1
   ,C3):TEMP$(C4)=TSCORE$(C5,C7):IF TEMP$
   ="      " THEN TEMP$="0"
KB 2280 A(C3)=VAL(TEMP$):? "K":POSITION 1
   3,0:? "Update Scores"
GL 2290 TEMP$="                ":FOR Z=14
    TO C1 STEP -C1:IF GAME$(Z,Z)<>" " THE
   N POP :GOTO 2310
MB 2300 NEXT Z
KD 2310 Q=C9-INT(Z/C2):TEMP$(Q,Q+Z-C1)=GA
   ME$:U=USR(RV,ADR(TEMP$))
RY 2320 POSITION 12,2:? TEMP$:POSITION 12
   ,4:? SCORE$;" ";LEVEL$;" ";NAME$
VF 2330 POSITION 12,6:? SSCORE$;" ";SLEVE
   L$;" ";SNAME$
AQ 2340 POSITION 12,8:? TSCORE$;" ";TLEVE
   L$;" ";TNAME$
OO 2350 POSITION 3,12:? " NEW SCORE    N
   EW LEVEL     NAME "
IF 2360 XP=0:YP=20:FB=0:LC=4:GOSUB 310:PO
   SITION 2,22:? "Press  RETURN  only to
   exit"
YC 2370 POSITION 2,20:? "Enter Score ";:L
   =6:GOSUB 120:IF PP=C1 AND DOWRITE=C0 T
   HEN 430
QQ 2380 IF PP=C1 THEN 2760
WC 2390 A(C4)=VAL(N$):IF A(C4)>A(C3) THEN
    2460
EV 2400 YP=20:LC=4:GOSUB 310:POSITION 2,2
   0:? "Score ";N$;" to low to be entered
   ."
CB 2410 POSITION 2,22:? " R  Re-enter   A
   A  Abort";
DZ 2420 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C65 AND A<>82 THEN 2420
VZ 2430 IF A=C65 AND DOWRITE=C0 THEN 430
CH 2440 IF A=C65 THEN 2760
SH 2450 GOTO 2360
TM 2460 GOSUB 280:USCORE$=TEMP$:POSITION
   6,14:? USCORE$
WV 2470 GOSUB 310:POSITION 2,22:? "Press
   RETURN  only to leave blank"
FH 2480 POSITION 2,20:? "Enter Level ";:L
   =2:GOSUB 120:ULEVEL$=N$:IF N$=" " THEN
    ULEVEL$="  "
GS 2490 POSITION 22,14:? ULEVEL$
LW 2500 POSITION 2,20:GOSUB 310:? "Enter
   Name ";:L=5:GOSUB 20:IF A$=" " THEN 25
   00
SD 2510 UNAME$=A$:POSITION 32,14:? UNAME$
WG 2520 GOSUB 310:POSITION 2,20:? " O  al
   l Ok     R  Re-enter":? :? " A  Abort"
```

```
   ;
BF 2530 POKE 702,64:POKE 694,0:GET #1,A:I
   F A=65 AND DOWRITE=0 THEN 430
CJ 2540 IF A=C65 THEN 2760
TD 2550 IF A=82 THEN 2230
CK 2560 IF A=79 THEN 2580
SA 2570 GOTO 2530
AU 2580 DOWRITE=C1:IF A(C4)<A(C1) THEN 26
   10
OJ 2590 TSCORE$=SSCORE$:TLEVEL$=SLEVEL$:T
   NAME$=SNAME$:SSCORE$=SCORE$:SLEVEL$=LE
   VEL$:SNAME$=NAME$
BR 2600 SCORE$=USCORE$:LEVEL$=ULEVEL$:NAM
   E$=UNAME$:GOTO 2640
GC 2610 IF A(C4)<A(C2) THEN 2630
PE 2620 TSCORE$=SSCORE$:TLEVEL$=SLEVEL$:T
   NAME$=SNAME$:SSCORE$=USCORE$:SLEVEL$=U
   LEVEL$:SNAME$=UNAME$:GOTO 2640
PX 2630 TSCORE$=USCORE$:TLEVEL$=ULEVEL$:T
   NAME$=UNAME$
AQ 2640 I=GN:SCRS$(I,I+13)=GAME$:SCRS$(I+
   21-LEN(SCORE$),I+20)=SCORE$:SCRS$(I+23
   -LEN(LEVEL$),I+22)=LEVEL$
WI 2650 SCRS$(I+23,I+27)=NAME$:SCRS$(I+35
   -LEN(SSCORE$),I+34)=SSCORE$:SCRS$(I+37
   -LEN(SLEVEL$),I+36)=SLEVEL$
MN 2660 SCRS$(I+37,I+41)=SNAME$
PQ 2670 SCRS$(I+49-LEN(TSCORE$),I+48)=TSC
   ORE$:SCRS$(I+51-LEN(TLEVEL$),I+50)=TLE
   VEL$:SCRS$(I+51,I+55)=TNAME$
IN 2680 XP=0:YP=4:LC=19:FB=0:GOSUB 310
JL 2690 POSITION 12,4:? SCRS$(I+14,I+20);
   " ";SCRS$(I+21,I+22);" ";SCRS$(I+23,I+
   27)
TM 2700 POSITION 12,6:? SCRS$(I+28,I+34);
   " ";SCRS$(I+35,I+36);" ";SCRS$(I+37,I+
   41)
BU 2710 POSITION 12,8:? SCRS$(I+42,I+48);
   " ";SCRS$(I+49,I+50);" ";SCRS$(I+51,I+
   55)
OT 2720 POSITION 2,20:? "Other Games to U
   pdate (Yes/No) ";
LH 2730 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C89 AND A<>C78 THEN 2730
WI 2740 IF A=C89 THEN 2200
PO 2750 GOTO 1400
EI 2760 XP=0:YP=18:LC=6:GOSUB 310:POSITIO
   N 2,18:? "Aborted.":GOTO 2720
GL 2770 TEMP$="Delete ":GOSUB 1800:IF GN=
   C0 AND DOWRITE=C0 THEN 430
EH 2780 IF GN=C0 THEN 3020
PJ 2790 GN=(GN-C1)*C56+C1
LW 2800 GAME$=SCRS$(GN,GN+13):SCORE$=SCRS
   $(GN+14,GN+20):LEVEL$=SCRS$(GN+21,GN+2
   2):NAME$=SCRS$(GN+23,GN+27)
DH 2810 SSCORE$=SCRS$(GN+28,GN+34):SLEVEL
   $=SCRS$(GN+35,GN+36):SNAME$=SCRS$(GN+3
   7,GN+41):TSCORE$=SCRS$(GN+42,GN+48)
HP 2820 TLEVEL$=SCRS$(GN+49,GN+50):TNAME$
   =SCRS$(GN+51,GN+55)
ZA 2830 ? "K":POSITION 13,0:? "Delete Gam
   e"
PD 2840 TEMP$="                    ":FOR Z=14
    TO C1 STEP -C1:IF GAME$(Z,Z)<>" " THE
   N POP :GOTO 2860
NA 2850 NEXT Z
LC 2860 Q=C9-INT(Z/C2):TEMP$(Q,Q+Z-C1)=GA
   ME$:U=USR(RV,ADR(TEMP$))
SX 2870 POSITION 12,2:? TEMP$:POSITION 12
   ,4:? SCORE$;" ";LEVEL$;" ";NAME$
WE 2880 POSITION 12,6:? SSCORE$;" ";SLEVE
   L$;" ";SNAME$
BP 2890 POSITION 12,8:? TSCORE$;" ";TLEVE
   L$;" ";TNAME$
BZ 2900 POSITION 2,17:? "Type full word D
   ELETE to delete this"
WT 2905 ? "game from the list. Anything e
   lse willAbort the process."
UH 2910 POSITION 2,21:L=6:GOSUB 20:IF A$<
   >"DELETE" AND DOWRITE=C0 THEN 430
OR 2920 IF A$<>"DELETE" THEN 3020
IS 2930 IF GN=2297 THEN 2950
IJ 2940 SCRS$(GN,2296)=SCRS$(GN+56,2352)
OQ 2950 TEMP$="                    ":SCRS$(2297
   ,2310)=TEMP$
PN 2960 SCRS$(2311,2324)=TEMP$:SCRS$(2325
   ,2338)=TEMP$:SCRS$(2339,2352)=TEMP$:NX
   GM=NXGM-1:DOWRITE=1
YJ 2970 YP=17:LC=C7:FB=0:GOSUB 310:POSITI
   ON 2,17:? "Game Deleted."
JT 2980 POSITION 2,21:? "Any more games t
   o Delete (Yes/No) ? ";
IT 2990 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C78 AND A<>C89 THEN 2990
FI 3000 IF A=C89 THEN 2770
ZI 3010 DOWRITE=C0:POSITION 2,21:? "QUpda
   ting Disk File...":GOTO 1420
FC 3020 YP=17:LC=C7:FB=C0:GOSUB 310:POSIT
   ION 2,17:? "↓Process Aborted.":GOTO 29
   80
UT 3030 ? "K":POSITION 15,0:? "HIGH SCORE
   ":POSITION 14,2:? "PRINT MENU↓↓↓"
NO 3040 ? "1 All Games with Scores↓↓":?
    "2 All Games with Top Score only↓↓"
CO 3050 ? "3 List of Game Names Only↓↓"
   :? "4 Return to Main Menu"
VU 3060 POSITION 2,21:? "Press number of
   your choice >";
OE 3070 POKE C694,C0:GET #C1,A:IF A<49 OR
    A>52 THEN 3070
ZX 3080 F=A-48:IF A=52 THEN 430
ZY 3090 POSITION 2,21:? "F ourty or E
   ighty Column Printer";
EV 3100 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>70 AND A<>69 THEN 3100
HH 3110 CLM=80:IF A=70 THEN CLM=40
QL 3120 ? "K":POSITION 12,0:? "Function S
   elected":ON F GOTO 3130,3140,3150
KL 3130 POSITION 7,2:? "Print All Games W
   ith Scores":GOTO 3160
EN 3140 POSITION 3,2:? "Print All Games W
   ith Top Score Only":GOTO 3160
TF 3150 POSITION 4,2:? "Print List of Gam
   e Names On File"
ZO 3160 POSITION 2,6:? "Please be sure pr
   inter is ON-LINE and ready to print."
BA 3170 POSITION 8,11:? "P Print":POSIT
   ION 8,13:? "A Abort print"
GC 3180 POSITION 2,17:? "Press Letter of
   Choice >";
NK 3190 POKE C702,C64:POKE C694,C0:GET #C
   1,A:IF A<>C65 AND A<>80 THEN 3190
UH 3200 IF A=C65 THEN 3030
PC 3210 TRAP 3520:OPEN #C4,C8,C0,"P:":TEM
   P$="                    ":POSITION 2,17:?
   "QPrinting...."
BY 3220 ON F GOTO 3230,3370,3470
PI 3230 S=112:S1=C1:? #C4;TEMP$(C1,C6);:I
   F CLM=80 THEN ? #C4;TEMP$;TEMP$(C1,C4)
   ;:S=224:S1=C3
SX 3240 ? #C4;"LIST OF ALL GAMES WITH SCO
   RES":? #C4:? #C4
SS 3250 FOR I=C1 TO NXGM*C56 STEP S:FOR W
   =C0 TO S1:P=W*C56+I:IF P>2352 THEN POP
    :? #C4:GOTO 3290
IR 3260 GAME$=SCRS$(P,P+13):FOR Z=14 TO C
   1 STEP -C1:IF GAME$(Z,Z)<>" " THEN POP
    :GOTO 3280
MW 3270 NEXT Z:POP :? #C4:GOTO 3290
EA 3280 TEMP$="                    ":Q=C9-INT
   (Z/C2):TEMP$(Q,Q+Z-C1)=GAME$:? #C4;"
    ";TEMP$;:NEXT W:? #C4
XI 3290 FOR W=C0 TO S1:P=W*C56+I:IF P>235
   2 THEN POP :? #C4:GOTO 3310
EO 3300 ? #C4;"   ";SCRS$(P+14,P+C20);" "
   ;SCRS$(P+21,P+22);" ";SCRS$(P+23,P+27)
```

```
   ;:NEXT W:? #C4
BK 3310 FOR W=C0 TO S1:P=W*C56+I:IF P>235
   2 THEN POP :? #C4:GOTO 3330
MX 3320 ? #C4;"   ";SCRS$(P+28,P+34);" ";
   SCRS$(P+35,P+36);" ";SCRS$(P+37,P+41);
   :NEXT W:? #C4
GO 3330 FOR W=C0 TO S1:P=W*C56+I:IF P>235
   2 THEN POP :? #C4:GOTO 3350
MK 3340 ? #C4;"   ";SCRS$(P+42,P+48);" ";
   SCRS$(P+49,P+50);" ";SCRS$(P+51,P+55);
   :NEXT W
HY 3350 ? #C4:? #C4:NEXT I:? #C4
WX 3360 CLOSE #C4:GOTO 3030
MB 3370 S=112:S1=C1:? #C4;"    ";:IF CLM=
   80 THEN ? #C4;"                    ";:
   S=224:S1=C3
ZL 3380 ? #C4;"LIST OF ALL GAMES WITH TOP
    SCORE":? #C4:? #C4
FA 3390 FOR I=C1 TO NXGM*C56 STEP S:FOR W
   =C0 TO S1:P=W*C56+I:IF P>2352 THEN POP
    :? #C4:GOTO 3430
UV 3400 GAME$=SCRS$(P,P+13):FOR Z=14 TO C
   1 STEP -C1:IF GAME$(Z,Z)<>" " THEN POP
    :GOTO 3420
YJ 3410 NEXT Z:? #C4:POP :GOTO 3430
FA 3420 TEMP$="                ":Q=9-INT(
   Z/C2):TEMP$(Q,Q+Z-C1)=GAME$:? #C4;"
   ";TEMP$;:NEXT W:? #C4
DJ 3430 FOR W=C0 TO S1:P=W*C56+I:IF P>235
   2 THEN POP :GOTO 3450
MF 3440 ? #C4;"   ";SCRS$(P+14,P+C20);" "
   ;SCRS$(P+21,P+22);" ";SCRS$(P+23,P+27)
   ;:NEXT W
IA 3450 ? #C4:? #C4:NEXT I:? #C4
WZ 3460 CLOSE #C4:GOTO 3030
MT 3470 S=112:S1=C1:? #C4;"        ";:IF C
   LM=80 THEN ? #C4;"
   ";:S=224:S1=C3
PO 3480 ? #C4;"LIST OF GAME NAMES ON FILE
   ":? #C4:? #C4
DX 3490 FOR I=C1 TO NXGM*C56 STEP S:FOR W
   =C0 TO S1:P=W*C56+I:IF P>2352 THEN POP
    :GOTO 3510
DM 3500 ? #C4;"   ";SCRS$(P,P+13);:NEXT W
   :? #C4:? #C4:NEXT I:? #C4:CLOSE #C4:GO
   TO 3030
SD 3510 ? #C4:? #C4:NEXT I:? #C4:CLOSE #C
   4:GOTO 3030
TU 3520 A=PEEK(195):? "↖↓↓↓ERROR ↳→ ";A:?
    :? :CLOSE #C4
CT 3530 ? "The Printer is not responding.
    Be sureit is on and in an on-line sta
   te."
RH 3540 IF A<>130 AND A<>138 AND A<>139 T
   HEN ? :? "This error might not be the
   printers  fault. Check program."
EC 3550 ? :? "Error occured in line ";PEE
   K(186)+PEEK(187)*C256
WF 3560 ? :? "M Main Menu   P Print M
   enu"
VI 3570 ? :? "Press letter of choice >";
   :POKE C764,C255
TJ 3580 IF PEEK(C764)=C10 THEN POKE C764,
   C255:GOTO 3030
RV 3590 IF PEEK(C764)=37 THEN POKE C764,C
   255:GOTO 430
UG 3600 GOTO 3580
WQ 3610 A=PEEK(195):? "↖↓↓↓ERROR ↳→";A;" I
   N LINE ";PEEK(186)+PEEK(187)*C256:? :?
    :POKE C752,C0:POKE 16,192
CM 3620 POKE 53774,247:IF A<>130 AND A<>1
   38 AND A<>139 AND A<>140 AND A<>142 AN
   D A<>143 AND A<>144 THEN 3640
VN 3630 ? "The problem seems to be with t
   he disk drive. It is not responding pr
   operly. Check drive.":END
WX 3640 IF A=169 THEN ? "The directory on
    the disk is full. No room for the dat
   a file.":END
XA 3650 IF A<>162 THEN 3680
HP 3660 ? "There is not enough room on th
   e disk  for the data file. You need at
    least  19 free sectors in Single ";
GY 3670 ? "or 10 free  sectors in Double
   Density.":END
JN 3680 IF A=167 THEN ? "The data file is
    locked. I am unable to update the inf
   ormation.":END
SO 3690 ? "Please Check manual for explai
   nation  of error.":END
PD 3700 DIM USCORE$(7),ULEVEL$(2),UNAME$(
   5),D$(1),SCRS$(2352),SCORE$(7),GAME$(1
   4),LEVEL$(2),NAME$(5),A$(14),N$(6)
DS 3710 DIM FILE$(10),RV$(22),TNAME$(5),T
   EMP$(16),A(4),SSCORE$(7),TSCORE$(7),SL
   EVEL$(2),TLEVEL$(2),SNAME$(5)
MM 3720 DIM SP$(105),LF$(61),MKS$(192)
OY 3800 RV=ADR(RV$):SP=ADR(SP$):LF=ADR(LF
   $):MKS=ADR(MKS$)
IB 3810 C0=0:C1=1:C2=2:C3=3:C4=4:C5=5:C6=
   6:C7=7:C8=8:C9=9:C10=10:C255=255:C256=
   256:C42=42:C694=694:C702=702
OV 3820 C16=16:C20=20:C764=764:C56=56:C65
   =65:C82=82:C89=89:C78=78:C64=64:C40=40
   :C752=752:C128=128
BA 3830 RETURN
```

●

Create-a-base

Make your own databases with this versatile program.

by C.F. Fogarty, III

Create-a-base is a versatile file and retrieve program that allows you to easily define your own personal databases. It has facilities for creating databases, adding new records to a database, updating records already on your database, searching on multiple key fields, and simple reporting.

Before we go into explanations of how to use **Create-a-base**, here's a quick overview of some common database buzzwords.

A *byte* or character, is the smallest piece of data that **Create-a-base** deals with. It's a single character, like the letter *A*.

A *field* is a collection of bytes and usually contains data pertaining to a single subject or item, like a name or address.

A *record* is a logical collection of fields. For instance, a record in a database called Phone Book, might contain the following fields: name, street, city, state, zip code and phone number.

A *file* (in this case, the database) is a collection of records.

A *database* is merely a collection of related data, usually in multiple files. Large mainframe databases can share data between files, avoiding the need to enter all the fields for each record or entity. However, this type of data sharing is beyond the scope of **Create-a-base**. Remember, most of us (I have an Atari 800) have only 48K of RAM built into our computers. It's this fixed amount of memory that's the main limitation when working with a database.

Limitations.

The input area for each record is limited to a single graphics 0 screen. You can enter up to sixteen fields per record, and each field can be up to 31 bytes long. This gives you a maximum record size of 496 bytes.

$$\frac{31 \text{ bytes}}{\text{field}} \times \frac{16 \text{ fields}}{\text{record}} = \frac{496 \text{ bytes}}{\text{record}}$$

The size of each database is limited only by the capacity of your disk drive and a single disk. A 1050 drive using DOS 2.5 will hold a database one and one-half times the size that an 850 drive will hold. It's also important to note that, the smaller you define your records, the more records you can fit on a disk.

Typing Create-a-base.

The instructions below should be followed exactly to create your copy of **Create-a-base**.

Type in Listing 1, using the **BASIC Editor II** (in issue 47 of **ANALOG Computing**) to verify your work. Be sure to save a backup copy.

Place a disk containing DOS in drive 1, and run the program created from Listing 1. Two files, ML1.LST and ML2.LST, will be written to your disk. Leave this disk in drive 1 until all the steps below have been completed.

After clearing your computer's memory, type in Listing 2 using the **BASIC Editor II** to verify your work. Be sure to save a backup copy. Run the program created from Listing 2. Two files, AUTORUN.SYS and CHSET.PMG, will be written to your disk.

After clearing your computer's memory, type in Listing

3 using the **BASIC Editor II** to verify your work. Save a copy to disk.

Load the program created from Listing 3 into memory and merge the file ML1.LST by typing *ENTER "D:ML1.LST"* and pressing RETURN. Save the resultant program to disk under the filename CREATEAB.ASE.

After clearing your computer's memory, type in Listing 4 using **BASIC Editor II** to verify. Save a copy to disk.

Load the program created from Listing 4 into memory and merge the file ML2.LST by typing *ENTER "D:ML2.LST"* and pressing RETURN. Save the resultant program to disk under the filename SORT.

Getting started.

Once you've typed in all the listings (no simple task) and created a master disk, boot your system with the **Create-a-base** master disk in drive 1. The main program loads automatically and prompts you to insert your database disk. Since this is your first time using **Create-a-base**, remove the master disk and insert a blank disk (no need to format it first). Once you've done this, press START, and **Create-a-base** will inform you that this isn't a valid database disk. Press *Y* to format it.

Now you can define your first database. Here's an example everyone can use. At the prompt for *Database Name*, type *Phone Book* and press RETURN. *Note:* remember to press RETURN after all entries, or **Create-a-base** will ignore that input. Next, it will ask you for a LABEL; enter *NAME*. Now **Create-a-base** will ask you to define the size of the field for NAME, enter *25*. This gives you an input area of 25 bytes for NAME. When you press RETURN **Create-a-base** does some processing on your input and prints the label *NAME* to the screen, followed by twenty-five underline characters. Meanwhile, you're prompted in the status window. Press OPTION to define the next field, or press START when the whole record is defined. This time, press OPTION and use the following list to completely define your Phone Book record:

LABEL	FIELD SIZE	CONSOLE KEY
STREET	25	OPTION
CITY	25	OPTION
STATE	2	OPTION
ZIP CODE	5	OPTION
PHONE#	16	START

Once you've pressed START, **Create-a-base** does some processing and writes to the disk. When it's done, you'll have a database disk called Phone Book and **Create-a-base** will go into the "add records" mode.

The next time you boot the **Create-a-base** master disk, insert this database disk at the prompt, and **Create-a-base** will go directly to the add records mode.

Using the edit screen.

To add records to the database, simply type in the person's name and press RETURN. The cursor automatically moves down to the street field and so on. . . When you've entered all the data for the first record, press START and you've written the first record to your Phone Book!

Advanced editing.

Pressing RETURN alone, without typing any text, moves the cursor down to the next field and leaves that field blank. However, if any text was on that line (as in update mode), it will be erased. Pressing SELECT allows you to move the cursor to the next field without erasing any text.

The OPTION key changes modes. There are five modes —add, search, update, report and create. By pressing OPTION five times, you can cycle through each mode. In all modes except create, the screen looks exactly the same, except for the "mode" in the status window. In create mode, pressing OPTION will bring you back to the add mode, while pressing START takes you to where you defined your database. If you accidentally press START, when you meant to press OPTION in create mode, press ESCape to return to the edit screen. Normally, while editing, you press START only when you're done editing the record on the screen. It tells **Create-a-base** to process your input.

Searching a database.

To search the database for a certain record, press OPTION until you're in search mode. Then type in the information you want to search for. *Remember to press RETURN after each field you enter*, and press START to begin the search. For example, if you wanted to search for Charles Fogarty, you could enter *Charles Fogarty, Fogarty*, or even *F*.

Create-a-base will search the database until it finds a match or comes to the end-of-file. If it finds a match, the record prints to the screen and prompts you to *Continue (Y/N)*. Pressing *Y* continues the search, and any other key brings you back to the edit screen (still in search mode). You can then search for different records. By the way, if you don't type in any information for **Create-a-base** to search with before you press START, it defaults to *all* records, so everything's a match.

You can also search on multiple fields. So, if you wanted to find everyone with a last name of Fogarty, who lives in Hartford, with a zip code of 06118, you could enter that information in the appropriate fields (name, city and zip code), and then press START. Only those records matching all three fields will show up on the screen.

Reporting.

Report mode works exactly like search mode, except all the output goes to the printer.

Updating records.

Update mode also works like search mode, until it finds a matching record. Once it finds a match, you can make any changes to that record displayed on the screen. Press START and the new, updated record is written to the database. The old record is written over by the new one. To delete a record completely, press CTRL-D.

Creating new databases.

Create mode was used to create your Phone Book database. You define labels and fields to create new and different databases. Remember, you can only put one database on a disk. If you try to create a new database on the same disk as Phone Book, it will erase the old Phone Book database and start a new one. You may, however, create as many different databases as you want, as long as they're on separate disks.

Other functions.

Pressing CTRL-P with a printer attached will print out the data currently displayed on the screen. This is sometimes called a "screen dump."

Pressing CTRL-S (for Sort) will prompt you to press OPTION to resume editing, or press START to sort the database. Remember to insert the **Create-a-base** master disk before pressing START, because the sort program is separate from the editor.

Once the sort program's running, it will read the whole database, sort it, and write it out to a new disk. So, after the sort, you'll have two copies of that particular database, the original and the sorted version. This gives you a backup copy, in case of any problems during the sort (like a power outage). The sort also gets rid of any "deleted" records and recovers lost disk space. These "deleted" records are still taking up space on the disk, even though they don't show up when you search.

A small database (one that can be sorted completely in RAM), takes a minute or two. The disk I/O takes considerably longer than the sort itself. Sometimes a large database won't fit into RAM all at once (only 48K), so I tried to use the available memory as efficiently as possible.

After reading and writing the database in blocks (approximately 25K on my system), **Create-a-base** reads the database a second time. This time it notes the position of each record on the disk and keeps only the sort field and pointers. Then it sorts the pointers and reorganizes the file on the disk. This pointer sort allows you to sort files much larger than your main memory could possibly hold. The sort program will also scale down the length of the sort field, to accommodate a very large database. What this means is, if the number of records multipled by the sort field length is greater than the number of records that will fit in RAM, the sort will systematically make the sort field 1 byte smaller, until all the records fit into RAM. A worst case would be that the sort field was only 1 byte long. The records would still be sorted in alphabetical order, only with less precision.

The following is a list of possible databases:

Phone Book		Bowler Stats		Sports Stats	
NAME	25	TEAM	25	TEAM	25
STREET	25	NAME	25	NAME	25
CITY	25	DATE	8	NUMBER	3
STATE	2	SCORE#1	3	etc. . .	
ZIP CODE	5	SCORE#2	3		
PHONE#	16	SCORE#3	3		

Articles		Disk Catalog		Home Inventory	
MAGAZINE	25	DISKNAME	25	LOCATION	25
TITLE	31	FILENAME	12	SERIAL#	25
AUTHOR	25	TYPE	10	DATE	8
MONTH	9	AUTHOR	25	DESCRIPT	31
YEAR	4	COMMENTS	31	VALUE	7
PAGE#	4				
DESCRIPT	31				

Subroutines.

Create-a-base has a number of relocatable machine language routines which can be used in other BASIC programs.

MATCH$ checks if two BASIC string variables are equal. *Call it with: X=USR(ADR(MATCH$),ADR(the first variable),ADR(second variable),LENGTH(to compare).* It com-

SCHEMA$

(1-51) boot program.

(52-64) to check for valid cb disk.

(52-91) display for accidental boot.

(92-106) database name 15 characters

(107-108) number of fields in use. MAX. 16

(109-140) length of corresponding field. MAX. 31

(141-143) record size MAX. 496 bytes

(145-272) label for each field 8 bytes each. MAX. of 16 labels

SECTOR #1

	1	2	3	4	5	6	7	8
1								
9								
17				boot				
25				program				
33								
41								
49				C	r	e	a	t
57	e	-	a	-	b	a	s	e
65		(	c	)		1	9	8
73	5		C	.	F	.	F	o
81	g	a	r	t	y		l	l
89	l			0	1	2	3	4
97	5	6	7	8	9	A	B	C
105	D	E	1	6	0	0	1	1
113	2	2	3	3	4	4	5	5
121	6	6	7	7	8	8	9	9

SECTOR #2

	1	2	3	4	5	6	7	8
1	A	A	B	B	C	C	D	D
9	E	E	F	F	4	9	6	
17	L	A	B	E	L	#	0	1
25	L	A	B	E	L	#	0	2
33	L	A	B	E	L	#	0	3
41	L	A	B	E	L	#	0	4
49	L	A	B	E	L	#	0	5
57	L	A	B	E	L	#	0	6
65	L	A	B	E	L	#	0	7
73	L	A	B	E	L	#	0	8
81	L	A	B	E	L	#	0	9
89	L	A	B	E	L	#	1	0
97	L	A	B	E	L	#	1	1
105	L	A	B	E	L	#	1	2
113	L	A	B	E	L	#	1	3
121	L	A	B	E	L	#	1	4

SECTOR #3

	1	2	3	4	5	6	7	8
1	L	A	B	E	L	#	1	5
9	L	A	B	E	L	#	1	6
17								
25								
33				reserved				
41				for				
49				future				
57				expansions.				
65								
73								
81								
89								
97								
105								
113								
121								

pares from left to right and returns *0* if they're equal, or *1* if they don't match.

Note: it considers underline characters as wildcards (always a match).

MOVEMEM$ moves memory. Call with: *X=USR(ADR(MOVEMEM$),FROM,TO,number of bytes to move,ADR(CONVERT$))*. *ADR(CONVERT$)* is optional—if used, it converts ATASCII to screen display code (what you see on the screen).

PARSE$ checks for valid input. Call with: *X=USR(ADR(PARSE$),IN,TYPE)*,where: *IN* is an ATASCII value (like *GET #1,IN*); *TYPE = ASC("A")* checks for alphanumeric; or *TYPE = ASC("N")* checks for numeric. The values returned in *X* are: 0 = invalid input; 1 = backspace was pressed; 2 = RETURN was pressed; and 3 = valid input.

SCANKB$ scans the keyboard and console keys. Call with: *X=USR(ADR(SCANKB$))*. It exits to BASIC when a key is pressed. The values returned in *X* are: 1 = a key was pressed; 2 = OPTION was pressed; 3 = SELECT was pressed; and 4 = START was pressed.

SECTORIO$ reads/writes disk sectors. Call with: *X=USR(ADR(SECTORIO$),sector number,operation,ADR (buffer))*, where *sector number* is any valid sector, and *operation* is ASC("R") for read or ASC("W") for write; the *buffer* must be at least 128 bytes long.

SORT$ is a bubble sort in machine language. It's not the best sort in the world, but it will sort 25K in about one minute. It sorts on one key field, which can be a maximum of 255 bytes long. The records can be any size. Call with: *X=USR(ADR(SORT$),ADR(file),number of records to sort,record length,sort-field length,starting position of the sort-field within a record)*. It returns *0* for success, or *nonzero* when invalid parameters are passed to it.

STRIP$ strips trailing underline characters from a string variable. Call with: *X=USR(ADR(STRIP$),ADR(string), LEN(string)-1))*. The string must be at least 2 bytes long. It returns the position of the last nonunderline character in *X*.

Well, that's it. Go forth and **Create-a-base**.

C.F. Fogarty worked in OP operations at Aetna for six years and is now a software programmer trainee. He bought his Atari 800 in 1982, and his CompuServe I.D. is 74206,3453. He's married, has a son and enjoys trout fishing and trail riding.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
ES 10 DIM M$(120),L$(120)
FD 20 GRAPHICS 0:POKE 710,0:? "PLACE FORM
   ATTED DISK IN DRIVE":? "THEN PRESS RET
   URN":INPUT L$:OPEN #1,8,0,"D:ML1.LST"
FQ 30 L$="780 CONVERT$=":L$(LEN(L$)+1)=CH
   R$(34):N=29:GOSUB 210:GOSUB 220
NQ 40 L$="790 DLI$=":L$(LEN(L$)+1)=CHR$(3
   4):N=24:GOSUB 210:GOSUB 220
QT 50 L$="795 MATCH$=":L$(LEN(L$)+1)=CHR$
   (34):N=42:GOSUB 210:GOSUB 220
GX 60 L$="800 MOVEMEM$=":L$(LEN(L$)+1)=CH
   R$(34):N=99:GOSUB 210:GOSUB 220
PL 70 L$="805 MOVEMEM$(100)=":L$(LEN(L$)+
   1)=CHR$(34):N=19:GOSUB 210:GOSUB 220
FI 80 L$="820 PARSE$=":L$(LEN(L$)+1)=CHR$
   (34):N=57:GOSUB 210:GOSUB 220
TJ 90 L$="850 SCANKB$=":L$(LEN(L$)+1)=CHR
   $(34):N=35:GOSUB 210:GOSUB 220
EN 100 L$="855 SECTORIO$=":L$(LEN(L$)+1)=
   CHR$(34):N=31:GOSUB 210:GOSUB 220
WL 110 L$="860 STRIP$=":L$(LEN(L$)+1)=CHR
   $(34):N=30:GOSUB 210:GOSUB 220
OH 120 L$="1050 SCHEMA$=":L$(LEN(L$)+1)=C
   HR$(34):N=74:GOSUB 210:GOSUB 220
MF 130 CLOSE #1:OPEN #1,8,0,"D:ML2.LST"
EU 140 L$="505 SORT$=":L$(LEN(L$)+1)=CHR$
   (34):N=74:GOSUB 210:GOSUB 220
VM 150 L$="510 SORT$(75)=":L$(LEN(L$)+1)=
   CHR$(34):N=75:GOSUB 210:GOSUB 220
PI 160 L$="515 SORT$(150)=":L$(LEN(L$)+1)
   =CHR$(34):N=75:GOSUB 210:GOSUB 220
QD 170 L$="520 SORT$(225)=":L$(LEN(L$)+1)
   =CHR$(34):N=75:GOSUB 210:GOSUB 220
LR 180 L$="525 SORT$(300)=":L$(LEN(L$)+1)
   =CHR$(34):N=29:GOSUB 210:GOSUB 220
DH 190 L$="535 SEARCH$=":L$(LEN(L$)+1)=CH
   R$(34):N=77:GOSUB 210:GOSUB 220
YS 200 RESTORE 440:L$="500 SECTORIO$=":L$
   (LEN(L$)+1)=CHR$(34):N=31:GOSUB 210:GO
   SUB 220:END
TB 210 FOR X=1 TO N:READ A:M$(X)=CHR$(A):
   NEXT X:RETURN
VK 220 L$(LEN(L$)+1)=M$:L$(LEN(L$)+1)=CHR
   $(34):? #1;L$:M$="":RETURN
AO 230 REM ****** CONVERT$ ******
DC 240 DATA 24,201,32,144,12,201,96,144,1
   6,201,128,144,15,201,160,176,4,105,64,
   208,7,201,224,176,3,56,233,32,96
FL 250 REM ****** DLI$ ******
VA 260 DATA 72,138,72,169,176,162,44,236,
   11,212,144,2,169,161,141,10,212,141,24
   ,208,104,170,104,64
KN 270 REM ****** MATCH$ ******
LS 280 DATA 104,104,133,204,104,133,203,1
   04,133,206,104,133,205,160,0,132,212,1
   32,213,162,1,104,104,240,14
VB 290 DATA 133,207,177,203,209,205,208,6
   ,200,196,207,208,245,202,134,212,96
PJ 300 REM ****** MOVEMEM$ ******
CD 310 DATA 104,133,214,201,4,240,4,201,3
   ,208,93,104,133,204,104,133,203,104,13
   3,206,104,133,205,104,133
WX 320 DATA 208,104,133,207,165,214,201,4
   ,208,13,169,76,133,214,104,133,216,104
   ,133,215,24,144,4,169,96
FD 330 DATA 133,214,160,0,166,207,240,24,
   132,207,177,203,32,214,0,145,205,230,2
   03,208,2,230,204,230,205
RL 340 DATA 208,2,230,206,202,208,234,166
   ,208,240,9,202,134,208,162,255,230,207
   ,208,221,166,207,208,209,134
VJ 350 DATA 212,134,213,96,170,240,5,104,
   104,202,208,251,134,213,232,134,212,96
UI 360 REM ****** PARSE$ ******
XY 370 DATA 104,104,104,133,203,104,104,1
   33,204,162,0,134,212,134,213,232,169,1
   26,197,203,240,32,232,169,32
NV 380 DATA 197,203,240,25,232,169,78,197
   ,204,240,6,169,31,160,123,208,4,169,32
   ,160,59,197,203,176,6
PG 390 DATA 196,203,144,2,134,212,96
```

```
ZW 400 REM ****** SCANKB$ ******
KQ 410 DATA 104,160,0,132,213,162,1,173,2
   52,2,201,255,208,18,232,173,31,208,201
   ,3,240,10,232,201,5
SF 420 DATA 240,5,232,201,6,208,229,134,2
   12,96
XL 430 REM ****** SECTORIO$ ******
JM 440 DATA 104,104,141,11,3,104,141,10,3
   ,104,104,141,2,3,104,141,5,3,104,141,4
   ,3,169,1,141,1,3,32,83,228,96
JS 450 REM ****** STRIP$ ******
LP 460 DATA 104,104,133,204,104,133,203,1
   04,104,133,205,169,0,133,213,164,205,1
   77,203,201,95,208,3,136,208
EK 470 DATA 247,200,132,212,96
ZM 480 REM ****** SCHEMA$ ******
IT 490 DATA 0,3,0,7,6,7,162,0,160,120,189
   ,50,7,201,32,144,12,201,96,144,16,201,
   128,144,15
ID 500 DATA 201,160,176,4,105,64,208,7,20
   1,224,176,3,56,233,32,145,88,200,232,2
   24,56,208,218,24,0
OT 510 DATA 32,195,242,229,225,244,229,17
   3,225,173,226,225,243,229,32,40,99,41,
   32,49,57,56,53,32
SL 520 REM ****** SORT$ ******
NR 530 DATA 216,162,1,134,231,202,134,232
   ,134,209,104,201,5,240,12,170,240,5,10
   4,104,202,208,251,232,134
YF 540 DATA 212,96,104,133,204,133,208,10
   4,133,203,133,207,104,133,225,104,133,
   224,104,133,227,104,133,226,104
OZ 550 DATA 104,133,230,104,133,229,104,5
   6,233,1,133,228,165,229,233,0,133,229,
   165,225,208,6,165,224,201
HR 560 DATA 2,144,201,165,227,208,6,165,2
   26,201,1,144,191,165,230,201,1,144,185
   ,24,165,228,101,230,133
DT 570 DATA 214,165,229,105,0,133,215,165
   ,227,197,215,144,166,165,226,197,214,1
   44,160,165,207,133,205,165,208
OD 580 DATA 133,206,24,165,205,101,226,13
   3,207,165,206,101,227,133,208,24,165,2
   05,101,228,133,214,165,206,101
CU 590 DATA 229,133,215,24,165,207,101,22
   8,133,216,165,208,101,229,133,217,160,
   0,177,216,209,214,144,75,208
LI 600 DATA 5,200,196,230,208,243,24,165,
   231,105,1,133,231,165,232,105,0,133,23
   2,165,231,197,224,208,175
JJ 610 DATA 165,232,197,225,208,169,166,2
   09,240,36,134,231,202,134,232,134,209,
   165,203,133,207,165,204,133,208
HG 620 DATA 56,165,224,233,1,133,224,165,
   225,233,0,133,225,208,135,165,224,201,
   1,208,129,96,208,188,165
UK 630 DATA 226,133,212,165,227,133,213,1
   65,205,133,214,165,206,133,215,165,207
   ,133,216,165,208,133,217,160,0
PU 640 DATA 166,212,240,27,132,212,177,21
   4,72,177,216,145,214,104,145,216,230,2
   14,208,2,230,215,230,216,208
DD 650 DATA 2,230,217,202,208,231,166,213
   ,240,9,202,134,213,162,255,230,212,208
   ,218,166,212,208,206,232,134
DB 660 DATA 209,208,175
DK 670 REM ****** SEARCH$ ******
SR 680 DATA 104,104,133,204,104,133,203,1
   04,133,206,104,133,205,104,104,133,207
   ,169,0,133,212,133,213,162,1
SM 690 DATA 24,165,203,101,207,133,203,16
   5,204,105,0,133,204,24,165,212,105,1,1
   33,212,165,213,105,0,133
JL 700 DATA 213,224,1,208,8,202,24,165,20
   7,105,3,133,207,160,0,177,203,209,205,
   208,210,200,192,3,208,245,96
```

Listing 2.
BASIC listing.

```
LP 10 GRAPHICS 0
UF 20 ? "CREATE-A-BASE MASTER DISK MAKER"
XY 30 ? :? "USE A FORMATTED DISKETTE WITH
    DOS.SYS AND DUP.SYS"
HA 40 ? :? "PRESS [START] TO CONTINUE..."
GW 50 IF PEEK(53279)<>6 THEN 50
GH 60 ? :? "WRITING AUTORUN.SYS..."
SG 70 OPEN #1,8,0,"D:AUTORUN.SYS":RESTORE
    100
DP 80 READ A:IF A=-999 THEN 200
TB 90 PUT #1,A:GOTO 80
GH 100 DATA 255,255,160,6,162,6
KH 101 DATA 76,175,6,175,6,251
BE 102 DATA 6,160,11,185,0,228
OD 103 DATA 153,163,6,136,16,247
ZN 104 DATA 169,222,141,167,6,169
IA 105 DATA 6,141,168,6,172,170
LP 106 DATA 6,174,169,6,232,208
XG 107 DATA 1,200,142,246,6,140
MA 108 DATA 247,6,169,163,141,33
ET 109 DATA 3,169,6,141,34,3
YL 110 DATA 96,172,0,6,208,10
FE 111 DATA 169,0,141,33,3,169
AD 112 DATA 228,141,34,3,185,1
GF 113 DATA 6,206,0,6,72,32
CG 114 DATA 251,6,104,160,1,96
GD 115 DATA 253,6,255,6,108,250
JK 116 DATA 191,68,2,68,2,0
UK 117 DATA 9,0,9,0,1,226
ZZ 118 DATA 2,227,2,160,6,224
DJ 119 DATA 2,225,2,253,6,0
SH 120 DATA 6,19,6,18,155,69
WW 121 DATA 83,65,46,66,65,69
YU 122 DATA 84,65,69,82,67,58
FT 123 DATA 68,34,78,85,82
MR 124 DATA -999
KL 200 CLOSE #1:OPEN #1,8,0,"D:CHSET.PMG"
   :RESTORE 1000
IO 210 ? :? "WRITING CHSET.PMG..."
TC 220 READ A:IF A=-999 THEN 240
VL 230 PUT #1,A:GOTO 220
NW 240 FOR X=1 TO 510:PUT #1,0:NEXT X:PUT
    #1,155
UL 250 ? :? "DON'T FORGET TO PUT FILES:":
   ? "CREATEAB.ASE & SORT.":? "ON THIS DI
   SK."
PD 1000 DATA 0,0,0,0,0,0
AN 1001 DATA 0,0,0,24,24,24
PC 1002 DATA 24,0,24,0,0,102
HR 1003 DATA 102,102,0,0,0,0
LO 1004 DATA 0,102,255,102,102,255
EB 1005 DATA 102,0,24,62,96,60
PI 1006 DATA 6,124,24,0,0,102
XW 1007 DATA 108,24,48,102,70,0
FQ 1008 DATA 28,54,28,56,111,102
CC 1009 DATA 59,0,0,24,24,24
IB 1010 DATA 0,0,0,0,0,14
ZF 1011 DATA 28,24,24,28,14,0
DW 1012 DATA 0,112,56,24,24,56
UP 1013 DATA 112,0,0,102,60,255
PK 1014 DATA 60,102,0,0,0,24
SE 1015 DATA 24,126,24,24,0,0
JU 1016 DATA 0,0,0,0,0,24
YL 1017 DATA 24,48,0,0,0,126
QM 1018 DATA 0,0,0,0,0,0
EC 1019 DATA 0,0,0,24,24,0
NE 1020 DATA 0,6,12,24,48,96
MP 1021 DATA 64,0,0,60,102,110
PD 1022 DATA 118,102,60,0,0,24
YJ 1023 DATA 56,24,24,24,126,0
YN 1024 DATA 0,60,102,12,24,48
TO 1025 DATA 126,0,0,126,12,24
LP 1026 DATA 12,102,60,0,0,12
XM 1027 DATA 28,60,108,126,12,0
```

```
CX 1028 DATA 0,126,96,124,6,102
CY 1029 DATA 60,0,0,60,96,124
OO 1030 DATA 102,102,60,0,0,126
HD 1031 DATA 6,12,24,48,48,0
PY 1032 DATA 0,60,102,60,102,102
TD 1033 DATA 60,0,0,60,102,62
FF 1034 DATA 6,12,56,0,0,0
VU 1035 DATA 24,24,0,24,24,0
AK 1036 DATA 0,0,24,24,0,24
IZ 1037 DATA 24,48,6,12,24,48
C5 1038 DATA 24,12,6,0,0,0
R5 1039 DATA 126,0,0,126,0,0
KG 1040 DATA 96,48,24,12,24,48
SE 1041 DATA 96,0,0,60,102,12
XX 1042 DATA 24,0,24,0,0,60
YF 1043 DATA 102,110,110,96,62,0
W5 1044 DATA 0,24,60,102,102,126
RX 1045 DATA 102,0,0,124,102,124
MY 1046 DATA 102,102,124,0,0,60
BQ 1047 DATA 102,96,96,102,60,0
HI 1048 DATA 0,120,108,102,102,108
DC 1049 DATA 120,0,0,126,96,124
DH 1050 DATA 96,96,126,0,0,126
UC 1051 DATA 96,124,96,96,96,0
DA 1052 DATA 0,62,96,96,110,102
RM 1053 DATA 62,0,0,102,102,126
OU 1054 DATA 102,102,102,0,0,126
WV 1055 DATA 24,24,24,24,126,0
NB 1056 DATA 0,6,6,6,6,102
QG 1057 DATA 60,0,0,102,108,120
XQ 1058 DATA 120,108,102,0,0,96
TF 1059 DATA 96,96,96,96,126,0
AU 1060 DATA 0,99,119,127,107,99
CF 1061 DATA 99,0,0,102,118,126
LQ 1062 DATA 126,110,102,0,0,60
QN 1063 DATA 102,102,102,102,60,0
KA 1064 DATA 0,124,102,102,124,96
QY 1065 DATA 96,0,0,60,102,102
TY 1066 DATA 102,108,54,0,0,124
CE 1067 DATA 102,102,124,108,102,0
ON 1068 DATA 0,60,96,60,6,6
XR 1069 DATA 60,0,0,126,24,24
OS 1070 DATA 24,24,24,0,0,102
ZB 1071 DATA 102,102,102,102,126,0
TG 1072 DATA 0,102,102,102,102,60
PA 1073 DATA 24,0,0,99,99,107
AZ 1074 DATA 127,119,99,0,0,102
OL 1075 DATA 102,60,60,102,102,0
UA 1076 DATA 0,102,102,60,24,24
VA 1077 DATA 24,0,0,126,12,24
ZK 1078 DATA 48,96,126,0,0,30
UD 1079 DATA 24,24,24,24,30,0
HY 1080 DATA 0,64,96,48,24,12
WG 1081 DATA 6,0,0,120,24,24
VK 1082 DATA 24,24,120,0,0,8
H5 1083 DATA 28,54,99,0,0,0
QR 1084 DATA 0,0,0,0,0,0
VU 1085 DATA 255,0,127,192,135,132
VW 1086 DATA 132,135,192,127,255,0
NO 1087 DATA 119,37,38,37,0,255
MA 1088 DATA 254,3,65,65,65,113
K5 1089 DATA 3,254,24,24,24,248
XK 1090 DATA 248,0,0,0,24,24
MU 1091 DATA 24,248,248,24,24,24
FA 1092 DATA 0,0,0,248,248,24
XJ 1093 DATA 24,24,127,192,135,133
WE 1094 DATA 133,135,192,127,255,0
NI 1095 DATA 119,82,114,66,0,255
VQ 1096 DATA 255,0,93,85,85,93
EL 1097 DATA 0,255,254,3,33,161
TA 1098 DATA 97,33,3,254,127,192
XS 1099 DATA 135,132,129,135,192,127
ON 1100 DATA 255,0,116,100,68,119
AA 1101 DATA 0,255,255,0,119,100
IG 1102 DATA 68,119,0,255,254,3
BH 1103 DATA 113,33,33,33,3,254
PA 1104 DATA 127,192,135,134,132,135
YN 1105 DATA 192,127,255,0,119,68
BG 1106 DATA 20,119,0,255,255,0
P5 1107 DATA 39,85,119,84,0,255
YP 1108 DATA 0,0,0,31,31,24
YA 1109 DATA 24,24,0,0,0,255
WY 1110 DATA 255,0,0,0,254,3
LR 1111 DATA 113,97,65,113,3,254
BG 1112 DATA 0,0,60,126,126,126
GE 1113 DATA 60,0,0,0,0,0
EB 1114 DATA 255,255,255,255,127,192
OH 1115 DATA 131,130,128,131,192,127
YM 1116 DATA 255,0,185,18,147,146
RV 1117 DATA 0,255,255,0,59,169
PM 1118 DATA 177,169,0,255,254,3
BE 1119 DATA 129,1,1,1,3,254
RN 1120 DATA 24,24,24,31,31,0
EH 1121 DATA 0,0,120,96,120,96
QQ 1122 DATA 126,24,30,0,0,24
VH 1123 DATA 60,126,24,24,24,0
ZL 1124 DATA 0,24,24,24,126,60
CJ 1125 DATA 24,0,0,24,48,126
AP 1126 DATA 48,24,0,0,0,24
PE 1127 DATA 12,126,12,24,0,0
CL 1128 DATA 0,24,60,126,126,60
HP 1129 DATA 24,0,0,0,60,6
AD 1130 DATA 62,102,62,0,0,96
YX 1131 DATA 96,124,102,102,124,0
UZ 1132 DATA 0,0,60,96,96,96
MM 1133 DATA 60,0,0,6,6,62
J5 1134 DATA 102,102,62,0,0,0
BU 1135 DATA 60,102,126,96,60,0
AB 1136 DATA 0,14,24,62,24,24
SZ 1137 DATA 24,0,0,0,62,102
ER 1138 DATA 102,62,6,124,0,96
VX 1139 DATA 96,124,102,102,102,0
BV 1140 DATA 0,24,0,56,24,24
PL 1141 DATA 60,0,0,6,0,6
UP 1142 DATA 6,6,6,60,0,96
BW 1143 DATA 96,108,120,108,102,0
BO 1144 DATA 0,56,24,24,24,24
RT 1145 DATA 60,0,0,0,102,127
YK 1146 DATA 127,107,99,0,0,0
VA 1147 DATA 124,102,102,102,102,0
MD 1148 DATA 0,0,60,102,102,102
OT 1149 DATA 60,0,0,0,124,102
BP 1150 DATA 102,124,96,96,0,0
YK 1151 DATA 62,102,102,62,6,6
JH 1152 DATA 0,0,124,102,96,96
NA 1153 DATA 96,0,0,0,62,96
WK 1154 DATA 60,6,124,0,0,24
UX 1155 DATA 126,24,24,24,14,0
LU 1156 DATA 0,0,102,102,102,102
ME 1157 DATA 62,0,0,0,102,102
PD 1158 DATA 102,60,24,0,0,0
IW 1159 DATA 99,107,127,62,54,0
TI 1160 DATA 0,0,102,60,24,60
GQ 1161 DATA 102,0,0,0,102,102
JD 1162 DATA 102,62,12,120,0,0
BN 1163 DATA 126,12,24,48,126,0
CL 1164 DATA 0,24,60,126,126,24
YP 1165 DATA 60,0,24,24,24,24
YE 1166 DATA 24,24,24,24,0,126
PP 1167 DATA 120,124,110,102,6,0
GP 1168 DATA 8,24,56,120,56,24
FA 1169 DATA 8,0,16,24,28,30
YH 1170 DATA 28,24,16,0,0,0
QC 1171 DATA -999
```

●

Listing 3.
BASIC listing.

```
GQ 1 REM Create-a-base
GE 2 REM (c) 1985 C.F.Fogarty III
LP 3 REM Version 3.1 Nov. 02, 1985
NJ 4 REM
```

```
CU 10 POKE 1664,104:POKE 1665,64:POKE 566
   ,128:POKE 567,6
RO 100 GOTO 745
PC 105 REM PROCESS
RP 110 TRAP ERRORHANDLER
PD 115 KBIP$="_":KBIP$(RECSIZE)="_":KBIP$
   (1+(RECSIZE>1))=KBIP$:DISKIP$=KBIP$:FI
   ELD=NUMFIELDS
KZ 120 X=USR(MOVEMEM,ADR(KSCRN$),SCREEN,9
   60,CONVERT)
IO 125 GOSUB SELECT:GOSUB 200
VX 130 KBIP$(FX(FIELD-1)+1,FX(FIELD))=TMP
   2$
PF 135 GOTO 125
XV 140 REM DISK-I/O
JT 145 IF START=640 THEN POINT #2,SECT,CH
   AR:GOTO 155
HG 150 IF IO=12 THEN NOTE #2,SECT,CHAR
QC 155 ICBLH=INT(RECSIZE/256):ICBLL=RECSI
   ZE-ICBLH*256
TU 160 POKE 866,ICCOM:POKE 872,ICBLL:POKE
    873,ICBLH:POKE 868,ICBAL:POKE 869,ICB
   AH
TQ 165 X=USR(CIO,32):IF PEEK(867)=136 THE
   N 180
FU 170 IF PEEK(867)>3 THEN POP :GOTO ERRO
   RHANDLER
AA 175 RETURN
AR 180 SOUND 0,50,10,10
FD 185 X=USR(MOVEMEM,ADR(PROMPT$(161)),SC
   REEN+720,40,CONVERT):CLOSE #2:OK=0
AA 190 FOR X=1 TO 50:NEXT X:SOUND 0,0,0,0
   :FOR X=1 TO 200:NEXT X
KP 195 POP :GOTO PROCESS
CR 200 REM GET-A-LINE
EC 205 TEMP$=""
QQ 210 POKE KEYBD,255:X=USR(SCANKB)
TH 215 ON X GOSUB KEYPRESS,OPTION,SELECT,
   START
MG 220 GOTO 210
RP 225 REM KEYPRESS
PK 230 GET #1,IN:IF IN=27 THEN POP :POP :
   CLOSE #2:OK=0:GOTO PROCESS
NG 235 X=USR(PARSE,IN,TYPE):ON X GOTO BAC
   KSPACE,EOL,LEGALIP
EF 240 IF IN=19 THEN GOSUB 1325
OF 245 IF IN=16 THEN GOSUB 650:REM SCREEN
    DUMP
YE 250 IF  NOT (IN=4 AND START=640) THEN
   280
LR 255 KBIP$="|":KBIP$(RECSIZE)="|":KBIP$
   (1+(RECSIZE>1))=KBIP$
LC 260 SOUND 0,100,10,10
YM 265 X=USR(MOVEMEM,ADR(PROMPT$(201)),SC
   REEN+720,40,CONVERT)
ZX 270 FOR X=1 TO 50:NEXT X:SOUND 0,0,0,0
   :FOR X=1 TO 200:NEXT X
EU 275 GOTO START
EZ 280 POKE 702,64:POKE 694,0:RETURN
MU 285 REM BACKSPACE
VX 290 TEMP=LEN(TEMP$):IF TEMP<2 THEN TEM
   P$="":GOTO PRTTOSCREEN
TA 295 TEMP$=TEMP$(1,TEMP-1):GOTO PRTTOSC
   REEN
AN 300 REM EOL
CN 305 POP :GOTO PRTTOSCREEN
QY 310 REM LEGAL-I/P
IP 315 TEMP=LEN(TEMP$):IF TEMP=MAX THEN R
   ETURN
II 320 TEMP$(TEMP+1)=CHR$(IN)
ID 325 REM PRINT-TO-SCREEN
SM 330 TMP2$=TEMP$:IF LEN(TEMP$)<MAX THEN
    TMP2$(LEN(TMP2$)+1,MAX)=UNDERLINE$
SZ 335 X=USR(MOVEMEM,ADR(TMP2$),SCREEN+LO
   C,MAX,CONVERT)
ZH 340 RETURN
SZ 345 REM SELECT
OX 350 X=USR(MOVEMEM,ADR(OFF$),PMBASE,256
   )
FP 355 FIELD=FIELD+1:IF FIELD>NUMFIELDS T
   HEN FIELD=1
BU 360 X=USR(MOVEMEM,PLAYER,PMBASE+40+FIE
   LD*8,8):POKE 53277,3
FM 365 LOC=49+FIELD*40:TEMP$=""
NU 370 MAX=FL(FIELD)
AC 375 RETURN
DO 380 REM OPTION
OJ 385 CLOSE #2:OK=0
IC 390 OX=OX+1:IF OX>4 THEN OX=0
MB 395 X=USR(MOVEMEM,ADR(OPTABLE$(OX*6+1)
   ),ADR(KSCRN$(767)),6):DSCRN$=KSCRN$
KY 400 X=USR(MOVEMEM,ADR(KSCRN$),SCREEN,9
   60,CONVERT)
UT 405 FIELD=NUMFIELDS:GOSUB SELECT
PW 410 IF OX=4 THEN X=USR(MOVEMEM,WINDOW,
   SCREEN+680,280,CONVERT):OK=1
ZE 420 RETURN
YI 425 REM START
KT 430 POP :POP :IF OK THEN 455
VV 435 IO=4:IF OX=0 THEN IO=9
YA 440 IF OX=2 THEN IO=12
BH 445 CLOSE #2:OPEN #2,IO,0,"D1:DATABASE
   "
CO 450 OK=1
YP 455 ON OX GOTO 495,495,495,955
SA 460 REM ADD
CK 465 ICCOM=11:ICBAH=INT(ADR(KBIP$)/256)
   :ICBAL=ADR(KBIP$)-ICBAH*256:IF KBIP$<>
   DISKIP$ THEN 480
RM 470 SOUND 0,200,10,10:X=USR(MOVEMEM,AD
   R(PROMPT$(241)),SCREEN+720,40,CONVERT)
RO 475 FOR X=1 TO 50:NEXT X:SOUND 0,0,0,0
   :GOTO 485
TB 480 GOSUB 140
EP 485 OK=0:CLOSE #2:GOTO PROCESS
KV 490 REM SEARCH, UPDATE, REPORT
FV 495 ICCOM=7:ICBAH=INT(ADR(DISKIP$)/256
   ):ICBAL=ADR(DISKIP$)-ICBAH*256
SM 500 GOSUB 140
ED 505 REM COMPARE
WW 510 IF DISKIP$(1,1)="|" THEN 500
MH 515 NG=0:AMATCH=1
WM 520 FOR I=0 TO NUMFIELDS-1:TEMP$=KBIP$
   (FX(I)+1,FX(I+1)):IF TEMP$(1,1)="_" TH
   EN 545
JX 525 IF LEN(TEMP$)=1 THEN PTR=1:GOTO 53
   5
TR 530 PTR=USR(STRIP,ADR(TEMP$),FL(I+1)-1
   ):IF PEEK(764)=28 THEN POP :GOTO PROCE
   SS
SR 535 FOR J=1 TO FL(I+1)-PTR+1:NG=USR(MA
   TCH,ADR(DISKIP$(FX(I)+J)),ADR(TEMP$),P
   TR):IF NG=0 THEN J=FL(I+1)+1
RR 540 NEXT J:IF NG THEN I=NUMFIELDS:AMAT
   CH=0
GF 545 NEXT I:IF  NOT AMATCH THEN 500
HK 550 REM A MATCH
JA 555 X=USR(MOVEMEM,SCREEN,ADR(TSCRN$),9
   60):REM SAVE SCREEN
OC 560 FOR I=0 TO NUMFIELDS-1:X=I*40:DSCR
   N$(90+X,89+X+FL(I+1))=DISKIP$(FX(I)+1,
   FX(I+1)):NEXT I
FV 565 X=USR(MOVEMEM,ADR(DSCRN$),SCREEN,9
   60,CONVERT)
TP 570 IF OX=2 THEN 605:REM UPDATE-CONT
WI 575 IF OX=3 THEN GOSUB 650:GOTO 495:RE
   M REPORT-CONT
DF 580 X=USR(MOVEMEM,ADR(PROMPT$),SCREEN+
   720,40,CONVERT):REM CONTY/N
VU 585 POKE 702,64:POKE 694,0:GET #1,IN
XW 590 IF IN=ASC("Y") THEN X=USR(MOVEMEM,
   ADR(TSCRN$),SCREEN,960):GOTO 495
VL 595 IF IN=16 THEN IN=0:GOSUB 650:GOTO
   585
DM 600 OK=0:CLOSE #2:GOTO PROCESS
VP 605 REM UPDATE-CONT
```

```
WZ 610 X=USR(MOVEMEM,ADR(PROMPT$(41)),SCR
   EEN+720,40,CONVERT):REM CHANGE IT Y/N
VH 615 POKE 702,64:POKE 694,0:GET #1,IN
NH 620 IF IN<>ASC("Y") THEN 580
YP 625 REM CHANGE IT
JA 630 X=USR(MOVEMEM,ADR(PROMPT$(81)),SCR
   EEN+720,40,CONVERT):REM CHANGE OR CTRL
   /D
EQ 635 TIP$=KBIP$:KBIP$=DISKIP$:FIELD=NUM
   FIELDS:START=640:OPTION=340:GOTO PROCE
   SS+20
DU 640 POP :POP :ICCOM=11:ICBAH=INT(ADR(K
   BIP$)/256):ICBAL=ADR(KBIP$)-ICBAH*256:
   GOSUB 140:START=425:KBIP$=TIP$
LB 645 OPTION=385:GOTO 580
IT 650 REM REPORT-CONT
QN 655 TRAP 690
NL 660 CLOSE #7:OPEN #7,8,0,"P:"
VN 665 FOR I=0 TO NUMFIELDS-1:IF PEEK(KEY
   BD)=28 THEN POP :GOTO PROCESS
WO 670 IF IN=16 THEN ? #7;SCHEMA$(145+I*8
   ,152+I*8);">";KBIP$(FX(I)+1,FX(I+1)):G
   OTO 680
IN 675 ? #7;SCHEMA$(145+I*8,152+I*8);">";
   DISKIP$(FX(I)+1,FX(I+1))
XI 680 NEXT I:? #7:CLOSE #7:TRAP ERRORHAN
   DLER
AH 685 RETURN
OH 690 REM PRINTER ERROR
AD 695 X=USR(MOVEMEM,ADR(PROMPT$(121)),SC
   REEN+720,40,CONVERT)
HP 700 REM ERRORHANDLER
KJ 705 SW$="|
              || Error #    occurred on li
   ne #         |"
YB 710 SW$(81)="|
                || Press |┬┤ to Recover
   ...          |"
LY 715 TEMP=PEEK(195):TEMP$=STR$(TEMP):SW
   $(50,49+LEN(TEMP$))=TEMP$
YS 720 TEMP=PEEK(186)+PEEK(187)*256:TEMP$
   =STR$(TEMP):SW$(72,71+LEN(TEMP$))=TEMP
   $
WW 725 X=USR(MOVEMEM,ADR(SW$),SCREEN+760,
   160,CONVERT)
AY 730 IF PEEK(53279)<>6 THEN 730
LO 735 CLOSE #2:CLOSE #7:OK=0
OA 740 GOTO PROCESS
RW 745 REM INITIALIZE
FV 750 DIM BLANK$(40),CIO$(7),CONVERT$(29
   ),DISKIP$(496),DLI$(24),DSCRN$(960)
PZ 755 DIM FL(16),FX(16)
RX 760 DIM KBIP$(496),KSCRN$(960),MATCH$(
   42),MOVEMEM$(118),OFF$(256),OPTABLE$(3
   0),PARSE$(57),PLAYER$(8),PROMPT$(280)
NG 765 DIM SCANKB$(35),SCHEMA$(384),SECTO
   RIO$(31),STRIP$(30),SW$(160),TEMP$(40)
   ,TIP$(496),TMP2$(40),TSCRN$(960)
MC 770 DIM UNDERLINE$(40),WINDOW$(280)
EM 775 BLANK$=" ":BLANK$(40)=" ":BLANK$(2
   )=BLANK$:CIO$="hhh▣LV▣"
YX 785 DISKIP$="_":DISKIP$(496)="_":DISKI
   P$(2)=DISKIP$
JO 810 OFF$="♥":OFF$(256)="♥":OFF$(2)=OFF
   $
YZ 815 OPTABLE$="Add    SearchUpdateReport
   Create"
FW 825 PARSE$(25,25)=CHR$(155):PARSE$(44,
   44)=CHR$(34)
PR 830 PLAYER$=CHR$(255):PLAYER$(8)=CHR$(
   255):PLAYER$(2)=PLAYER$
AL 835 PROMPT$="┌──────** Continue?  (Y
   /N) **──────┐ └──────** Change it?
   (Y/N) **──────┘"
SH 840 PROMPT$(81)="┌─* Make changes or ♥
   ├|&D to delete. *─┐ └──────** Printer
   not online **──────┘"
SU 845 PROMPT$(161)="┌──────*** End-of
   -File ***──────┐ └──────** DE
   LETED. **──────┘"
UR 846 PROMPT$(241)="┌──────*** Empty
   Record ***──────┐"
VV 865 UNDERLINE$="_":UNDERLINE$(40)="_":
   UNDERLINE$(2)=UNDERLINE$
PL 870 WINDOW$="┌──────────────────────
   ──────────────────────┐└──────────────
   ──────────────────────┘"
YF 875 WINDOW$(81)="||Press |┬┤ to  ||
   Press ▲◢▗ to use|||create a NEW  -or
   - the CURRENT    |||"
AN 880 WINDOW$(161)="||database.      ||
    database.       |└──────────────┘"
RD 885 WINDOW$(241)="└──────────────────
   ──────────────────────────────────┘"
RW 890 CIO=ADR(CIO$):CONVERT=ADR(CONVERT$
   ):MATCH=ADR(MATCH$):MOVEMEM=ADR(MOVEME
   M$):PARSE=ADR(PARSE$)
OQ 895 PLAYER=ADR(PLAYER$):SCANKB=ADR(SCA
   NKB$):SECTORIO=ADR(SECTORIO$):STRIP=AD
   R(STRIP$):WINDOW=ADR(WINDOW$)
UK 900 KEYBD=764:KEYPRESS=230:BACKSPACE=2
   90:EOL=305:LEGALIP=315:PROCESS=105:PRT
   TOSCREEN=330
UZ 905 OPTION=380:SELECT=350:START=425:ER
   RORHANDLER=705
VK 910 REM INIT.CHSET & PMG.
NK 915 PM=PEEK(106)-8:CHSET=PM*256:POKE 1
   06,PM:GRAPHICS 0:POKE 756,PM
TP 920 CLOSE #1:OPEN #1,4,0,"D:CHSET.PMG"
EE 925 POKE 853,PM:POKE 852,0:POKE 857,6:
   POKE 856,0:POKE 850,7:X=USR(CIO,16)
XJ 930 CLOSE #1:OPEN #1,4,0,"K:"
EO 935 PMBASE=CHSET+1024:POKE 704,212:POK
   E 559,62:POKE 623,1:POKE 53256,3:POKE
   54279,PM:POKE 53248,48
BH 940 X=USR(MOVEMEM,ADR(DLI$),1536,24):D
   L=PEEK(560)+256*PEEK(561):POKE DL+6,13
   0:POKE DL+22,130:POKE 512,0
WN 945 POKE 513,6:POKE 54286,192:POKE 710
   ,176
XG 950 REM CREATE MODE
IU 955 GOSUB 1225:X=USR(MOVEMEM,ADR(KSCRN
   $),SCREEN,960,CONVERT):POKE 53277,0
BM 960 SW$="|  Insert database diskette i
   nto          ||          Disk Drive #1,
                |"
WT 965 SW$(81)="|
                ||   and press |┬┤ to c
   ontinue...   |"
WT 970 X=USR(MOVEMEM,ADR(SW$),SCREEN+760,
   160,CONVERT)
OH 975 IF PEEK(53279)<>6 THEN 975
LZ 980 TRAP 1025
AO 985 SCHEMA$="♥":SCHEMA$(384)="♥":SCHEM
   A$(2)=SCHEMA$
SW 990 REM READ SCHEMA
XZ 995 FOR I=0 TO 2:X=USR(SECTORIO,I+1,82
   ,ADR(SCHEMA$(I*128+1))):NEXT I
EE 1000 IF SCHEMA$(52,64)<>"Create-a-base
   " THEN 1025
UP 1005 IF  NOT RESTART THEN RESTART=1:GO
   SUB 1285:OX=5:GOSUB OPTION:GOTO PROCES
   S
ZV 1010 SW$="| This is a Create-a-base di
   skette.     |"
OR 1015 SW$(41)="| The database name is _
   ______________.|"
SI 1020 X=USR(MOVEMEM,ADR(SCHEMA$(92)),AD
   R(SW$(64)),15):GOTO 1030
NL 1025 SW$="|    Not a Create-a-base dis
   kette.      ||
                |"
CV 1030 SW$(81)="|
                ||          Format it?
   (Y/N)        |"
```

```
EQ 1035 X=USR(MOVEMEM,ADR(SW$),SCREEN+760
   ,160,CONVERT):GET #1,IN:IF IN=27 AND R
   ESTART THEN GOSUB 1285:GOTO PROCESS
AV 1040 IF IN<>ASC("Y") THEN 960
SW 1045 CLOSE #2:XIO 254,#2,0,0,"D1:*.*"
CG 1055 SCHEMA$(75)="C.F.Fogarty III  012
   3456789ABCDE16001122334455667788 99AABB
   CCDDEEFF496♥"
LA 1060 SCHEMA$(384)="♥":SCHEMA$(145)=SCH
   EMA$(144)
KO 1065 SW$="| Enter your database name
            || (Up to 15 Characters) __
   _____________|"
QT 1070 SW$(81)="|
                ||
                 |"
SW 1075 X=USR(MOVEMEM,ADR(SW$),SCREEN+760
   ,160,CONVERT)
PK 1080 NUM=0:RECSIZE=0:OPTION=340:SELECT
   =OPTION:START=OPTION:TYPE=ASC("A"):MAX
   =15:LOC=824:GOSUB 200
SP 1085 SCHEMA$(92,106)=TMP2$
TP 1090 SW$="| Enter a label:
            || (Up to 8 characters) ___
   _____.      |"
RT 1095 SW$(81)="|
                ||
                 |"
RJ 1100 X=USR(MOVEMEM,ADR(SW$),SCREEN+760
   ,160,CONVERT)
YL 1105 TYPE=ASC("A"):MAX=8:LOC=823:GOSUB
    200:TEMP=LEN(TEMP$)
MJ 1110 IF TEMP<MAX THEN TEMP$(TEMP+1,MAX
   )=BLANK$
AP 1115 SCHEMA$(145+NUM*8,144+(NUM+1)*8)=
   TEMP$:NUM=NUM+1:TEMP$=STR$(NUM)
XQ 1120 IF LEN(TEMP$)=1 THEN TEMP$(2)=TEM
   P$:TEMP$(1,1)="0"
NQ 1125 SCHEMA$(107,108)=TEMP$
YP 1130 SW$="| Enter size of field for
          __.||        (Maximum size is 31
    bytes)      |"
IT 1135 SW$(27,41)=TMP2$:X=USR(MOVEMEM,AD
   R(SW$),SCREEN+760,160,CONVERT)
US 1140 TYPE=ASC("N"):LOC=796:MAX=2:GOSUB
    200:TRAP 1130:TEMP=VAL(TEMP$):IF TEMP
   >31 OR TEMP<1 THEN TEMP$="31":TEMP=31
DF 1145 TRAP 955:RECSIZE=RECSIZE+TEMP:IF
   TEMP<10 THEN TMP2$=TEMP$:TEMP$="0":TEM
   P$(2)=TMP2$
KH 1150 TEMP=109+(NUM-1)*2:SCHEMA$(TEMP,T
   EMP+1)=TEMP$
TD 1155 TEMP$=STR$(RECSIZE)
KM 1160 IF LEN(TEMP$)<3 THEN TMP2$=TEMP$:
   TEMP$="0":TEMP$(2)=TMP2$:GOTO 1160
LU 1165 SCHEMA$(141,143)=TEMP$
OR 1170 GOSUB 1285:IF NUM=16 THEN 1200
UA 1175 SW$="| Press ▲ to define anoth
   er field   ||              -OR-
                |"
AB 1180 SW$(81)="| Press |┬┤ after defin
   ing your LAST  ||            field (Ma
   ximum 16 fields).|"
TB 1185 X=USR(MOVEMEM,ADR(SW$),SCREEN+760
   ,160,CONVERT)
AI 1190 IF PEEK(53279)=3 THEN 1090
ZY 1195 IF PEEK(53279)<>6 THEN 1190
LD 1200 REM WRITE SCHEMA
KC 1205 FOR I=0 TO 2:X=USR(SECTORIO,I+1,8
   7,ADR(SCHEMA$(I*128+1))):NEXT I
CN 1210 CLOSE #2:OPEN #2,8,0,"D1:DATABASE
   ":CLOSE #2
UV 1215 OPTION=380:SELECT=350:START=425:R
   ESTART=0:TYPE=ASC("A")
WR 1220 GOTO 985
CW 1225 REM INIT-KSCRN
QW 1230 KSCRN$=" Create-a-base (c) 1985 C
   .F.Fogarty III "
PK 1235 KSCRN$(41)="____________________
   ____________________ "
JR 1240 KSCRN$(720)=" ":KSCRN$(82)=KSCRN$
   (81)
HI 1245 KSCRN$(721)="┌──────────────────
   ──────────────────┐"
TY 1250 KSCRN$(761)="|▲ Create mode.
       ♥├|&S-Sort db. |"
ZE 1255 KSCRN$(801)="|▶ Field to EDIT.
       ♥├|&P-Printout.|"
OV 1260 KSCRN$(841)="||┬┤ When DONE Edit
   ing.<DATABASE NAME>|"
GJ 1265 KSCRN$(881)="|┬┤ CANCEL Operati
   on. ______________|"
KA 1270 KSCRN$(921)="└──────────────────
   ──────────────────┘"
LU 1275 SCREEN=PEEK(88)+256*PEEK(89)
BB 1280 RETURN
IF 1285 GOSUB 1225:NUMFIELDS=VAL(SCHEMA$(
   107,108)):FL(0)=0:FX(0)=0
JD 1290 FOR I=0 TO NUMFIELDS-1
BP 1295 X=I*40:KSCRN$(81+X,88+X)=SCHEMA$(
   145+I*8,152+I*8):KSCRN$(89+X,89+X)=">"
ON 1300 IN=109+I*2:IN=VAL(SCHEMA$(IN,IN+1
   )):KSCRN$(90+X,89+X+IN)=UNDERLINE$:FL(
   I+1)=IN:FX(I+1)=FX(I)+IN
HJ 1305 NEXT I:KSCRN$(905,919)=SCHEMA$(92
   ,106):DSCRN$=KSCRN$:TSCRN$=KSCRN$
AU 1310 RECSIZE=VAL(SCHEMA$(141,143))
IF 1315 X=USR(MOVEMEM,ADR(KSCRN$),SCREEN,
   960,CONVERT)
AL 1320 RETURN
BV 1325 REM SORT
ZA 1330 SW$="| Insert Create-a-base Maste
   r disk and || Press |┬┤ to SORT datab
   ase.        |"
LQ 1335 SW$(81)="|                - OR -
                || Press ▲ to use CU
   RRENT database.  |"
RZ 1340 X=USR(MOVEMEM,ADR(SW$),SCREEN+760
   ,160,CONVERT)
WV 1345 IF PEEK(53279)=3 THEN RETURN
BJ 1350 IF PEEK(53279)<>6 THEN 1345
EK 1355 POP :RUN "D:SORT"
```

●

Listing 4.
BASIC listing.

```
WQ 1 REM Sort for Create-a-base
AR 2 REM (c)1985 C.F.Fogarty III
SZ 3 REM Version 3.4 August 9,1985
NJ 4 REM
AE 100 POKE 106,PEEK(106)+8:POKE 53277,0
KT 105 GOSUB 475:GOTO 215
RC 110 REM DISK I/O
LL 115 ICBAH=INT(ICBAL/K256):ICBAL=ICBAL-
   ICBAH*K256:POKE 850,ICCOM:POKE 852,ICB
   AL:POKE 853,ICBAH:POKE 856,ICBLL
RZ 120 POKE 857,ICBLH:X=USR(CIO,K16):IF P
   EEK(851)>K3 AND PEEK(851)<>136 THEN ST
   OP
ZQ 125 RETURN
RA 130 REM REORG
NU 135 ? #6:? #6;"Reorg in progress...":
   CLOSE #1:OPEN #1,K12,K0,"D1:DATABASE":
   PTR=LENGTH+K1:PTR2=FX+K1:EOF=CTR:K=K1
AL 140 S=ASC(BUFR$(PTR))+ASC(BUFR$(PTR+K1
   ))*K256:C=ASC(BUFR$(PTR+K2)):POINT #K1
   ,S,C:ICCOM=K7:ICBAL=TEMP:GOSUB 115
TN 145 POSITION K0,4:? #K6;"COUNTDOWN..."
   ;EOF;" "
BL 150 IF BUFR$(PTR,PTR+K2)=BUFR$(PTR2,PT
   R2+K2) THEN 185
JS 155 S2=ASC(BUFR$(PTR2))+ASC(BUFR$(PTR2
   +K1))*256:C2=ASC(BUFR$(PTR2+2)):POINT
```

```
   #1,52,C2:ICCOM=K7:ICBAL=TMP2:GOSUB 115
FY 160 IF BUFR$(PTR-K1,PTR-K1)="|" THEN 1
   85
XU 165 POINT #K1,52,C2:ICCOM=K11:ICBAL=TE
   MP:GOSUB 115:BUFR$(PTR-K1,PTR-K1)="|":
   EOF=EOF-1
PW 170 I=USR(SEARCH,ADR(BUFR$),ADR(BUFR$(
   PTR2)),LENGTH):PTR=I*(LENGTH+K3)-K2:TE
   MP$=TMP2$:PTR2=FX+K1+(I-K1)*K3
QJ 175 GOTO 145
UQ 180 REM FIND NEXT VALID TABLE ENTRY
MR 185 OK=K0:IF BUFR$(PTR-K1,PTR-K1)<>"|"
    THEN BUFR$(PTR-K1,PTR-K1)="|":EOF=EOF
   -K1
SR 190 FOR I=K TO CTR:J=I*(LENGTH+K3)-K2:
   IF BUFR$(J-K1,J-K1)<>"|" THEN PTR=J:PT
   R2=FX+K1+(I-K1)*K3:K=I:I=CTR+K1:OK=K1
XM 195 NEXT I:IF OK THEN 140
VL 200 CLOSE #K1
GL 205 FX=41:GOSUB 440:TRAP 205:RUN "D1:C
   REATEAB.ASE"
NT 210 END
KN 215 REM GET SORT FIELD
NE 220 FX=K1:GOSUB 440:IO=82:GOSUB 425:IF
    SCHEMA$(52,64)<>"Create-a-base" THEN
   ? CHR$(253):GOTO 215
NV 225 J=VAL(SCHEMA$(107,108)):? #K6;"↖so
   rt on which field?"
DZ 230 FOR I=K0 TO J-K1:? #K6;I;" ";SCHEM
   A$(145+I*8,152+I*8):NEXT I
EU 235 TRAP 235:? "What is your choice";:
   INPUT K:TRAP 40000:IF K>J-K1 OR K<0 TH
   EN ? CHR$(253):GOTO 235
KH 240 TOO=K0:FOR I=K0 TO K:TEMP=109+I*K2
   :LENGTH=VAL(SCHEMA$(TEMP,TEMP+K1)):TOO
   =TOO+LENGTH:NEXT I
JA 245 FROM=TOO-LENGTH+K1:RECSIZE=VAL(SCH
   EMA$(141,143))
CI 250 DIM TEMP$(RECSIZE),TMP2$(RECSIZE):
   ICBLH=INT(RECSIZE/K256):ICBLL=RECSIZE-
   ICBLH*K256
KE 255 TEMP$="♥":TEMP$(RECSIZE)="♥":TEMP$
   (K2)=TEMP$:TMP2$=TEMP$:TEMP=ADR(TEMP$)
   :TMP2=ADR(TMP2$)
YN 260 I=INT((FRE(K0)-512)/RECSIZE):BUFSI
   ZE=I*RECSIZE:DIM BUFR$(BUFSIZE):BUFR$=
   "":K=K0:EOF=K0
YN 265 REM CONDENSE
MB 270 CLOSE #K1:OPEN #K1,K12,K0,"D1:DATA
   BASE":? #K6;"↖condensing database":CTR
   =K0
SB 275 ICCOM=K7:ICBAL=TEMP:GOSUB 115:IF P
   EEK(851)=136 THEN CLOSE #K1:EOF=K1:GOT
   O 300
RZ 280 IF TEMP$(K1,K1)="┤" THEN 275
XS 285 BUFR$(LEN(BUFR$)+K1)=TEMP$:CTR=CTR
   +K1:POSITION K0,K3:? #K6;"READING RECO
   RD #";CTR
DD 290 IF LEN(BUFR$)=BUFSIZE THEN K=K+1:G
   OSUB 300
SF 295 GOTO 275
YL 300 REM WRITE NEW DATABASE
PC 305 ? #K6:? #K6;"sorting records...":X
   =USR(ADR(SORT$),ADR(BUFR$),LEN(BUFR$)/
   RECSIZE,RECSIZE,LENGTH,FROM)
ZC 310 FX=21:IF LEN(BUFR$)=K0 THEN GOSUB
   450:GOTO 350
QL 315 IF FORMATTED THEN GOSUB 445:GOTO 3
   35
NO 320 ? #K6;"↖>>>>> warning! <<<<<disk g
   ets formatted!>>>>>          <<<<<"
HQ 325 GOSUB 450:? #K6;"FORMATTING...":CL
   OSE #K2:XIO 254,#K2,K0,K0,"D1:*.*":LET
    FORMATTED=1:IO=87:GOSUB 425
RX 330 CLOSE #K2:OPEN #K2,8,K0,"D1:OUTPUT
   "
OZ 335 ? #K6;"↖writing new database":IF
   NOT EOF THEN ? #K6;"BLOCK #";K
DD 340 ICBAL=ADR(BUFR$):ICBAH=INT(ICBAL/K
   256):ICBAL=ICBAL-ICBAH*K256:POKE 866,K
   11:POKE 868,ICBAL:POKE 869,ICBAH
SK 345 POKE 873,INT(LEN(BUFR$)/K256):POKE
    872,LEN(BUFR$)-PEEK(873)*K256:X=USR(C
   IO,32):IF PEEK(867)>K1 THEN STOP
MU 350 BUFR$="":IF  NOT EOF THEN FX=K1:GO
   TO 445
WU 355 CLOSE #K2:XIO 32,#K2,K0,K0,"D1:OUT
   PUT,DATABASE"
OU 360 IF  NOT K THEN 200
GN 365 REM READ POINTERS
WQ 370 FX=CTR*(LENGTH+K3):J=CTR*K3:IF BUF
   SIZE>=FX+J THEN 385
HC 375 LENGTH=LENGTH-K1:TOO=TOO-K1:IF  NO
   T LENGTH THEN STOP
PR 380 GOTO 370
LM 385 PTR=K1:PTR2=FX+K1:BUFR$="♥":BUFR$(
   FX+J)="♥":BUFR$(K2)=BUFR$:CLOSE #K1:OP
   EN #K1,K12,K0,"D1:DATABASE"
AN 390 ? #K6;"↖reading pointers..."
ZG 395 FOR I=K1 TO CTR:POSITION K0,K3:? #
   K6;"RECORDS TO GO...";CTR-I;" "
PN 400 NOTE #K1,S,C:X=INT(S/K256):P$=CHR$
   (S-X*K256):P$(K2)=CHR$(X):P$(K3)=CHR$(
   C):ICCOM=K7:ICBAL=TEMP:GOSUB 115
BT 405 BUFR$(PTR,PTR+LENGTH-K1)=TEMP$(FRO
   M,TOO):BUFR$(PTR+LENGTH,PTR+LENGTH+K2)
   =P$:BUFR$(PTR2,PTR2+K2)=P$
CQ 410 PTR=PTR+LENGTH+K3:PTR2=PTR2+K3:NEX
   T I:? #K6:? #K6;"sorting pointers..."
PA 415 X=USR(ADR(SORT$),ADR(BUFR$),FX/(LE
   NGTH+K3),LENGTH+K3,LENGTH,K1)
MU 420 GOTO 130
XX 425 REM READ/WRITE SCHEMAS
SM 430 FOR I=K0 TO K2:X=USR(ADR(SECTORIO$
   ),I+K1,IO,ADR(SCHEMA$(I*128+K1))):NEXT
    I
ZV 435 RETURN
RR 440 REM INSERT DISK
PV 445 GRAPHICS K1:POKE 710,K0
PS 450 POSITION 7,5:? #K6;"insert":? #K6:
   ? #K6;TABLE$(FX,FX+19):? #K6;"
   disk":? #K6:? #K6
PU 455 ? #K6;"into disk drive #1":? #K6:?
    #K6;"and press [start] to":? #K6;"con
   tinue..."
FW 460 IF PEEK(53279)<>K6 THEN 460
XL 465 ? #K6;"↖";
ZO 470 RETURN
VG 475 REM INIT
CH 480 K0=0:K1=1:K2=2:K3=3:K6=6:K7=7:K11=
   11:K12=12:K16=16:K256=256
SV 485 DIM CIO$(K7),P$(K3),SECTORIO$(31),
   SCHEMA$(384),SORT$(328),TABLE$(60),SEA
   RCH$(77)
OX 490 SCHEMA$="♥":SCHEMA$(384)="♥":SCHEM
   A$(K2)=SCHEMA$
UQ 495 CIO$="hhh*LVd"
WM 530 TABLE$="      database
   output        create-a-base master"
WZ 540 CIO=ADR(CIO$):SEARCH=ADR(SEARCH$)
ZY 545 RETURN
```


GAMES

Squeeze

Amontillado in Action!

by David Plotkin

Squeeze is a fast-action game written in Action! Your objective is to control the gun in the center of the screen and keep the advancing rows of multicolored bricks from meeting in the middle.

The bricks grow faster and the action speeds up in the upper levels. You can choose your own level of difficulty and which score will end the game (your goal).

The gun moves up and down under joystick control. Pushing the stick left and right aims the gun in the appropriate direction. And, of course, pressing the fire button unleashes a stream of bullets to obliterate the bricks.

If two lines of bricks in the same row manage to meet in the center of the screen, the game is over. So keep the lines of bricks from reaching the center—especially tough because the line from the opposite side will grow faster to try and meet its partner!

Each PROCedure is commented to tell what it does. Each level is 1000 points.

Good luck!

David Plotkin has his Masters in Chemical Engineering and works as a Design Engineer for Chevron U.S.A. He owns a 130XE and a 520ST, and is currently a heavy Pascal user on the ST. His interests (on computers) lie in programming, games and tutorials.

Listing 1.
Action! listing.

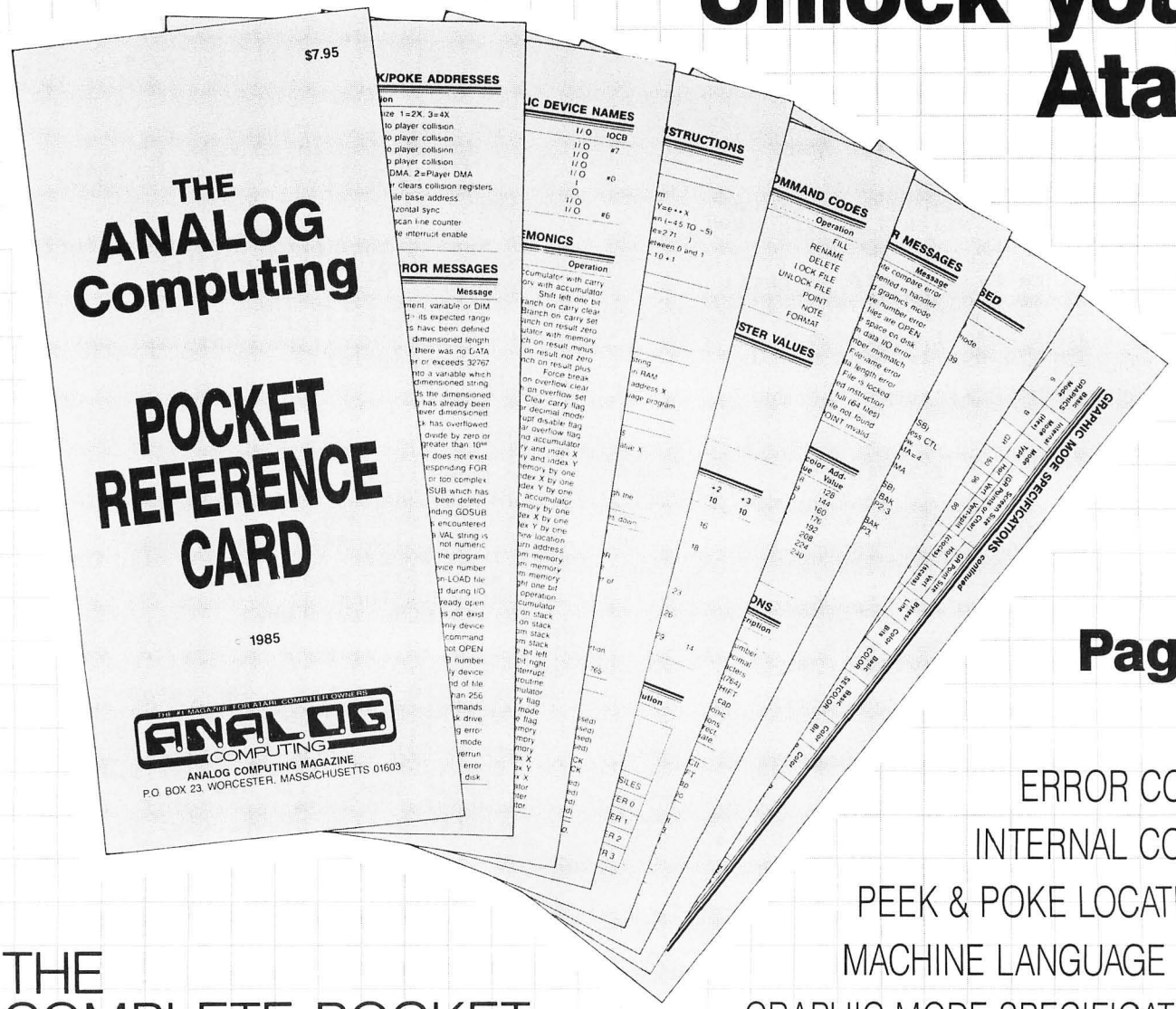
```
MODULE; SQUEEZE by David Plotkin

;       CHECKSUM DATA
;[9D B7 4C F7 52 58 31 F9
; 40 38 55 8D C3 54 96 2B
; 04 40 20 7F 1B 2A 55 57
; 0A 61 51 8F 9F 58 F4 4B
; B6 B4 B5 E9 ]

BYTE ChrBase=756,Max,Bkgrnd=710,
     Fate=53770,Level=[1],CursIn=752,
     Gunx=[19],Guny=[12],Ps=[1],
     Loud=[0],Dly=[3],Hard=[1]

CARD Scrn=88,RamSet,HiMem=$2E5,
     Score=[0],Target

CARD ARRAY Linept(24),L1(30)

BYTE ARRAY Charset,Shotstatus(30),
     Shotx(30),Shoty(30),EndL(24),
     EndR(24),
     ShapeTable(0)=
    [104 208 208 213 213 208 208 104
     10  8   7   87  87  7   7   10
     255 255 255 255 255 255 255 255
     170 170 170 170 170 170 170 170
     85  85  85  85  85  85  85  85
     87  87  87  87  87  87  87  87
     175 175 175 175 175 175 175 175
     170 255 170 255 170 255 170 255
     85  255 85  255 85  255 85  255
     85  170 85  170 85  170 85  170]
```

```
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 Gr0Init()
;Set up the address of each screen
;line and initialize
CARD xx
GRAPHICS(0) CursIn=1 PRINT(" ")
FOR xx=0 TO 23
DO
  Linept(xx)=Scrn+(40*xx)
  EndL(xx)=0 EndR(xx)=39
OD
FOR xx=0 TO 29
DO Shotstatus(xx)=0 Shotx(xx)=0
  Shoty(xx)=0 L1(xx)=xx*1000
OD  Bkgrnd=0
RETURN

PROC Plot0(BYTE x,y,ch)
;Plot a char at location x,y
BYTE ARRAY line
line=Linept(y) line(x)=ch
RETURN

PROC Modify()
;Modify the RAM character set
CARD xx
FOR xx=0 TO 79
DO
  Charset(xx+8)=ShapeTable(xx)
OD
RETURN

PROC UpdateScore()
;Print the score and Level
POSITION(1,23) PRINT("SCORE:")
POSITION(8,23) PRINTC(Score)
POSITION(16,23) PRINT("LEVEL:")
POSITION(23,23) PRINTB(Level)
POSITION(27,23) PRINT("Targ: ")
POSITION(33,23) PRINTC(Target)
RETURN

PROC Noise()
;the explosions when a block is hit
IF Loud=0 THEN RETURN FI
Loud==-2 SOUND(1,90,8,Loud)
RETURN

PROC NewLevel()
;set up a more difficult level
BYTE time=20,lp
SOUND(1,0,0,0) PUT(125) Level==+1
POSITION(9,12)
PRINT("New Level:") POSITION(20,12)
PRINTB(Level) time=0
  DO SOUND(0,time,10,4)
    UNTIL time>100
  OD
PUT(125) SOUND(0,0,0,0)
UpdateScore()
FOR lp=0 TO 29
DO Shotstatus(lp)=0 OD
FOR lp=0 to 23
DO EndL(lp)=0 EndR(lp)=39 OD
IF Level>8 THEN Dly=1 ELSEIF Level>3
  THEN Dly=2 ELSE Dly=3
FI Loud=0
RETURN

PROC Choice()
;choose the difficulty level
BYTE lp=[1],time=20,trig=644,stick=632
POSITION(2,13)
PRINT("Select Difficulty with Joystick")
POSITION(2,14)
PRINT("Then press fire.")
POSITION(7,16)
PRINT("1. Easy - Goal 8000 points")
POSITION(7,17)
PRINT("2. Medium - Goal 12000 points")
POSITION(7,18)
PRINT("3. Hard - Goal 14000 points")
DO Plot0(5,lp+15,0)
  IF stick=14 AND lp>1 THEN lp==-1
    ELSEIF stick=13 AND lp<3 THEN
    lp==+1
  FI Plot0(5,lp+15,84)
  time=0 DO UNTIL time=20 OD
  UNTIL trig=0
OD Hard=lp
IF lp=1 THEN Target=8000 ELSEIF
  lp=2 THEN Target=12000 ELSE
  Target=14000
FI
RETURN

PROC Intro()
;The introduction
BYTE time=20,lp,xx
BYTE ARRAY hello(0)=[51 49 53 37 37
37 37 58 37 1 1 1]
POSITION(7,5)
PRINT("ANALOG PRESENTS")
FOR lp=0 TO 11
DO Plot0(lp+9,8,hello(lp))
  SOUND(0,hello(lp) LSH 1,10,4)
  time=0 DO UNTIL time=9 OD
OD SOUND(0,0,0,0) POSITION(7,9)
PRINT("written in ACTION")
POSITION(7,10)
PRINT("by David Plotkin")
Choice()
FOR lp=0 TO 11
DO xx=lp+9
  DO Plot0(xx,8,0) xx==-1
    IF xx<1 THEN EXIT FI
    Plot0(xx,8,hello(lp))
    SOUND(0,xx LSH 3,10,4)
    time=0 DO UNTIL time=1 OD
  OD
OD SOUND(0,0,0,0) PUT(125)
RETURN

PROC EndGame()
;the game over routines
BYTE time=20,lp,trig=644,xx,yy
BYTE ARRAY gameover(0)=[39 97 109 101
0 47 118 101 114]
PUT(125) SOUND(1,0,0,0)
FOR lp=0 TO 8
DO Plot0(lp+7,12,gameover(lp)) OD
IF Score>=Target THEN POSITION(5,7)
  PRINT("You met your goal!!!")
FI Updatescore()
time=0
DO SOUND(0,time,10,8)UNTIL time=60 OD
SOUND(0,0,0,0) Choice() Level=0
FOR lp=0 TO 8
DO xx=lp+7 yy=12
  DO Plot0(xx,yy,0) xx==+1 yy==-1
    IF (xx>39 or yy<1) THEN EXIT FI
    Plot0(xx,yy,gameover(lp))
    SOUND(0,xx LSH 3,10,4)
    time=0 DO UNTIL time=1 OD
  OD
OD Score=0  NewLevel()
RETURN
```

```
PROC Movegun()
;Read joystick and move the gun
BYTE stick=632
Plot0(Gunx,Guny,0);erase the gun
IF stick=14 THEN;this is a stick up
  Guny==-1 ELSEIF stick=13;stick down
  THEN Guny==+1
FI
IF stick=7 THEN Ps=1 ELSEIF stick=11
  THEN Ps=2;stick right(1) or left(2)
FI
IF Guny<1 THEN Guny=1 ELSEIF;out of
  Guny>21 THEN Guny=21;       Bounds
FI
Plot0(Gunx,Guny,Ps);redraw the gun
RETURN

PROC Testcol(BYTE wh)
;see if bullet wh hit anything
BYTE qq
qq=Shoty(wh)
IF Shotstatus(wh)=1 THEN
  IF EndR(qq)<=Shotx(wh) THEN
    Plot0(Shotx(wh),Shoty(wh),0)
    Shotstatus(wh)=0
    EndR(qq)==+1 Loud=6 Score==+2
  FI ELSE
  IF EndL(qq)>=Shotx(wh) THEN
    Plot0(Shotx(wh),Shoty(wh),0)
    Shotstatus(wh)=0
    EndL(qq)==-1 Loud=6 Score==+2
  FI
FI
IF Score>L1(Level) THEN NewLevel() FI
RETURN

PROC Shoot()
;check the trigger and fire if pushed
BYTE trig=644,lp
IF trig=1 THEN RETURN FI
FOR lp=0 to 29;find an empty shot
DO
  IF Shotstatus(lp)=0 THEN;got one
    IF Ps=1 THEN;gun facing right
      Shotstatus(lp)=1
      Shotx(lp)=Gunx+1 ELSE
      Shotstatus(lp)=2
      Shotx(lp)=Gunx-1
    FI  Shoty(lp)=Guny
    Plot0(Shotx(lp),Shoty(lp),84)
    Testcol(lp) EXIT
  FI
OD
RETURN

PROC MoveShots()
;move the fired bullets
BYTE lp
FOR lp=0 TO 29;for each shot
DO
  IF Shotstatus(lp)=1 THEN;going right
    Plot0(Shotx(lp),Shoty(lp),0)
    Shotx(lp)==+1
    IF Shotx(lp)=39 THEN
      Shotstatus(lp)=0 ELSE
      Plot0(Shotx(lp),Shoty(lp),84)
      Testcol(lp)
    FI
  FI
  IF Shotstatus(lp)=2 THEN;going left
    Plot0(Shotx(lp),Shoty(lp),0)
    Shotx(lp)==-1
    IF Shotx(lp)=0 THEN
      Shotstatus(lp)=0 ELSE
      Plot0(Shotx(lp),Shoty(lp),84)
      Testcol(lp)
    FI
  FI
OD
RETURN

PROC GrowWalls()
;grow squares from both sides
BYTE lvl,lp,dum,y
BYTE ARRAY lmore(24),rmore(24)
FOR lp=0 TO 23
DO lmore(lp)=0 rmore(lp)=0 OD
IF Level>10 THEN
  lvl=10 ELSE lvl=Level
FI
FOR lp=1 to lvl+Hard
DO
IF Fate>210-lvl LSH 2 THEN;grow
  dum=RAND(8)+3 y=RAND(21)+1
  IF Fate>128 AND EndR(y)>20 AND
    rmore(y)=0 THEN rmore(y)=1
    EndR(y)==-1 Plot0(EndR(y),y,dum)
    ELSEIF EndL(y)<18 AND
    lmore(y)=0 THEN lmore(y)=1
    EndL(y)==+1 Plot0(EndL(y),y,dum)
  FI
FI
OD
FOR lp=1 to 22
DO IF EndL(lp)=18 AND EndR(lp)=20
  THEN EndGame() EXIT FI
OD
IF Score>=Target THEN EndGame() FI
RETURN

PROC Main()
BYTE time=20
Gr0Init() Intro()
Download()
Modify() UpdateScore()
DO Movegun() GrowWalls()
Shoot() MoveShots() Noise()
time=0 POSITION(8,23) PRINTC(Score)
DO UNTIL time=Dly OD OD
RETURN
```

●

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

ONLY $7.95 ea.

ANALOG COMPUTING
P.O. BOX 23, WORCESTER, MA 01603
(617) 892-3488 • (617) 892-9230

THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS
ANALOG COMPUTING
P.O. BOX 23, WORCESTER, MA 01603

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

☐ CASH ☐ CHECK ☐ CHARGE

Name _____ Card # _____

Address _____ Exp. date _____

City _____ State ____ Zip _____ Signature _____

48K Cassette or Disk

GAME

Surface Run

A game that helps get you into Action!

by David Plotkin

Action!, the high-speed, high-level language from OSS, is a really excellent tool for game writing. In fact, once you've learned its structured approach (and some of its idiosyncracies) and tasted its dazzling speed, you may never go back to the (normally) slow crawl of BASIC.

Surface Run, included at the end of this article, is a sample of what Action! can do. It's also the first game I've ever written where too much speed was a significant problem. Of course, it's a lot easier to get *rid* of excess speed than to add it.

I've found that there are two ways to program in Action!. The first way can be thought of as "high level," using the many functions and keywords that Action! provides. While this is straightfoward, and even allows for pseudo-translation of BASIC programs (many of the commands or keywords are the same in both Atari BASIC and Action!), it suffers from some loss in speed. An example is found in high-resolution graphics.

PLOT and DRAWTO are available in Action!, but use the Atari's CIO routines, the same ones BASIC uses. This is *not* to say that even "high level" Action! isn't fast... compared to BASIC, it's fast indeed. Still, there are ways to considerably increase the speed of slower Action! functions, to a point approaching true machine speed. This is what I refer to as "low level" Action!. What you do is write your own special routines to do the job. This generally involves direct byte manipulation to the screen, use of shifts instead of multiply/divide, and construction of tables in the program initialization phase, so that results of complex calculations can simply be looked up.

An example is seen in the graphics routines in **Surface Run**. To fill a graphics 7 screen with color using PLOT takes about 27 seconds. Use of my procedure PLOT7, which does some complex direct byte manipulation (bit twiddling?), takes about 4.25 seconds.

The reason that even this procedure still takes so long is that there's a fair amount of math going on before each 2-bit pair is modified. If you can define your picture ahead of time and just place the bytes on-screen using a procedure like FASTDRAW, the process takes about two jiffies (or 1/30 second). In the latter part of this article, we'll talk about some of the more interesting procedures included in **Surface Run**, and what purposes they serve.

Surface Running.

To play **Surface Run**, punch in the listing that follows. Before you run it, save it to disk or cassette (SHIFT CTRL-W, followed by the filename or C: for cassette), then enter the monitor (SHIFT CTRL-M) and compile (C). When the computer beeps at you, plug your joystick into port 1 and run the program by pressing *R*.

You're in control of a space fighter, zooming low on patrol over the scrolling surface of Stripes, your home planet. Pulling back on your joystick causes you to climb; pushing foward makes you dive toward the planet's sur-

face, although your flight computer won't let you crash (at least, not into the surface).

Pressing the joystick left and right will cause the fighter to respond in the appropriate direction. It will also respond to diagonals, for added maneuvering.

And you're not defenseless. Pressing the fire button unleashes missiles which emerge from your wingtips and converge in the distance. You may have up to four missiles on the screen at any one time.

The enemy is a massive "mother ship," which emerges from hyperspace with a roar and moves rather unpredictably about the screen, launching tracking fireballs at you. You must neutralize all these fireballs with your missiles, while destroying the mother ship—by first shooting out the left engine, then the right, and, finally, the main center one.

Strategy is something of a problem: to destroy the mother ship, you must move in close, but the fireballs are more dangerous if you do. You start out with four ships. The number of ships left and your score are kept in the window at the bottom of the screen. My high score is about 7000, so good luck and good hunting!

The real Action!

Some of the PROCedures used in **Surface Run** are quite interesting, and they enhance the speed of the program considerably. Let's touch on some of these programming techniques.

(1) The use of the DEFINE statement to equate assembly code statements (such as RTI or PHA) to the actual hex codes that represent these instructions make the listing more readable and understandable.

(2) SAVETEMPS and GETTEMPS are found whenever an interrupt (such as VBI or System Timer) is used, to save and retrieve the temporary math variables needed by the main program. Thus, the interrupt doesn't change these variables, which could cause some unpredictable results in the main program. The line of hex codes is two short machine language routines to do the job.

(3) PROC DLINT is a display list interrupt (DLI) routine written in Action! Note the use of the assembly code blocks DEFINEd earlier to save the accumulator, and the X- and Y-registers during the interrupt. We did *not* use SAVETEMPS and GETTEMPS, because there isn't enough time, but it seems to work okay. The DLI changes the background color by displaying a hue taken from the byte array CLRS.

(4) PROC INIT7 does the program initialization. The real purpose is do some drawn out math to find screen addresses, then store the results in an array—because it's much faster to look those results up than to calculate them, which would slow down program execution. Thus, the low byte of the address of each screen line is stored in array YLOCL, and the high byte in array YLOCH. The array RSH2 holds which of the 40 bytes on the line is actually referred to by the X-coordinate range from 0 to 159. There's a little trickiness going on here, to break up the 2-byte address held in SCREEN into the two 1-byte numbers needed by YLOCL and YLOCH. By making CARD SCREEN have the same address as BYTE LOW1, each time SCREEN is changed, LOW1 and HIGH1 are also automatically changed. This is sneaky, but *very* fast.

(5) PROC DLSETUP modifies the display list, to turn on the high byte of each instruction on each line where a DLI is required. The instruction *VDSLST=DLINT* installs the DLI.

(6) PROC ROTATE is a routine executed each time the system timer interrupt is called (more on this later). It rotates the elements of the array CLRS, so that the colors displayed by the DLI appear to move down the screen.

(7) INT FUNCs HSTICK and VSTICK are used to read the joystick. They're taken directly from the Programmer's Aid Disk (PAD).

(8) PROC DRAW7 allows you to plot a point on the screen in any of the graphics 7 colors. This is much quicker than using the PLOT function. You pass the x- and y-coordinates, and the color number to the procedure. There's some major speed enhancement here. First, note the BYTE variable declarations. When byte variables are passed to a procedure, they are passed on page 0 in locations $A0 to $AF. So, declaring byte variable X1 to reside at location $A1 equates it to the passed variable X. But, because it's a 0 page quantity, operations using X1 will be faster. Note also that the variables LOW and HIGH are equated to the proper element of YLOCL and YLOCH. This automatically moves byte array LINE to the proper line on the screen, because the variables LOW and HIGH reside in the memory location that defines where byte array LINE will be (see the MODULE statement at the beginning of the program). The last line of this procedure looks pretty horrendous, but what it does is directly manipulate the proper screen byte by punching a 2-bit hole in the byte with a bit mask (array BM), then filling in the hole with the proper color via a color mask (array CM).

(9) PROC FASTDRAW is the fastest of the drawing routines. It takes data contained in a byte array and places it directly on-screen, byte by byte. The variables WIDTH and HEIGHT determine the limits for picture drawing, and XX and YY are the position to draw the picture on the screen. The drawback to using this procedure is that you have to figure out how to draw a picture and convert it to a string of bytes. FASTDRAW is set up to use a picture drawn with **DrawPic**, from Artworx. When you construct a picture with **DrawPic**, you can save the image to disk as BASIC program lines containing a string of bytes. **DrawPic** also automatically saves the width and height. It is then simple to enter these program lines into an Action! program and modify them to the proper format. The byte arrays SHIP, NOLEFT and NOENG declared at the beginning of the program are constructed in just this manner.

The rest of the procedures are fairly straightforward, PMGRAPHICS, PMCLEAR, and PMADR are from the PAD, although PMGRAPHICS is a cut-down version of the general routine provided on PAD.

ERASESHIP removes the mother ship from the screen and increments the difficulty each time you triumph over one. WINDOW draws the text window at the bottom of the screen. UPDATE prints the new score, while UPDATESHIP keeps track of the number of ships you have left.

TESTHIT checks to see if your missiles have hit the proper spot on the mother ship. SHIPFLY moves your little space fighter in response to the joystick. MISSILEFIRE fires off a missile (don't you just *love* descriptive names?) when you press the fire button, provided there aren't four missiles on-screen already. MISSILEMOVE converges the missiles in the distance. The distance the missile has traveled from its original Y-coordinate is used to determine the X-coordinate.

SHIPDRAW places the mother ship on the screen, while SHIPMOVE bounces it around. DARKEN checks to see if the background is lit up from an explosion. It progressively darkens the area, so the explosions can continue while other things happen.

SHOOTBACK causes the mothership to launch any unused fireballs at your fighter. ALIGN determines the direction the fireballs have to move to reach your ship and how far they will move each time (based on the difficulty level). BALLMOVE moves fireballs in the appropriate direction. HITBALL figures out which fireball was hit and removes it from the screen, making an explosion and lighting up the background.

ENDGAME displays the end-of-game message and restarts the game when you hit the fire button. BLOWNAWAY checks to see if a fireball has hit your fighter and blows it apart if one has.

Finally, MAIN does the initial setup and calls each of the other procedures as needed. One popular misconception is that the "driving" PROC must be called MAIN. This is not true; the procedure may be called anything. What's different about MAIN is that it is the PROCedure which calls all the others.

Action! is a very nice midpoint between BASIC and assembly—and, as you can probably tell, I'm a big fan. Programming in Action! is more fun than in BASIC, with far better results. And it's much easier than learning assembly language.

Listing 1.
Action! listing.

```
;        CHECKSUM DATA
;[E9 50 29 34 E5 0E ED 98
; C0 79 81 0C 97 C2 39 01
; 40 0B 36 1C 52 FF 63 91
; 11 AE 88 3A 75 2B E0 7D
; 47 BD 52 23 D2 71 86 9A
; D6 D5 1D E1 65 60 86 47
; D9 64 A9 70 A1 2F BC 97
; F6 90 64 7D 8B 76 C9 E0
; 54 32 97 68 20 1A 37 FB
; 5A 59 ]

MODULE; SURFACE.RUN
DEFINE RTI="$40",
       PHA="$48",
       PLA="$68",
       TXA="$8A",
       TAX="$AA",
       TYA="$98",
       TAY="$A8",

       SAVETEMPS="[$A2 7 $B5 $A8 $48
                  $CA $10 $FA]",

       GETTEMPS ="[$A2 0 $68 $95 $A8
                  $E8 $E0 8 $D0 $F8]"

CARD OLD,SDLST=560,VDSLST=512,
     SCRLOC=88,CDTMV2=$21A,
     CDTMA2=$228,HIMEM=$2E5,
     PM_BASEADR,ADRES,ADRESB,
     SCORE=[0]

INT  SX=[1],SY=[1]

INT ARRAY BXDR=[0 0 0 0],
          BYDR=[0 0 0 0]

BYTE NMIEN=$D40E,COLBK=$D01A,T=$DA,
     VCOUNT=$D40B,WSYNC=$D40A,
     COUNT=[0],
     PMHITCLR=$D01E,DMACTL=$22F,
     GRACTL=$D01D,PMBASE=$D407,
     PRIORITY=$26F,X0,Y0,COLWND=$D018,
     SHIPX,SHIPY,SHIPSTAT=[0],
     PCLR0=704,PCLRM=711,COLR0=708,
     COLR1=709,COLR2=710,COLR4=712,
     FATE=53770,NUMSHIP=[4],CURSH=752,
     TXTROW=656,TXTCOL=657,LVL=[10],
     LVL1=[10],SND1=$D20F,SND2=$D208

BYTE ARRAY DLIST,YLOCL(80),
     YLOCH(80),RSH2(160),
     PMHPOS(8)=$D000,
     PMVPOS(8)=[0 0 0 0 0 0 0 0],
     PM_WIDTH(5)=$D008,PLPTR,
     PM_MISMASK(4)=[$FC $F3 $CF $3F],
    BALL1=[0 0 0 0 165 90 36 90 90 36
          90 165 0 0 0 0],
    PCOLR(4)=704,
    BALL2=[0 0 0 0 90 165 219 165 165
          219 165 90 0 0 0 0],
    BSTAT=[0 0 0 0],
    BX=[0 0 0 0],BY=[0 0 0 0],
    BLANK=[0 0 0 0 0 0 0 0 0 0 0 0 0
          0 0 0]
BYTE ARRAY CLRS(0)=[64 66 68 70 72 74
     64 66 68 70 72 74 64 66 68 70 72
     74 64 66 68 70 72 74 64 66 68]

BYTE ARRAY BM(0)=[$C0 $30 $C $3],
           CM(0)=[$0 $55 $AA $FF],
  SHIPSHAPE(0)=[0 0 0 0 66 36 24 165
                231 165 24 36 66 0 0
                0 0],
  MSTATUS(0)=[0 0 0 0],
  MX(0)=[0 0 0 0],
  MY(0)=[0 0 0 0],
  MXOLD(0)=[0 0 0 0],
  MYOLD(0)=[0 0 0 0],
SHIP(100)=
['♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'@'♥'|'▜'X'█'♥
 '├'♥'♥'@'P'♥'➔'Ѵ'♡'┐'├'♥'♥'A'ŧ'U'▶'◩
 'U'→'A'♥'♥'W'◩'j'▶'◩'⁆'▶'⊔'♥'♥'W'◩'j
 '▶'◩'⁆'▶'⊔'♥'♥'A'ŧ'U'▶'◩'U'→'A'♥'♥'@
 'P'♥'➔'Ѵ'♡'┐'├'♥'♥'@'♥'▐'▜'X'█'♥'├'♥
 '♥'♥'♥'♥'♥'♥'♥'♥'♥'♥],
;BYT=10:LIN=10
NOLEFT(100)=
['♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'|'▜'X'█'♥
 '├'♥'♥'♥'♥'♥'➔'Ѵ'♡'┐'├'♥'♥'♥'┤'U'▶'◩
 'U'→'A'♥'♥'♥'├'j'▶'◩'⁆'▶'⊔'♥'♥'♥'┤'j
 '▶'◩'⁆'▶'⊔'♥'♥'♥'┌'U'▶'◩'U'→'A'♥'♥'♥
 '♥'♥'➔'Ѵ'♡'┐'├'♥'♥'♥'♥'▐'▜'X'█'♥'├'♥
 '♥'♥'♥'♥'♥'♥'♥'♥'♥'♥],
;BYT=10:LIN=10
NOENG(100)=
['♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'♥'|'▜'X'█'♥
 '♥'♥'♥'♥'♥'♥'➔'Ѵ'♡'♥'♥'♥'♥'♥'┤'U'▶'◩
 'U'D'♥'♥'♥'♥'├'j'▶'◩'⁆'♣'♥'♥'♥'♥'┤'j
 '▶'◩'⁆'♣'♥'♥'♥'♥'┌'U'▶'◩'U'D'♥'♥'♥'♥
 '♥'♥'➔'Ѵ'♡'♣'♥'♥'♥'♥'♥'▐'▜'X'█'♥'♥'♥
```

```
 '♥'♥'♥'♥'♥'♥'♥'♥'♥]

BYTE ARRAY LINE
BYTE LOW=LINE,HIGH=LINE+1

PROC DLINT()
BYTE DUM
[PHA TXA PHA TYA PHA]
IF VCOUNT>94 THEN
 WSYNC=1
 COLBK=0 COLWND=0
 ELSE DUM=CLRS(COUNT)
 WSYNC=1
 COLBK=DUM
FI
COUNT=COUNT+1
IF COUNT=27
THEN COUNT=0
FI
[PLA TAY PLA TAX PLA RTI]

PROC INIT7()
BYTE LOW1,HIGH1,I
CARD SCREEN=LOW1
GRAPHICS(7)
COLR0=44 COLR1=102
COLR2=52 COLR4=0
SCREEN=SCRLOC
I=0
WHILE I<80 DO
YLOCL(I)=LOW1
YLOCH(I)=HIGH1
SCREEN=SCREEN+40
I=I+1
OD
I=0
WHILE I<160 DO
RSH2(I)=I RSH 2
I=I+1
OD
RETURN

PROC DLSETUP()
BYTE I
INIT7()
NMIEN=$40
DLIST=SDLST
VDSLST=DLINT
FOR I=30 TO 40
DO DLIST(I)=141 OD
FOR I=42 TO 54 STEP 2
DO DLIST(I)=141 OD
FOR I=57 TO 72 STEP 3
DO DLIST(I)=141 OD
FOR I=76 TO 84 STEP 4
DO DLIST(I)=141 OD
NMIEN=$C0
RETURN

PROC ROTATE()
BYTE HOLD,CTR,CNTR
[PHA TXA PHA TYA PHA]
SAVETEMPS
HOLD=CLRS(26)
FOR CTR=0 TO 25
DO CNTR=25-CTR
CLRS(CNTR+1)=CLRS(CNTR) OD
CLRS(0)=HOLD
CDTMV2=2
GETTEMPS
[PLA TAY PLA TAX PLA]
RETURN

INT FUNC HSTICK(BYTE PORT)
BYTE ARRAY PORTS(4)=$278
INT ARRAY VALUE(4)=[0 1 $FFFF 0]
RETURN (VALUE((PORTS(PORT)&$C) RSH 2))
INT FUNC VSTICK(BYTE PORT)
BYTE ARRAY PORTS(4)=$278
INT ARRAY VALUE(4)=[0 1 $FFFF 0]
RETURN (VALUE(PORTS(PORT)&3))

PROC DRAW7(BYTE X,Y,CLR)
BYTE X1=$A0,Y1=$A1,CLR1=$A2
LOW=YLOCL(Y1)
HIGH=YLOCH(Y1)
T=RSH2(X1)
LINE(T)=(((BM(X1&3)!$FF)&LINE(T))%
        (BM(X1&3)&CM(CLR1)))
RETURN

PROC FASTDRAW(BYTE ARRAY PICTURE
        BYTE WIDTH,HEIGHT,XX,YY)
BYTE LCTR1,LCTR2
CARD LCTR3
FOR LCTR1=0 TO HEIGHT-1
DO LOW=YLOCL(YY+LCTR1) HIGH=YLOCH(YY+
       LCTR1)
 LCTR2=XX+WIDTH
 LCTR3=(LCTR1+1)*WIDTH-1
 DO
 LINE(LCTR2)=PICTURE(LCTR3)
 LCTR3==-1
 LCTR2==-1
 UNTIL LCTR2=XX
 OD
OD RETURN

PROC PMGRAPHICS()
ZERO(PMHPOS,8) ZERO(PMVPOS,8)
ZERO(PM_WIDTH,5)
DMACTL=$3E
PM_BASEADR=(HIMEM-$800)&$F800
PMBASE=PM_BASEADR RSH 8
HIMEM=PM_BASEADR+768
PRIORITY==&$C0%17 GRACTL=3
RETURN

CARD FUNC PMADR(BYTE N)
IF N>=4 THEN N=0 ELSE N==+1 FI
RETURN(PM_BASEADR+768+(N*$100))

PROC PMCLEAR(BYTE N)
CARD CTR
BYTE ARRAY PLAYADR
PLAYADR=PMADR(N)
IF N<4 THEN ZERO(PLAYADR,$100)
 ELSE N==-4
FOR CTR=0 TO $100-1
DO PLAYADR(CTR)==&PM_MISMASK(N) OD
FI
RETURN

PROC ERASESHIP()
BYTE LOOPX,LOOPY,LL
LL=SHIPX LSH 2
FOR LOOPY=SHIPY TO SHIPY+10
 DO
  FOR LOOPX=LL TO LL+39
  DO DRAW7(LOOPX,LOOPY,0) OD
 OD
LVL==+2 IF LVL>20 THEN LVL=20 FI
LVL1==+5 IF LVL1>200 THEN LVL1=200 FI
RETURN

PROC WINDOW()
BYTE LOOPS
TXTROW=0 TXTCOL=0 CURSH=1
PRINT("┌──────────────────────────────")
PRINT("──────────┐")
FOR LOOPS=1 TO 2 DO
TXTROW=LOOPS TXTCOL=0 PRINT("|")
TXTCOL=38 PRINT("|")
OD  TXTROW=3 TXTCOL=0
PRINT("└──────────────────────────────")
PRINT("──────────┘")
```

```
TXTROW=1 TXTCOL=5 PRINT("SCORE: ")
TXTCOL=12 PRINTC(SCORE)
TXTCOL=20 PRINT("SHIPS LEFT: ")
FOR LOOP5=1 TO 5 DO TXTCOL=31+LOOP5
IF NUMSHIP>=LOOP5 THEN PRINT("♠")
   ELSE PRINT(" ")
FI OD
RETURN

PROC UPDATE()
BYTE LOOP5
TXTROW=1 TXTCOL=12 PRINTC(SCORE)
RETURN

PROC UPDATESHIP()
BYTE LOOP5
TXTROW=1
FOR LOOP5=1 TO 5 DO TXTCOL=31+LOOP5
IF NUMSHIP>=LOOP5 THEN PRINT("♠")
   ELSE PRINT(" ")
FI OD
RETURN

PROC TESTHIT(BYTE MISSUL)
BYTE MISSLY,MISSLX,XSHIP
IF SHIPSTAT=0 THEN RETURN FI
MISSLY=(MY(MISSUL)-30) RSH 1
MISSLX=MX(MISSUL)-48
XSHIP=SHIPX LSH 2
IF MISSLY<SHIPY+4 OR
   MISSLY>SHIPY+7 THEN RETURN FI
IF SHIPSTAT=1 THEN
 IF MISSLX>XSHIP+9 AND
    MISSLX<XSHIP+15
    THEN SHIPSTAT=2 COLR4=14
    SCORE==+20
    UPDATE()
 FI
 RETURN
FI
IF SHIPSTAT=2 THEN
 IF MISSLX>XSHIP+31 AND
    MISSLX<XSHIP+37
    THEN SHIPSTAT=3 COLR4=14
    SCORE==+20
    UPDATE()
 FI
 RETURN
FI
IF MISSLX>XSHIP+20 AND
   MISSLX<XSHIP+26 THEN
   SHIPSTAT=0 SCORE==+50
   COLR4=14 SOUND(1,COLR4 LSH 4,8,4)
   SOUND (2,0,0,0) ERASESHIP()
   UPDATE()
FI
RETURN

PROC SHIPFLY()
BYTE STCK=632
SOUND (0,Y0,8,2)
IF STCK=15 THEN RETURN FI
X0=X0+HSTICK(0) LSH 1
Y0=Y0+VSTICK(0) LSH 1
IF X0>190 THEN X0=190 FI
IF X0<50 THEN X0=50 FI
IF Y0>170 THEN Y0=170 FI
IF Y0<50 THEN Y0=50 FI
ADRES=PMADR(0)+Y0
MOVEBLOCK(ADRES,SHIPSHAPE,17)
PMHPOS(0)=X0
RETURN

PROC MISSILEFIRE()
BYTE TRIGGER=644,INDX,MASK
IF TRIGGER=1 THEN RETURN FI
;TRIGGER IS NOT 1, SO FIRE!
FOR INDX=0 TO 3 DO
IF MSTATUS(INDX)=0 THEN
   MSTATUS(INDX)=1  MY(INDX)=Y0+6
   MYOLD(INDX)=MY(INDX)
   MX(INDX)=X0
   IF INDX=1 OR INDX=3 THEN
      MX(INDX)=X0+15  FI
   MXOLD(INDX)=MX(INDX)
   MASK=PM_MISMASK(INDX)!$FF
   PLPTR(MY(INDX))==%MASK
   PMHPOS(INDX+4)=MX(INDX)
   EXIT
FI  OD  RETURN

PROC MISSILEMOVE()
BYTE INDX,MASK,DELTA
FOR INDX=0 TO 3   DO
IF MSTATUS(INDX)=1 THEN
  PLPTR(MY(INDX))==&PM_MISMASK(INDX)
  MY(INDX)==-2
  MASK=PM_MISMASK(INDX)!$FF
 IF MYOLD(INDX)-MY(INDX)>44 THEN
  MSTATUS(INDX)=0 SOUND (2,0,0,0)
  ELSE PLPTR(MY(INDX))==%MASK;REDRAW
  DELTA=(MYOLD(INDX)-MY(INDX))/6
  IF INDX=0 OR INDX=2 THEN
   MX(INDX)=MXOLD(INDX)+DELTA
   ELSE MX(INDX)=MXOLD(INDX)-DELTA
  FI
  PMHPOS(INDX+4)=MX(INDX)
  SOUND(2,DELTA LSH 2,10,4)
  TESTHIT(INDX)
 FI
FI   OD RETURN
PROC SHIPDRAW()
BYTE TIME=20
IF SHIPSTAT<>0 OR FATE<250
THEN RETURN FI
SHIPSTAT=1
COLR0=14 COLR1=14 COLR2=14 COLR4=14
SHIPX=RAND(24)+2  SHIPY=RAND(30)+2
FASTDRAW(SHIP,10,10,SHIPX,SHIPY)
TIME=0 DO SOUND(1,100,8,12-TIME RSH 1)
IF TIME=4 OR TIME=8 OR TIME=12 THEN
   SHIPFLY() MISSILEMOVE()
FI
UNTIL TIME=16 OD
WHILE COLR4>0
DO COLR4==-1        COLR2=RAND(250)
   COLR0=RAND(250) COLR1=RAND(250)
   TIME=0 DO UNTIL TIME=2 OD
   SOUND(1,COLR4 LSH 4,8,4)
   SHIPFLY() MISSILEMOVE()
OD
COLR0=44 COLR1=102 COLR2=52
SOUND(1,0,0,0)
RETURN

PROC SHIPMOVE()
IF SHIPSTAT=0 THEN RETURN FI
SHIPX==+SX SHIPY==+SY
IF SHIPX<2 OR SHIPX>28 THEN SX=-SX
 ELSEIF FATE>(255-LVL) THEN SX=-SX FI
IF SHIPY<2 OR SHIPY>55 THEN SY=-SY
 ELSEIF FATE<LVL THEN SY=-SY FI
IF SHIPSTAT=1 THEN
FASTDRAW(SHIP,10,10,SHIPX,SHIPY)
ELSEIF SHIPSTAT=2 THEN
FASTDRAW(NOLEFT,10,10,SHIPX,SHIPY)
ELSE FASTDRAW(NOENG,10,10,SHIPX,SHIPY)
FI
RETURN

PROC DARKEN()
IF COLR4=0 THEN RETURN FI
COLR4==-1 SOUND(1,COLR4 LSH 4,8,4)
IF COLR4=0 THEN SOUND(1,0,0,0) FI
RETURN
```

```
PROC SHOOTBACK()
BYTE LLP
IF SHIPSTAT=0 OR FATE>LVL1
   THEN RETURN FI
FOR LLP=1 TO 3 DO
IF BSTAT(LLP)=0 THEN
  BSTAT(LLP)=1
  BX(LLP)=(SHIPX LSH 2)+68
  BY(LLP)=(SHIPY LSH 1)+34
  PCOLR(LLP)=RAND(15) LSH 4;RND COLOR
  PCOLR(LLP)==+10; LIGHTEN COLOR
  ADRESB=PMADR(LLP)+BY(LLP)
  MOVEBLOCK(ADRESB,BALL1,16)
  PMHPOS(LLP)=BX(LLP)
  EXIT
FI
OD
RETURN

PROC ALIGN()
BYTE LLL,CLUNK=[0]
IF LVL1>50 THEN CLUNK=1
   ELSEIF LVL1>150 THEN CLUNK=2
FI
FOR LLL=1 TO 3 DO
IF BSTAT(LLL)<>0 THEN
  IF BX(LLL)>(X0+4) THEN
     BXDR(LLL)=-2-CLUNK
   ELSEIF BX(LLL)<(X0+4) THEN
     BXDR(LLL)=2+CLUNK
   ELSE BXDR(LLL)=0
  FI
  IF BY(LLL)>(Y0+4) THEN
   BYDR(LLL)=-2-CLUNK
   ELSEIF BY(LLL)<(Y0+4) THEN
   BYDR(LLL)=2+CLUNK
   ELSE BYDR(LLL)=0
  FI
FI
OD
RETURN

PROC BALLMOVE()
BYTE LLP
FOR LLP=0 TO 3 DO
IF BSTAT(LLP)<>0 THEN
 IF BSTAT(LLP)=1 THEN BSTAT(LLP)=2
    ELSE BSTAT(LLP)=1
 FI
 BX(LLP)==+BXDR(LLP)
 BY(LLP)==+BYDR(LLP)
 ADRESB=PMADR(LLP)+BY(LLP)
 IF BX(LLP)<50 OR BX(LLP)>190 OR
    BY(LLP)<34 OR BY(LLP)>182 THEN
    BSTAT(LLP)=0
    MOVEBLOCK(ADRESB,BLANK,16)
 FI
 PMHPOS(LLP)=BX(LLP)
 IF BSTAT(LLP)=1 THEN
    MOVEBLOCK(ADRESB,BALL1,16)
    ELSEIF BSTAT(LLP)=2 THEN
    MOVEBLOCK(ADRESB,BALL2,16)
 FI
FI
OD
RETURN

PROC HITBALL()
BYTE ARRAY MISCOL(3)=$D008
BYTE IND,PLY,DUMMI
FOR IND=0 TO 3 DO
IF MISCOL(IND)>1 THEN MSTATUS(IND)=0
   PLPTR(MY(IND))==&PM_MISMASK(IND)
   DUMMI=MISCOL(IND)
   IF (DUMMI&2)=2 THEN PLY=1
     ELSEIF (DUMMI&4)=4 THEN PLY=2
     ELSE PLY=3
   FI
   ADRESB=PMADR(PLY)+BY(PLY)
   MOVEBLOCK(ADRESB,BLANK,16)
   COLR4=10 SOUND(1,COLR4 LSH 4,8,4)
   BSTAT(PLY)=0 PMHITCLR=1
   SCORE==+10 UPDATE()
FI
OD RETURN

PROC ENDGAME()
BYTE TRIGGER=644
ERASESHIP()
TXTROW=2 TXTCOL=2
PRINT("GAME OVER..PRESS FIRE TO PLAY")
PRINT(" AGAIN")
DO UNTIL TRIGGER=0 OD
NUMSHIP=4 SCORE=0 TXTROW=2 TXTCOL=2
LVL=10 LVL1=10 SHIPSTAT=0
PRINT("                                  ")
PRINT("        ")
TXTROW=1 TXTCOL=12 PRINT("        ")
UPDATE() UPDATESHIP()
RETURN

PROC BLOWNAWAY()
BYTE ARRAY SHIPH(0)=53260
BYTE LQ,TIMER=20
IF SHIPH(0)=0 THEN RETURN FI
PM_WIDTH(0)=0
FOR LQ=0 TO 3 DO
IF MSTATUS(LQ)=1 THEN MSTATUS(LQ)=0
   PLPTR(MY(LQ))==&PM_MISMASK(LQ)
   SOUND (2,0,0,0)
FI
PMCLEAR(LQ) BSTAT(LQ)=1 BX(LQ)=X0
BY(LQ)=Y0 ADRESB=PMADR(LQ)+BY(LQ)
MOVEBLOCK(ADRESB,BALL1,16)
PMHPOS(LQ)=BX(LQ)
PCOLR(LQ)=RAND(15) LSH 4+10
OD
COLR4=14 SOUND(1,COLR4 LSH 4,8,8)
BXDR(0)=2 BYDR(0)=2 BXDR(1)=2
BYDR(1)=-2 BXDR(2)=-2 BYDR(2)=2
BXDR(3)=-2 BYDR(3)=-2
DO
IF BSTAT(0)=0 AND BSTAT(1)=0 AND
   BSTAT(2)=0 AND BSTAT(3)=0 THEN
   EXIT
FI
BALLMOVE()
TIMER=0 DO UNTIL TIMER=3 OD
OD
COLR4=0
SOUND(1,0,0,0) PMHITCLR=1 NUMSHIP==-1
UPDATESHIP()
IF NUMSHIP=0 THEN ENDGAME() FI
X0=120 Y0=170
PM_WIDTH(0)=1 PCOLR(0)=170
ADRES=PMADR(0)+Y0
MOVEBLOCK(ADRES,SHIPSHAPE,17)
PMHPOS(0)=X0
RETURN

PROC MAIN()
BYTE XX,COUNT,TIMER=20
SND1=3 SND2=0
DLSETUP()
PMGRAPHICS()
FOR XX=0 TO 7 DO PMCLEAR(XX) OD
Y0=120 X0=120 PCOLR(0)=170 PCLRM=14
ADRES=PMADR(0)+Y0 PLPTR=PMADR(4)
MOVEBLOCK(ADRES,SHIPSHAPE,17)
PMHPOS(0)=X0 PM_WIDTH(0)=1
WINDOW()
CDTMA2=ROTATE
CDTMV2=2
DO
SHIPDRAW()
SHIPMOVE() SHOOTBACK()
ALIGN()    BALLMOVE()
FOR COUNT=1 TO 3
 DO
 TIMER=0 DO UNTIL TIMER=1 OD
 SHIPFLY() MISSILEFIRE() MISSILEMOVE()
 DARKEN() HITBALL() BLOWNAWAY()
 OD
OD
RETURN
```

●

Spy Plane II

Try saving the Earth ...again.

by Mark Comeau

Mission #2, should you choose to accept it, is to stop the production of the enemy's killer satellites. They're being manufactured at this moment, in the secret enemy base in the Commodore mountains. If production doesn't stop, they'll be launched—and demolish the Earth for sure. The coordinates for the base are listed in your secret agent handbook. Land your **Spy Plane** immediately and get to work!

On your last mission, the enemy had somehow managed to photocopy plans to your top-secret satellites. Cases and cases of the plans will be found now in the caverns of the base. Confiscate as many as you can, but don't let that deter you from your main mission.

Once inside the caverns, look for some small portals. Inside are the factories producing the enemy satellites. Drop a radio-controlled robot into them and maneuver it with your hand-held spy computer. To disable the factory's machines, just unplug them and turn off the water supply. After all the machines have been sabotaged, an exit will appear in the lower right-hand corner.

As part of their protective system, the factories have tubes which emit radioactive mist. The mist is dispersed in straight lines, at irregular intervals. The caverns also contain mist portals, but, here, once the mist hits the ground it spreads out a little. The mist causes death on contact. Avoid it at all costs!

In the caverns are empty tubes which you can use to travel up and down. Do *not* travel off the top of the tubes.

On the first level of **Spy Plane II**, you can use your spy jump boots. These will let you "fall" down to lower surfaces without injury. After the first level, though, the boots will become useless. Any fall will result in death.

There are two factories on each cavern screen. When you've sabotaged the first, it will blow up and disappear. Both of the factories must be destroyed before an exit appears.

Each level of the game has four cavern screens, all of which must be completed before the satellites are produced. On the first level, there's a 500-second time limit. In every succeeding level, the time it takes to produce satellites is decreased by 50 seconds. If the factories are not destroyed and your exit accomplished in time, you'll see the satellites launch and destroy the Earth. When the Earth is destroyed, you lose a life.

The fate of the world rests heavily upon your shoulders. You have only four lives in which to complete your mission, so live them with care.

Running and playing the game.

Type in Listing 1 exactly as it appears. Be careful with the data statements.

Type *RUN*, and the screen will go blank for about 30 seconds. Then the **Spy Plane** will land, and your man will get out. After that, the familiar **Spy Plane** logo will appear. Press the fire button to start the game.

Spy Plane II *continued*

After you press the fire button, the score display will appear. Press the fire button again to get to the first screen. During the score display, if you press START, the computer will end the game.

On the first screen, your man will automatically climb out of the plane and down a tube. Cases of plans (worth 10 points) are located all around him. Each screen is worth 100 points. The destruction of a factory will win you 100 points, while the sabotage of each plug or faucet inside is good for 20 points.

Programming tips.

When the program is run, it turns the screen display off, reading and initializing all the necessary stuff. "Why in the world would you want to turn the Atari's superb graphics display off?" you may ask. Because the initialization process takes a while—anything that can make it speed up is A-OK. When the screen is turned off, the computer is freed from graphics—everything else speeds up.

To do this, just *POKE 559,0*. I even use it to turn the screen off when displaying screens. Instead of a flicker, you get a split-second of black, then a quick display.

If you look at the **Spy Plane II** program, you'll see that every number from *0* to *20* has a *C* in front of it. This is done to conserve memory. The computer has an easier time handling variables than numbers. I saved 2263 bytes by using constants on this program. The variables are defined with an unusually large read-data combination. Look at Lines 2510 and 2520.

With the kind of character-set/player-missile graphics used in this game, everything is displayed in 8*8 squares, in order to make things manageable enough for BASIC. But you'll notice that the man moves around pretty smoothly. If moved in steps of 8, he would skip around and wouldn't look too good. Instead, he moves in steps of 2 until he gets to the 8th pixel, because he has to match the character set graphics display.

The trick is to set two variables to the joystick position to determine the direction of the player. Each direction has a variable X- and Y-step, which is either *2* or *-2*. A FOR. . .NEXT loop from *1* to *4* displays the player/missile character each time, adding the X- and Y-step values. Whoa! Did you catch all that?

Each direction the spy may go must have a bit-mapped graphic stored in a string array. This is so that the player/missile graphics routine can display it nice and quickly. The only drawback is that each graphic has to have a different string. "But that's a little too slow for BASIC," you say. Fear not.

The way to get around it is to put all your player/missile graphics into one string. Use a variable for the string pointer of your intended graphic. It goes in steps of 8, because each character should take up 8 bytes. The pointer is set to whichever graphic you want. All of the data for your player/missile can be read in with one FOR. . .NEXT loop, as in Line 2580.

If you were to have an IF. . .THEN statement for every joystick position, your player/missile wouldn't go very fast at all. I use what's called Boolean algebra. What the heck is that? Well, it's really simple. Here's an example. . .

```
100 S=STICK(0):SX=(S=7 AND X<456)*2-(S=11 AND X>304)*2
```

If the conditions inside the parentheses are met (*S=7* and *X<456*), then the value will be a *1*. If the conditions are not met, the value will be a *0*. If the stick position is a *7* and X is not too high, then it will be multiplied by *2*. This particular expression will give a result of either a *2* or *-2*.

The other Boolean expression used is in Line 110. It's only purpose is to determine what the pointer for the player/missile array will be.

Program breakdown.

Lines 10-70 — A little credit, please!.
Line 80 — Branch to initialization.
Lines 90-230 — Main loop. Movement, etc.
Lines 240-270 — Vaporize case. Make it blow up!
Lines 280-400 — Fall down and/or figure out if it is a fatal fall.
Lines 410-430 — Death. Figure out if it is the last man.
Lines 440-510 — Emit radiation.
Lines 520-530 — Next screen and see if it is the last.
Lines 540-700 — Display score, then display screen and go to main loop.
Lines 710-770 — Walk out of plane.
Lines 780-940 — Small init for factory.
Lines 950-1020 — Main loop for factory.
Lines 1030-1180 — Display factory radiation.
Lines 1190-1200 — Check to see if RC robot is stepping on something harmful.
Lines 1210-1250 — Make RC robot die.
Lines 1260-1290 — Unplug machines.
Lines 1300-1330 — Go back to main loop.
Lines 1340-1510 — Launch satellites and destroy Earth.
Lines 1520-2070 — PRINT #6 values for your graphics screens.
Lines 2080-2230 — Display title screen.
Lines 2240-2350 — Land the **Spy Plane**, and then display the logo.
Lines 2360-2470 — GAME OVER message with high score and last score.
Lines 2480-2520 — Start initializing.
Lines 2530-2560 — P/M mover — by Tom Hudson.
Lines 2570-2590 — Set up data for radiation and exits.
Lines 2600-2660 — Character set initializer — created by Steven Pogatch.
Lines 2670-2690 — Character set init DATA.
Lines 2700-3010 — Character set graphics DATA.
Lines 3020-3060 — P/M mover DATA.
Lines 3070-3120 — P/M graphics DATA.
Lines 3130-3180 — Radiation DATA.
Lines 3190-3200 — Exit DATA.

Okay, the game's up. I hope you get hours of fun from **Spy Plane II.**

---

*Mark Comeau is a self-taught BASIC programmer from Piscataway, New Jersey. This is his fourth program published by **ANALOG Computing**. The original **Spy Plane** appeared in issue 21. His interests include graffiti art, rock & roll music, Atari and video games.*

Variables used.

A	For P/M mover.
C	ATASCII value of the character that the player is on.
D	ATASCII value of the character the player is stepping on.
DF	Flag for death fall.
GTM	Time limit.
GX	GR. 2 horizontal position of player.
GY	GR. 2 verticle position of player.
IT	For READing IT.
L(T)	All the DATA for the radiation.
MEN	Number of men left.
MN$	Graphics data for player/missile.
PMD	Position of MN$ in memory.
S	STICK value.
SC	Current screen number.
SCO	Score.
T	Counter for radiation.
TM	Time left to complete screen.
WX	Horizontal position of factory portal.
X	Horizontal position of player.
Y	Vertical position of player.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
WE 10 REM
NB 20 REM          SPY PLANE
UH 30 REM             II
HR 40 REM
HX 50 REM        BY : Mark Comeau
CU 60 REM        FOR: A.N.A.L.O.G
WK 70 REM
KT 80 GOTO 2510
YV 90 REM ▒▒MAIN LOOP▒▒
AG 100 S=STICK(C0):SX=(S=C7 AND X<456)*C2
   -(S=C11 AND X>304)*C2
MF 110 PS=(S=C7)*C0+(S=C11)*C8+(S=C15)*PS
YS 120 SY=C0:IF C=230 THEN SY=(S=C13 AND
   Y<104 AND D<>71 AND D<>72)*C2-(S=C14 A
   ND Y>C16)*C2:PS=C16
WI 130 FOR IT=C1 TO C4
OA 140 X=X+SX:Y=Y+SY:A=USR(MOVE,C0,PMB,PM
   D+PS,X,Y,C8)
PP 150 NEXT IT
EG 160 TM=TM+C1:GX=(X-304)/C8:GY=(Y-C16)/
   C8:LOCATE GX,GY,C
NI 170 IF C=217 THEN GOSUB 250
ES 180 IF C=71 OR C=72 THEN 370
BM 190 IF C=122 THEN 530
HL 200 IF C=105 THEN 790
VT 210 LOCATE GX,GY+C1,D:IF D=32 AND C<>2
   30 THEN 290
WY 220 T=T+RND(C0)*C2:IF T>C10 THEN GOSUB
    450
LN 230 GOTO 100
HX 240 REM ▒▒VAPORIZE CASE▒▒
WF 250 SCO=SCO+C10:COLOR 203:PLOT GX,GY
YR 260 FOR IT=C14 TO C0 STEP -C1:SOUND C0
   ,IT,C0,IT:NEXT IT
ZC 270 COLOR 32:PLOT GX,GY:SOUND C0,C0,C0
   ,C0:RETURN
ZA 280 REM ▒▒FALL▒▒
RX 290 IF DF=C1 THEN 370
IK 300 FOR IT=GY TO C11:LOCATE GX,IT,C
AM 310 IF C<>32 AND C<>217 THEN 330
MR 320 NEXT IT:GOTO 370
PW 330 IT=IT-C1:FOR Y=Y TO IT*C8+C16:SOUN
   D C0,Y,C14,C4:SOUND C1,Y+C1,C14,C4
WI 340 A=USR(MOVE,C0,PMB,PMD+C16,X,Y,C8)
QO 350 NEXT Y:Y=Y-C1:SOUND C0,C0,C0,C0:SO
   UND C1,C0,C0,C0
LU 360 GOTO 100
ZM 370 FOR Y=Y TO 134
WQ 380 A=USR(MOVE,C0,PMB,PMD+C16,X,Y,C8)
OX 390 SOUND C0,Y+121,C14,C14
KB 400 NEXT Y:SOUND C0,C0,C0,C0
JC 410 REM ▒▒DEATH▒▒
HY 420 WX=-C1:WX2=-C1:TM=C0:MEN=MEN-C1:IF
    MEN<C0 THEN 2370
PG 430 GOTO 550
WF 440 REM ▒▒FIRE LAZER▒▒
NS 450 IF TM>GTM THEN 1350
KY 460 T=((INT(RND(C0)*L(SC)))*C3)+L(SC+C
   4)
EG 470 LX=L(T):LY=L(T+C1):LN=L(T+C2):T=C0
SZ 480 COLOR 107:GOSUB 510:SOUND C0,C2,C4
   ,C14
HN 490 LOCATE GX,GY,C:IF C=107 THEN 370
TI 500 COLOR C20+C12:GOSUB 510:SOUND C0,C
   0,C0,C0:T=C0:RETURN
MS 510 PLOT LX,LY-LN:DRAWTO LX,LY:DRAWTO
   LX-C1,LY:DRAWTO LX+C1,LY:RETURN
FT 520 REM ▒▒NEXT SCREEN▒▒
VN 530 TM=C0:WX=-C1:WX2=-C1:SC=SC+C1:IF S
   C=C5 THEN SC=C1:SCO=SCO+100:DF=C1:FL=C
   1:GTM=GTM-50:GOSUB 550:GOTO 720
CQ 540 REM ▒▒GOTO SCREEN▒▒
IL 550 POKE 77,C0:? #C6;CHR$(125):POSITIO
   N C7,C3:? #C6;SCO:COLOR 108:PLOT C7,C5
   :DRAWTO C7+MEN,C5:POSITION C0,C0
EP 560 A=USR(MOVE,C0,PMB,PMD,C0,C0,C0)
FE 570 ? #C6;"NNNNNNN=>?@VNNNNNNNN";
FV 580 ? #C6;"UUUUUUUUUTUUUUUUUUU";
ZO 590 POSITION C0,C10
PP 600 ? #C6;"NNNNNNN=>?@VNNNNNNNN";
QG 610 ? #C6;"UUUUUUUUUTUUUUUUUUU";
NI 620 IF PEEK(53279)=C6 THEN 2370
BB 630 POKE 708,RND(C0)*255:POKE 710,RND(
   C0)*255:IF STRIG(C0)=C1 THEN 620
QN 640 POKE 708,52:POKE 710,164:POKE 559,
   C0:POSITION C0,C0:ON SC GOSUB 1530,167
   0,1810,1950:COLOR 32
DK 650 IF WX>-C1 THEN PLOT WX,WY
WC 660 IF WX2>-C1 THEN PLOT WX,WY
GN 670 HWX=WX:HWX2=WX2
OT 680 IF WX<C0 OR WX2<C0 THEN PLOT D((SC
   -C1)*C2),D((SC-C1)*C2+C1)
AW 690 IF SF=C1 THEN SF=C0:RETURN
PS 700 POKE 559,46:GOTO 100
SS 710 REM ▒▒WALK OUT OF PLANE▒▒
UQ 720 POKE 559,46:Y=32:FOR X=368 TO 383
KB 730 A=USR(MOVE,C0,PMB,PMD,X,Y,C8)
LZ 740 NEXT X
NB 750 FOR Y=32 TO 55
WQ 760 A=USR(MOVE,C0,PMB,PMD+C16,X,Y,C8)
IG 770 NEXT Y:GOTO 100
KF 780 REM ▒▒FACTORY▒▒
WC 790 POKE 559,C0:POSITION C0,C0
KT 800 ? #C6;"eeeeeeeeeeeeeeeeeeee";
TR 810 ? #C6;"e  CCCCC    CCCCC  e";
AM 820 ? #C6;"exrCCpCCxrrrCCpCC  e";
OF 830 ? #C6;"esrCC CCsrrrCC CC  e";
NA 840 ? #C6;"e  JJRJJ    JJRJJ  e";
PP 850 ? #C6;"ew                 e";
PR 860 ? #C6;"ew                 e";
HO 870 ? #C6;"e  QQRQQ    QQRQQ  e";
OP 880 ? #C6;"esrCC CCsrrrCC CC  e";
BA 890 ? #C6;"exrCCpCCxrrrCCpCC  e";
TQ 900 ? #C6;"e  CCCCC    CCCCC  e";
KW 910 ? #C6;"eeeeeeeeeeeeeeeeeeee";
SX 920 POKE 559,46
RE 930 FOR I=C0 TO C3:SOUND I,(RND(C0)*C5
   )+C10,C8,C2:NEXT I
PL 940 RX=448:RY=96:P=C0:IT=C0
BD 950 REM ▒▒RC ROBOT MOVEMENT▒▒
ZR 960 S=STICK(C0):RX=RX+(S=C7)*C8-(S=C11
```

```
    )*C8:RY=RY+(S=C13)*C8-(S=C14)*C8
AO 970 A=USR(MOVE,C0,PMB,PMD+24,RX,RY,C8)
SA 980 GX=(RX-304)/C8:GY=(RY-C16)/C8:LOCA
   TE GX,GY,C
OV 990 IF C=105 THEN 1310
OP 1000 IF C=243 THEN COLOR 239:PLOT GX,G
   Y:GOSUB 1270
SW 1010 IF C=248 THEN COLOR 237:PLOT GX,G
   Y:SOUND IT,C0,C0,C0:IT=IT+C1:GOSUB 127
   0
TB 1020 TM=TM+C1:IF C<>32 THEN 1200
SX 1030 REM ▇▇▇RADIATION▇▇▇
PX 1040 B=B+RND(C0)*C1:IF B<C10 THEN 960
EL 1050 B=C0:T=INT(RND(C0)*C6)+C1:SOUND C
   0,C14,C2,C14
HG 1060 IF T=C1 THEN LX=C2:MX=C18:LY=C5:M
   Y=C5
KV 1070 IF T=C2 THEN LX=C2:MX=C18:LY=C6:M
   Y=C6
OG 1080 IF T=C3 THEN B=C1:LX=C3
SB 1090 IF T=C4 THEN B=C1:LX=C6
PY 1100 IF T=C5 THEN B=C1:LX=C12
TW 1110 IF T=C6 THEN B=C1:LX=C15
QD 1120 COLOR 107:IF B=C0 THEN PLOT LX,LY
   :DRAWTO MX,MY
QE 1130 IF B=C1 THEN PLOT LX,C5:DRAWTO LX
   ,C6:DRAWTO LX+C1,C6:DRAWTO LX+C1,C5
ZK 1140 LOCATE GX,GY,C:IF C=107 THEN 1220
WU 1150 COLOR 32:IF B=C0 THEN PLOT LX,LY:
   DRAWTO MX,MY
NW 1160 IF B=1 THEN PLOT LX,C5:DRAWTO LX,
   C6:DRAWTO LX+C1,C6:DRAWTO LX+C1,C5
JR 1170 SOUND C0,C0,C0,C0:IF IT=C0 THEN S
   OUND C0,C4,C8,C2
TW 1180 GOTO 960
LK 1190 REM ▇▇▇DEATH?▇▇▇
SG 1200 IF C=243 OR C=248 OR C=237 OR C=2
   39 OR C=242 THEN 960
AM 1210 REM ▇▇▇DEATH▇▇▇
BN 1220 FOR IT=C1 TO C3:SOUND IT,C0,C0,C0
   :NEXT IT:SF=C0
AN 1230 FOR IT=C8 TO C0 STEP -1:POKE 712,
   RND(C0)*255:POKE 707,RND(C0)*255:SOUND
    C0,RND(C0)*255,C4,C14
UA 1240 A=USR(MOVE,C0,PMB,PMD+24,RX,RY+C8
   -IT,IT)
IF 1250 NEXT IT:SOUND C0,C0,C0,C0:POKE 71
   2,C0:POKE 707,C14:WX=-C1:WX2=-C1:GOTO
   420
OR 1260 REM ▇▇▇PLUGS OR FAUCETS▇▇▇
DB 1270 FOR V=C0 TO C10:SOUND C0,V,C0,V+C
   4:NEXT V:SOUND C0,C0,C0,C0
DH 1280 SCO=SCO+C20:P=P+C1:IF P=C8 THEN C
   OLOR 105:PLOT C18,C10
BE 1290 RETURN
GT 1300 REM ▇▇▇GO BACK▇▇▇
HJ 1310 RX=X:RY=Y
QC 1320 X=RX:Y=RY:WX2=WX:WY2=WY:SF=C1:GOS
   UB 550:X=RX:Y=RY:WX=(X-304)/C8:WY=(Y-C
   16)/C8
ON 1330 COLOR 217:PLOT WX,WY:POKE 559,46:
   SCO=SCO+100:GOTO 100
PX 1340 REM ▇▇▇DESTROY EARTH▇▇▇
MU 1350 ? #C6;"K":POSITION C0,C9
YT 1360 ? #C6;"B                AB   ";
JB 1370 ? #C6;"CB  AB     ABABACCBAB";
NS 1380 ? #C6;"CCBACCBABACCCCCCCCCC";
KM 1390 A=USR(MOVE,C0,PMB,PMD,C0,C0,C0)
IV 1400 FOR IT=C0 TO C15:SOUND C0,RND(C0)
   *255,C8,IT
QB 1410 FOR X=200 TO 100 STEP -C10:SOUND
   C1,X,C14,IT:NEXT X
VN 1420 NEXT IT:SOUND C1,C0,C0,C0
BV 1430 FOR IT=C0 TO C19:X=RND(C0)*C19:CO
   LOR 107:PLOT X,C9:PLOT X,C8
GC 1440 FOR Y=C8 TO C0 STEP -C1:COLOR 240
   :PLOT X,Y:COLOR 32:PLOT X,Y+C1:SOUND C
   0,Y+C4,C0,Y+C4
MM 1450 NEXT Y:NEXT IT
QC 1460 FOR IT=C0 TO 50:X=RND(C0)*C19:SOU
   ND C0,C0,C0,C0:LOCATE X,C0,C
JY 1470 IF C=240 THEN SOUND C0,C9,C4,C14:
   COLOR 107:PLOT X,C1:DRAWTO X,C11:COLOR
    32:PLOT X,C1:DRAWTO X,C11
SP 1480 NEXT IT
HJ 1490 SOUND C0,C0,C0,C0:FOR I=C14 TO C0
    STEP -C1:X=RND(C0)*255:POKE 712,X:POK
   E 710,I:SOUND C0,X,C8,C14
EZ 1500 NEXT I
CP 1510 POKE 712,C0:SOUND C0,C0,C0,C0:GOT
   O 420
PP 1520 REM ▇▇▇SCREEN #1▇▇▇
DM 1530 ? #C6;"B                ABAB  A";
EP 1540 ? #C6;"CB  AB       ACCCCBAC";
XK 1550 ? #C6;"CCBACCBd  fACCCCCCCC";
MU 1560 ? #C6;"CCCCCCCeeefCCCCCCCCC";
GY 1570 ? #C6;"HGHGJHGHGHfHGHGHGHGH";
EY 1580 ? #C6;"fY i  f Y   Y Y Y Yf";
VL 1590 ? #C6;"fGHG  fGHGHGHGHGJHGf";
NB 1600 ? #C6;"f Y   fY iYf Yf    f";
EG 1610 ? #C6;"fHJGHGHG GGf Gf   Yf";
ZX 1620 ? #C6;"f           f  fHGHGH";
UM 1630 ? #C6;"f   Y Y Y Yf  f Y Yz";
GD 1640 ? #C6;"GHGHGHGHGHGHG GHGHGH";
TK 1650 X=384:Y=56:RETURN
QY 1660 REM ▇▇▇SCREEN #2▇▇▇
DP 1670 ? #C6;"  Y f fYiYf f Y Y Yf";
TC 1680 ? #C6;"GHGHf fGJHf fHGJHGHf";
XJ 1690 ? #C6;"fY YfYf   f f     Yf";
MB 1700 ? #C6;"fG HGHfGJHGHGHGHGHGH";
TD 1710 ? #C6;"fY Yf f  Yf Y Y Yf  ";
NK 1720 ? #C6;"fHGHf f  HfGHGHJGfYf";
KZ 1730 ? #C6;"fY Yf f   f     HHGf";
HW 1740 ? #C6;"HGHGfYf   fY Y  Yi f";
LE 1750 ? #C6;" Y YfGHGHGHGHGH GHGf";
FX 1760 ? #C6;"GHGHf            f";
RR 1770 ? #C6;"Y Y f Y Y zY Y    Yf";
UH 1780 ? #C6;"HGHGHGHGHGHGHGHGHGHG";
XT 1790 X=304:Y=C16:RETURN
RF 1800 REM ▇▇▇SCREEN #3▇▇▇
JO 1810 ? #C6;"f Y Y f Y Yf  f Y Yf";
XG 1820 ? #C6;"fGHGH fGJGHf  fHGJHf";
VP 1830 ? #C6;"fY Y YfY Y fY ff   f";
XA 1840 ? #C6;"fGHGHGHG GHGJHGfY Yf";
AX 1850 ? #C6;"fY Yf         fG HG";
VP 1860 ? #C6;"fGHGf      f    f   ";
LD 1870 ? #C6;"f   HHHGHGfY  Yf   f";
AK 1880 ? #C6;"ff ifY Y YfG HGHGHGf";
EG 1890 ? #C6;"Gf GfGGGGGG     f i f";
VT 1900 ? #C6;"  f  fZf        fGHGH";
WN 1910 ? #C6;"YfY fGfY Y    Yf Y Y";
TA 1920 ? #C6;"HGHGH GHGHGHGHGHGHGH";
ZL 1930 X=328:Y=C16:RETURN
SO 1940 REM ▇▇▇SCREEN #4▇▇▇
AM 1950 ? #C6;"f YiY f Y Y f  z Y f";
HN 1960 ? #C6;"fGGGG fGGGGGf  GGGGf";
PY 1970 ? #C6;"fY Y ffY  Y f Yf Y f";
XS 1980 ? #C6;"fGJG fGG  GGGJGf GGf";
OM 1990 ? #C6;"fY fYfY Y fY   ff Yf";
FY 2000 ? #C6;"fG fGGG GGfGGGGGf Gf";
AV 2010 ? #C6;"f  fY f  ffY Y Yf Yf";
SE 2020 ? #C6;"fY GGGf  fGGGGGGG Gf";
KO 2030 ? #C6;"fG Y Yff f YiY f Y f";
XV 2040 ? #C6;"f  GJGGf GGJGGGf GGf";
GK 2050 ? #C6;"f     Yf Y   Y f Y f";
FP 2060 ? #C6;"GGGGGGGG GGGGGGGGGGG";
WP 2070 X=312:Y=C16:RETURN
QQ 2080 REM ▇▇▇TITLE SCREEN▇▇▇
UF 2090 GRAPHICS C18:POKE 559,C0:POKE 756
   ,PEEK(106)+C1:POKE 704,C14:POKE 708,C0
   :POKE 709,C15:POKE 710,C0:POKE 711,52
JY 2100 ? #C6;"NNNNNNN=>?@VNNNNNNNN";
SB 2110 ? #C6;"UUUUUUUUUTUUUUUUUUUU";
PO 2120 ? #C6;"NNNNNNN=>?@VNNNNNNNN";
XR 2130 ? #C6;"UUUUUUUUUTUUUUUUUUUU";
KK 2140 ? #C6;"NNNNNNN=>?@VNNNNNNNN";
```

```
SN 2150 ? #C6;"UUUUUUUUUTUUUUUUUUUU";
VH 2160 ? #C6;"bNNNNNN=>?@VNNNNNNab";
WV 2170 ? #C6;"cbabUUUUUTUUUUUUUacc";
YJ 2180 ? #C6;"ccccb          abaccc";
TV 2190 ? #C6;"cccccb        accccccc";
ID 2200 ? #C6;"cccccccb    Faccccccc";
FD 2210 ? #C6;"cccccccceeeeFcccccccc";
XH 2220 POKE 559,46:B=C10:Y=C0
FB 2230 IF STRIG(C0)=C0 OR S=0.1 THEN 227
   0
MV 2240 REM ▒▒▒LAND▒▒▒
KT 2250 FOR I=C0 TO 33:B=B*0.91:Y=Y+B:A=U
   SR(MOVE,C0,PMB,PMD+32,360,Y,C8)
DS 2260 SOUND C0,Y*2.6,C14,C14:SOUND C1,Y
   *2.6,C14,C14:SOUND C1,Y,C8,C14:NEXT I
QP 2270 COLOR 100:PLOT C7,C10:A=USR(MOVE,
   C0,PMB,PMD,C0,C0,C0)
ZL 2280 SOUND C1,C0,C0,C0:SOUND C0,C0,C0,
   C0
JH 2290 FOR I=C0 TO 100:NEXT I
AH 2300 SOUND C0,100,C13,C14:SOUND C0,C0,
   C0,C0:COLOR 108:PLOT C8,C10
RM 2310 FOR I=C0 TO C14:POKE 708,I:POKE 7
   10,I:FOR B=C0 TO C10:NEXT B:NEXT I:IT=
   C0:S=0.1
SU 2320 POKE 708,RND(0)*255:POKE 710,RND(
   C0)*255
TA 2330 IF STRIG(C0)=C0 THEN 2440
KE 2340 IT=IT+C1:IF IT>250 AND SC>C0 THEN
    2370
QF 2350 GOTO 2320
WT 2360 REM ▒▒▒GAME OVER▒▒▒
UO 2370 IF HI<SCO THEN HI=SCO
KK 2380 A=USR(MOVE,C0,PMB,PMD,C0,C0,C0)
EM 2390 GRAPHICS 18:? #C6;"     GAME OVER"
   :? #C6:? #C6;"       score":? #C6;"
    ";SCO:? #C6:? #C6;"    high score"
ZP 2400 ? #C6;"       ";HI:IT=C0:POKE 709,
   C14:POKE 708,100
DP 2410 IT=IT+C1:POKE 710,RND(C0)*255
IN 2420 IF STRIG(C0)=C0 OR IT>100 THEN 20
   90
QA 2430 GOTO 2410
TF 2440 IF STRIG(C0)=C0 THEN 2440
LN 2450 POKE 708,52:POKE 710,164:POKE 711
   ,70
UP 2460 SC=C1:WX=-C1:WX2=WX:SCO=C0:MEN=C3
   :DF=C0:SF=C1:TM=C0:GTM=500
IR 2470 GOSUB 550:GOTO 720
RV 2480 REM ▒▒▒INITALIZATION▒▒▒
JG 2490 REM
XR 2500 REM ▒▒▒P/M MOVER▒▒▒
LG 2510 READ C0,C1,C2,C3,C4,C5,C6,C7,C8,C
   9,C10,C11,C12,C13,C14,C15,C16,C17,C18,
   C19,C20
EJ 2520 DATA 0,1,2,3,4,5,6,7,8,9,10,11,12
   ,13,14,15,16,17,18,19,20
SB 2530 DIM PMMOV$(100),MN$(40),XFR$(38),
   L(50),D(C7)
HN 2540 POKE 559,C0:POKE 712,52:MOVE=ADR(
   PMMOV$):RESTORE 3030:FOR B=C1 TO 100:R
   EAD IT:PMMOV$(B)=CHR$(IT):NEXT B
GR 2550 FOR B=C1 TO 40:READ IT:MN$(B)=CHR
   $(IT):NEXT B
BW 2560 PMBASE=INT((PEEK(145)+C3)/C4)*4:P
   OKE 54279,PMBASE:PMB=PMBASE*256:PMD=AD
   R(MN$):POKE 53277,C3
JO 2570 REM ▒▒▒DATA SETUP▒▒▒
CQ 2580 RESTORE 3140:FOR B=C1 TO 50:READ
   IT:L(B)=IT:NEXT B
II 2590 FOR B=C0 TO C7:READ IT:D(B)=IT:NE
   XT B
JY 2600 REM ▒▒▒Char.Set INIT▒▒▒
MC 2610 POKE 712,190:POKE 106,PEEK(106)-C
   5:START=(PEEK(106)+C1)*256:POKE 756,ST
   ART/256
TR 2620 RESTORE 2680:FOR B=C1 TO 38:READ
   IT:XFR$(B)=CHR$(IT):NEXT B
PV 2630 A=USR(ADR(XFR$)):B=232:READ IT
HO 2640 IF IT=-C1 THEN 2090
WO 2650 FOR Y=C0 TO C7:POKE B+Y+START,IT:
   READ IT:NEXT Y
XV 2660 B=B+C8:GOTO 2640
VG 2670 REM ▒▒▒DATA→Char.Set Init▒▒▒
QA 2680 DATA 104,169,0,133,203,133,205,16
   9,224,133,206,165,106,24,105,1,133,204
   ,160,0,177,205,145,203,200,208
TT 2690 DATA 249,230,204,230,206,165,206,
   201,228,208,237,96
PC 2700 REM ▒▒▒Char.Set SHAPE DATA▒▒▒
NU 2710 DATA 120,248,195,242,122,27,251,2
   43
SC 2720 DATA 0,0,172,172,172,188,24,24
UY 2730 DATA 120,108,109,109,121,97,97,97
CK 2740 DATA 0,0,157,149,149,157,213,213
EJ 2750 DATA 0,5,5,23,23,95,95,255
FI 2760 DATA 0,64,192,208,248,250,254,255
MV 2770 DATA 255,255,255,255,255,255,255,
   255
WK 2780 DATA 0,0,0,192,118,63,112,192
EZ 2790 DATA 255,17,255,136,255,17,255,13
   6
AW 2800 DATA 129,129,195,195,129,129,195,
   195
UL 2810 DATA 255,153,255,255,239,170,34,0
TD 2820 DATA 255,153,255,255,221,213,69,0
IE 2830 DATA 0,0,24,60,52,60,60,60
WO 2840 DATA 255,153,255,255,219,24,126,1
   26
MI 2850 DATA 84,130,37,74,145,36,80,9
VV 2860 DATA 24,52,24,58,92,24,100,70
ZA 2870 DATA 96,32,112,255,255,112,0,0
UE 2880 DATA 0,0,0,0,0,0,255,255
NR 2890 DATA 0,56,252,63,63,252,56,0
EP 2900 DATA 129,90,60,126,126,60,90,129
BG 2910 DATA 126,126,24,219,255,255,153,2
   55
ME 2920 DATA 0,0,0,255,255,0,0,0
MH 2930 DATA 0,224,240,255,255,240,224,0
JO 2940 DATA 0,126,126,36,36,36,126,126
HP 2950 DATA 0,255,255,0,0,0,0,0
AS 2960 DATA 0,0,215,214,215,246,119,119
KR 2970 DATA 128,240,131,255,255,131,240,
   128
YA 2980 DATA 48,32,112,255,255,112,0,0
HU 2990 DATA 0,0,0,60,36,126,126,126
DH 3000 DATA 60,231,66,195,66,195,66,195
EJ 3010 DATA -1
AY 3020 REM ▒▒▒DATA FOR P/M MOVER▒▒▒
MA 3030 DATA 216,104,104,104,133,213,104,
   24,105,2,133,206,104,133,205,104,133,2
   04,104,133,203,104,104,133,208
BM 3040 DATA 104,104,133,209,104,104,24,1
   01,209,133,207,166,213,240,16,165,205,
   24,105,128,133,205,165,206,105
TM 3050 DATA 0,133,206,202,208,240,160,0,
   162,0,196,209,144,19,196,207,176,15,13
   2,212,138,168,177,203,164
UK 3060 DATA 212,145,205,232,169,0,240,4,
   169,0,145,205,200,192,128,208,224,166,
   213,165,208,157,0,208,96
BD 3070 REM ▒▒▒P/M SHAPE DATA▒▒▒
VM 3080 DATA 24,52,24,58,92,24,100,70
JQ 3090 DATA 24,44,24,92,58,24,38,98
SP 3100 DATA 24,36,24,60,90,24,36,102
PH 3110 DATA 4,132,100,164,60,126,165,126
QH 3120 DATA 0,0,192,118,63,112,192,0
NH 3130 REM ▒▒▒LAZER DATA▒▒▒
EQ 3140 DATA 3,4,3,4,9,18,30,39
RY 3150 DATA 2,10,1,4,7,2,16,8,1
CA 3160 DATA 8,2,0,15,2,0,8,7,3,15,10,4
CC 3170 DATA 8,5,3,12,10,6,17,6,4
OW 3180 DATA 2,10,6,4,10,0,11,10,0,13,4,0
QD 3190 REM ▒▒▒EXIT DATA▒▒▒
UU 3200 DATA 19,10,10,10,5,9,15,0
```

Make the connection!

ANALOG Computing on Delphi puts you on-line with the world.

Delphi, an on-line, full-service information network, offers news and sports from the Associated Press, weather reports, movie reviews, shopping services, travel information, and more.

ANALOG Computing, the #1 magazine for Atari owners, brings you the Atari Users' Group on Delphi. We offer a message forum and an extensive database for up- or downloading—all from as little as 10 cents per minute from most U.S. cities, with no additional telephone charges and no extra charge for 1200 or 2400 bps. We'll use the group's conference feature for electronic meetings with well known Atarians and, of course, **ANALOG** staff. Bring on your toughest questions!

Special rates

Subscribers to **ANALOG Computing** or **ST-Log** may join without charge, and will receive a free lifetime Delphi membership, a Delphi Command Card and $5.00 of line-time credit applicable to their account. If you purchase the *Delphi Handbook*—the highly detailed manual on using the whole Delphi system—for $29.95, you will get an additional $20.00 of line-time credit. And you can subscribe to either **ANALOG Computing** or **ST-Log** directly, while on-line, to be eligible for these bonuses.

How to connect

First, select a data communications network: Telenet or Tymnet (in the U.S.), or DataPac (in Canada). In the Boston area, dial Delphi direct (617-576-0862). To determine your local Telenet number, dial 800-TELENET or 703-689-5700 (in Alaska, 907-264-7391). To obtain a Tymnet number, call 800-336-0149. If you have difficulty, call Delphi at 800-544-4005 (in Massachusetts, 617-491-3393). Current subscribers to **ANALOG Computing** or **ST-Log** should type *JOINATARI* when asked for user name. When asked for a password, type *ANALOG*. Those who wish to subscribe to either magazine on-line should, instead, type *SUBSCRIBE* at the password prompt. Once on Delphi, you'll find our group on the "Groups & Clubs" menu. To get there, just type *GR ATARI* from the main menu prompt.

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

P.O. BOX 23, WORCESTER, MA 01603

48K Cassette or Disk

GAME

Reversi in Action!

by Paul T. Sprague

Reversi is a strategy game written in Action!, a wonderful language from OSS. It's not only very fast in compilation and execution, but also has the best editor I've ever seen. Action! makes it possible to write games such as this one in a high-level language—and yet still be able to realize the speed of assembly language (or very close to it).

The rules of **Reversi** are quite easy to grasp. The board starts out with two white pieces and two black pieces in the center (as you'll see when you start up the program). White moves first, then black, then white, etc. . . . until all squares are taken up, or neither player can move.

A move consists of placing your piece on an empty square, thereby capturing all your opponent's pieces between your played piece and another piece of your color. Your pieces must be flanking those of the opponent, with no squares left empty between the pieces.

These captures may take place horizontally, vertically, or diagonally. Also, you may capture pieces in more than one direction in a single move (even in all eight directions).

The pieces thus taken become your color; so ends your turn.

One important point: you must capture at least one piece in order to make a legal move. If you can't do this, you must pass and allow your opponent to move again.

The winner is the player with the most pieces of their color on the board when the game ends.

That's all you need to know to play **Reversi**. The rules may seem quite simple, but, the more you play, the more strategies you find which are important for good play.

The fine points.

This **Reversi** program allows for three different modes of operation. A menu of these appears after the board has been drawn at the beginning of the game.

The computer will ask you to choose a playing mode: 1 for computer vs computer, 2 for human vs computer and 3 for human vs human. Pressing either *1*, *2*, or *3* at this point will select the appropriate mode. Note that you don't need to (nor should you) press RETURN after entering the number.

In mode 1 (computer vs computer), you'll be asked to select the strategy level for the white and black sides. The game will then begin, and you'll see white and black exchanging moves on-screen until the game's over. This probably isn't really helpful in learning game strategies, but it *is* quite interesting to watch.

Mode 2 (human vs computer) first prompts you to select the color (white or black) you wish to play. To do this, simply press *W* for white or *B* for black. (The computer automatically plays the opposite color; it never argues—well, almost never.) Once colors are selected and you choose the skill level of the computer (more on this later), play begins. If you're white, you'll go first. Otherwise, the

computer will make the first move. Regardless of which color you pick, you'll always use joystick 1 in this mode.

Mode 3 (human vs human) allows you to play against a friend. In this mode, joystick 1 is the white player and joystick 2 is the black.

To move, you must have a joystick plugged into the correct port. The cursor appears on-screen and may be moved around via joystick. Place the cursor on the square to which you wish to move and press the fire button. If the square is a legal one for your move, a piece of your color will appear there, while all pieces which your move captured will be changed by the computer. If you have no legal move, then you must forfeit your turn by pressing *P* (Pass) on the keyboard.

At the end of each game, the computer will ask whether or not you'd like to play another game. If you want to play again, press the *Y* key. This will cause the game board to be reinitialized and the starting menu to appear.

As mentioned above, in each case where the computer plays one or both sides, it will ask you to select a skill level for each color. Here are the basic strategies for each level.

Good: The second level, using the least strategy of the three, plays simply for capture of the most pieces. This is the way most beginners play. Soon, however, it becomes evident that more thought is necessary.

Better: The second level combines the previous method with a knowledge of which squares are better to hold. The map of numbers you see at the beginning of the program (Listing 1) accomplishes this. However, in this level the map is static (it doesn't change as the game progresses).

Best: Our third level also uses the map, but has map updates in special cases, to account for possible changes in the strategic value of a square. Although, in play against humans, this level seems quite a bit better than the second, when the two levels are played head-to-head, the difference is not particularly evident. The third level seems to win a majority of the time—but not a large majority, by any means. Another interesting change in this level's strategy is that, for the first part of the game, it doesn't try to capture the most pieces, but the least! This may seem backwards, but usually plays well. See if you can figure out why.

Here's a quick summary of each function and procedure in the program.

SET_CHIP: Places a piece of the current color into the board array at XC,YC.

TEST_SQR: Returns the value of the square XC,YC in the board array.

PLACE_CHIP: Places a piece of the current color into the board array at XC,YC and draws it on the screen board.

PSCORE: Switches inverse lettering to the current player color and prints the score.

GET_LEVEL: Inputs strategy level.

INITIALIZE: Sets up screen and array board, gets mode and levels, sets initial score and prints it.

FLIPPER: If *FLIP_FLAG=0*, then count the number of chips captured by the move XC,YC. If *FLIP_FLAG=1*, then actually capture the chips for the move XC,YC.

UPDATE_VALUES: If a move is made to a corner, then make the squares adjacent to the move valuable.

COMPUTER: Get a computer move.

PLAYER: Get a human player move.

MAKE_MOVE: As the name implies. . .

MAIN: The primary game loop, with end-of-game checking.

I hope that some of you will look at the code, figure out how the strategies work and try to come up with stronger ones. It really is fun to program a strategy, then pit it against one of the other strategies. If you come up with a really good one, or you have any questions or comments, please write to **Reader Comment** in the pages of **ANALOG Computing**.

Good luck. Hope your life is filled with lots of Action!

Paul T. Sprague has his bachelor of science degree in Electrical Engineering and works as an Associate Engineer of Design and Development for Raytheon. He's had his Atari 800 for seven years and Action! for two and one-half. They make a great pair!

Listing 1.
Action! listing.

```
; REVERSI in Action!
; Written by Paul T. Sprague
;
;       CHECKSUM DATA
;[A8 A8 F6 7E 6B 8B CA 8A
; 6D 02 1D F0 F3 49 A6 9A
; C1 E1 E3 44 20 99 F5 50
; 30 D8 B3 E9 B9 83 64 6D
; 97 61 DA 34 AD 0E 84 5B
; DA 61 EB ]

BYTE WHITE_SCORE,BLACK_SCORE,
     KEY=$2FC,CURSOR=$2F0,ATTRACT=77,
     PRO_COLOR,OPP_COLOR,
     MOVEX,MOVEY,MOVE,JOYX,JOYY
BYTE ARRAY FRESH_BOARD(128) =

     [   15  0  6  6  6  6  0 15
          0  0  1  1  1  1  0  0
          6  1  4  2  2  4  1  6
          6  1  2  0  0  2  1  6
          6  1  2  0  0  2  1  6
          6  1  4  2  2  4  1  6
          0  0  1  1  1  1  0  0
         15  0  6  6  6  6  0 15

         15  0  6  6  6  6  0 15
          0  0  1  1  1  1  0  0
          6  1  4  2  2  4  1  6
          6  1  2  0  0  2  1  6
          6  1  2  0  0  2  1  6
          6  1  4  2  2  4  1  6
          0  0  1  1  1  1  0  0
         15  0  6  6  6  6  0 15    ] ,

     VALUE_BOARD(128) ,
     BOARD(64) , WB(2) , LEVEL(2) ,
     WHITE="WHITE" , BLACK="BLACK"

PROC SET_CHIP(BYTE XC,YC)
  BOARD(XC+YC*8)=PRO_COLOR
RETURN
```

```
BYTE FUNC TEST_SQR(BYTE XC,YC)
  RETURN (BOARD(XC+YC*8))

PROC PLACE_CHIP(BYTE XC,YC)
  SET_CHIP(XC,YC)
  XC=(XC+2)*4+17  YC=(YC+1)*4
  COLOR=PRO_COLOR+1
  PLOT(XC,YC)          DRAWTO(XC+2,YC)
  DRAWTO(XC+2,YC+2)    DRAWTO(XC,YC+2)
  PLOT(XC,YC+1)        PLOT(XC+1,YC+1)
RETURN

PROC PSCORE()
  PRINT("↑↑")
  IF PRO_COLOR=1 THEN
    PRINT("          WHITE       ")
    PRINTE("BLACK")
   ELSE
    PRINT("          WHITE       ")
    PRINTE("BLACK")
  FI
PRINTF("             %B          %B  %E"
  ,WHITE_SCORE,BLACK_SCORE)
RETURN

PROC GET_LEVEL ( CHAR ARRAY
            COLOR_STR , BYTE TEMP1 )
  BYTE CHOICE
  DO
    PRINT("↰PRESS NUMBER TO SELECT")
    PRINTF(" %S LEVEL:%E",COLOR_STR)
      PRINTE("  1 - GOOD")
      PRINTE("  2 - BETTER")
      PRINT ("  3 - BEST")
      CHOICE=GETD(7)
    UNTIL (CHOICE>$30)AND(CHOICE<$34)
  OD
    LEVEL(TEMP1)=CHOICE-$30
RETURN
PROC INITIALIZE()
  CHAR TEMP
  BYTE I,J,CHOICE
  JOYX=38  JOYY=17  KEY=255
  CLOSE(7)  OPEN(7,"K:",4,0)
  GRAPHICS(5)       SETCOLOR(4,12,5)
  SETCOLOR(2,0,0)   SETCOLOR(1,0,12)
  SETCOLOR(0,0,8)
  FOR I=0 TO 63 DO
    BOARD(I)=0
  OD
  FOR I=0 TO 127 DO
    VALUE_BOARD(I)=FRESH_BOARD(I)
  OD
  COLOR=1
  FOR I=24 TO 56 STEP 4 DO
    PLOT(I,3)        DRAWTO(I,35)
    PLOT(25,I-21)    DRAWTO(55,I-21)
  OD
  PRO_COLOR=1
    PLACE_CHIP(3,3)  PLACE_CHIP(4,4)
  PRO_COLOR=2
    PLACE_CHIP(3,4)  PLACE_CHIP(4,3)
  DO
    PRINTE("↰PRESS NUMBER TO SELECT:")
    PRINT("  1 - Computer vs. ")
    PRINTE("Computer")
    PRINT("  2 - Human     vs. ")
    PRINTE("Computer")
    PRINT ("  3 - Human     vs. ")
    PRINT("Human      ")
    CHOICE=GETD(7)
  UNTIL (CHOICE>$30)AND(CHOICE<$34) OD
  IF CHOICE=$31 THEN
    WB(0)=2     WB(1)=2
    GET_LEVEL ( WHITE , 0 )
    GET_LEVEL ( BLACK , 1 )
   ELSEIF CHOICE=$33 THEN
    WB(0)=0     WB(1)=1
   ELSE
    DO
      PRINT("↰ WHICH COLOR DO YOU")
      PRINT(" WANT [W/B] ?")
      TEMP=GETD(7)
    UNTIL (TEMP='W)OR(TEMP='B) OD
    IF TEMP='W THEN
      WB(0)=0  WB(1)=2
      GET_LEVEL ( BLACK , 1 )
     ELSE
      WB(0)=2  WB(1)=0
      GET_LEVEL ( WHITE , 0 )
    FI
  FI
  PRINT("↰")  CURSOR=1
  WHITE_SCORE=2  BLACK_SCORE=2
  PRO_COLOR=1 OPP_COLOR=2
  PSCORE()
RETURN

BYTE FUNC FLIPPER(BYTE XC,YC,
                  FLIP_FLAG)
  BYTE TMPX,TMPY,FLIPS,COUNT,FLAG,TEMP
  INT I,J
  FLIPS=0
  IF TEST_SQR(XC,YC)=0 THEN
    FOR J=-1 TO 1 DO FOR I=-1 TO 1 DO
      IF (I#0)OR(J#0) THEN
        TMPX=XC   TMPY=YC
        COUNT=0   FLAG=0
        DO
          TMPX==+I  TMPY==+J
          IF (TMPX<8)AND(TMPY<8) THEN
            TEMP=TEST_SQR(TMPX,TMPY)
            IF TEMP=0 THEN
              FLAG=2
             ELSEIF TEMP=OPP_COLOR
              THEN
              COUNT==+1
             ELSE
              FLAG=1
            FI
           ELSE FLAG=2
          FI
        UNTIL FLAG#0 OD
        IF FLAG=1 THEN
          FLIPS==+COUNT
          IF FLIP_FLAG=1 THEN
            TMPX=XC   TMPY=YC
            FLAG=0
            DO
              TMPX==+I  TMPY==+J
              TEMP=TEST_SQR(TMPX,TMPY)
              IF TEMP=OPP_COLOR THEN
                 PLACE_CHIP(TMPX.TMPY)
               ELSE
                FLAG=1
               FI
             UNTIL FLAG#0 OD
          FI
        FI
      FI
    OD OD
  FI
RETURN (FLIPS)

PROC UPDATE_VALUES()
  IF (MOVEX%MOVEY)=0 THEN
    VALUE_BOARD((PRO_COLOR-1)*64+1)=8
    VALUE_BOARD((PRO_COLOR-1)*64+8)=8
    VALUE_BOARD((PRO_COLOR-1)*64+9)=8
   ELSEIF (MOVEX=0)AND(MOVEY=7) THEN
    VALUE_BOARD((PRO_COLOR-1)*64+48)=8
    VALUE_BOARD((PRO_COLOR-1)*64+49)=8
    VALUE_BOARD((PRO_COLOR-1)*64+57)=8
   ELSEIF (MOVEX=7)AND(MOVEY=0) THEN
    VALUE_BOARD((PRO_COLOR-1)*64+6)=8
    VALUE_BOARD((PRO_COLOR-1)*64+14)=8
```

```
      VALUE_BOARD((PRO_COLOR-1)*64+15)=8
    ELSEIF (MOVEX=7)AND(MOVEY=7) THEN
      VALUE_BOARD((PRO_COLOR-1)*64+54)=8
      VALUE_BOARD((PRO_COLOR-1)*64+55)=8
      VALUE_BOARD((PRO_COLOR-1)*64+62)=8
  FI
RETURN

PROC COMPUTER()
  BYTE BEST,SCORE,COUNT,XC,YC,TEMP
  BYTE ARRAY CHOICEX(19)
  BYTE ARRAY CHOICEY(19)
  BEST=0 COUNT=0
  FOR YC=0 TO 7 DO FOR XC=0 TO 7 DO
    SCORE=FLIPPER(XC,YC,0)
    IF SCORE>0 THEN
      IF LEVEL(PRO_COLOR-1)=2 THEN
        SCORE==+VALUE_BOARD(
           (PRO_COLOR-1)*64+YC*8+XC)
       ELSEIF LEVEL(PRO_COLOR-1)=3
          THEN
        IF WHITE_SCORE+BLACK_SCORE<30
           THEN
          SCORE=(25-SCORE)/3+
             VALUE_BOARD((PRO_COLOR-
             1)*64+YC*8+XC)
         ELSE
          SCORE==+VALUE_BOARD(
             (PRO_COLOR-1)*64+YC*8+XC)
        FI
        IF VALUE_BOARD((PRO_COLOR-1)*
           64+8*YC+XC)=0 THEN
          SCORE=1
        FI
      FI
      IF SCORE=BEST THEN
        CHOICEX(COUNT)=XC
        CHOICEY(COUNT)=YC
        COUNT==+1
       ELSEIF SCORE>BEST THEN
        COUNT=1
        CHOICEX(0)=XC
        CHOICEY(0)=YC
        BEST=SCORE
      FI
    FI
  OD OD
  IF BEST=0 THEN
    MOVEX=8  MOVEY=8
   ELSE
    TEMP=RAND(COUNT)
    MOVEX=CHOICEX(TEMP)
    JOYX=(MOVEX+2)*4+18
    MOVEY=CHOICEY(TEMP)
    JOYY=(MOVEY+1)*4+1
    IF LEVEL(PRO_COLOR-1)=3 THEN
      UPDATE_VALUES()
    FI
  FI
RETURN

PROC PLAYER(BYTE STICK_NUM)
  BYTE TEMP,SX,SY,FLAG,R,I,J
  KEY=255  TEMP=LOCATE(JOYX,JOYY)
  IF TEMP=0 THEN
    COLOR=1
   ELSE
    COLOR=5-TEMP
  FI
  PLOT(JOYX,JOYY)  SX=JOYX SY=JOYY
  DO
    R=STICK(STICK_NUM)
    IF (R&$8)=0 THEN  JOYX==+4  FI
    IF (R&$4)=0 THEN  JOYX==-4  FI
    IF (R&$2)=0 THEN  JOYY==+4  FI
    IF (R&$1)=0 THEN  JOYY==-4  FI
    IF R#15 THEN
      IF      JOYX<26 THEN JOYX=54
       ELSEIF JOYX>54 THEN JOYX=26
      FI
      IF      JOYY<5  THEN JOYY=33
       ELSEIF JOYY>33 THEN JOYY=5
      FI
      POSITION(SX,SY)  PUTD(6,TEMP)
      SX=JOYX  SY=JOYY
      TEMP=LOCATE(JOYX,JOYY)
      IF TEMP=0 THEN
        COLOR=1
       ELSE
        COLOR=5-TEMP
      FI
      PLOT(JOYX,JOYY)
      SOUND(0,200,10,8)
      FOR I=0 TO 200 DO FOR J=0 TO 10
         DO OD OD
      SNDRST()
      FOR I=0 TO 200 DO FOR J=0 TO 50
         DO OD OD
    FI
    FLAG=0
    IF STRIG(STICK_NUM)=0 THEN
      MOVEX=(JOYX-18)/4-2
      MOVEY=(JOYY-1)/4-1
      IF FLIPPER(MOVEX,MOVEY,0)>0 THEN
        FLAG=1
      FI
    FI
    IF KEY=10 THEN
      FLAG=2
      FOR I=0 TO 7 DO FOR J=0 TO 7 DO
        IF FLIPPER(I,J,0)>0 THEN
          FLAG=0
          I=7  J=7
        FI
      OD OD
      KEY=255
      MOVEX=8
      MOVEY=8
    FI
  UNTIL (FLAG#0) OD
  POSITION(SX,SY)  PUTD(6,TEMP)
  KEY=255
RETURN

PROC MAKE_MOVE(BYTE XC,YC)
  BYTE NF
  CARD I
  NF=FLIPPER(XC,YC,0)
  IF PRO_COLOR=1 THEN
    WHITE_SCORE==+NF+1
    BLACK_SCORE==-NF
   ELSE
    BLACK_SCORE==+NF+1
    WHITE_SCORE==-NF
  FI
  NF=FLIPPER(XC,YC,1)
  PLACE_CHIP(XC,YC)
  SOUND(0,80,10,8)
  FOR I=0 TO 800 DO OD
  SOUND(0,0,0,0)
  ATTRACT=0
RETURN

PROC MAIN()
  BYTE PASS
  CHAR TEMP
  DO
    INITIALIZE()
    PSCORE()
    DO
      IF WB(PRO_COLOR-1)=2 THEN
        COMPUTER()
       ELSE
        PLAYER(WB(PRO_COLOR-1))
      FI
      IF MOVEX=8 THEN
```

```
        PASS==+1
       ELSE
        PASS=0
        MAKE_MOVE(MOVEX,MOVEY)
      FI
      PRO_COLOR=OPP_COLOR
      OPP_COLOR=3-OPP_COLOR
      PSCORE()
    UNTIL (WHITE_SCORE+BLACK_SCORE=64)
       OR(PASS=2) OD
    PRINTE(" ")
    IF WHITE_SCORE>BLACK_SCORE THEN
      PRINT("White wins!...")
      ELSEIF BLACK_SCORE>WHITE_SCORE
         THEN
       PRINT("Black wins!...")
      ELSE
       PRINT("Tie!...")
     FI
     PRINT("play again?")
     TEMP=GETD(7)
  UNTIL TEMP='N  OD
RETURN
```

●

Back Issues

All back issues are priced at $4.00 each.

Send your check or money order to
ANALOG Computing Back Issues,
P.O. Box 625, Holmes, PA 19043.
MasterCard and VISA orders,
call 1-800-345-8112
(in Pennsylvania, 1-800-662-2444).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

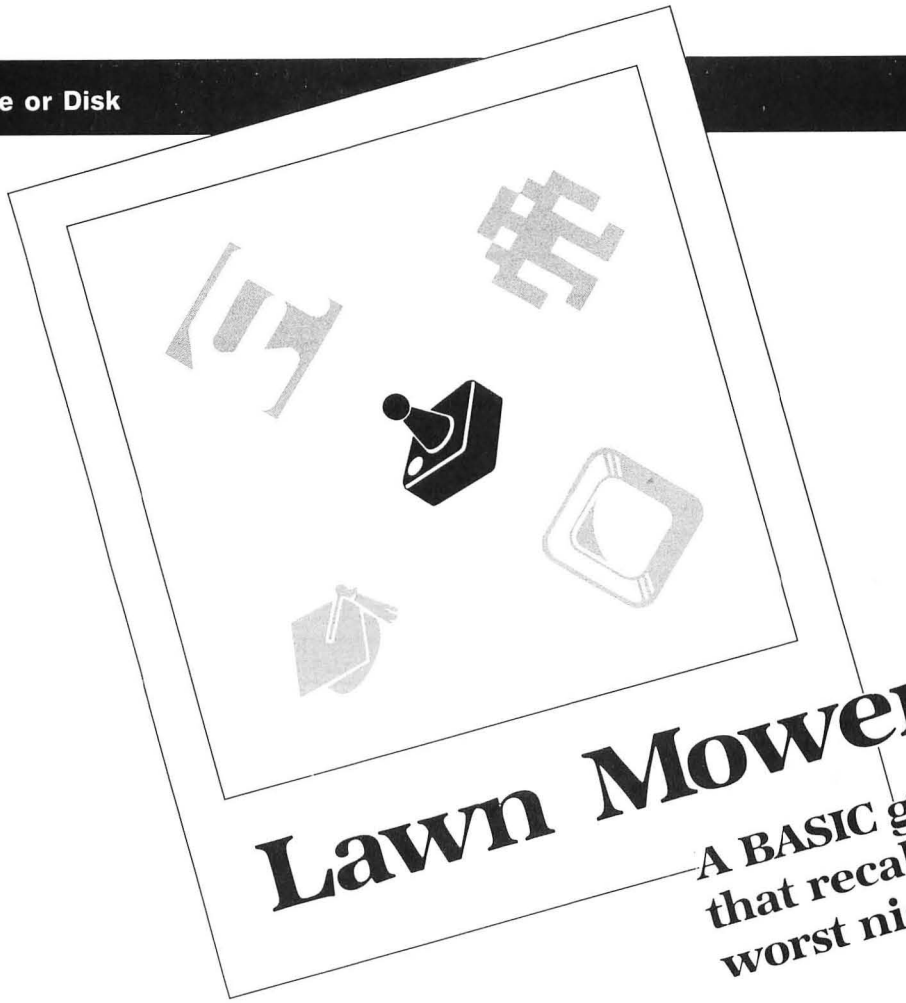
ISSUE 50 • Krazy Katerpillars • Atari Picture Storage Techniques • Trails in Action! • Scroll-It • Screen Scroller

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

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS
ANALOG COMPUTING™
P.O. BOX 23 WORCESTER, MASSACHUSETTS 01603

48K Cassette or Disk

GAME



Lawn Mower

A BASIC game that recalls every boy's worst nightmare.

by Paul Tupaczewski

In the game of **Lawn Mower**, you're Tommy, a boy hired to mow lawns all around the town of Atariville. Since you have signed contracts with the people you're going to mow for, you can't escape the dangers that crop up while trimming the greens.

The object of **Lawn Mower** is to clear the screen of grass. Whenever you go over a strip of grass, it turns darker to show it's been cut. There are also trees impeding your way. If you hit a tree, you'll bounce back.

On board 1, the Joneses' house, you must avoid Hi-Leggers. These creatures move from side to side, while randomly bouncing up and down. If they hit you, you lose one of your three lives. When you've lost all of your lives, the game ends.

On board 2, Cursor Park, holes suddenly appear! These are made by gophers who are afraid to show themselves. If you fall into one of the holes, you lose a life.

Board 3, the golf course, introduces the Mad Planter. He's a little orange man who plants grass where you've already mowed. The only way to get rid of him is either to run him over, or to plant a land mine—and make him run into it. This will make him disappear. . . for a while.

To plant a land mine, you simply press your joystick button. An explosive which you've buried in the ground will look just like a piece of mowed lawn. The number of land mines is shown at the bottom of the screen. You get an extra mine every time you clear a board, with a maximum of five. If *you* run into a mine, you won't be killed, but you will destroy the charge, rendering that mine useless against the enemy.

In the final board, John's orchard, the Mad Planter reappears. And there's also a new problem. The orchard is separated into two parts by a superhighway. You must get across this road to travel from one side of the orchard to the other.

If you run into a car while crossing, you'll lose a life. Also, you can't plant land mines on the road. If you mow all of *this* board, you'll go back to board 1, but at a harder level.

Scoring is as follows: mowing a piece of lawn=250 points; making the Mad Planter run into a land mine=250 points; running over the Mad Planter=500 points; and mowing all of a board=500 points times the level at which you played.

Your score is shown in the upper left on the screen. The level is in the upper right, and your number of lives remaining is shown by the number of circles next to the level number. The number of mines can be seen at the bottom of the screen.

I used Tom Hudson's excellent player mover subroutine from issue 10 and found it very easy and *fast*. I hope you have as much fun with **Lawn Mower** as I did.

Paul Tupaczewski attends school in Boonton, New Jersey. He's had his Atari 400 for three years, with an Indus disk drive and an Epson RX-80 printer, which he received as a Christmas present.

(Listing starts on next page)

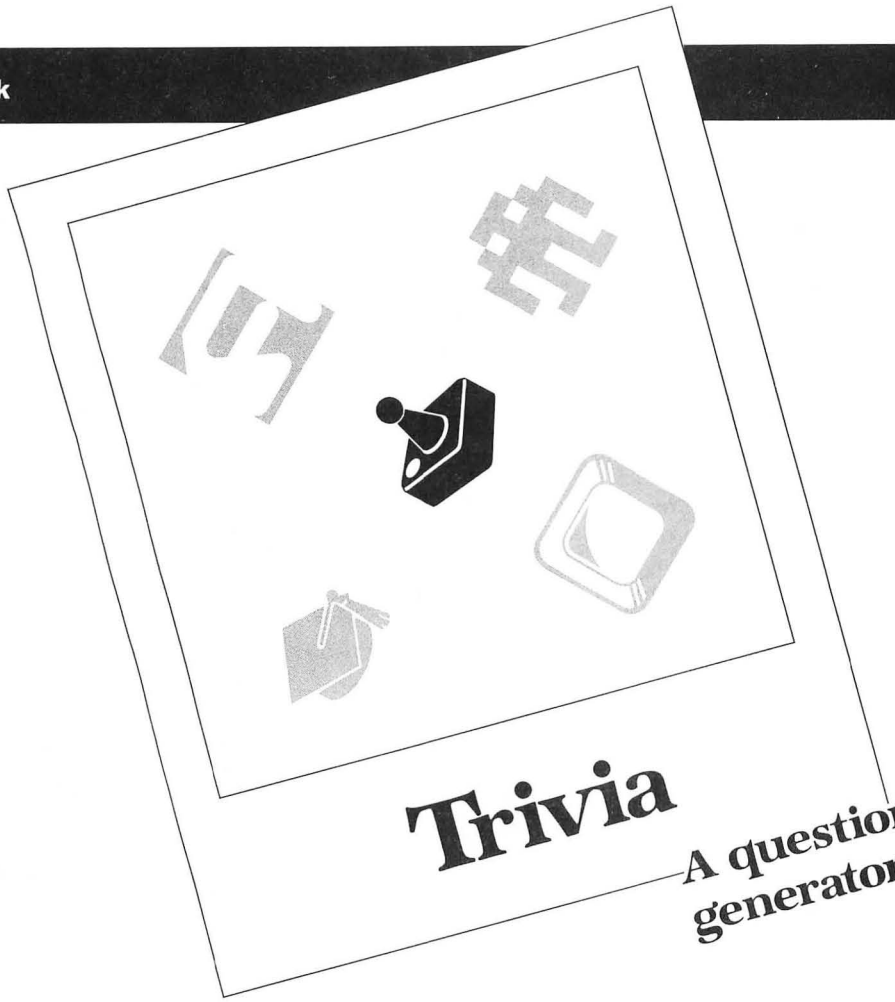
The two-letter checksum code preceding the line numbers here is *not* a part of the BASIC program. For further information, see the *BASIC Editor II*, in issue 47 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
NJ 0 REM ************************
QP 1 REM *      Lawn Mower      *
SY 2 REM * by Paul Tupaczewski *
FG 3 REM *  ANALOG Computing    *
NN 4 REM ************************
OM 5 DIM LOC(5)
NL 7 K0=0:K1=1:K2=K1+K1:K3=K2+K1:K4=K2+K2
   :K5=K2+K3:K6=K3+K3:K7=K4+K3:K8=K4+K4
FR 10 GOSUB 815:GOSUB 720:GOSUB 875
MR 15 LIV=K3:SC=K0:LEV=K1:LEV2=K1:MIN=K5:
   HARD=K0
CV 20 GRAPHICS K3*K6:DL=PEEK(560)+PEEK(56
   1)*256+K4:POKE DL-K1,70:POKE DL+12,6:P
   OKE DL+K2,6:POKE 709,216:POKE 623,K1
ZT 25 PMBASE=INT((PEEK(145)+K3)/K4)*K4:PO
   KE 54279,PMBASE:PMB=PMBASE*256:POKE 55
   9,46:POKE 53277,K3:POKE 756,ST/256
PH 30 POKE 708,194
EE 35 POKE 704,148:MIN=MIN+K1:IF MIN>K5 T
   HEN MIN=K5
HC 40 POKE 705,252:POKE 706,160:POKE 707,
   54
GW 45 POSITION K1,K0:? #6;"SCORE";CHR$(15
   4);SC:POSITION 13,K0:FOR R=K1 TO LIV:?
    #6;CHR$(138);:NEXT R
SK 50 POSITION 17,K0:? #6;LEV:POSITION K6
   ,11:? #6;"MINES";MIN
MN 55 FOR R=K2 TO K3*K3:COLOR K3:PLOT K2,
   R:DRAWTO 17,R:NEXT R
JA 60 POSITION K5*K2,K5:? #6;"#":GRA=K0
AD 65 ON LEV2 GOSUB 430,450,475,500
SJ 67 GOSUB 910
KK 70 X=128:Y=K3*K8*K2:X1=K5+K5:Y1=K5:M=K
   1:TMWT=K0:X3=K5+K5:Y3=K2:MY3=K3*K8:MX3
   =128
OY 75 A=USR(MOVE,K0,PMB,ADR(M$(M*K8-K7,M*
   K8)),X,Y,K8)
TT 80 OX=X:OY=Y:OX1=X1:OY1=Y1
GU 85 POKE 53278,K1
WN 90 ON LEV2 GOSUB 195,205,240,250
YS 95 IF PEEK(764)<>33 THEN 115
WH 100 POKE 764,255
ID 105 IF PEEK(764)<>33 THEN 105
WJ 110 POKE 764,255
HD 115 IF STRIG(0)=K0 AND MIN>K0 AND LEV2
   >K2 AND LEV2<K5 THEN GOSUB 565
VC 120 S=STICK(K0):XAD=(S=K7)-(S=11):YAD=
   (S=13)-(S=14)
PU 125 X=X+XAD*K8:Y=Y+YAD*K8:X1=X1+XAD:Y1
   =Y1+YAD
MF 130 IF XAD=-K1 THEN M=K2
OD 135 IF XAD=K1 THEN M=K1
NN 140 IF YAD=-K1 THEN M=K3
RE 145 IF YAD=K1 THEN M=K4
TO 150 IF X1<K2 OR X1>17 OR Y1<K2 OR Y1>9
    THEN X1=OX1:Y1=OY1:X=OX:Y=OY
YA 155 LOCATE X1,Y1,LOC
QJ 160 IF OX1<>X1 OR OY1<>Y1 THEN IF LOC=
   K3 THEN POSITION X1,Y1:? #6;"#":SC=SC+
   10:GOSUB 585:GRA=GRA+K1
GU 165 IF OX1<>X1 OR OY1<>Y1 THEN IF LOC=
   36 THEN POSITION X1,Y1:? #6;"#":GOSUB
   580
XT 170 IF OX1<>X1 OR OY1<>Y1 THEN IF LOC=
   K7 THEN X1=OX1:Y1=OY1:X=OX:Y=OY:GOSUB
   575
JZ 175 IF GRA=GR5 THEN 685
UM 180 IF LOC=32 AND LEV2<>K4 THEN 525
US 185 GOTO 75
TB 195 GOSUB 310
YY 200 RETURN
RA 205 TMWT=TMWT+K1:IF TMWT=25 AND HARD<>
   K1 THEN 220
HN 210 IF TMWT=18 AND HARD=K1 THEN 220
ZP 215 RETURN
VS 220 TMWT=K0:R=INT(RND(0)*14)+K3:T=INT(
   RND(0)*K5)+K3:LOCATE R,T,Z:GOSUB 930:I
   F FG=1 THEN 220
PO 225 IF Z=32 THEN 220
XF 230 IF Z=K3 THEN GR5=GR5-K1
SP 235 POSITION R,T:? #6;" ":SOUND K0,K0,
   K8,K6:FOR R=K1 TO K5:NEXT R:SOUND K0,K
   0,K0,K0:RETURN
DG 240 TMWT=TMWT+K1:IF TMWT>55 THEN GOSUB
    390
ZV 245 RETURN
OJ 250 A=USR(MOVE,K1,PMB,ADR(CL$),CX1,56,
   K8):A=USR(MOVE,K2,PMB,ADR(CR$),CX2,70,
   K8)
YK 255 CX1=CX1-K4-HARD*K4:IF CX1<65 THEN
   CX1=184
HZ 260 CX2=CX2+K6+HARD*K4:IF CX2>184 THEN
    CX2=65
SM 265 IF PEEK(53260)=K2 OR PEEK(53260)=K
   4 THEN 280
ZH 270 TMWT=TMWT+K1:IF TMWT>60 THEN GOSUB
    390
AB 275 RETURN
MY 280 FOR R=15 TO K0 STEP -0.2:POKE 704,
   R:SOUND K0,100,K0,R:NEXT R
QC 285 FOR R=K1 TO 100:NEXT R
VW 290 POSITION 12+LIV,K0:? #6;" ":LIV=LI
   V-K1
MX 295 FOR R=15 TO K0 STEP -K1:SOUND K0,1
   21,10,R:NEXT R
NG 300 IF LIV=K0 THEN 595
YA 305 FOR R=K1 TO 100:NEXT R:CX1=120:CX2
   =65:POKE 704,148:GOTO 70
HV 310 A=USR(MOVE,K1,PMB,ADR(GT$),MX1,MY1
   ,K8):A=USR(MOVE,K2,PMB,ADR(GT$),MX2,MY
   2,K8):OMY1=MY1:OMY2=MY2
HD 315 MX1=MX1+K4+HARD*K4:IF MX1>184 THEN
    MX1=64
EO 320 MX2=MX2-K4-HARD*K4:IF MX2<64 THEN
   MX2=184
QR 325 ADD=INT(RND(0)*K3)-K1:ADD=ADD*(K3+
   HARD):MY1=MY1+ADD:IF MY1<24 THEN MY1=2
   4
HM 330 IF MY1>80 THEN MY1=80
ZT 335 ADD=INT(RND(0)*K3)-K1:ADD=ADD*(K3+
   HARD):MY2=MY2+ADD:IF MY2<24 THEN MY2=2
   4
IU 340 IF MY2>80 THEN MY2=80
XY 345 IF PEEK(53260)=K2 OR PEEK(53260)=K
   4 THEN 355
ZJ 350 RETURN
TQ 355 FOR R=15 TO K0 STEP -0.2:SOUND K0,
   100,K0,R:POKE 704,R:NEXT R
PK 360 FOR R=K1 TO 100:NEXT R
WG 365 POSITION 12+LIV,K0:? #6;" ":LIV=LI
   V-K1
FM 370 FOR R=15 TO K0 STEP -1:SOUND K0,12
   1,10,R:NEXT R
OJ 375 IF LIV=K0 THEN 595
KK 380 FOR R=K1 TO 100:NEXT R:POKE 704,14
   8:GOSUB 680:POKE 53278,K1
IJ 385 MX1=64:MX2=184:MY1=48:MY2=48:GOTO
   70
HL 390 A=USR(MOVE,K3,PMB,ADR(PL$),MX3,MY3
   ,K8):OX3=X3:OMX3=MX3:LOCATE X3,Y3,ZZ
LR 395 IF ZZ=36 THEN SC=SC+250:GOSUB 590:
   GOTO 425
EP 400 IF ZZ=35 THEN POSITION X3,Y3:? #6;
   CHR$(K3):GRA=GRA-K1
```

```
ML 405 Y3=Y3+K1:MY3=MY3+K8:IF MY3>80 THEN
    TMWT=K0:A=USR(MOVE,3,PMB,ADR("♥"),0,0
   ,1):X3=10:Y3=K2:MX3=128:MY3=24:RETURN
BD 410 ADD=INT(RND(0)*K3)-K1:X3=X3+ADD:MX
   3=MX3+ADD*K8:IF X3<K2 OR X3>17 THEN X3
   =OX3:MX3=OMX3
JQ 415 IF PEEK(53260)=K8 THEN SC=SC+500:G
   OSUB 590:FOR R=15 TO K0 STEP -K3:SOUND
    K0,200,10,R:NEXT R:MY3=90:GOTO 405
ZE 420 RETURN
SQ 425 POSITION X3,Y3:? #6;"#":FOR R=15 T
   O K0 STEP -1.5:SOUND K0,200,K8,R:NEXT
   R:MY3=90:GOTO 405
MH 430 GR5=115:COLOR K7:PLOT K3,K3:PLOT K
   3,K4:PLOT K4,K3:PLOT K3,K8:PLOT K3,K7:
   PLOT K4,K8:MX1=64:MX2=184
FV 435 PLOT 15,K3:PLOT 16,K3:PLOT 16,K4:P
   LOT 15,K8:PLOT 16,K8:PLOT 16,K7:MY1=48
   :MY2=48
OG 440 POSITION K4,K1:? #6;"JONES% HOUSE"
ZX 445 RETURN
XY 450 GR5=109:COLOR K7
KI 455 PLOT K3,K8:DRAWTO K8,K3:PLOT 11,K3
   :DRAWTO 16,K8
TR 460 PLOT K8,K8:PLOT 9,K7:PLOT 10,K7:PL
   OT 11,K8:PLOT K3,K3:PLOT 16,K3
MV 465 POSITION K5,K1:? #6;"CURSOR PARK"
ZO 470 RETURN
SA 475 POSITION K5,K1:? #6;"GOLF COURSE":
   X3=10:Y3=K2:MY3=24:MX3=128:COLOR K7
RV 480 FOR R=K1 TO 15
SO 485 A=INT(RND(0)*16)+K2:B=INT(RND(0)*K
   7)+K2:LOCATE A,B,C:GOSUB 920:IF FG=1 T
   HEN 485
AF 490 IF C=K7 OR (A=10 AND B=K5) THEN 48
   5
SC 495 PLOT A,B:NEXT R:GR5=112:RETURN
RY 500 POSITION K3,K1:? #6;"JOHN%S ORCHAR
   D"
CE 505 COLOR K7:FOR R=K3 TO 17 STEP 2:FOR
    T=K3 TO 9 STEP K2:PLOT R,T
GG 510 NEXT T:NEXT R:GR5=55:CX1=120:CX2=6
   5
BL 515 COLOR 32:PLOT K2,K8:DRAWTO 17,K8:P
   LOT K2,K6:DRAWTO 17,K6:COLOR 45:PLOT K
   2,K7:DRAWTO 17,K7
ZF 520 RETURN
TE 525 A=USR(MOVE,K0,PMB,ADR(M$(K1,K8)),X
   ,Y,K8)
NV 530 FOR R=15 TO K0 STEP -0.1:SOUND K0,
   60-(R*K2),10,R:POKE 704,R:NEXT R
PV 535 FOR R=K1 TO 100:NEXT R
VP 540 POSITION 12+LIV,K0:? #6;" ":LIV=LI
   V-K1
MQ 545 FOR R=15 TO K0 STEP -K1:SOUND K0,1
   21,10,R:NEXT R
NS 550 IF LIV=K0 THEN 595
PX 555 FOR R=K1 TO 100:NEXT R:POKE 704,14
   8
SA 560 GOTO 70
IA 565 POSITION X1,Y1:? #6;"$":MIN=MIN-K1
   :POSITION 12,11:? #6;MIN:FOR R=15 TO K
   0 STEP -K1
SB 570 SOUND K0,25,10,R:NEXT R:RETURN
EE 575 SOUND K0,200,8,6:FOR R=K1 TO K3:NE
   XT R:SOUND K0,K0,K0,K0:RETURN
QK 580 SOUND K0,100,K0,10:FOR R=K1 TO K3:
   NEXT R:SOUND K0,K0,K0,K0:RETURN
TV 585 FOR R=10 TO K0 STEP -2.5:SOUND K0,
   R,K8,R:NEXT R:SOUND K0,K0,K0,K0
IK 590 POSITION K7,K0:? #6;SC:RETURN
RV 595 GOSUB 680:POSITION K6,K5:? #6;"gam
   e over"
XH 600 RESTORE 670
YV 605 READ O,P,DLY:IF O=-K1 THEN 615
JM 610 FOR R=15 TO K0 STEP -DLY:SOUND K0,
   O,10,R:SOUND K1,P,10,R:NEXT R:GOTO 605
ON 615 FOR R=K1 TO 100:NEXT R:? #6;"↖":PO
   SITION K2,K3:? #6;"your score";CHR$(15
   4);SC
PZ 620 POSITION K2,K5:? #6;"high score";C
   HR$(26);HS:FOR R=K1 TO 400:NEXT R
RV 625 IF SC<=HS THEN 655
BE 630 FOR R=K1 TO K4:POSITION K2,K3:? #6
   ;"your score";CHR$(26):POSITION K2,K5:
   ? #6;"high score";CHR$(154)
MQ 635 FOR T=15 TO K0 STEP -1.5:SOUND K0,
   60,10,T:NEXT T:POSITION K2,K3:? #6;"yo
   ur score";CHR$(154)
WQ 640 POSITION K2,K5:? #6;"high score";C
   HR$(26):FOR T=15 TO K0 STEP -1.5:SOUND
    K0,121,10,T:NEXT T:NEXT R
YF 645 FOR R=K1 TO 200:NEXT R:FOR R=HS TO
    SC STEP 50
KG 650 POSITION 13,K5:? #6;R:POKE 53279,K
   0:NEXT R:HS=SC:POSITION 13,K5:? #6;HS
QZ 655 POSITION K5,K0:? #6;"PRESS start":
   POSITION K4,K1:? #6;"TO PLAY AGAIN"
IE 660 IF PEEK(53279)<>K6 THEN 660
IW 665 LIV=K3:SC=K0:LEV=K1:LEV2=K1:MIN=K5
   :HARD=K0:? #6;"↖":GOTO 30
HB 670 DATA 121,96,1,96,81,1,108,91,1,91,
   72,1,96,81,1,72,60,1,72,60,5,81,64,5
MR 675 DATA 91,72,5,96,81,5,108,91,5,121,
   96,1,-1,0,0
SB 680 FOR R=K0 TO K3:A=USR(MOVE,R,PMB,AD
   R("♥"),K0,K0,K1):NEXT R:RETURN
YQ 685 RESTORE 715
SJ 690 READ O,P,DLY:IF O=-K1 THEN 700
WA 695 FOR R=15 TO K0 STEP -DLY:SOUND K0,
   O,10,R:SOUND K1,P,10,R:NEXT R:GOTO 690
EJ 700 LEV2=LEV2+K1:IF LEV2>K4 THEN LEV2=
   K1:HARD=K1
GV 705 SC=SC+500*LEV:LEV=LEV+K1
DP 710 FOR R=K0 TO K3:A=USR(MOVE,R,PMB,AD
   R("♥"),K0,K0,K1):NEXT R:GOTO 20
HV 715 DATA 81,64,1,91,72,3,96,81,1,108,9
   1,3,121,96,1,60,47,1,-1,0,0
FI 720 DIM PMMOV$(100):MOVE=ADR(PMMOV$):R
   ESTORE 755
PC 725 FOR X=K1 TO 100:READ N:PMMOV$(X)=C
   HR$(N):NEXT X
MV 730 DIM M$(32),PL$(K8),CL$(K8),CR$(K8)
   ,GT$(K8)
OK 735 FOR R=K1 TO K8:READ D:CL$(R)=CHR$(
   D):NEXT R:FOR R=K1 TO K8:READ D:CR$(R)
   =CHR$(D):NEXT R
TO 740 FOR R=K1 TO K8:READ D:GT$(R)=CHR$(
   D):NEXT R:FOR R=K1 TO K8:READ D:PL$(R)
   =CHR$(D):NEXT R
NS 745 FOR R=K1 TO 32:READ D:M$(R)=CHR$(D
   ):NEXT R
ZN 750 RETURN
US 755 DATA 216,104,104,104,133,213,104,2
   4,105,2,133,206,104,133,205,104,133,20
   4,104,133,203,104,104,133,208
IF 760 DATA 104,104,133,209,104,104,24,10
   1,209,133,207,166,213,240,16,165,205,2
   4,105,128,133,205,165,206,105
IZ 765 DATA 0,133,206,202,208,240,160,0,1
   62,0,196,209,144,19,196,207,176,15,132
   ,212,138,168,177,203,164
LE 770 DATA 212,145,205,232,169,0,240,4,1
   69,0,145,205,200,192,128,208,224,166,2
   13,165,208,157,0,208,96
QU 775 DATA 0,14,18,34,127,127,54,54
KQ 780 DATA 0,112,72,68,254,254,108,108
RN 785 DATA 28,42,54,28,20,20,20,54
VU 790 DATA 130,68,56,84,108,56,40,108
TS 795 DATA 0,96,96,240,236,226,71,98
LQ 800 DATA 0,6,6,15,55,71,226,70
JT 805 DATA 64,224,80,16,8,190,254,56
LA 810 DATA 28,127,125,16,8,10,7,2
QZ 815 POKE 106,PEEK(106)-K5:GRAPHICS K0:
   ST=(PEEK(106)+K1)*256:POKE 756,ST/256:
   POKE 752,K1
```

```
HJ 820 RESTORE 940:DIM XFR$(38):FOR R=K1
   TO 38:READ D:XFR$(R,R)=CHR$(D):NEXT R
WP 825 Z=USR(ADR(XFR$)):RESTORE 845
QC 830 POSITION 14,12:? "Initializing"
RV 835 READ X:IF X=-K1 THEN RETURN
OR 840 FOR Y=K0 TO K7:READ Z:POKE X+Y+ST,
   Z:NEXT Y:GOTO 835
RD 845 DATA 24,255,255,255,255,255,255,25
   5,255
QE 850 DATA 32,255,255,255,255,255,255,25
   5,255
OW 855 DATA 56,255,199,163,21,65,171,199,
   255
ES 860 DATA 40,0,24,24,24,0,0,0,0
TZ 865 DATA 80,0,0,28,62,62,62,28,0
GR 870 DATA -1
OR 875 GRAPHICS 17:DL=PEEK(560)+PEEK(561)
   *256+K4:POKE DL-K1,71:POKE DL+K2,K7:PO
   KE DL+K3,K7
WG 880 COLOR 138:PLOT K4,K0:DRAWTO 15,K0:
   PLOT K4,K2:DRAWTO 15,K2
IO 885 POSITION K4,K1:? #6;CHR$(138);"lawn mower";CHR$(138)
PS 890 POSITION 9,12:? #6;"BY":POSITION K
   3,14:? #6;"PAUL TUPACZEWSKI"
VY 895 POSITION K5,18:? #6;"PRESS start"
EG 900 IF PEEK(53279)<>K6 THEN 900
ZU 905 RETURN
EW 910 FOR I=K8 TO 12:LOCATE I,K5,Z:LOC(I
   -K7)=Z:NEXT I:POSITION K8,K5:? #6;"REA
   DY"
SD 915 FOR I=K1 TO 200:NEXT I:FOR I=K8 TO
    12:POSITION I,K5:? #6;CHR$(LOC(I-K7))
   :NEXT I:RETURN
SG 920 FG=0:FOR I=A-K1 TO A+K1 STEP K2:FO
   R J=B-K1 TO B+K1 STEP K2:LOCATE I,J,Z:
   IF Z=K7 THEN FG=1
AW 925 NEXT J:NEXT I:RETURN
KH 930 FG=0:FOR I=R-K1 TO R+K1 STEP K2:FO
   R J=T-K1 TO T+K1 STEP K2:LOCATE I,J,P:
   IF P=32 THEN FG=1
AY 935 NEXT J:NEXT I:RETURN
EB 940 DATA 104,169,0,133,203,133,205,169
   ,224,133,206,165,106,24,105,1,133,204,
   160,0,177,205,145,203,200,208,249
WX 945 DATA 230,204,230,206,165,206,201,2
   28,208,237,96
```



Trivia

A question-and-answer generator and sample game.

by Jan Iverson

Trivia seems to be a "hot" item nowadays. There are board games on the market shelves and even some games on the more popular computers.

With the program in **Trivia**, you can generate a question and four possible answers. Use the second listing as a sample of a game you may create. If you wish to create your own game, do so; the generator will assist you in setting up your trivia database.

The uses are only limited by your imagination. You could reserve a disk each for sports, TV, movies, science, history, the Bible, etc.; the list can go on and on.

Question-and-answer generator.

The main menu contains four options: "create," "edit," "play" and "print."

The create menu has four options: "continue," "edit," "print" and "menu."

After typing in your question, four answers and the correct number corresponding to the answer, press RETURN if you wish to continue entering. This will clear the screen, and you may enter a further trivia question with its answers. If you need to correct any of the data just entered, use the ARROW keys and page over to the edit option. Hit RETURN, and you may change any line.

If you're finished and want to print what you have in the database, you need not go back to the main menu. Just page over to the print option and press RETURN. This will save all the data you've entered thus far, so you'll be able to view it. Paging over to the menu option and pressing RETURN will take you back to the main menu, after you've saved the database just entered.

Our **Trivia** game is limited to 200 items. A count at the top of the screen indicates how many items you're entering and how many remain.

If you need to edit any item in your trivia database, use the second option from the main menu.

You'll be allowed to enter the question as a search item, or, if you wish to step through the file, use the asterisk (*), and each item on the database will be displayed.

The edit section has four options: "change," "delete," "next" and "menu."

When the item in question appears on-screen, press RETURN if you want to change any line. This routine will allow you to alter a line as many times as you wish. When finished, press OPTION to return to the edit menu. If you used the asterisk option to step through your database and want to see additional items, use the ARROW key to page over to the "next" option. The next item on the database will appear on-screen. The delete option will allow you to remove a single item from the database if you typed in the question name as a search message. If you used the asterisk option, it will delete the item and await your next request. When you're finished, page over to the menu option. All changes will be saved and you'll return to the main menu.

If you have enough questions to run the **Trivia** game,

use the play option. The screen will inform you that the game is loading.

The print menu has four of its own options: "screen," "printer," "both" and "menu."

Using the screen option allows you to view two complete items on your database at a time, with record numbers. Press START to continue viewing. Press ESC to terminate the operation. When you've looked at the complete database, you'll be prompted to press SELECT to return to the main menu.

You also may send the database to a printer. Page over to the printer option and press RETURN. A hard copy of your database will be printed. If you wish to see the database on-screen as it's printing, use the "both" option.

Paging over to the menu option will return you to the main menu.

The Trivia game.

When saving Listing 2, use the name D:TRIV.BAS. The game question generator looks for this name when you use the play option from the main menu.

The game program reads your database into an array with a limit of 200 items. When completed, the game will begin.

Questions are selected through a random number algorithm beginning on Line 1110. The same questions and answers will not be used again in your session. When the questions are exhausted, a session will terminate, and you'll be asked if you wish to play again. Pressing START will allow the database to be loaded for another session. The program has some sounds built into it, but, because we want enough questions and answers loaded into the array, the program is much simplified.

If you select an incorrect answer, a buzz will sound while the correct number flashes for a few seconds. If you choose the correct number, a nice "beep-beep" sound will play. At the end of each question and answer, you'll be asked to either press START to continue, or OPTION to finish.

A timer at the top left will count down from 10 to 0. If you don't answer the question in 10 seconds, the buzz will sound and a wrong answer will result. If the correct answer is given, the remaining seconds are transfered to the right-hand score. The screen will clear, and the running total of right and wrong responses will be printed at the top. The running total will always print at the end of each question/answer routine.

When the OPTION key is pressed, results will be printed at the top of the screen, an appropriate message will be printed, and a few bars of "The Entertainer" will play. If your current score is higher than the high score, it will be transfered to the HI-SCORE area. This way, you may compete against another person—or against your previous best score. You'll then be given the option to either end the session or play again.

Use the question-and-answer generator to update your database. My family has played the game a number of times, and—just when they think they're getting good at it—I put some new questions in and take out some old ones. It keeps them on their toes.

I have a number of trivia databases I've developed, including sports, TV, movies, commercials and ads, and general trivia. Have a happy **Trivia** hunt.

Jan Iverson is an applications programmer with Chevron Corp. He's been working with computers for eighteen years and is program chairman for his local user's group (DACE). He lives in Antioch, California with his wife and three children.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
HY 10 REM ******************************
ZR 20 REM *                            *
TQ 30 REM *      GAME GENERATOR        *
SK 40 REM *           by               *
MQ 50 REM *        Jan Iverson         *
ZV 60 REM *                            *
IE 70 REM ******************************
BF 80 REM
UK 90 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:
   K7=7:K8=8:K9=9:K10=10:K11=11:K12=12:K1
   3=13:K14=14:K15=15
MV 100 K16=16:K17=17:K18=18:K19=19:K20=20
   :K21=21:K22=22:K23=23:K24=24:K25=25:K2
   6=26:K28=28:K29=29:K30=30:K31=31
FI 110 K708=708:K709=709:K710=710:K711=71
   1:K712=712:K764=764:K752=752:K155=155:
   K255=255:K54286=54286:K64=64:K192=192
EL 120 K53761=53761:K45=45:K126=126:K5327
   9=53279:K125=125:K132=132:K196=196
QZ 130 K1729=1729:K1730=1730:K1731=1731:K
   1732=1732:K1733=1733:K152=152:K132=132
   :K32=32:K52=52:K198=198
RH 140 INIT=K0:COUNT=K0:PR=K0:MAX=200:CNT
   =K0:RMAIN=K0
DX 150 DIM AN(20),Q(30),A$(20),CR(2),C$(2
   ),Q5$(30)
LL 160 DIM QUEST$(30),AN51$(20),AN52$(20)
   ,AN53$(20),AN54$(20),IT$(1),ARR$(111),
   FILE1$(15),FILE2$(15),Q$(30)
VD 170 FILE1$="                ":FILE2$="
                 "
NR 180 FILE1$="D1:GAME.DAT":FILE2$="D1:GA
   ME.TMP"
BA 190 OPEN #K4,K4,K0,"K:"
DY 200 GOSUB 4920:POKE 39976,K6:POKE 3997
   7,K11:POKE K752,K1:POKE 53774,K64:POKE
    112,K64
YN 210 POSITION K7,K0:? "Use <ESC> to exi
   t program"
PD 220 POKE 1729,4:FOR I=1 TO 10:NEXT I:P
   OKE 1730,4:FOR I=1 TO 10:NEXT I
CN 230 POKE 1731,4:FOR I=1 TO 10:NEXT I:P
   OKE 1732,4
XS 240 POKE 1733,132
ME 250 POKE K708,218
JS 260 POSITION K1,K3:? "A TRIVIA QUIZ GA
   ME"
AY 270 POSITION K13,K6:? "By Jan Iverson"
ND 280 POSITION K10,K8:? "For Analog Comp
   uting"
BO 290 POSITION K2,K10:? "──────────────
   ──────────────────────"
ZC 300 POSITION K11,K14:? "Use the ↑← ↑→
   keys"
```

```
MI 310 POSITION K8,K16:? "to make your se
   lection"
SH 320 POSITION K11,K18:? "then press RET
   URN"
XY 330 POSITION K1,K22:? " CREATE "
OM 340 POSITION K11,K22:? "  EDIT  "
GA 350 POSITION K21,K22:? "  PLAY  "
KG 360 POSITION K31,K22:? " PRINT  "
KX 370 POKE K764,K255
YI 380 POSITION K1,K22:? " CREATE "
SK 390 IF PEEK(K764)=K6 THEN POSITION K1,
   K22:? " CREATE ":GOSUB 650:GOTO 580
ES 400 IF PEEK(K764)=K7 THEN POSITION K1,
   K22:? " CREATE ":GOSUB 650:GOTO 440
EO 410 IF PEEK(K764)=K28 THEN GRAPHICS 0:
   END
RT 420 IF PEEK(K764)=K12 THEN 660
QE 430 GOTO 390
KS 440 POKE K764,K255
TB 450 POSITION K11,K22:? "  EDIT  "
MP 460 IF PEEK(K764)=K6 THEN POSITION K11
   ,K22:? "  EDIT  ":GOSUB 650:GOTO 370
CD 470 IF PEEK(K764)=K7 THEN POSITION K11
   ,K22:? "  EDIT  ":GOSUB 650:GOTO 510
FC 480 IF PEEK(K764)=K28 THEN GRAPHICS 0:
   END
UM 490 IF PEEK(K764)=K12 THEN 1530
PC 500 GOTO 460
KN 510 POKE K764,K255
KI 520 POSITION K21,K22:? "  PLAY  "
MP 530 IF PEEK(K764)=K6 THEN POSITION K21
   ,K22:? "  PLAY  ":GOSUB 650:GOTO 440
BL 540 IF PEEK(K764)=K7 THEN POSITION K21
   ,K22:? "  PLAY  ":GOSUB 650:GOTO 580
EX 550 IF PEEK(K764)=K28 THEN GRAPHICS 0:
   END
EI 560 IF PEEK(K764)=K12 THEN 4860
OT 570 GOTO 530
LB 580 POKE K764,K255
PA 590 POSITION K31,K22:? " PRINT  "
GH 600 IF PEEK(K764)=K6 THEN POSITION K31
   ,K22:? " PRINT  ":GOSUB 650:GOTO 510
SL 610 IF PEEK(K764)=K7 THEN POSITION K31
   ,K22:? " PRINT  ":GOSUB 650:GOTO 370
ES 620 IF PEEK(K764)=K28 THEN GRAPHICS 0:
   END
OI 630 IF PEEK(K764)=K12 THEN 3100
NR 640 GOTO 600
VU 650 SOUND 0,45,10,8:FOR I=K1 TO K3:NEX
   T I:SOUND 0,0,0,0:RETURN
EN 660 POKE K1733,K4:? CHR$(K125)
WA 670 POKE K708,K152:POSITION K6,K3:? "C
   REATE"
AR 680 POSITION K2,K1:? "Max = ";MAX:POSI
   TION K15,K1:? "Curr = ":POSITION K29,K
   1:? "Rem = ":CNT=K0:RMAIN=K0
KF 690 POKE K1733,196:CH=1
ZM 700 POKE K54286,K64
NI 710 POSITION K10,K10:? "Reading file..
   ."
QO 720 CLOSE #K1:CLOSE #K2
EW 730 TRAP 760:OPEN #K1,K4,K0,FILE1$:TRA
   P 40000
ZH 740 OPEN #K2,K8,K0,FILE2$
OO 750 GOTO 800
XS 760 CLOSE #K1:OPEN #K1,K8,K0,FILE1$
EC 770 QUEST$="////////////////////////////
   ////"
JO 780 ? #K1;QUEST$:? #K1;AN51$:? #K1;AN5
   2$:? #K1;AN53$:? #K1;AN54$:? #K1;IT$
JL 790 CLOSE #K1:GOTO 720
WZ 800 INPUT #K1,QUEST$,AN51$,AN52$,AN53$
   ,AN54$,IT$
OP 810 IF QUEST$="////////////////////////
   ///////" THEN 850
SI 820 ? #K2;QUEST$:? #K2;AN51$:? #K2;AN5
   2$:? #K2;AN53$:? #K2;AN54$:? #K2;IT$
IH 825 POSITION K22,K1:? "   ":POSITION 3
   5,K1:? "   "
FT 830 CNT=CNT+1:POSITION K22,K1:? CNT:RM
   AIN=MAX-CNT:POSITION 35,K1:? RMAIN
ON 840 GOTO 800
MO 850 POKE K54286,K192
QQ 860 POSITION K10,K10:? "
   "
WB 870 POSITION K1,K0:? "Create your own
   questions and answers"
UI 880 GOSUB 4400
JS 890 POKE K1732,K52:FOR I=K1 TO K10:NEX
   T I:POKE K1731,K52:FOR I=K1 TO K10:NEX
   T I
WO 900 POKE K1730,K52:FOR I=K1 TO K10:NEX
   T I:POKE K1729,K52
QZ 910 GOSUB 4470:GOSUB 4760
NZ 920 POKE K764,K255:X=K8:Y=K6:POSITION
   X,Y:? "▌"
VK 930 GOSUB 3800
IX 940 IF LEN(Q$)<K30 THEN Q$(LEN(Q$)+1)=
   " ":GOTO 940
XT 950 IF Q$="
      " THEN 920
MG 960 QUEST$=Q$
XA 970 Q$(30)="♥":Q$(2)=" ":Q$=" "
OO 980 POKE K764,K255:X=K10:Y=K8:POSITION
    X,Y:? "▌"
YV 990 GOSUB 3950
QG 1000 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 1000
FN 1010 IF A$="                    " THEN
    980
JB 1020 AN51$=A$
VH 1030 A$(20)="♥":A$(2)=" ":A$=" "
ZQ 1040 POKE K764,K255:X=K10:Y=K10:POSITI
   ON X,Y:? "▌"
FB 1050 GOSUB 3950
CS 1060 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 1060
PD 1070 IF A$="                    " THEN
    1040
KC 1080 AN52$=A$
VZ 1090 A$(20)="♥":A$(2)=" ":A$=" "
BQ 1100 POKE K764,K255:X=K10:Y=K12:POSITI
   ON X,Y:? "▌"
ER 1110 GOSUB 3950
WK 1120 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 1120
JW 1130 IF A$="                    " THEN
    1100
KB 1140 AN53$=A$
VP 1150 A$(20)="♥":A$(2)=" ":A$=" "
ES 1160 POKE K764,K255:X=K10:Y=K14:POSITI
   ON X,Y:? "▌"
FJ 1170 GOSUB 3950
IW 1180 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 1180
UG 1190 IF A$="                    " THEN
    1160
KA 1200 AN54$=A$
VF 1210 A$(20)="♥":A$(2)=" ":A$=" "
MM 1220 POKE K764,K255:X=K16:Y=K16:POSITI
   ON X,Y:? "▌"
ZH 1230 GOSUB 4110
GP 1240 IF LEN(C$)<K1 THEN C$(LEN(C$)+1)=
   " ":GOTO 1240
CV 1250 IF C$=" " THEN 1220
YI 1260 IT$=C$(1,1)
WY 1270 C$(2)="♥":C$(2)=" ":C$=" "
AN 1280 POKE K764,K255
OW 1285 POSITION K22,K1:? "   ":POSITION
   35,K1:? "   "
IX 1290 CNT=CNT+1:POSITION K22,K1:? CNT:R
   MAIN=MAX-CNT:POSITION 35,K1:? RMAIN
JA 1300 POKE K764,K255:POSITION K1,K22:?
   "CONTINUE"
AG 1310 IF PEEK(K764)=K6 THEN POSITION K1
   ,K22:? "CONTINUE":GOSUB 650:GOTO 1470
```

```
5R 1320 IF PEEK(K764)=K7 THEN POSITION K1
   ,K22:? "CONTINUE":GOSUB 650:GOTO 1350
KG 1330 IF PEEK(K764)=K12 THEN POKE K5428
   6,K64:GOSUB 4260:POKE K54286,K192:GOSU
   B 4400:GOTO 920
PD 1340 GOTO 1310
AG 1350 POKE K764,K255
CS 1360 POSITION K11,K22:? "  EDIT  "
PR 1370 IF PEEK(K764)=K6 THEN POSITION K1
   1,K22:? "  EDIT  ":GOSUB 650:GOTO 1280
CC 1380 IF PEEK(K764)=K7 THEN POSITION K1
   1,K22:? "  EDIT  ":GOSUB 650:GOTO 1410
PC 1390 IF PEEK(K764)=K12 THEN 2180
RT 1400 GOTO 1370
ZW 1410 POKE K764,K255
JG 1420 POSITION K21,K22:? "  PRINT "
ZI 1430 IF PEEK(K764)=K6 THEN POSITION K2
   1,K22:? "  PRINT ":GOSUB 650:GOTO 1350
IW 1440 IF PEEK(K764)=K7 THEN POSITION K2
   1,K22:? "  PRINT ":GOSUB 650:GOTO 1470
BA 1450 IF PEEK(K764)=K12 THEN POKE K5428
   6,K64:GOSUB 4260:GOSUB 4280:POKE K5428
   6,K192:GOTO 3100
QX 1460 GOTO 1430
AO 1470 POKE K764,K255
UY 1480 POSITION K31,K22:? "  MENU  "
HU 1490 IF PEEK(K764)=K6 THEN POSITION K3
   1,K22:? "  MENU  ":GOSUB 650:GOTO 1410
WD 1500 IF PEEK(K764)=K7 THEN POSITION K3
   1,K22:? "  MENU  ":GOSUB 650:GOTO 1280
AN 1510 IF PEEK(K764)=K12 THEN POKE K5428
   6,K64:GOSUB 4260:GOSUB 4280:POKE K5428
   6,K192:? CHR$(125):GOTO 210
TN 1520 GOTO 1490
TC 1530 POKE K1733,K4:? CHR$(K125)
WQ 1540 POKE K708,70:POSITION K7,K3:? "ED
   IT"
ZM 1550 FOUND=K0:CH=2
VL 1560 POKE K1733,K32
VW 1570 POSITION K2,K0:? "Use the ↓ ↑ k
   eys then press RETURN"
BB 1580 GOSUB 4400
XW 1590 POKE K1732,K132:FOR I=K1 TO K10:N
   EXT I:POKE K1731,K132:FOR I=K1 TO K10:
   NEXT I
YK 1600 POKE K1730,K132:FOR I=K1 TO K10:N
   EXT I:POKE K1729,K132
OB 1610 GOSUB 4500:GOSUB 4600
SL 1620 POSITION K0,K6:? "▶"
CG 1630 POKE K764,K255:X=K8:Y=K6:POSITION
    X,Y:? "▌"
CF 1640 GOSUB 3800
FQ 1650 IF LEN(Q$)<K30 THEN Q$(LEN(Q$)+1)
   =" ":GOTO 1650
UI 1660 IF Q$="
       " THEN 1630
PO 1670 Q5$=Q$
NZ 1680 Q$(30)="♥":Q$(2)=" ":Q$=""
RZ 1690 POKE K54286,K64
TB 1700 CLOSE #K1:CLOSE #K2
MG 1710 OPEN #K1,K4,K0,FILE1$:OPEN #K2,K8
   ,K0,FILE2$
HI 1720 INPUT #K1,QUEST$,AN51$,AN52$,AN53
   $,AN54$,IT$
WP 1730 IF QUEST$="zzzzzzzzzzzzzzzzzzzzzzz
   zzzzzzzz" THEN 2120
QV 1740 IF Q5$(1,1)="*" THEN FOUND=1:GOTO
    1780
AT 1750 IF Q5$=QUEST$ THEN FOUND=1:GOTO 1
   780
BS 1760 ? #K2;QUEST$:? #K2;AN51$:? #K2;AN
   52$:? #K2;AN53$:? #K2;AN54$:? #K2;IT$
SD 1770 GOTO 1720
KY 1780 POSITION K8,K6:? QUEST$:POSITION
   K10,K8:? AN51$:POSITION K10,K10:? AN52
   $
YX 1790 POSITION K10,K12:? AN53$:POSITION
    K10,K14:? AN54$:POSITION K16,K16:? IT
   $
FR 1800 POKE K54286,K192
AE 1810 POKE K764,K255
TQ 1820 POSITION K1,K22:? "  CHANGE "
LO 1830 IF PEEK(K764)=K6 THEN POSITION K1
   ,K22:? " CHANGE ":GOSUB 650:GOTO 2000
AC 1840 IF PEEK(K764)=K7 THEN POSITION K1
   ,K22:? " CHANGE ":GOSUB 650:GOTO 1870
XM 1850 IF PEEK(K764)=K12 THEN CH=2:GOTO
   2180
TB 1860 GOTO 1830
AW 1870 POKE K764,K255
MO 1880 POSITION K11,K22:? "  DELETE "
JO 1890 IF PEEK(K764)=K6 THEN POSITION K1
   1,K22:? " DELETE ":GOSUB 650:GOTO 1810
UX 1900 IF PEEK(K764)=K7 THEN POSITION K1
   1,K22:? " DELETE ":GOSUB 650:GOTO 1940
CS 1910 IF PEEK(K764)=K12 AND Q5$(1,1)<>"
   *" THEN POKE K54286,K64:GOTO 2080
OZ 1920 IF PEEK(K764)=K12 AND Q5$(1,1)="*
   " THEN POKE K54286,K64:POSITION K11,K2
   2:? " DELETE ":GOTO 1720
VU 1930 GOTO 1890
AP 1940 POKE K764,K255
FU 1950 POSITION K21,K22:? "  NEXT  "
CY 1960 IF PEEK(K764)=K6 THEN POSITION K2
   1,K22:? "  NEXT  ":GOSUB 650:GOTO 1870
PS 1970 IF PEEK(K764)=K7 THEN POSITION K2
   1,K22:? "  NEXT  ":GOSUB 650:GOTO 2000
PF 1980 IF PEEK(K764)=K12 AND Q5$(1,1)="*
   " THEN POSITION K21,K22:? "  NEXT  ":P
   OKE K54286,K64:GOSUB 2060:GOTO 1720
VL 1990 GOTO 1960
ZM 2000 POKE K764,K255
TW 2010 POSITION K31,K22:? "  MENU  "
DP 2020 IF PEEK(K764)=K6 THEN POSITION K3
   1,K22:? "  MENU  ":GOSUB 650:GOTO 1940
TA 2030 IF PEEK(K764)=K7 THEN POSITION K3
   1,K22:? "  MENU  ":GOSUB 650:GOTO 1810
YP 2040 IF PEEK(K764)=K12 THEN POKE K5428
   6,K64:GOSUB 2060:GOTO 2080
OP 2050 GOTO 2020
BF 2060 ? #K2;QUEST$:? #K2;AN51$:? #K2;AN
   52$:? #K2;AN53$:? #K2;AN54$:? #K2;IT$
AV 2070 RETURN
HN 2080 INPUT #K1,QUEST$,AN51$,AN52$,AN53
   $,AN54$,IT$
WU 2090 IF QUEST$="zzzzzzzzzzzzzzzzzzzzzzz
   zzzzzzzz" THEN 2120
AP 2100 ? #K2;QUEST$:? #K2;AN51$:? #K2;AN
   52$:? #K2;AN53$:? #K2;AN54$:? #K2;IT$
RF 2110 GOTO 2080
HM 2120 IF FOUND=K0 THEN GOSUB 4360
AY 2130 ? #K2;QUEST$:? #K2;AN51$:? #K2;AN
   52$:? #K2;AN53$:? #K2;AN54$:? #K2;IT$
TC 2140 CLOSE #K1:CLOSE #K2
AP 2150 XIO 33,#K1,K0,K0,FILE1$
FJ 2160 XIO 32,#K1,K0,K0,"D:GAME.TMP,GAME
   .DAT"
DJ 2170 POKE K54286,K192:? CHR$(K125):GOT
   O 210
LO 2180 POKE K1733,K4
JG 2190 GOSUB 4810:POKE K1729,K4:GOSUB 48
   20:POKE K1730,K4
JG 2200 GOSUB 4830:POKE K1731,K4:GOSUB 48
   40:POKE K1732,K4
DT 2210 GOSUB 4650
IJ 2220 POKE 1757,K6:POKE K1733,K2:POSITI
   ON K12,K22:? "OPTION=RETURN"
CQ 2230 POKE K1732,K198:FOR I=K1 TO K5:NE
   XT I:POKE K1731,K198:FOR I=K1 TO K5:NE
   XT I
WB 2240 POKE K1730,K198:FOR I=K1 TO K5:NE
   XT I:POKE K1729,K198
DS 2250 GOSUB 4550
AE 2260 POKE K764,255
ST 2270 POSITION K0,K6:? "▶"
JG 2280 IF PEEK(K764)=K14 THEN POSITION K
```

```
   0,K6:? " ":GOSUB 650:GOTO 2960
IB 2290 IF PEEK(K764)=K15 THEN POSITION K
   0,K6:? " ":GOSUB 650:GOTO 2400
XB 2300 IF PEEK(K53279)=K3 AND CH=K2 THEN
    GOSUB 4710:POKE 1757,12:POKE K1733,K3
   2:GOTO 1810
BP 2310 IF PEEK(K53279)=K3 AND CH=K1 THEN
    GOSUB 4660:POKE 1757,12:POKE K1733,K1
   96:GOTO 1300
QW 2320 IF PEEK(K764)=K12 THEN GOTO 2340
SN 2330 GOTO 2280
BK 2340 POKE K764,K255:X=K8:Y=K6:POSITION
    K8,K6:? "...........................
   ..":GOSUB 3800
BO 2350 IF LEN(Q$)<K30 THEN Q$(LEN(Q$)+1)
   =" ":GOTO 2350
PV 2360 IF Q$="
       " THEN GOTO 2340
SW 2370 QUEST$=Q$
FA 2380 Q$(30)=" ":Q$(2)=" ":Q$=""
SF 2390 GOTO 2260
ZQ 2400 POKE K764,255
TR 2410 POSITION K0,K8:? "E>"
UE 2420 IF PEEK(K764)=K14 THEN POSITION K
   0,K8:? " ":GOSUB 650:GOTO 2260
XP 2430 IF PEEK(K764)=K15 THEN POSITION K
   0,K8:? " ":GOSUB 650:GOTO 2540
XP 2440 IF PEEK(K53279)=K3 AND CH=K2 THEN
    GOSUB 4710:POKE 1757,12:POKE K1733,K3
   2:GOTO 1810
CD 2450 IF PEEK(K53279)=K3 AND CH=K1 THEN
    GOSUB 4660:POKE 1757,12:POKE K1733,K1
   96:GOTO 1300
YH 2460 IF PEEK(K764)=K12 THEN GOTO 2480
QZ 2470 GOTO 2420
PY 2480 POKE K764,K255:X=K10:Y=K8:POSITIO
   N K10,K8:? "...................":GOSU
   B 3950
SW 2490 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 2490
YZ 2500 IF A$="                    " THEN
    GOTO 2480
JJ 2510 AN51$=A$
ZJ 2520 A$(20)=" ":A$(2)=" ":A$=""
PP 2530 GOTO 2400
AE 2540 POKE K764,255
OL 2550 POSITION K0,K10:? "E>"
FL 2560 IF PEEK(K764)=K14 THEN POSITION K
   0,K10:? " ":GOSUB 650:GOTO 2400
GM 2570 IF PEEK(K764)=K15 THEN POSITION K
   0,K10:? " ":GOSUB 650:GOTO 2680
YD 2580 IF PEEK(K53279)=K3 AND CH=K2 THEN
    GOSUB 4710:POKE 1757,12:POKE K1733,K3
   2:GOTO 1810
CR 2590 IF PEEK(K53279)=K3 AND CH=K1 THEN
    GOSUB 4660:POKE 1757,12:POKE K1733,K1
   96:GOTO 1300
SD 2600 IF PEEK(K764)=K12 THEN GOTO 2620
SX 2610 GOTO 2560
SP 2620 POKE K764,K255:X=K10:Y=K10:POSITI
   ON K10,K10:? "...................":GO
   SUB 3950
KK 2630 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 2630
SF 2640 IF A$="                    " THEN
    GOTO 2620
KG 2650 AN52$=A$
ZX 2660 A$(20)=" ":A$(2)=" ":A$=""
SP 2670 GOTO 2540
AS 2680 POKE K764,255
QN 2690 POSITION K0,K12:? "E>"
VG 2700 IF PEEK(K764)=K14 THEN POSITION K
   0,K12:? " ":GOSUB 650:GOTO 2540
YS 2710 IF PEEK(K764)=K15 THEN POSITION K
   0,K12:? " ":GOSUB 650:GOTO 2820
XP 2720 IF PEEK(K53279)=K3 AND CH=K2 THEN
    GOSUB 4710:POKE 1757,12:POKE K1733,K3
   2:GOTO 1810
CD 2730 IF PEEK(K53279)=K3 AND CH=K1 THEN
    GOSUB 4660:POKE 1757,12:POKE K1733,K1
   96:GOTO 1300
ZO 2740 IF PEEK(K764)=K12 THEN GOTO 2760
RJ 2750 GOTO 2700
ZF 2760 POKE K764,K255:X=K10:Y=K12:POSITI
   ON K10,K12:? "...................":GO
   SUB 3950
US 2770 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 2770
BT 2780 IF A$="                    " THEN
    GOTO 2760
LD 2790 AN53$=A$
ZJ 2800 A$(20)=" ":A$(2)=" ":A$=""
UN 2810 GOTO 2680
AE 2820 POKE K764,255
RN 2830 POSITION K0,K14:? "E>"
MD 2840 IF PEEK(K764)=K14 THEN POSITION K
   0,K14:? " ":GOSUB 650:GOTO 2680
PP 2850 IF PEEK(K764)=K15 THEN POSITION K
   0,K14:? " ":GOSUB 650:GOTO 2960
YD 2860 IF PEEK(K53279)=K3 AND CH=K2 THEN
    GOSUB 4710:POKE 1757,12:POKE K1733,K3
   2:GOTO 1810
CR 2870 IF PEEK(K53279)=K3 AND CH=K1 THEN
    GOSUB 4660:POKE 1757,12:POKE K1733,K1
   96:GOTO 1300
UM 2880 IF PEEK(K764)=K12 THEN GOTO 2900
UJ 2890 GOTO 2840
ET 2900 POKE K764,K255:X=K10:Y=K14:POSITI
   ON K10,K14:? "...................":GO
   SUB 3950
MG 2910 IF LEN(A$)<K20 THEN A$(LEN(A$)+1)
   =" ":GOTO 2910
TX 2920 IF A$="                    " THEN
    GOTO 2900
KY 2930 AN54$=A$
ZX 2940 A$(20)=" ":A$(2)=" ":A$=""
SZ 2950 GOTO 2820
AS 2960 POKE K764,255
TP 2970 POSITION K0,K16:? "E>"
FL 2980 IF PEEK(K764)=K14 THEN POSITION K
   0,K16:? " ":GOSUB 650:GOTO 2820
BK 2990 IF PEEK(K764)=K15 THEN POSITION K
   0,K16:? " ":GOSUB 650:GOTO 2260
WW 3000 IF PEEK(K53279)=K3 AND CH=K2 THEN
    GOSUB 4710:POKE 1757,12:POKE K1733,K3
   2:GOTO 1810
BK 3010 IF PEEK(K53279)=K3 AND CH=K1 THEN
    GOSUB 4660:POKE 1757,12:POKE K1733,K1
   96:GOTO 1300
NY 3020 IF PEEK(K764)=K12 THEN GOTO 3040
VO 3030 GOTO 2980
JZ 3040 POKE K764,K255:X=K16:Y=K16:POSITI
   ON K16,K16:? "":GOSUB 4110
JE 3050 IF LEN(C$)<K2 THEN C$(LEN(C$)+1)=
   " ":GOTO 3050
IP 3060 IF C$="  " THEN GOTO 3040
IF 3070 IT$=C$
AT 3080 C$(2)=" ":C$(2)=" ":C$=""
VG 3090 GOTO 2960
SN 3100 POKE K1733,K4:? CHR$(K125)
NC 3110 COUNT=0
OC 3120 POSITION K4,K0:? "Use <CTL> & 1 k
   ey to stop print"
FB 3130 POKE K708,K152:POSITION K7,K3:? "
   PRINT"
VY 3140 POSITION K12,K6:? "Printer Option
   s"
ED 3150 POSITION K2,K10:? "──────────────
   ──────────────────────────────"
DW 3160 POSITION K11,K14:? "Use the E< E>
    keys"
WO 3170 POSITION K8,K16:? "to make your s
   election"
JK 3180 POSITION K11,K18:? "then press RE
   TURN"
FB 3190 POKE K1729,K4:FOR I=K1 TO K10:NEX
```

```
   T I:POKE K1730,K4:FOR I=K1 TO K10:NEXT
    I
IS 3200 POKE K1731,K4:FOR I=K1 TO K10:NEX
   T I:POKE K1732,K4
XY 3210 POKE K1733,38
UT 3220 POSITION K1,K22:? " SCREEN "
AG 3230 POSITION K11,K22:? "PRINTER "
SR 3240 POSITION K21,K22:? "  BOTH  "
CR 3250 POSITION K31,K22:? "  MENU  "
AJ 3260 POKE K764,K255
VI 3270 POSITION K1,K22:? " SCREEN "
IQ 3280 IF PEEK(K764)=K6 THEN POSITION K1
   ,K22:? " SCREEN ":GOSUB 650:GOTO 3440
BB 3290 IF PEEK(K764)=K7 THEN POSITION K1
   ,K22:? " SCREEN ":GOSUB 650:GOTO 3320
KI 3300 IF PEEK(K764)=K12 THEN 3500
ST 3310 GOTO 3280
ZZ 3320 POKE K764,K255
SE 3330 POSITION K11,K22:? "PRINTER "
IR 3340 IF PEEK(K764)=K6 THEN POSITION K1
   1,K22:? "PRINTER ":GOSUB 650:GOTO 3260
SF 3350 IF PEEK(K764)=K7 THEN POSITION K1
   1,K22:? "PRINTER ":GOSUB 650:GOTO 3380
EN 3360 IF PEEK(K764)=K12 THEN PR=K1:GOTO
    3500
RX 3370 GOTO 3340
AR 3380 POKE K764,K255
LE 3390 POSITION K21,K22:? "  BOTH  "
UZ 3400 IF PEEK(K764)=K6 THEN POSITION K2
   1,K22:? "  BOTH  ":GOSUB 650:GOTO 3320
EN 3410 IF PEEK(K764)=K7 THEN POSITION K2
   1,K22:? "  BOTH  ":GOSUB 650:GOTO 3440
FK 3420 IF PEEK(K764)=K12 THEN PR=K2:GOTO
    3500
PZ 3430 GOTO 3400
AH 3440 POKE K764,K255
UR 3450 POSITION K31,K22:? "  MENU  "
EM 3460 IF PEEK(K764)=K6 THEN POSITION K3
   1,K22:? "  MENU  ":GOSUB 650:GOTO 3380
WU 3470 IF PEEK(K764)=K7 THEN POSITION K3
   1,K22:? "  MENU  ":GOSUB 650:GOTO 3260
ED 3480 IF PEEK(K764)=K12 THEN POKE 1733,
   4:? CHR$(125):GOTO 210
TR 3490 GOTO 3460
UD 3500 GOSUB 4340:POKE K54286,K64
JY 3510 X1=K3:Y1=K5
AN 3520 OPEN #K1,K4,K0,FILE1$
HJ 3530 INPUT #K1,QUEST$,ANS1$,ANS2$,ANS3
   $,ANS4$,IT$
NO 3540 IF QUEST$="zzzzzzzzzzzzzzzzzzzzzzz
   zzzzzzzz" THEN 3730
GQ 3550 COUNT=COUNT+1
VQ 3560 IF PR=K1 THEN GOSUB 3650:GOTO 353
   0
HW 3570 IF PR=K2 THEN GOSUB 3650
QI 3580 POSITION X1,Y1:? "Record Number:
   ";COUNT
ND 3590 Y1=Y1+1
HD 3600 POSITION X1,Y1:? QUEST$:Y1=Y1+1:P
   OSITION X1,Y1:? ANS1$:Y1=Y1+1:POSITION
    X1,Y1:? ANS2$:Y1=Y1+1
II 3610 POSITION X1,Y1:? ANS3$:Y1=Y1+1:PO
   SITION X1,Y1:? ANS4$:Y1=Y1+1:POSITION
   X1,Y1:? IT$
MK 3620 Y1=Y1+1
XQ 3630 IF Y1>K16 THEN POSITION K2,Y1:? "
   Press START to continue, ESC to end":P
   OKE K54286,K192:GOTO 3770
SF 3640 GOTO 3530
NZ 3650 LPRINT "Record Number: ";COUNT
KU 3660 LPRINT QUEST$
HZ 3670 LPRINT ANS1$
IS 3680 LPRINT ANS2$
JL 3690 LPRINT ANS3$
JC 3700 LPRINT ANS4$
LU 3710 LPRINT IT$
AV 3720 RETURN
PY 3730 CLOSE #K1:POKE K54286,K192
RD 3740 POSITION K4,Y1+1:? "Press SELECT
   to return to MENU"
JU 3750 IF PEEK(K53279)=K5 THEN ? CHR$(12
   5):GOTO 210
UL 3760 GOTO 3750
EZ 3770 IF PEEK(53279)=K6 THEN GOSUB 4340
   :Y1=5:POKE K54286,K64:GOTO 3640
QK 3780 IF PEEK(K764)=K28 THEN CLOSE #K1:
   ? CHR$(125):GOTO 210
VU 3790 GOTO 3770
NR 3800 E1=K0
GI 3810 FOR I=K1 TO K31
SJ 3820 IF I>K30 THEN I=I-K1:POSITION X,Y
   :? " ":X=X-K1:E1=K1
PE 3830 IF I<K1 THEN I=I+K1:POSITION X,Y:
   ? " ":X=X+K1:POSITION X,Y:? "▌"
LZ 3840 GET #4,Q
ZY 3850 IF Q=K126 THEN 3870:E1=K0
HY 3860 IF Q=K155 THEN 3930
MY 3870 IF Q=K126 THEN POSITION X,Y:? "."
   :X=X-K1:POSITION X,Y:? "▌":I=I-K1:E1=K
   0
JB 3880 IF I<K1 THEN 3830
PJ 3890 IF Q=K126 AND I>K0 THEN Q$(I)=" "
   :GOTO 3830
RO 3900 POSITION X,Y:? CHR$(Q)
VI 3910 IF I>K0 THEN Q$(I)=CHR$(Q)
QA 3920 X=X+K1:POSITION X,Y:? "▌":NEXT I
PG 3930 IF E1=K1 THEN RETURN
HG 3940 POSITION X,Y:? ".":RETURN
OI 3950 E1=K0
GG 3960 FOR I=K1 TO K21
SO 3970 IF I>K20 THEN I=I-K1:POSITION X,Y
   :? " ":X=X-K1:E1=K1
PV 3980 IF I<K1 THEN I=I+K1:POSITION X,Y:
   ? " ":X=X+K1:POSITION X,Y:? "▌"
UQ 3990 GET #4,AN
IM 4000 IF AN=K126 THEN 4030:E1=K0
KY 4010 IF AN=K155 THEN 4090
SO 4020 IF I=K21 THEN I=I-K1:POSITION X,Y
   :? " ":X=X-K1:POSITION X,Y:? "▌":GOTO
   3990
WZ 4030 IF AN=K126 THEN POSITION X,Y:? ".
   ":X=X-K1:POSITION X,Y:? "▌":I=I-K1:E1=
   K0
MV 4040 IF I<K1 THEN 3980
MN 4050 IF AN=K126 AND I>K0 THEN A$(I)="
   ":GOTO 3980
GE 4060 POSITION X,Y:? CHR$(AN)
HU 4070 IF I>K0 THEN A$(I)=CHR$(AN)
QB 4080 X=X+K1:POSITION X,Y:? "▌":NEXT I
PH 4090 IF E1=K1 THEN RETURN
GF 4100 POSITION X,Y:? ".":RETURN
TK 4110 FOR I=K1 TO K2
CR 4120 IF I>K2 THEN I=I-K1:POSITION X,Y:
   ? " ":X=X-K1
OR 4130 IF I<K1 THEN I=I+K1:POSITION X,Y:
   ? " ":X=X+K1:POSITION X,Y:? "▌"
WQ 4140 GET #4,CR
MQ 4150 IF CR=K126 THEN 4180
KZ 4160 IF CR=K155 THEN 4240
BG 4165 IF CR<49 OR CR>52 THEN POSITION X
   ,Y:? "▌":GOTO 4140
AD 4170 IF I=K2 THEN I=I-K1:POSITION X,Y:
   ? " ":X=X-K1:POSITION X,Y:? "▌":GOTO 4
   140
TT 4180 IF CR=K126 THEN POSITION X,Y:? "
   ":X=X-K1:POSITION X,Y:? "▌":I=I-K1:GOT
   O 4130
EA 4190 IF I<K1 THEN 4130
UU 4200 IF CR=K126 AND I>K0 THEN C$(I)="
   ":GOTO 4130
LX 4210 POSITION X,Y:? CHR$(CR)
PX 4220 IF I>K0 THEN C$(I)=CHR$(CR)
PQ 4230 X=X+K1:POSITION X,Y:? "▌":NEXT I
EI 4240 IF I<K3 THEN POSITION X,Y:? " "
AV 4250 RETURN
BL 4260 ? #K2;QUEST$:? #K2;ANS1$:? #K2;AN
```

```
   52$:? #K2;AN53$:? #K2;AN54$:? #K2;IT$
BB 4270 RETURN
AM 4280 QUEST$="ZZZZZZZZZZZZZZZZZZZZZZZZZZ
   ZZZZZ"
BU 4290 ? #K2;QUEST$:? #K2;AN51$:? #K2;AN
   52$:? #K2;AN53$:? #K2;AN54$:? #K2;IT$
SW 4300 CLOSE #K1:CLOSE #K2
AJ 4310 XIO 33,#K1,K0,K0,FILE1$
FD 4320 XIO 32,#K1,K0,K0,"D:GAME.TMP,GAME
   .DAT"
AR 4330 RETURN
YP 4340 FOR I=K5 TO K20:POSITION K1,I:? "

   ":NEXT I:RETURN
YS 4350 POKE K1730,K52:FOR I=K1 TO K10:NE
   XT I:POKE K1729,K52
OU 4360 FOR I=K1 TO K10:POSITION K0,K0:?
   "
   ":FOR J=K1 TO K10:NEXT J
XF 4370 POSITION K0,K0:? "           RECORD
    NOT FOUND        ":FOR K=K1 TO K10:NEXT
   K
FW 4380 NEXT I
BJ 4390 RETURN
II 4400 POSITION K1,K6:? "Quest? ........
   ......................."
DM 4410 POSITION K1,K8:? "Answer 1 ......
   ..............."
EW 4420 POSITION K1,K10:? "Answer 2 .....
   ..............."
HT 4430 POSITION K1,K12:? "Answer 3 .....
   ..............."
KQ 4440 POSITION K1,K14:? "Answer 4 .....
   ..............."
UI 4450 POSITION K1,K16:? "Correct number
    ."
BC 4460 RETURN
SY 4470 POSITION K2,K18:? "Input Question
   , Answers & Correct No."
MB 4480 POSITION K6,K20:? "Use E← E→ keys
    - press RETURN"
BL 4490 RETURN
YY 4500 POSITION K0,K18:? "Input the Ques
   tion to locate the record"
EY 4510 POSITION K0,K19:? "or use an '*'
   to step through the file."
PN 4520 POSITION K0,K20:? "Use the OPTION
   S below when the record"
OS 4530 POSITION K0,K21:? "is found. Pres
   s RETURN when changed."
AY 4540 RETURN
JC 4550 POSITION K1,K18:? "Use the E↓ E↑
   keys to page up and down"
HL 4560 POSITION K1,K19:? "until you find
    the line you wish to"
VP 4570 POSITION K1,K20:? "change.  Make
   the change and press the"
ZO 4580 POSITION K1,K21:? "RETURN key.  P
   ress <OPTION> when done"
BN 4590 RETURN
OX 4600 POSITION K1,K22:? " CHANGE "
TW 4610 POSITION K11,K22:? " DELETE "
NM 4620 POSITION K21,K22:? "  NEXT  "
CU 4630 POSITION K31,K22:? "  MENU  "
UV 4640 FOR I=K6 TO K16:POSITION K0,I:? "
    ":NEXT I:RETURN
GT 4650 POSITION K0,K22:? "
                           ":RETURN
JJ 4660 GOSUB 4810:POKE K1729,K4:GOSUB 48
   20:POKE K1730,K4
KL 4670 GOSUB 4830:POKE K1731,K4:GOSUB 48
   40:POKE K1732,K4
ST 4680 POKE K1732,K52:FOR I=K1 TO K5:NEX
   T I:POKE K1731,K52:FOR I=K1 TO K5:NEXT
    I
QV 4690 POKE K1730,K52:FOR I=K1 TO K5:NEX
   T I:POKE K1729,K52
MK 4700 GOSUB 4470:GOSUB 4650:GOSUB 4760:
```

```
   RETURN
IW 4710 GOSUB 4810:POKE K1729,K4:GOSUB 48
   20:POKE K1730,K4
JY 4720 GOSUB 4830:POKE K1731,K4:GOSUB 48
   40:POKE K1732,K4
TO 4730 POKE K1732,K132:FOR I=K1 TO K5:NE
   XT I:POKE K1731,K132:FOR I=K1 TO K5:NE
   XT I
MZ 4740 POKE K1730,K132:FOR I=K1 TO K5:NE
   XT I:POKE K1729,K132
AB 4750 GOSUB 4500:GOSUB 4650:GOSUB 4600:
   RETURN
ZD 4760 POSITION K1,K22:? "CONTINUE"
LK 4770 POSITION K11,K22:? "  EDIT  "
SL 4780 POSITION K21,K22:? "  PRINT "
DO 4790 POSITION K31,K22:? "  MENU  "
UN 4800 FOR I=K6 TO K16:POSITION K0,I:? "
    ":NEXT I:RETURN
FW 4810 POSITION K0,K18:? "
                          ":RETURN
GT 4820 POSITION K0,K19:? "
                          ":RETURN
AR 4830 POSITION K0,K20:? "
                          ":RETURN
BO 4840 POSITION K0,K21:? "
                          ":RETURN
YD 4850 FOR I=K18 TO K21:POSITION K0,I:?
   "
     ":NEXT I:RETURN
ZF 4860 GRAPHICS K1:POKE K710,K0:POKE K75
   2,K1:POKE K708,148:POKE K764,K255
DL 4870 POSITION 9,4:? #6;"A"
PV 4880 POSITION K7,K8:? #6;"trivia"
CQ 4890 POSITION K8,K12:? #6;"game"
KK 4900 ? "  Please wait - program is loa
   ding"
UW 4910 RUN "D:TRIV.BAS"
OW 4920 INIT=INIT+K1:IF INIT>K1 THEN 5010
TT 4930 GRAPHICS 0:POKE K752,K1:POKE K710
   ,K0:POSITION K10,K10:? "10 Seconds ple
   ase..."
NN 4940 RESTORE 5150:FOR N=0 TO 99:READ X
   :POKE 1664+N,X:NEXT N
VE 4950 COLTAB=1712:LUMTAB=COLTAB+24
JN 4960 REM START COUNTER AND RESET EVERY
    VBI
PI 4970 X=USR(1693)
AA 4980 REM TELL ANTIC WHERE DLI CODE IS
CX 4990 POKE 512,128
PF 5000 POKE 513,6
MS 5010 REM NOW SET INTERRUPT BITS
JF 5020 DSTART=PEEK(560)+256*PEEK(561)
AU 5030 FOR N=DSTART+6 TO DSTART+28
XG 5040 POKE N,130
HL 5050 NEXT N
VN 5060 REM SET INTERRUPT BIT ON FIRST LI
   NE
TA 5070 POKE DSTART+3,194
DK 5080 REM ENABLE DLI
CS 5090 POKE 54286,192
MD 5100 PRINT CHR$(125)
FT 5110 REM HANDLE LINE 0 AS BACKGROUND
TA 5120 POKE 710,PEEK(COLTAB)
JU 5130 POKE 709,PEEK(LUMTAB)
AR 5140 RETURN
NZ 5150 DATA 72,138,72,174,156,6,189,176,
   6,141
HX 5160 DATA 10,212,141,24,208,189,200,6,
   141,23
HS 5170 DATA 208,238,156,6,104,170,104,64
   ,1,104
NN 5180 DATA 169,7,160,168,162,6,32,92,22
   8,96
JP 5190 DATA 169,1,141,156,6,76,98,228,8,
   4
RL 5200 DATA 4,4,4,4,4,4,4,4,4,4
RO 5210 DATA 4,4,4,4,4,4,4,4,4,4
TJ 5220 DATA 4,4,0,12,12,12,12,12,12,12
```

```
WB 5230 DATA 12,12,12,12,12,12,12,12,12,1
   2
VY 5240 DATA 12,12,12,14,14,14,0,0,0,0
```

Listing 2.
BASIC listing.

```
HX 10 REM A TRIVIA GAME BY Jan Iverson
AZ 20 REM
BL 30 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:
   K7=7:K8=8:K9=9:K10=10:K11=11:K12=12:K1
   3=13:K14=14:K15=15:K16=16:K17=17
KN 40 K18=18:K19=19:K20=20:K21=21:K22=22:
   K29=29:K30=30:K34=34:K49=49:K50=50:K52
   =52:K60=60:K91=91:K100=100:K200=200
KI 50 K752=752:K35=35:K45=45:K53=53:K64=6
   4:K81=81:K96=96:K128=128:K540=540:K532
   79=53279:K764=764:K255=255
IE 60 DIM A$(30),B$(20),C$(20),D$(20),E$(
   20),F$(1),H$(1),R(3)
UT 70 DIM A1$(6000),B1$(4000),C1$(4000),D
   1$(4000),E1$(4000),F1$(200),FILE1$(15)
AJ 80 FILE1$="                    ":FILE1$="D:
   GAME.DAT":SCORE=K0:HSCORE=K0
PA 90 A=K0:B=K0:C=K0:D=K0:A1=K0:P=K0:Q=K0
   :R=K0:CNT=K0:CT=K0:XX=K0:XXX=K0
JO 100 CLOSE #K1:OPEN #K1,K4,K0,FILE1$:OP
   EN #K4,K4,K0,"K:":GOSUB 1350:POKE 5377
   4,K64:POKE K16,K64
XM 110 GOSUB 1500:POKE K752,K1:GOSUB 760
DQ 120 POSITION K2,K22:? "START=PLAY AGAI
   N  OPTION=END SESSION":POKE 53774,K64:
   POKE K16,K64:S5=45:L1=K9
AN 130 POKE 54286,192:GOSUB 1150:GOSUB 11
   70:GOSUB 1190
UA 140 POSITION K5,K3:? A$:SOUND K0,K45,K
   10,K8:FOR I=K1 TO K15:NEXT I
OT 150 POSITION K10,K9:? B$:SOUND K0,K53,
   K10,K8:FOR I=K1 TO K15:NEXT I
HO 160 POSITION K10,K11:? C$:SOUND K0,K64
   ,K10,K8:FOR I=K1 TO K15:NEXT I
IO 170 POSITION K10,K13:? D$:SOUND K0,K81
   ,K10,K8:FOR I=K1 TO K15:NEXT I
TC 180 POSITION K10,K15:? E$:SOUND K0,K96
   ,K10,K8:FOR I=K1 TO K15:NEXT I:SOUND K
   0,K0,K0,K0
QA 190 POSITION K6,K19:? "       Select 1
    - 4          "
HM 200 TM=K10:POKE K764,K255
VP 210 GOSUB 1920
XY 220 GET #K4,ANS
GS 230 IF ANS<K49 OR ANS>K52 THEN POKE K7
   64,K255:GOSUB 1920:GOTO 220
BJ 240 H$=CHR$(ANS)
VK 250 IF H$=F$ THEN SOUND K0,K45,K10,K12
   :FOR I=K1 TO K30:NEXT I:SOUND K0,K52,K
   10,K12:FOR I=K1 TO K30:NEXT I
GX 260 IF H$=F$ THEN GOSUB 700:GOTO 420
GM 270 GOSUB 710:GOTO 440
VB 280 GOSUB 730
PZ 290 POSITION K10,K3:? "RIGHT ";B;"   W
   RONG ";C
PT 300 POSITION K6,K19:? "Choose one of t
   he options ↵     "
MH 310 POSITION K2,K22:? "START":POSITION
    K20,K22:? "OPTION"
MX 320 FOR I=K1 TO K30
DV 330 IF PEEK(K53279)=K3 THEN POSITION K
   2,K22:? "START":POSITION K20,K22:? "OP
   TION":GOTO 890
RK 340 IF PEEK(K53279)=K6 THEN POSITION K
   2,K22:? "START":POSITION K20,K22:? "OP
   TION":GOSUB 730:GOTO 130
GD 350 NEXT I
CJ 360 POSITION K2,K22:? "START":POSITION
    K20,K22:? "OPTION"
NH 370 FOR I=K1 TO K30
EF 380 IF PEEK(K53279)=K3 THEN POSITION K
   2,K22:? "START":POSITION K20,K22:? "OP
   TION":GOTO 890
RU 390 IF PEEK(K53279)=K6 THEN POSITION K
   2,K22:? "START":POSITION K20,K22:? "OP
   TION":GOSUB 730:GOTO 130
FU 400 NEXT I
MQ 410 GOTO 310
OS 420 B=B+K1:SOUND K0,K0,K0,K0
DD 430 GOSUB 2050:GOTO 280
EG 440 C=C+K1:SOUND K0,K91,K12,K10:FOR DE
   LAY=K1 TO K200:NEXT DELAY:SOUND K0,K0,
   K0,K0
ZK 450 IF F$="1" THEN GOSUB 500
FD 460 IF F$="2" THEN GOSUB 550
BM 470 IF F$="3" THEN GOSUB 600
HF 480 IF F$="4" THEN GOSUB 650
PV 490 GOTO 280
EL 500 FOR I=K1 TO K4
MP 510 POSITION K7,K9:? "1.":SOUND K0,K96
   ,K10,K8:FOR DELAY=K1 TO K50:NEXT DELAY
RM 520 POSITION K7,K9:? "1.":SOUND K0,K0,
   K0,K0:FOR DELAY=K1 TO K50:NEXT DELAY
GB 530 NEXT I
ZJ 540 RETURN
EV 550 FOR I=K1 TO K4
KH 560 POSITION K7,K11:? "2.":SOUND K0,K9
   6,K10,K8:FOR DELAY=K1 TO K50:NEXT DELA
   Y
AS 570 POSITION K7,K11:? "2.":SOUND K0,K0
   ,K0,K0:FOR DELAY=K1 TO K50:NEXT DELAY
GL 580 NEXT I
ZT 590 RETURN
EM 600 FOR I=K1 TO K4
MI 610 POSITION K7,K13:? "3.":SOUND K0,K9
   6,K10,K8:FOR DELAY=K1 TO K50:NEXT DELA
   Y
CT 620 POSITION K7,K13:? "3.":SOUND K0,K0
   ,K0,K0:FOR DELAY=K1 TO K50:NEXT DELAY
GC 630 NEXT I
ZK 640 RETURN
EW 650 FOR I=K1 TO K4
PC 660 POSITION K7,K15:? "4.":SOUND K0,K9
   6,K10,K8:FOR DELAY=K1 TO K50:NEXT DELA
   Y
FN 670 POSITION K7,K15:? "4.":SOUND K0,K0
   ,K0,K0:FOR DELAY=K1 TO K50:NEXT DELAY
GM 680 NEXT I
ZU 690 RETURN
HX 700 POSITION 36,K15:? "└":RETURN
BP 710 POSITION 36,K15:? "┐":RETURN
AJ 720 POSITION K5,K9:? "
                    ":POSITION K7,K13:?
"                         ":RETURN
GA 730 POSITION K3,K3:? "
                    ":POSITION K10,K9:?
 "                  "
OS 740 POSITION K10,K11:? "
          ":POSITION K10,K13:? "
                    "
FT 750 POSITION K10,K15:? "
          ":RETURN
DW 760 FOR I=K0 TO K21:POSITION K0,I:? "|

   |":NEXT I
HB 770 POSITION K0,K5:? "├──────────────
──────────────────────────────┤"
BO 780 POSITION K0,K17:? "├──────────────
──────────────────────────────┤"
SP 790 POSITION K0,K21:? "└──────────────
──────────────────────────────┘"
TQ 800 POSITION K0,K0:? "┌──────────────
──────────────────────────────┐"
GU 810 POSITION K12,K1:? "A TRIVIA QUIZ"
NJ 820 POSITION K11,K6:? "MULTIPLE CHOICE
"
```

```
LQ 830 POSITION 35,K12:? "▄▄▄▄"
DS 840 POSITION 35,K13:? "▌..▐"
DB 850 POSITION 35,K14:? "▌  ▐"
DW 860 POSITION 35,K15:? "▌  ▐"
VC 870 POSITION 35,K16:? "▀▀▀▀"
ZU 880 RETURN
OX 890 POKE 54286,192:GOSUB 1180:GOSUB 11
   70
WZ 900 POSITION K3,K3:? "
                        "
PQ 910 POSITION K10,K3:? "RIGHT ";B;"   W
   RONG ";C
KK 920 P=B+C:Q=B/P:Q=Q*100
PD 930 IF SCORE>HSCORE THEN HSCORE=SCORE:
   POSITION 34,K7:? HSCORE
BN 940 IF Q<K50 THEN POSITION K7,K9:? "YO
   U NEED A TRIVIA CLASS":GOTO 1000
UG 950 IF Q>=75 THEN 970
JI 960 IF Q>=K50 THEN POSITION K9,K9:? "P
   RETTY GOOD, STUDY MORE   ":GOTO 1000
UM 970 IF Q>=90 THEN 990
VA 980 IF Q>=75 THEN POSITION K9,K9:? "GR
   EAT - ALMOST PERFECT    ":GOTO 1000
IW 990 IF Q>=90 THEN POSITION K5,K9:? "GR
   EAT!!! GO TO HEAD OF CLASS"
DC 1000 GOSUB 1760
AG 1010 POSITION K7,K13:? "DO YOU WISH TO
    PLAY AGAIN"
ZA 1020 POSITION K2,K22:? "START":POSITIO
   N K20,K22:? "OPTION"
GO 1030 FOR I=K1 TO K50
SI 1040 IF PEEK(K53279)=K3 THEN GRAPHICS
   0:END
GG 1050 IF PEEK(K53279)=K6 THEN GOSUB 113
   0:GOSUB 720:B=K0:C=K0:GOSUB 1180:GOSUB
    1330:GOSUB 1490:GOSUB 1350:GOTO 130
FH 1060 NEXT I
ND 1070 POSITION K2,K22:? "START":POSITIO
   N K20,K22:? "OPTION"
HD 1080 FOR I=K1 TO K50
SX 1090 IF PEEK(K53279)=K3 THEN GRAPHICS
   0:END
HP 1100 IF PEEK(K53279)=K6 THEN GOSUB 113
   0:GOSUB 730:B=K0:C=K0:GOSUB 1180:GOSUB
    1330:GOSUB 1490:GOSUB 1350:GOTO 130
EU 1110 NEXT I
NW 1120 GOTO 1020
YP 1130 POSITION K2,K22:? "START":POSITIO
   N K20,K22:? "OPTION":RETURN
PB 1140 GOTO 1130
RR 1150 POSITION K7,K9:? "1.":POSITION K7
   ,K11:? "2."
BB 1160 POSITION K7,K13:? "3.":POSITION K
   7,K15:? "4.":RETURN
NA 1170 FOR I=K9 TO K15:POSITION K9,I:? "
                          ":RETURN
AL 1180 FOR I=K9 TO K15:POSITION K3,I:? "
                          ":NEXT I:RETURN
NO 1190 CT=INT(RND(K0)*CNT)
MT 1200 IF CT=K0 THEN 1190
YZ 1210 IF XXX=CNT THEN 890
LF 1220 IF XX=CNT-K1 THEN CT=CNT:XXX=CNT
AM 1230 IF F1$(CT,CT)="X" THEN 1190
CY 1240 A$=A1$(CT*K30-K29)
CE 1250 B$=B1$(CT*K20-K19)
CW 1260 C$=C1$(CT*K20-K19)
DO 1270 D$=D1$(CT*K20-K19)
EG 1280 E$=E1$(CT*K20-K19)
LX 1290 F$=F1$(CT,CT)
SS 1300 F1$(CT,CT)="X"
VK 1310 XX=XX+K1
AL 1320 RETURN
ZQ 1330 A=K0:B=K0:C=K0:D=K0:A1=K0:P=K0:Q=
   K0:R=K0:CT=K0
AR 1340 RETURN
DI 1350 INPUT #K1,A$,B$,C$,D$,E$,F$
QO 1360 IF A$="ZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
   ZZZZ" THEN 1460
KV 1370 CNT=CNT+K1
IZ 1380 IF CNT=200 THEN 1460
KD 1390 A1$(CNT*K30-K29)=A$
IZ 1400 B1$(CNT*K20-K19)=B$
KF 1410 C1$(CNT*K20-K19)=C$
LL 1420 D1$(CNT*K20-K19)=D$
MR 1430 E1$(CNT*K20-K19)=E$
LX 1440 F1$(CNT,CNT)=F$
RI 1450 GOTO 1350
OP 1460 SCORE=K0:POSITION K34,K6:? "
   ":POSITION K34,K6:? "0"
AW 1470 CLOSE #K1
BF 1480 RETURN
GY 1490 POKE 54286,64:CT=0:CNT=0:XX=K0:XX
   X=K0:CLOSE #K1:OPEN #K1,K4,K0,FILE1$:R
   ETURN
YH 1500 GRAPHICS K0:POKE 752,K1:POKE 710,
   K0:POSITION K10,K10:? "20 Seconds plea
   se..."
CF 1510 FOR N=K0 TO 99:READ X:POKE 1664+N
   ,X:NEXT N
UK 1520 COLTAB=1712:LUMTAB=COLTAB+24
OL 1530 X=USR(1693)
BX 1540 POKE 512,128
JJ 1550 POKE 513,K6
JX 1560 DSTART=PEEK(560)+256*PEEK(561)
MT 1570 FOR N=DSTART+K6 TO DSTART+28
XY 1580 POKE N,130
ID 1590 NEXT N
SN 1600 POKE DSTART+3,194
CC 1610 POKE 54286,192
MP 1620 PRINT CHR$(125)
TJ 1630 POKE 710,PEEK(COLTAB)
KD 1640 POKE 709,PEEK(LUMTAB)
BA 1650 RETURN
OI 1660 DATA 72,138,72,174,156,6,189,176,
   6,141
IG 1670 DATA 10,212,141,24,208,189,200,6,
   141,23
ZU 1680 DATA 208,238,156,6,104,170,104,64
   ,18,104
NW 1690 DATA 169,7,160,168,162,6,32,92,22
   8,96
NO 1700 DATA 169,1,141,156,6,76,98,228,14
   4,144
MM 1710 DATA 144,144,144,144,196,196,196,
   196,196,196
PW 1720 DATA 196,196,196,196,196,64,64,64
   ,64,64
IS 1730 DATA 2,0,12,12,12,12,12,12,12,12
WK 1740 DATA 12,12,12,12,12,12,12,12,12,1
   2
AA 1750 DATA 12,12,12,12,6,6,0,0,0,0
RM 1760 RESTORE 1880
EP 1770 FOR I=K1 TO 24
SK 1780 READ P0,P1,P2,P3,DUR
WW 1790 POKE K540,DUR
IM 1800 SOUND K0,P0,K10,K12
JS 1810 SOUND K1,P1,K10,K12
KY 1820 SOUND K2,P2,K10,K12
ME 1830 SOUND K3,P3,K10,K12
UR 1840 IF PEEK(K540)<>K0 THEN 1840
FU 1850 NEXT I
PF 1860 SOUND K0,K0,K0,K0:SOUND K1,K0,K0,
   K0:SOUND K2,K0,K0,K0:SOUND K3,K0,K0,K0
BK 1870 RETURN
AN 1880 DATA 108,0,0,0,10,102,0,0,0,10,96
   ,243,0,0,10,60,121,162,0,20,96,0,0,0,1
   0,60,162,0,0,20,96,121,136,0,20
US 1890 DATA 60,0,0,0,20,60,182,0,0,20,60
   ,121,144,0,20,60,193,0,0,10,60,193,0,0
   ,10,53,121,162,0,10,50,121,162,0,10
WQ 1900 DATA 47,162,0,0,10,60,162,0,0,10,
   53,96,121,0,10,47,96,121,0,10,47,162,0
   ,0,10,64,162,0,0,10,53,91,128,0,20
IJ 1910 DATA 60,96,121,0,20,60,162,0,0,20
   ,0,243,0,0,30
GC 1920 POSITION K3,K6:? "TIME=":POSITION
```

```
    28,K6:? "SCORE=":POSITION 25,K7:? "HI
   -SCORE=":POSITION K8,K6:? TM
IV 1930 POKE K540,K60
XV 1940 IF PEEK(K540)=K0 THEN TM=TM-K1:PO
   SITION K8,K6:? "  ":POSITION K8,K6:? T
   M:SOUND K0,K128,K10,K10:GOTO 1980
SY 1950 IF PEEK(K764)<>K255 THEN 1990
KJ 1960 IF TM=K0 THEN 2010
UF 1970 GOTO 1940
KH 1980 FOR J=K1 TO K2:NEXT J:SOUND K0,K0
   ,K0,K0
KK 1990 IF PEEK(K764)<>K255 THEN ANS=PEEK
   (764):RETURN
SG 2000 GOTO 1930
SB 2010 SOUND K0,K96,K10,K12:GOSUB 2030:A
   NS=53:POSITION K8,K6:? "  ":POP :GOTO
   240
AG 2020 RETURN
JR 2030 FOR K=K1 TO K30:NEXT K
NH 2040 SOUND K0,K0,K0,K0:RETURN
EF 2050 IF TM=K0 THEN RETURN
ZY 2060 TM=TM-K1:POSITION K8,K6:? "  ":PO
   SITION K8,K6:? TM:SOUND K0,K128,K10,K6
   :FOR I=K1 TO K3:NEXT I
SS 2070 SOUND K0,K0,K0,K0
FY 2080 FOR I=K1 TO K10:NEXT I
XB 2090 SCORE=SCORE+K1:POSITION K34,K6:?
   SCORE:SOUND K0,162,K10,K6:FOR I=K1 TO
   K3:NEXT I:SOUND K0,K0,K0,K0
FP 2100 FOR I=K2 TO K10:NEXT I
PS 2110 GOTO 2050
AI 2120 RETURN
```

Invasion III

Defeat the Zorcrons

by Jerry Lemaitre

In this game, you're the lowly Anthort, struggling to defend your planet against the evil Zorcron empire. If the rock-eating Zorcrons manage to penetrate your defenses, they'll gobble up your entire planet. To prevent this, you're armed with the mystical Fyreballs, which ignite anything in their path. You may have three of these flying at one time, but shoot carefully!

You're not the only one with weapons, though. The Zorcrons have discovered machinery! There are three types of machines that they build with the iron ore they can't digest.

Their Eggbarge is a bulky space vessel which incubates Zorcron eggs during flight. When it reaches its destination, the newly hatched Zorcrons help to replenish the fighting troops.

A Whizzer is a warp-speed vessel which transports and launches the most deadly Zorcron offense of all—the Zingbomb. You'll know the Zingbombs when you see them. These menaces head straight for your planet at incredible speeds. On impact, they create a shock wave that will pulverize your delicate Anthortian insides.

Now, don't get me wrong. This isn't just another one-dimensional shoot-'em-up. Constantly changing colors and totally animated characters add to the visual appeal. You can move your Anthort in eight (count 'em, eight) directions. There's also horizontal wraparound, so you're not confined by the sides of the screen.

Even though **Invasion III** is written in BASIC, there can be as many as twenty-three characters on-screen at a time—at speeds that'll make you sweat! Enjoy!

Jerry Lamaitre has owned his Atari 400 for four years. He's very interested in robotics and artificial intelligence, and sells his own programs and accessories as a small mail-order business.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
VK 100 GOSUB 570
MP 110 SC=K0:MEN=K2:GOSUB 720
```

```
AE 120 POKE P+SCR,K2:IF STICK(K0)=K15 AND
    STRIG(K0) THEN 120
QD 130 FOR Z=K0 TO K15:P1=P:S=M(STICK(K0)
   ):P=P+S:P=P-S*(P>439 OR P<40)
QO 140 C=INT(K5*RND(K0)):SETCOLOR C,Z,K8*
   (C<K4)
PN 150 CH= NOT CH:POKE 756,CH(CH):L=PEEK(
   P+SCR):POKE P1+SCR,K0:POKE P+SCR,K2:IF
    L AND L<>K2 THEN 410
XV 160 L=K0:IF  NOT STRIG(K0) THEN GOSUB
   330
ZZ 170 FOR A=K0 TO K2:F1=F(A)
RJ 180 IF F1 THEN POKE F1+SCR,K0:F1=F1-K2
   0:FL=PEEK(F1+SCR):IF FL THEN F=F1:F1=K
   0:GOSUB 350
EB 190 IF F1 THEN POKE F1+SCR,132
SZ 200 F(A)=F1:NEXT A
SB 210 IF ZP(Z) THEN ZP1=ZP(Z):ZP(Z)=ZP(Z
   )+INT(K3*RND(K0)+19):L=PEEK(ZP(Z)+SCR)
   :POKE ZP1+SCR,K0:POKE ZP(Z)+SCR,195
WB 220 IF L THEN GOSUB 460
ZB 230 IF  NOT E THEN IF RND(K0)<0.01 THE
   N E=K20:EM=K1:SOUND K1,E,12,K8:IF RND(
   K0)<0.5 THEN E=39:EM=-K1
QG 240 IF E THEN E=E+EM:POKE E+SCR,K6:POK
   E E-EM+SCR,K0:IF E=K20 OR E=39 THEN PO
   KE E+SCR,K0:E=K0:SOUND K1,K0,K0,K0
ZN 250 IF E THEN IF E>21 AND E<38 THEN IF
    NOT ZP(E-22) THEN ZP(E-22)=E+K20:POK
   E ZP(E-22)+SCR,195:SOUND K1,E,12,K8
JD 260 IF W=K0 THEN IF RND(K0)<0.01 THEN
   W=K20:WM=K1:SOUND K2,W,10,10:POKE 77,K
   0:IF RND(K0)<0.5 THEN W=39:WM=-K1
KN 270 IF W THEN W=W+WM:POKE W+SCR,133:PO
   KE W-WM+SCR,K0:IF W=K20 OR W=39 THEN P
   OKE W+SCR,K0:W=K0:SOUND K2,K0,K0,K0
VX 280 IF W THEN IF  NOT B AND RND(K0)<0.
   06 THEN B=W+K20:SOUND K3,B,K4,12
ML 290 IF B THEN SOUND K3,B,K4,12:B=B+K20
   :BL=PEEK(SCR+B):POKE SCR+B,71:POKE SCR
   +B-K20,K0
XN 300 IF B THEN IF BL=K2 OR B>439 THEN S
   OUND K3,K0,K0,K0:POKE SCR+B,K0:B=K0:GO
   TO 410
FZ 310 IF B THEN IF BL=132 THEN FL=71:GOS
   UB 390
KE 320 NEXT Z:GOTO 130
ZK 330 FOR J=K0 TO K2:IF  NOT F(J) THEN F
   OR I=K0 TO 31:POKE 53761,I:NEXT I:F(J)
   =P-K20:RETURN
NB 340 NEXT J:RETURN
DC 350 IF FL=195 THEN FOR I=K0 TO K15:IF
   ZP(I)=F THEN SC=SC+25:GOSUB 530:Z=I:GO
   SUB 460
VZ 360 IF FL=195 THEN NEXT I:RETURN
VN 370 IF FL=133 THEN SC=SC+1000:GOSUB 53
   0:Q=W:W=K0:SOUND K2,K0,K0,K0:GOTO 550
FW 380 IF FL=K6 THEN SC=SC+1000:GOSUB 530
   :Q=E:E=K0:SOUND K1,K0,K0,K0:GOTO 550
NL 390 IF FL=71 THEN SC=SC+500:GOSUB 530:
   Q=B:B=K0:SOUND K3,K0,K0,K0:GOTO 550
ZA 400 RETURN
EH 410 POKE P+SCR,K8:FOR I=K1 TO K20:SOUN
   D K3*RND(K0),200*RND(K0),12,K8:NEXT I
ZV 420 MEN=MEN-K1:POKE P+SCR,9:FOR I=K1 T
   O K20
YC 430 SOUND K2*RND(K0),200*RND(K0),12,K8
   :NEXT I:FOR I=K0 TO K3:SOUND I,K0,K0,K
   0:NEXT I
XJ 440 POKE P+SCR,K0:FOR I=K0 TO 100:SETC
   OLOR K4*RND(K0),K15*RND(K0),K15*RND(K0
   ):NEXT I:IF MEN<K0 THEN 500
CY 450 GOSUB 720:GOTO 120
YL 460 IF ZP(Z)=K0 OR L=195 THEN RETURN
NX 470 IF L=74 OR L=75 OR L=K2 THEN 410
GU 480 POKE ZP(Z)+SCR,200:FOR I=K0 TO K15
   :SOUND K0,K0,K4,I:NEXT I:POKE ZP(Z)+SC
   R,201
YU 490 FOR I=K0 TO K15:SOUND K0,K0,K5,I:N
   EXT I:POKE ZP(Z)+SCR,K0:ZP(Z)=K0:RETUR
   N
UW 500 GOSUB 720:POSITION K5,K6:? #K6;"Ga
   ME OvER!":POSITION K5,K8:? #K6;"PRESS
   FIRE":POKE 17+SCR,K0
DI 510 IF STRIG(K0) THEN 510
LZ 520 GOTO 110
UI 530 POSITION K4,23:? #K6;SC:IF SC>HSC
   THEN HSC=SC:POSITION 14,23:? #K6;HSC
ZJ 540 RETURN
XO 550 SOUND K2,34,10,10:FOR I=K0 TO 16:F
   OR J=K0 TO K1:POKE SCR+Q,K8+J:SOUND K1
   ,I+J,10,11+J:NEXT J:NEXT I
FU 560 POKE SCR+Q,K0:SOUND K1,K0,K0,K0:SO
   UND K2,K0,K0,K0:RETURN
EL 570 K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K8=8
   :K15=15:K20=20
YN 580 GRAPHICS 18:POSITION K4,K2:? #K6;"
   InVASiON JiI":POSITION 9,K5:? #K6;"by"
   :POSITION K3,7
ZX 590 ? #K6;"JERRY LEMAITRE":COLOR 138:P
   LOT K0,K0:DRAWTO 19,K0:DRAWTO 19,11:DR
   AWTO K0,11:DRAWTO K0,K0
KQ 600 DIM CH(K1),M(K15),ZP(K15),F(K2),J$
   (39):M(K5)=21:M(K6)=-19:M(7)=K1:M(9)=1
   9:M(10)=-21:M(11)=-K1:M(13)=K20
AB 610 M(14)=-K20:M(K15)=K0:FOR I=K1 TO 3
   9:READ A:J$(I)=CHR$(A):NEXT I
TK 615 DATA 104,104,133,215,104,133,214,1
   04,133,217,104,133
LU 620 DATA 216,104,133,218,104,170,160,0
   ,177,214,145,216,200,208,4,230,215,230
   ,217,202,208,242,198,218,16,238,96
DX 630 CH(K0)=PEEK(106)-K8:CH(K1)=CH(K0)-
   K8:I=USR(ADR(J$),57344,CH(K0)*256,511)
XW 640 A=CH(K0)*256:FOR I=K0 TO 95:READ B
   :POKE A+I,B:FOR J=K0 TO K3:SETCOLOR J,
   K15*RND(K0),J+J+K4:NEXT J:NEXT I
HA 650 I=USR(ADR(J$),CH(K0)*256,CH(K1)*25
   6,511)
HY 660 A=CH(K1)*256:FOR I=16 TO 63:READ B
   :POKE A+I,B:NEXT I:RETURN
QW 670 DATA 0,0,0,0,0,0,0,0,7,15,30,56,48
   ,0,192,192,129,90,60,219,126,36,72,144
   ,129,129,165,219,126,60,36,36
OI 680 DATA 0,20,8,20,8,0,8,0,255,129,189
   ,181,181,181,133,253,60,102,255,24,255
   ,171,255,126
NO 690 DATA 0,14,24,24,24,60,60,24,0,34,2
   0,104,22,40,68,0,65,34,0,192,3,0,68,13
   0,0,16,60,127,255,255,255,0
OQ 700 DATA 0,0,65,225,251,255,255,0,66,9
   0,60,219,126,36,18,9,0,0,36,90,255,189
   ,153,153,0,8,20,8,0,8,0,8
MA 710 DATA 191,161,173,173,173,189,129,2
   55,60,102,255,24,255,213,255,126,0,112
   ,24,24,24,60,60,24
QK 720 GRAPHICS 17:POSITION K0,K0:? #K6;"
    INVASION III ":POSITION K1,23:?
    #K6;"SCR";SC:POSITION 10,23
ID 730 ? #K6;"HSCR";HSC:POKE 756,CH(K0):F
   OR I=K0 TO 19:COLOR INT(K2*RND(K0)+10)
CK 740 PLOT I,22:SOUND K0,I+I+I,10,10:NEX
   T I
ZO 750 SOUND K0,K0,K0,K0:SCR=PEEK(88)+256
   *PEEK(89):P=350:FOR I=K0 TO K15:ZP(I)=
   K0:NEXT I:IF MEN THEN POKE 17+SCR,130
ZF 760 IF MEN=K2 THEN POKE 18+SCR,130
CO 770 B=K0:W=K0:E=K20:EM=K1:F(K0)=K0:F(K
   1)=K0:F(K2)=K0:RETURN
```

48K Cassette or Disk

GAME

Dragon Chase

Get the dragon away from your princess.

by David Huff

The game of **Dragon Chase** depends more on a sharp mind than on quick reflexes. The object here is to rescue the princess before an evil dragon can reach her.

In order to save the princess, you must remove the black castle which surrounds her. To accomplish this feat, you must find certain objects—such as diamonds, swords and rings. Unfortunately, these items are hidden from view until you move over them. And, if the dragon reaches your fair lady, the game is over.

The game rules.

You begin each level in the lower left corner, marked by a square pink cursor. As you move with the joystick, objects hidden below become visible. A row of the things you must find is seen at the upper left, and you must uncover the objects in the order shown. When you locate one, stay over it until its color changes, then move on to find the next one. After you've retrieved all the required items, the castle is automatically removed. To rescue the princess and advance to the next level, move your pink cursor over her.

Also hidden are various objects which can help or hinder you. The squares can make the whole field visible for a few seconds, giving you time to locate other needed items.

Wild cards are also randomly hidden. These are marked with a *W*. Finding one is the same as locating the next object you were searching for.

Also hidden are dragons. If you hover above one, your movement is stopped—and the dragon takes another step toward the princess. A tombstone marks this event. Sometimes the dragon may be sleeping, in which case you can step right over him.

For help in finding things, a scanner is provided. Press the fire button to activate it, and a portion of the screen around you becomes visible. Using the scanner costs you 10 points and advances the dragon one step.

Each round of **Dragon Chase** has five levels. On higher levels, the dragon moves faster—while you must find more objects.

Scores are tallied as follows. You receive 200 points for saving the princess, or 100 points for finding an object. You lose points in this way: 50 off for finding a dragon, 5 are subtracted for advancing the dragon, and using the scanner eats up 10.

About the program.

Dragon Chase takes advantage of Atari's character color assignment in graphics mode 2. The same character is easily displayed in different colors, and there are sixty-four characters in the graphics 2 character set. You have a choice of four colors. In choosing one, a specific number is added to the character number, as shown in Figure 1.

To display a character, POKE its number into the dis-

play memory, adding the indicated amount to choose a color register. Character numbers are shown on page 55 of Atari's BASIC Reference Manual.

Figure 1.

COLOR REGISTER				
Character No.	708	709	710	711
0 - 63	+0	+64	+128	+192

Dragon Chase uses a redefined character set at Line 1000. Line 340 pokes random characters into display memory. Their number has *192* added to it, specifying color register 711.

Similarly, Line 280 randomly selects the characters for you to find; by adding *128* to them, color register 710 is chosen.

The princess awaits your help.

David Huff, a D.D.S., is currently studying for a specialty degree in Orthodontics. He's had his Atari 1200XL for three years and is currently working on a program for orthodontic x-ray analysis.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
XY 10 REM DRAGON CHASE
RA 15 REM DAVID E. HUFF
LF 20 GRAPHICS 18:PMBASE=PEEK(106)-16:CHS
   ET=256*(PMBASE+8)
ST 25 DIM E$(50)
HS 30 DM=PEEK(88)+256*PEEK(89):DL=PEEK(56
   0)+256*PEEK(561)
QL 40 P0=PMBASE*256+512:P1=P0+128:P2=P1+1
   28:P3=P2+128
AT 50 POKE 54279,PMBASE:POKE 53277,3:POKE
    559,46
OV 60 POKE 53257,1:POKE 53258,1:POKE 5325
   9,3
II 70 POKE 712,20:POKE 709,12:POKE 708,20
   0
WU 80 POSITION 4,5:? #6;"dragon chase"
WK 90 GOSUB 1000:GOSUB 1500
PP 100 REM SET UP DISPLAY LIST INTERRUPT
CD 110 POKE DL+15,135:POKE DL+16,160:POKE
    DL+17,7:POKE DL+18,65
BP 120 POKE DL+19,PEEK(560):POKE DL+20,PE
   EK(561)
BM 130 POKE 512,197:POKE 513,6:POKE 54286
   ,192
KS 140 REM INITIAL VALUES
HO 150 DIF=50:SCORE=0:LEVEL=0:FIELD=2:R=1
VV 160 REM START NEW LEVEL HERE
AK 170 CLEAR=USR(ADR(E$),PMBASE*256)
MO 180 LEVEL=LEVEL+1:FIELD=FIELD+1:IF FIE
   LD=8 THEN GOSUB 2400
UB 190 POSITION 0,0:? #6;"K"
PZ 200 POSITION 1,11:? #6;"ROUND ";R;"
   LEVEL ";LEVEL
AI 210 REM COLORS
CG 220 POKE 623,0:POKE 705,0:POKE 706,88:
   POKE 707,34:POKE 708,12
MB 230 POKE 709,200:POKE 710,120:POKE 711
   ,20:POKE 712,12
OU 250 REM SET UP FIELD
BM 260 FOR I=20 TO 219:POKE DM+I,1+192:NE
   XT I
TH 270 FOR I=0 TO FIELD-1
CI 280 POKE DM+I*2,INT(RND(0)*10)+2+128
GK 290 NEXT I
TE 300 FOR I=0 TO 60
OU 310 RP=INT(RND(0)*200)+20
OI 320 RC=INT(RND(0)*12)+2+192
YF 330 SOUND 2,RP,10,4
SX 340 POKE RP+DM,RC
GD 350 NEXT I
WV 360 SOUND 2,0,0,0
NE 370 FOR I=1 TO 10:RPD=INT(RND(0)*200)+
   20:POKE DM+RPD,15+192:NEXT I
PR 380 FOR I=1 TO 3:RPW=INT(RND(0)*200)+2
   0:POKE DM+RPW,20+192:NEXT I
TB 390 POKE 712,20:REM HIDE CHARACTERS
UP 400 GOSUB 1900
TT 410 GOSUB 1700
IM 450 COUNT=0:CHR=PEEK(DM+COUNT)-128
CD 460 DRAGON=0:HID=0
TS 470 GOSUB 2500
JZ 500 REM START MAIN PROGRAM LOOP
VD 505 POKE 53278,1
MU 510 POKE 1790,1
XU 515 IF PEEK(53261)=4 THEN 2000
MN 520 MEM=DM+20+20/8*(PEEK(203)-23)+(PEE
   K(205)-48)/8
IC 525 CHR1=PEEK(MEM)-192
ZF 530 IF CHR1=CHR THEN GOSUB 700
ZD 535 IF CHR1=20 THEN GOSUB 700
BW 540 IF CHR1=15 THEN GOSUB 800
CJ 545 IF CHR1=-174 THEN GOTO 900
EZ 550 IF CHR1=12 THEN POKE 712,16:FOR I=
   0 TO 350:SOUND 0,350-I,10,10:NEXT I:PO
   KE 712,20:POKE MEM,12+128
OV 555 IF STRIG(0)=0 THEN GOSUB 2100
LL 560 LOOP=LOOP+1:POKE 77,0
HT 570 IF LOOP<DIF THEN 515
CP 580 LOOP=0:GOSUB 600
QJ 590 GOTO 515
HV 600 REM MOVE DRAGON
KY 610 FOR I=15 TO 17:POKE DM+20+DRAGON,I
   :FOR J=0 TO 15:SOUND 2,100,12,10:NEXT
   J:SOUND 2,100+I,12,20-I:NEXT I
YR 620 FOR I=16 TO 15 STEP -1:POKE DM+20+
   DRAGON,I:FOR J=0 TO 15:SOUND 2,100,12,
   10:NEXT J:SOUND 2,100+I,12,20-I
GC 630 NEXT I
RW 640 FOR I=0 TO 15:NEXT I
BC 650 SCORE=SCORE-5:GOSUB 2500:SOUND 2,0
   ,0,0
RM 660 POKE DM+20+DRAGON,19:POKE DM+21+DR
   AGON,15
QV 670 DRAGON=DRAGON+1
NT 680 IF DRAGON=18 THEN POP :GOTO 2200
ZU 690 RETURN
HD 700 REM YOU FOUND ONE
MU 710 IF CHR+64<0 THEN RETURN
YL 720 SOUND 0,10,10,10
PH 730 POKE DM+COUNT,CHR+64:POKE MEM,CHR+
   64
OB 740 COUNT=COUNT+2
MV 750 CHR=PEEK(DM+COUNT)-128
UG 755 GOSUB 2500
PK 760 FOR I=0 TO 20:NEXT I
UV 770 SCORE=SCORE+100:GOSUB 2500:SOUND 0
   ,0,0,0
PX 780 HID=HID+1:IF HID=FIELD THEN GOSUB
   1800
ZV 790 RETURN
VD 800 REM YOU FOUND A DRAGON
MI 810 POKE 1790,0
JR 820 FOR I=15 TO 17:POKE MEM,I:FOR J=0
   TO 15:SOUND 2,100,12,10:NEXT J:SOUND 2
```

```
   ,100+I,12,20-I:NEXT I
TP 825 GOSUB 600
NA 830 FOR I=16 TO 15 STEP -1:POKE MEM,I:
   FOR J=0 TO 15:SOUND 2,100,12,10:NEXT J
   :SOUND 2,100+I,12,20-I:NEXT I
SA 850 FOR I=0 TO 15:NEXT I
VI 860 SCORE=SCORE-50:GOSUB 2500
OG 870 POKE MEM,14:SOUND 2,0,0,0
NL 880 POKE 1790,1
ZW 890 RETURN
WA 900 REM YOU SAVED PRINCESS
MJ 910 POKE 1790,0
QD 920 FOR I=1 TO 10:SOUND 0,10*I,10,I
IR 930 FOR K=1 TO 5:NEXT K:POKE 712,20
RU 940 FOR J=14 TO 0 STEP -1
GC 950 SOUND 0,10,10,J
ZN 960 SOUND 1,14,10,J+2:NEXT J
XW 970 POKE 712,12:SCORE=SCORE+20:GOSUB 2
   500:NEXT I
EE 980 SOUND 0,0,0,0:SOUND 1,0,0,0
OU 990 GOTO 160
JB 995 REM REDEFINED CHRACTER SET
II 1000 FOR I=0 TO 511:POKE CHSET+I,PEEK(
   57344+I):NEXT I
NT 1010 POKE 756,CHSET/256
VG 1020 READ N:IF N=-1 THEN RETURN
DZ 1030 FOR I=0 TO 7:READ D:POKE CHSET+N*
   8+I,D:NEXT I:GOTO 1020
QO 1035 DATA 0,0,0,0,0,0,0,0
AG 1040 DATA 1,0,0,0,24,24,0,0,0
TO 1050 DATA 2,0,126,90,126,126,36,60,0
NT 1060 DATA 3,73,42,28,119,28,42,73,0
GI 1070 DATA 4,165,66,165,24,24,165,66,16
   5
YK 1080 DATA 5,0,24,60,126,126,60,24,0
LP 1090 DATA 6,0,24,36,66,66,36,24,0
ZG 1100 DATA 7,0,2,4,8,80,32,80,0
YN 1110 DATA 8,0,54,127,127,62,28,8,0
OB 1120 DATA 9,195,195,36,24,24,36,195,19
   5
ZR 1130 DATA 10,0,224,160,255,255,170,234
   ,0
ZG 1140 DATA 11,60,126,219,255,90,102,60,
   0
IB 1150 DATA 14,60,66,153,189,153,153,129
   ,255
RE 1160 DATA 15,112,208,255,170,213,127,0
   ,0
FR 1170 DATA 16,112,208,255,234,128,213,1
   27,0
SF 1180 DATA 17,112,208,255,234,128,208,1
   17,31
ZY 1200 DATA 18,60,60,24,126,24,24,36,66
OK 1210 DATA 19,24,60,127,255,255,254,68,
   68
KI 1220 DATA 12,126,129,153,165,165,153,1
   29,126
DD 1230 DATA 13,90,189,90,231,231,90,189,
   90
RV 1240 DATA 20,0,195,195,219,255,231,195
   ,0
DV 1250 DATA 21,0,0,0,240,152,240,176,152
GN 1260 DATA 22,0,0,0,234,170,234,138,142
HI 1270 DATA 23,0,0,0,138,200,170,154,138
RR 1280 DATA 24,0,0,0,0,0,0,0,0
GS 1290 DATA 25,0,0,134,135,126,126,100,6
   8
EK 1300 DATA -1
ZQ 1495 REM VBLANK ROUTINE
EE 1500 FOR I=0 TO 250:READ D:POKE 1536+I
   ,D
OR 1510 SOUND 2,D,10,4:NEXT I
FK 1515 FOR I=1 TO 42:READ D:E$(I,I)=CHR$
   (D):NEXT I
HR 1520 FOR I=3 TO 15
LK 1530 SOUND 2,100,12,I
VZ 1540 FOR J=15 TO 17
WX 1550 POKE DM+100+I,J:FOR N=0 TO 10:SOU
   ND 1,200,12,8:NEXT N
WV 1560 POKE DM+100+I,19:SOUND 1,0,0,0
XB 1570 NEXT J:NEXT I
JC 1580 SOUND 2,0,0,0
BK 1590 RETURN
OK 1600 DATA 216,169,0,141,3,210,173,13,2
   08,208,8,173,255,6,240,6,206,255,6,76,
   98,228,169,15,141
QP 1610 DATA 255,6,173,254,6,240,243,174,
   120,2,224,7,240,14,224,11,240,24,224,1
   4,240,37,224,13,240
SS 1620 DATA 50,208,222,165,205,201,200,2
   40,216,24,165,205,105,8,76,76,6,165,20
   5,201,48,240,202,56,233
BL 1630 DATA 8,133,205,141,2,208,76,138,6
   ,165,203,201,15,240,185,32,123,6,56,16
   5,203,233,8,76,115
LI 1640 DATA 6,165,203,201,95,240,168,32,
   123,6,24,165,203,105,8,133,203,32,128,
   6,76,138,6,169,0
PR 1650 DATA 76,130,6,169,240,160,8,145,2
   03,136,208,251,96,169,168,141,3,210,16
   9,96,141,2,210,76,19
FX 1660 DATA 6,104,104,133,204,104,133,20
   3,104,104,133,205,104,133,207,104,133,
   206,230,208,165,208,201,13,208
SX 1670 DATA 12,169,213,160,5,145,206,24,
   105,1,136,208,248,160,0,162,6,169,7,76
   ,92,228,72,169,26
CB 1680 DATA 141,10,212,141,26,208,169,21
   8,141,0,2,169,6,141,1,2,104,64,72,138,
   72,169,224,162,200
WY 1690 DATA 141,10,212,141,9,212,142,24,
   208,142,26,208,169,197,141,0,2,169,6,1
   41,1,2,104,170,104,64
SM 1695 DATA 104,104,133,207,104,133,206,
   162,4,169,0,168,145,206,136,208,251,23
   0,207,202,208,246,96
PL 1696 DATA 104,104,133,204,104,133,203,
   104,104,160,0,145,203,200,192,220,208,
   249,96
MI 1700 REM CASTLE DATA
RZ 1710 Y1=17:FOR I=0 TO 16:READ D:POKE P
   1+I+Y1,D:NEXT I
NJ 1720 RESTORE 1730
HJ 1730 DATA 24,153,153,153,255,195,129,1
   29,129,129,129,129,129,129,129,255,255
QO 1740 POKE DM+38,18:POKE 53249,188
BC 1750 RETURN
WZ 1800 REM REMOVE CASTLE
VN 1810 FOR I=0 TO 40:NEXT I
XG 1820 FOR I=16 TO 0 STEP -1
DB 1830 POKE P1+I+Y1,0
VX 1840 SOUND 0,10,10,I
VV 1850 SOUND 1,12,10,I+1
YK 1860 SOUND 2,14,10,I+2
WG 1870 FOR J=0 TO 5:NEXT J:NEXT I
ZR 1880 FOR I=0 TO 2:SOUND I,0,0,0:NEXT I
BQ 1890 RETURN
FA 1900 REM INITIAL PLAYER POSITION
RM 1910 Y=95:X2=48:POKE 53250,X2
AY 1920 FOR I=1 TO 8
UA 1930 POKE P2+I+Y,240:NEXT I
PI 1940 D=USR(1687,P2+Y,X2,DM+210)
BG 1950 RETURN
DU 2000 REM PLAYER TOUCHES CASTLE
TF 2010 POKE 1790,0
TV 2020 FOR I=0 TO 80:SOUND 2,127-I,8,6
QG 2030 POKE P2+I,0:NEXT I
BJ 2040 GOSUB 1900
UN 2050 SOUND 2,0,0,0:GOTO 500
UP 2100 REM SCANNER
HY 2110 POKE 53251,PEEK(205)-12:Y3=PEEK(2
   03)-10
IO 2120 FOR I=0 TO 30:POKE P3+I+Y3,255:SO
   UND 0,I,10,8:NEXT I
HI 2130 SOUND 0,0,0,0
XO 2140 GOSUB 600
```

```
VM 2150 FOR I=0 TO 15:NEXT I:SCORE=SCORE-
   5:GOSUB 2500
OM 2160 FOR I=0 TO 30:POKE P3+I+Y3,0:SOUN
   D 0,30-I,10,10:NEXT I
AS 2165 POKE 53278,1:RETURN
JF 2200 REM DRAGON GETS PRINCESS
WP 2210 POKE 1790,0:CLEAR=USR(ADR(E$),PMB
   ASE*256)
DN 2220 GOSUB 2300:POSITION 1,11:? #6;"
     PRESS FIRE    "
VV 2230 FOR I=0 TO 35:IF STRIG(0)=0 THEN
   POP :GOTO 140
FG 2240 NEXT I
JT 2260 GOSUB 2300:GOSUB 2500
WH 2270 FOR I=0 TO 35:IF STRIG(0)=0 THEN
   POP :GOTO 140
FS 2280 NEXT I
QD 2290 GOTO 2220
VK 2300 FOR J=15 TO 17
VU 2310 SOUND 3,186+J,10,6:SOUND 2,185+J,
   10,6
XH 2320 X=USR(ADR(E$)+23,DM,J)
PH 2340 FOR I=0 TO 25:NEXT I:NEXT J
WZ 2350 SOUND 3,0,0,0:SOUND 2,0,0,0
AY 2360 RETURN
RN 2400 REM NEXT ROUND
PI 2410 DIF=DIF-10:IF DIF<10 THEN DIF=10
GT 2420 FIELD=3:LEVEL=1
CB 2430 POSITION 0,11:? #6;"K"
SJ 2440 POSITION 2,11:? #6;"ROUND ";R;" C
   OMPLETE":FOR I=0 TO 200 STEP 2
RP 2450 SOUND 0,254-I,10,8:SOUND 1,252-I,
   10,8
ZI 2460 SOUND 2,250-I,10,8:SOUND 3,248-I,
   10,8
FT 2470 NEXT I
AB 2480 FOR I=0 TO 3:SOUND I,0,0,0:NEXT I
WZ 2490 R=R+1:RETURN
PU 2500 REM PRINT SCORE
DB 2510 IF SCORE<0 THEN SCORE=0
MM 2520 POSITION 1,11:? #6;"
        ":REM 17 SPACES
XK 2530 POSITION 4,11:? #6;"SCORE ";SCORE
AW 2540 RETURN
```

●

Krebs removal

Get these fission inhibitors out of the power plant.

by Chuck Rosko

The nuclear industry is looking for an adventurous individual who's willing to tackle a high-risk job. Surprise! That's you. Your task is to remove the krebs located in the reactor cores of the Nittany Memorial Power Plant. A kreb, of course, is a uranium pellet which is no longer radioactive. The krebs inhibit fission, so they must be removed and replaced with new radioactive uranium.

Your joystick (port 1) controls your atomic core scrubber. Move over each kreb (in green), and your scrubber will remove it, replacing it with a radioactive uranium pellet (in red). If you run into any of the radioactive pellets, your scrubber will be destroyed. The big spenders of the industry are paying you $5.00 for every kreb removed.

At the bottom right-hand side of the screen is a readout of the amount of energy in your scrubber. You must replace all the krebs before your energy runs out, or—again—your scrubber will be annihilated.

You'll start work on each successive core with less energy. After you've restored two cores, you'll have to avoid the deadly hudnall. This creature, who's trapped inside the core, is attracted to the noise of your scrubber and will attack it whenever possible. Avoid the hudnall at all costs.

Whenever your scrubber is destroyed, a chain reaction takes place—causing a reactor meltdown. The game (or rather, your job) is over after three meltdowns.

How it works.

Here's a description of the **Krebs removal** program.

Lines 78-85 — update your energy usage.

Lines 98-100 — the sound routine, heard when you hit the core wall.

Lines 108-120 — scoring routine.

Lines 198-285 — moves the hudnall. The logic routine is a modified version, adapted from the **Basic Training** series (this one was in issue 18).

Lines 288-640 — reads the joystick and moves the scrubber, first checking what the scrubber will hit, then going to the appropriate subroutine.

Lines 748-770 — the scrubber is destroyed; the core melts down; and the number of scrubbers decreases.

Lines 773-795 — game-over message. Returns you to the title page.

Lines 798-820 — core-is-secured routine. Allotted energy decreases, so the difficulty level increases.

Lines 848-860 — plot the scrubber's initial position (random).

Lines 998-1040 — plot 30 krebs (random).

Lines 1098-1150 — plot 10 uranium pellets (random).

Lines 10000-10030 — initialize, then start game.

Lines 29098-30060 — draw main screen.

Lines 30198-30260 — draw title screen and initialize variables.

Lines 31098-32239 — redefines two character sets.

Lines 32000-32040 — move character set from ROM to two different locations in RAM.

Lines 32050-32230 — read in data for the first character set.

Lines 32231-32239 — read in data for the second character set.

Table 1.

LIST OF VARIABLES	
BX,BY	Hudnall's X- and Y-position
BUG	Flag; if less than 5, hudnall moves.
CLOCK	Timer; used to determine when to decrease energy level.
EN	Scrubber's energy level.
LEVEL	Amount of energy you initially enter the core with.
PC	Flag; indicates the number of krebs cleared.
SC	Score.
SCRUB	Number of scrubbers or lives.
XV,YV	Holds direction hudnall is to move.
X,Y	Scrubbers X- and Y-position.

Krebs removal was written without player/missile graphics, and with only two short machine language routines. One routine is used to move the character set from ROM to RAM; the other produces the rainbow effect when the core melts down.

I did this in order to show that you can make a relatively fast game primarily out of BASIC. I'm not saying that player/missile graphics and machine language routines aren't helpful. In fact, they're very useful, and can enhance a game tremendously. I just wanted to write a game without them.

I did, of course, redefine the character set. In fact, I redefined two character sets. Each contains a different view of the scrubber, krebs, radioactive pellets and hudnall. All you have to do to animate them is quickly flip between the two character sets. This technique is useful when you want to animate a large number of the *same* objects (i.e., krebs and radioactive pellets), regardless of where they are on-screen.

Since I was using the technique for krebs and pellets, I also used it for the scrubber, hudnall and title page. To see what **Krebs removal** would be like without this technique, change Line 290 to read *J = STICK(0)*. One note, if you redefined a character but don't want it animated (like the core walls), you must put the same view in each character set.

The routine which moves the hudnall towards the scrubber was taken (and slightly modified) from issue 18's **Basic Training**. I highly recommend that you read these articles. They contain many valuable programming tips.

Another way to pick up some knowledge is to analyze other people's games. So take a look at **Krebs removal**. Maybe you'll find something you can use in your next game.

Chuck Rosko is a microbiologist from Pittsburgh, Pennsylvania, the proud father of a baby boy. His interests include his wife and son, hockey, and writing educational programs.

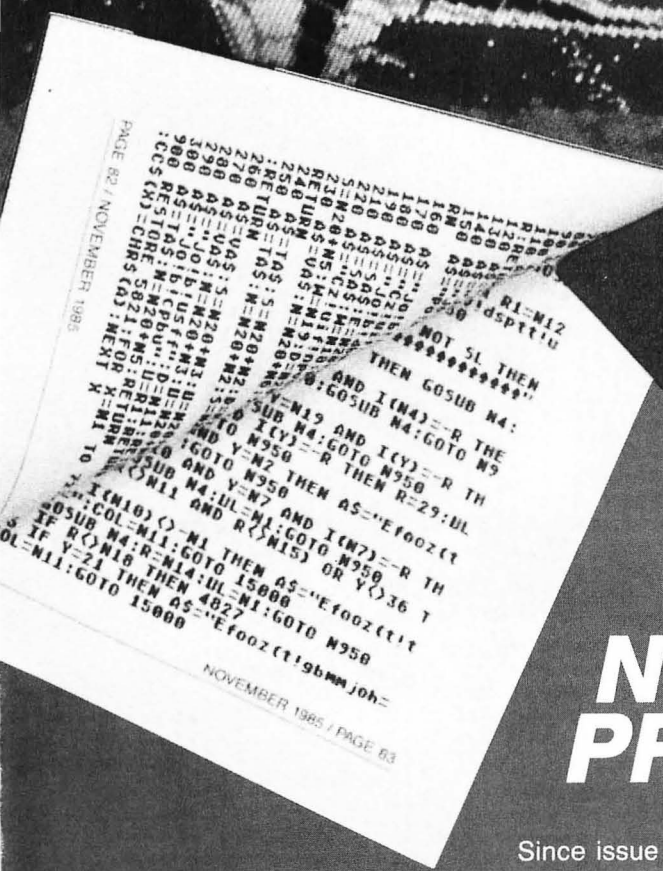
The two-letter checksum code preceding the line numbers here is *not* a part of the BASIC program. For further information, see the *BASIC Editor II*, in issue 47 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
WU 1 REM  KREBS REMOVAL
QD 2 REM  BY  CHUCK ROSKO
QG 10 K1=1:K2=2:K3=3:K5=5:K6=6:K10=10:K15
   =15:K18=18:K50=50:K255=255:POKE 559,K0
   :GOTO 10000
RF 78 REM UPDATE ENERGY USAGE
YP 80 EN=EN-K5:POSITION 16,23:? #K6;EN;"
   ":IF EN<=K0 THEN 750
YN 85 CLOCK=K10:GOTO 290
UK 98 REM HITTING WALL SOUND
YL 100 POKE 710,K15:SOUND K0,125,12,K6:FO
   R C=K1 TO K50:NEXT C:SOUND K0,K0,K0,K0
   :POKE 710,148:GOTO 290
VK 108 REM SCORE ROUTINE
EE 110 SC=SC+K5:POSITION K1,23:? #K6;SC:S
   OUND K0,75,K10,K10:SOUND K0,K0,K0,K0:P
   C=PC+K1:Z(K2)=Z(K1):IF PC=30 THEN 800
PP 120 GOTO 290
KN 198 REM MOVE HUDNALL
TX 200 COLOR 94:PLOT X,Y:SOUND K0,25,K10,
   K6:SOUND K0,K0,K0,K0:M=K1-M
XJ 210 CLOCK=CLOCK-0.2:IF CLOCK<=K0 THEN
   80
KL 220 IF LEVEL>44 THEN 290
YN 240 BUG=INT(RND(K0)*LEVEL):IF BUG>4 TH
   EN 290
GM 260 XV=SGN(X-BX):YV=SGN(Y-BY)
UJ 265 LOCATE BX+XV,BY+YV,Z1:TEMP1=Z1
OA 270 COLOR TEMP2:PLOT BX,BY:BX=BX+XV:BY
   =BY+YV:COLOR 220:PLOT BX,BY:IF Z1=94 T
   HEN 750
AS 285 TEMP2=TEMP1
MW 288 REM READ JOYSTICK
JG 290 J=STICK(K0):POKE 756,PEEK(106)+K1+
   M+M:C=XD(J):IF C=K5 THEN 200
LJ 300 LOCATE X+XM(C),Y+YM(C),Z:IF Z>185
   AND Z<192 THEN 100
ML 310 COLOR Z(K2):PLOT X,Y:X=X+XM(C):Y=Y
   +YM(C):COLOR 94:PLOT X,Y:M=K1-M
SU 320 IF Z=14 THEN Z(K1)=255:GOTO 110
YJ 330 IF Z<>255 THEN Z(K2)=160:GOTO 210
EW 748 REM SCRUBBER DESTROYED
MD 750 POSITION K3,K0:? #K6;"   MELTDOWN
     ":FOR Z=K0 TO K3:SOUND Z,255-Z,14,K5
   :NEXT Z
UW 752 U=USR(ADR(RB$)):P=PEEK(560):FOR Z=
   K0 TO K3:SOUND Z,K0,K0,K0:NEXT Z
KY 753 FOR C=K1 TO K15:FOR Z=K0 TO K3:R=I
   NT(RND(0)*30):POKE 712,PEEK(53770):POK
   E 560,P+Z:SOUND K0,R,8,14:NEXT Z
UR 754 NEXT C:POKE 560,P:BX=K10:BY=K10:TE
   MP2=191
VK 760 SCRUB=SCRUB-K1:COLOR 160:PLOT 7+SC
   RUB+SCRUB,22:POKE 559,K0:POKE 712,14
NS 762 FOR Z=K1 TO K10:SOUND K0,K10*Z,K10
   -Z,K10-Z
```

```
DJ 765 FOR C=K1 TO K10:NEXT C:NEXT Z:POKE
    712,K0:POKE 559,46:GOSUB 30001
JM 770 IF SCRUB THEN CLOCK=K10:EN=LEVEL:P
   C=K0:GOSUB 1000:GOSUB 850:GOTO 290
SN 773 REM GAME OVER
PN 775 POSITION K3,K6:? #K6;"JOB TERMINAT
   ED":POSITION K3,14:? #K6;"YOU MADE $";
   SC
NF 780 FOR Z=K0 TO K3:SOUND Z,K255-Z,14,K
   5:NEXT Z:U=USR(ADR(RB$))
RA 790 FOR Z=K0 TO K3:SOUND Z,K0,K0,K0:NE
   XT Z:? #K6;CHR$(125):GOSUB 30200
OT 795 GOSUB 30000:GOSUB 1000:GOSUB 850:G
   OTO 290
FC 798 REM COMPLETED CORE ROUTINE
LC 800 P$="          ALL KREBS CLEARED     C
   ore is secured                    ":BX
   =K10:BY=K10:TEMP2=191
CQ 805 FOR Z=K1 TO K5:SOUND K0,75,K10,K10
   :POKE 712,14:FOR C=K1 TO K50:NEXT C:SO
   UND K0,150,K10,K10:POKE 712,K0
QF 807 FOR C=K1 TO K50:NEXT C
QP 810 NEXT Z:FOR F=K1 TO 45:POSITION K1,
   9:? #K6;P$(K1,K18):Q$=P$(K2):Q$(LEN(Q$
   )+K1)=P$:P$=Q$:SOUND K0,200,11,14
NC 815 FOR Z=K1 TO K10:NEXT Z:SOUND K0,K0
   ,K0,K0:NEXT F:PC=K0:LEVEL=LEVEL-K5:IF
   LEVEL<30 THEN LEVEL=30
DD 820 EN=LEVEL:Z(K1)=160:Z(K2)=160:GOSUB
    30005:GOSUB 1000:CLOCK=K10:GOSUB 850:
   GOTO 290
GM 848 REM INITIALIZE SCRUBBER
LJ 850 X=INT(RND(K0)*K18)+K1:Y=INT(RND(K0
   )*K18)+K2:LOCATE X,Y,Z
IQ 852 IF (Z>185 AND Z<192) OR Z=14 OR Z=
   64 OR Z=K255 THEN 850
VJ 860 COLOR 94:PLOT X,Y:RETURN
YJ 998 REM PLOT KREBS
US 1000 FOR CC=K1 TO 30
WR 1005 A=INT(RND(K0)*K18)+K1:B=INT(RND(K
   0)*K18)+K2:LOCATE A,B,Z
CM 1010 IF (Z>185 AND Z<192) OR Z=14 OR Z
   =64 THEN 1005
MK 1040 COLOR 14:PLOT A,B:NEXT CC
BB 1098 REM PLOT URANIUM
ZW 1100 FOR CC=K1 TO K10
WT 1105 A=INT(RND(K0)*K18)+K1:B=INT(RND(K
   0)*K18)+K2:LOCATE A,B,Z
GS 1110 IF (Z>185 AND Z<192) OR Z=14 OR Z
   =64 OR Z=K255 THEN 1105
YP 1150 COLOR K255:PLOT A,B:NEXT CC:POSIT
   ION 16,23:? #K6;EN;" ":RETURN
VI 10000 DIM ZZ$(32),Z(K2):GOSUB 32000:GR
   APHICS 17:POKE 756,PEEK(106)+K1:GOSUB
   30200:GOSUB 30000
IS 10010 DIM XD(K15),P$(65),Q$(65):FOR X=
   K5 TO K15:XD(X)=K5:NEXT X:XD(7)=K2:XD(
   11)=4:XD(14)=K1:XD(13)=K3
WZ 10020 DIM XM(4),YM(4):RESTORE 10030:FO
   R I=K1 TO 4:READ X,Y:XM(I)=X:YM(I)=Y:N
   EXT I
SH 10030 DATA 0,-1,1,0,0,1,-1,0
PV 10040 DIM RB$(21):RESTORE 10050:FOR I=
   K1 TO 21:READ X:RB$(I)=CHR$(X):NEXT I
ZV 10050 DATA 104,169,0,133,20,133,19,105
   ,1,232,142,22,208,142,10,212,197,19,20
   8,245,96
FF 10060 GOSUB 1000:GOSUB 850:GOTO 290
TM 29098 REM DRAW SCREEN
YY 30000 POSITION K0,22:? #K6;"Money  ^ ^
   ^  energy$"
VK 30001 POSITION K3,K0:? #K6;"KREBS  REM
   OVAL"
GU 30005 POSITION K0,K1:? #K6;"      :::::
   :::::     eeee>          ?eeeeeee>
            ?eee";
UO 30010 ? #K6;"ee>                  ?eee>
                ?e<                     =";
IF 30020 ? #K6;"<                     =<
                =<                      =";
TG 30030 ? #K6;"<          >?         =<
      ?>        =<                      =";
IN 30040 ? #K6;"<                     =<
                =<                      =";
BG 30050 ? #K6;"e?                    >eee?
                >eeeee?               >eee";
QW 30060 ? #K6;"eeee?            >eeee
   ::::::::::      ":RETURN
SF 30198 REM DRAW INITIAL SCREEN
SX 30200 SC=K0:SCRUB=K3:EN=K50:CLOCK=K10:
   PC=K0:LEVEL=K50:BX=K10:BY=K10:TEMP2=19
   1:Z(K1)=160:Z(K2)=160:POKE 559,34
BD 30205 POKE 710,148:POKE 709,202:POKE 7
   08,54:POKE 711,70
NU 30210 POSITION K0,4:? #K6;"Z Z ZZZ ZZZ
    ZZZ ZZZ Z Z Z Z Z   Z Z Z   Z Z Z Z Z
       Z Z Z   ";
LC 30212 ? #K6;"ZZ  ZZZ ZZZ ZZ  ZZZ ZZ  Z
   Z  Z   Z Z   Z Z Z Z Z Z   Z Z   Z ";
WK 30214 ? #K6;"Z Z Z Z ZZZ ZZZ ZZZ
          REMOVAL "
GZ 30240 POSITION K3,14:? #K6;"BY chuck r
   osko":POSITION K0,21:? #K6;"to apply p
   ress start"
AK 30250 IF PEEK(53279)=K6 THEN ? #K6;CHR
   $(125):RETURN
JA 30260 POKE 756,PEEK(106)+K1+M+M:M=K1-M
   :SOUND K0,200,K10,K10:SOUND K0,K0,K0,K
   0:GOTO 30250
CR 31098 REM DEFINE 2 CHARACTER SETS
KA 32000 RESTORE 32010:FOR I=K1 TO 32:REA
   D A:ZZ$(I)=CHR$(A):NEXT I
BR 32010 DATA 104,104,133,204,104,133,203
   ,104,133,206,104,133,205,162,4,160,0
QV 32020 DATA 177,203,145,205,136,208,249
   ,230,204,230,206,202,208,240,96
QN 32030 POKE 106,PEEK(106)-K5:GRAPHICS 1
   7:START=(PEEK(106)+K1)*256:POKE 752,K1
KD 32035 POSITION 4,K10:? #K6;"PLEASE  WA
   IT"
FD 32040 A=USR(ADR(ZZ$),57344,START):A=US
   R(ADR(ZZ$),57344,START+512)
BV 32050 READ X:IF X=-K1 THEN 32300
NY 32060 FOR Y=K0 TO 7:READ Z:POKE X+Y+ST
   ART,Z:NEXT Y:GOTO 32050
EZ 32200 DATA 208,0,0,0,0,0,0,0,255,216,2
   55,0,0,0,0,0,0,0,0,224,1,1,1,1,1,1,1,2
   32,128,128,128,128,128,128,128,128
IZ 32210 DATA 240,1,2,4,8,16,32,64,128,24
   8,128,64,32,16,8,4,2,1
XG 32220 DATA 496,24,60,102,195,195,102,6
   0,24,504,0,60,126,126,126,126,60,0
RZ 32225 DATA 112,60,126,15,252,63,240,12
   6,60,256,255,129,129,129,129,129,129,2
   55
CS 32230 DATA 464,255,195,195,195,195,195
   ,195,255,480,36,24,165,126,126,126,189
   ,24,-1
TA 32300 READ X:IF X=-K1 THEN RETURN
MT 32310 FOR Y=K0 TO 7:READ Z:POKE X+Y+ST
   ART+512,Z:NEXT Y:GOTO 32300
FK 32320 DATA 208,0,0,0,0,0,0,0,255,216,2
   55,0,0,0,0,0,0,0,0,224,1,1,1,1,1,1,1,2
   32,128,128,128,128,128,128,128,128
JK 32330 DATA 240,1,2,4,8,16,32,64,128,24
   8,128,64,32,16,8,4,2,1
OP 32340 DATA 496,0,0,24,60,60,24,0,0,504
   ,60,126,255,255,255,255,126,60
RW 32350 DATA 112,60,126,240,63,252,15,12
   6,60,256,255,129,129,129,129,129,129,2
   55
CI 32360 DATA 464,255,255,255,255,255,255
   ,255,255,480,102,24,165,126,126,126,18
   9,153,-1
```

IT'S ALL IN THE DISK



NO PROGRAMMING

Since issue 1, **ANALOG Computing**'s disk subscriptions have eliminated the need for you to spend hours typing in programs from the magazine. *All* of the programs in the magazine are on the disk version.

A 1-year subscription (12 issues) is $105.00; a ½-year (6 issues) is $59.00.

To subscribe on disk, send your check or money order to:

DISK SUBSCRIPTION, P.O. BOX 625, HOLMES, PA 19043.

For fast service, call our toll-free U.S. order line: 800-345-8112 (in PA, call 800-662-2444).

GENERAL

Integer BASIC

For the 64K Atari XL computers.

by Barry Green

This program was developed quite unintentionally. I was busy hacking away on a new machine-language animation system for the Atari, that works with BASIC. One evening, I actually got around to testing it with Atari BASIC. The program worked flawlessly and made animation feasible from BASIC, but it was still pitifully slow.

After hours of poring over the code, trying to trim and streamline it to make it run faster, I realized the fault was not with my program at all. It was working as fast as it could. Still, I thought, there *has* to be a better way! One evening when I was reading through my *Atari BASIC Sourcebook*, the dim light in my brain flickered on for a moment.

The incredible slothfulness of BASIC—from a game programmer's point of view—could be traced to the fact that no provision for integers had been made. Only floating-point math was supported, and it's very slow. Integer math is very fast, so somehow my Atari had to be made to understand integer math. This was, of course, much easier said than done.

And how?

One very underexplored feature of the Atari 800XL and 1200XL (and the expanded 600XL) is that the operating systems can be placed in the upper 16K bank of memory and modified to no end.

This was my chance. I could load the OS into RAM, then replace the old floating-point math package with a much faster integer package. Every operation BASIC did, from POKE to FOR-NEXT, would run faster.

That's what I did. The following program places the OS in RAM, then replaces the old math package with a much more efficient integer math package. I've run some BASIC test programs, to show the speed gained by using integer math instead of floating point.

Typing it in.

Listings 1 and 2 are the BASIC data to create your copies of **Integer BASIC**. Please refer to the **M/L Editor** on page 4 for typing instructions. You should create the program in Listing 1 under the filename INTBASIC.OBJ. Create the program in Listing 2 under the filename INTBASI2.OBJ.

The test.

Is **Integer BASIC** that much faster? To prove that it is, I devised three very simple BASIC programs to test the speed difference. Each was run in both languages, and the jiffy-count was printed next to each.

	Floating-Point	Integer	%Faster
FOR-NEXT	1459	1044	30%
MATH TEST	947	510	46%
SCREEN FILL	3754	1869	51%

The source listings for **Integer BASIC** follow this article. As you can see, the **Integer BASIC** can be up to 50%

faster. That's a serious improvement for something so easily accessible.

The side benefit: those of you using an **Integer BASIC** compiler might consider how handy it would be to be able to interactively debug your own programs.

How to integrate it.

Here are instructions for using the INTBASIC integer math package for 64K Atari XL computers.

To use this package, just load the INTBASIC.OBJ file from DOS menu item L, or from OS/A+ type in *LOAD INTBASIC.OBJ*. Your computer will now be in **Integer BASIC**. Do this only if no BASIC program is in memory at the time.

The INTBASIC package has been tested, with no bugs found. However, some knowledge of integer math is required to use the package effectively. Since the numbers being dealt with are integers, BASIC will no longer recognize a decimal point as valid. Only numbers in the range of 0-65535 or, in version 2, −32768 to +32767 will be accepted. This also means that division is treated slightly differently. In integer math, the expression *10/3* evaluates into *3*, not *3.3333*. . . All numbers are rounded down; the digits past the decimal point are simply dropped.

This means that special care must be taken in using the built-in functions such as COS(X) and RND(0). Math functions like COS(X) will simply not work correctly. RND(0) should not be used, because it now only returns a value between *0* and *3*. It's a better idea to use PEEK(53770) to get a random number.

Negative numbers in version 1 are different. They are now expressed in 65536-X terms. This means that a negative number is subtracted from 65536 and the result is printed. Therefore, when printing a negative 1 *(-1)* on the screen, you will get *65535 (65536-1)*. Negative numbers are fully usable; they just print differently from what you'd expect.

One last restriction: BASIC programs developed under the integer math package cannot be loaded into the floating-point BASIC; nor can floating point programs be loaded into **Integer BASIC**.

The solution to transferring programs from one format to the other is simply to LIST them onto disk or cassette, then ENTER them into the other version of BASIC. When transferring programs from floating-point BASIC to integer, remember that decimal points will be flagged as errors, and you must fix all RND(0) usage.

Have fun with **Integer BASIC** and enjoy its refreshing speed.

Barry Green of Out of the Blue Associates bought his first Atari (an 800) in 1982 and taught himself BASIC and assembly language. Since then, he's worked on many conversions and originals for various companies. His main interest lies in system software and utilities.

Listing 1.

```
1000 DATA 255,255,0,6,101,6,32,70,6,17
7,203,145,205,200,208,249,1480
1010 DATA 230,204,230,206,202,224,48,2
08,3,32,91,6,224,0,208,233,8696
1020 DATA 120,169,0,141,14,212,169,254
,141,1,211,32,70,6,177,205,7512
1030 DATA 145,203,200,208,249,230,204,
230,206,202,224,48,208,3,32,91,502
1040 DATA 6,224,0,208,233,88,169,64,14
1,14,212,96,169,0,133,203,8047
1050 DATA 169,192,133,204,169,0,133,20
5,169,64,133,206,162,64,160,0,7732
1060 DATA 96,160,8,230,204,230,206,202
,136,208,248,96,226,2,228,2,1480
1070 DATA 0,6,96,0,216,78,216,32,161,2
19,169,64,133,212,169,0,8184
1080 DATA 133,213,133,214,133,215,56,8
,164,242,177,243,201,48,144,52,341
1090 DATA 201,58,176,48,40,24,8,41,15,
72,165,214,133,215,165,213,8715
1100 DATA 10,38,215,10,38,215,24,101,2
13,133,213,165,215,101,214,133,1444
1110 DATA 214,6,213,38,214,104,24,101,
213,133,213,144,2,230,214,230,2251
1120 DATA 242,76,17,216,40,96,230,216,
91,217,32,81,218,165,213,133,1323
1130 DATA 220,165,214,133,221,160,0,16
2,0,165,213,217,87,217,165,214,2880
1140 DATA 249,82,217,144,19,165,213,56
,249,87,217,133,213,165,214,249,5171
1150 DATA 82,217,133,214,232,76,245,21
6,138,9,48,145,243,200,192,5,499
1160 DATA 208,213,136,177,243,9,128,14
5,243,160,0,177,243,201,48,208,2167
1170 DATA 4,200,76,39,217,132,222,165,
243,56,229,222,133,218,165,244,5799
1180 DATA 233,0,133,219,177,243,145,21
8,200,192,5,208,247,165,220,133,5010
1190 DATA 213,165,221,133,214,96,39,3,
0,0,0,16,232,100,10,1,9645
1200 DATA 170,217,183,217,166,212,164,
213,169,64,133,212,134,213,132,214,447
1
1210 DATA 24,96,210,217,219,217,165,21
3,133,212,165,214,133,213,24,96,2487
1220 DATA 96,218,98,218,76,117,218,102
,218,146,218,165,213,24,101,225,2362
1230 DATA 133,213,165,214,101,226,133,
214,24,96,165,213,56,229,225,133,2628
1240 DATA 213,165,214,229,226,133,214,
6,212,165,214,10,102,212,165,213,3024
1250 DATA 5,214,208,2,133,212,24,96,21
9,218,2,219,169,0,133,219,9685
1260 DATA 133,218,162,16,208,13,24,165
,219,101,225,133,219,165,218,101,2080
1270 DATA 226,133,218,70,218,102,219,1
02,214,102,213,202,48,4,144,243,1188
1280 DATA 176,228,24,96,40,219,86,219,
165,225,5,226,240,39,169,0,8939
1290 DATA 133,219,133,218,160,16,6,213
,38,214,38,219,38,218,56,165,8328
1300 DATA 219,229,225,170,165,218,229,
226,144,6,134,219,133,218,230,213,5973
1310 DATA 136,208,227,24,96,56,96,137,
221,151,221,134,252,132,253,160,4240
1320 DATA 2,177,252,153,212,0,136,16,2
48,96,152,221,166,221,134,252,3994
1330 DATA 132,253,160,2,177,252,153,22
4,0,136,16,248,96,167,221,181,2025
1340 DATA 221,134,252,132,253,160,2,18
5,212,0,145,252,136,16,248,96,607
1350 DATA 182,221,193,221,160,2,185,21
2,0,153,224,0,136,16,247,96,8467
1360 DATA 224,2,225,2,228,2,0,0,0,0,0,
0,0,0,0,0,3423
```

Listing 2.

```
1000 DATA 255,255,0,6,101,6,32,70,6,17
```

```
7,203,145,205,200,208,249,1480
1010 DATA 230,204,230,206,202,224,48,2
08,3,32,91,6,224,0,208,233,8696
1020 DATA 120,169,0,141,14,212,169,254
,141,1,211,32,70,6,177,205,7512
1030 DATA 145,203,200,208,249,230,204,
230,206,202,224,48,208,3,32,91,502
1040 DATA 6,224,0,208,233,88,169,64,14
1,14,212,96,169,0,133,203,8047
1050 DATA 169,192,133,204,169,0,133,20
5,169,64,133,206,162,64,160,0,7732
1060 DATA 96,160,8,230,204,230,206,202
,136,208,248,96,226,2,228,2,1480
1070 DATA 0,6,96,0,216,78,216,32,161,2
19,169,64,133,212,169,0,8184
1080 DATA 133,213,133,214,133,215,56,8
,164,242,177,243,201,48,144,52,341
1090 DATA 201,58,176,48,40,24,8,41,15,
72,165,214,133,215,165,213,8715
1100 DATA 10,38,215,10,38,215,24,101,2
13,133,213,165,215,101,214,133,1444
1110 DATA 214,6,213,38,214,104,24,101,
213,133,213,144,2,230,214,230,2251
1120 DATA 242,76,17,216,40,96,230,216,
132,217,32,81,218,165,213,133,1692
1130 DATA 220,165,214,133,221,16,29,16
5,213,73,255,24,105,1,133,213,8100
1140 DATA 165,214,73,255,105,0,133,214
,160,0,169,45,145,243,230,243,2604
1150 DATA 208,2,230,244,160,0,162,0,16
5,213,217,128,217,165,214,249,4825
1160 DATA 123,217,144,19,165,213,56,24
9,128,217,133,213,165,214,249,123,4897
1170 DATA 217,133,214,232,76,20,217,13
8,9,48,145,243,200,192,5,208,109
1180 DATA 213,136,177,243,9,128,145,24
3,160,0,177,243,201,48,208,4,9712
1190 DATA 200,76,70,217,132,222,165,24
3,56,229,222,133,218,165,244,233,7075
1200 DATA 0,133,219,177,243,145,218,20
0,192,5,208,247,165,220,133,213,5700
1210 DATA 165,221,133,214,16,8,165,243
,208,2,198,244,198,243,96,39,1337
1220 DATA 3,0,0,0,16,232,100,10,1,170,
217,183,217,166,212,164,716
1230 DATA 213,169,64,133,212,134,213,1
32,214,24,96,210,217,219,217,165,4440
1240 DATA 213,133,212,165,214,133,213,
24,96,96,218,98,218,76,117,218,1105
1250 DATA 102,218,146,218,165,213,24,1
01,225,133,213,165,214,101,226,133,356
9
1260 DATA 214,24,96,165,213,56,229,225
,133,213,165,214,229,226,133,214,6544
1270 DATA 6,212,165,214,10,102,212,165
,213,5,214,208,2,133,212,24,8786
1280 DATA 96,219,218,2,219,169,0,133,2
19,133,218,162,16,208,13,24,6991
1290 DATA 165,219,101,225,133,219,165,
218,101,226,133,218,70,218,102,219,421
8
1300 DATA 102,214,102,213,202,48,4,144
,243,176,228,24,96,40,219,86,8678
1310 DATA 219,165,225,5,226,240,39,169
,0,133,219,133,218,160,16,6,7494
1320 DATA 213,38,214,38,219,38,218,56,
165,219,229,225,170,165,218,229,6048
1330 DATA 226,144,6,134,219,133,218,23
0,213,136,208,227,24,96,56,96,9978
1340 DATA 137,221,151,221,134,252,132,
253,160,2,177,252,153,212,0,136,1950
1350 DATA 16,248,96,152,221,166,221,13
4,252,132,253,160,2,177,252,153,4501
1360 DATA 224,0,136,16,248,96,167,221,
181,221,134,252,132,253,160,2,2836
1370 DATA 185,212,0,145,252,136,16,248
,96,182,221,193,221,160,2,185,2265
1380 DATA 212,0,153,224,0,136,16,247,9
6,226,2,227,2,228,2,0,4969
```

●

Listing 3.
BASIC listing.

```
DE 5 POKE 19,0:POKE 20,0
GJ 10 FOR X=1 TO 10000
MV 20 NEXT X
XG 30 PRINT PEEK(19)*256+PEEK(20)
```

●

Listing 4.
BASIC listing.

```
MM 10 POKE 19,0:POKE 20,0
SR 20 J=5
XJ 30 FOR X=1 TO 1000
DF 40 J=J*2:J=J/2
MY 50 NEXT X
XT 60 PRINT PEEK(19)*256+PEEK(20)
```

●

Listing 5.
BASIC listing.

```
JG 10 GRAPHICS 8+16:SCREEN=PEEK(88)+256*P
   EEK(89)
MM 20 POKE 19,0:POKE 20,0
YP 30 FOR X=0 TO 7679:POKE SCREEN+X,255:N
   EXT X
IX 40 VV=PEEK(19)*256+PEEK(20)
BO 50 GRAPHICS 0:PRINT VV
```

●

Listing 6.
Assembly listing.

```
;-----------------------------
; SAVE @D:INTBASIC.ASM
; ASM,,@D:INTBASIC.OBJ
;-----------------------------
; by Barry Green
; Out of the Blue Associates
;-----------------------------
; This is a math package designed
; to replace the floating-point
; package in the ATARI 64K XL
; computers. This package is
; entirely integer-based so it
; is much faster. Its internal
; representation of numbers is
; still referred to as Floating-
; point, merely for convenience.
; This representation differs
; from standard 6502 integers in
; that there is an exponent byte
; of $40 added to it. Integers can
; still be used in FPI and IFP
; because these routines restore
; the numbers to proper order.
; This version will handle numbers
; in the range of 0-65535. Any
; number above 32767 will be
; treated as a negative number in
; any mathematical computations
; but is valid as a number for
; such things as FOR-NEXT loops
; and so forth. Any number that
; is INPUTted or so forth with a
; minus sign in front of it will
; be converted to this format,
; which is simply 65536-X.
;-----------------------------
        .OPT NO LIST
FRO     =   $D4
FRE     =   $DA
FR1     =   $E0
INBUFF  =   $F3
CIX     =   $F2
SKIP.BLANKS = $DBA1
INTLBF  =   $DA51
LBUFF   =   $0580
FLPTR   =   $FC
;-----------------------------
        *=  $0600
START
;-----------------------------
; MOVES THE XL OS TO RAM
; THIS CODE IS BASED ON THE
; OFFICIAL RELEASE FROM ATARI INC.
; AND WAS WRITTEN BY THEM.
;-----------------------------
ROM     =   $CB
RAM     =   ROM+2
OSROM   =   $C000
```

```
OSRAM     =    $4000
NMIEN     =    $D40E
PORTB     =    $D301
          JSR INIT      SET POINTERS
MOV1      LDA (ROM),Y   MOVE THE ROM
          STA (RAM),Y   INTO RAM
          INY
          BNE MOV1
          INC ROM+1
          INC RAM+1
          DEX
          CPX #$30
          BNE :M1
          JSR SKIP      SKIP I/O
:M1       CPX #$00
          BNE MOV1
*-------------------------------
          SEI
          LDA #$00      DISABLE THE
          STA NMIEN     INTERRUPTS
          LDA #$FE
          STA PORTB     FLIP ROM OUT
*-------------------------------
          JSR INIT
MOV2      LDA (RAM),Y   MOVE THE OS
          STA (ROM),Y   BACK INTO THE
          INY           RIGHT ADDRESSES
          BNE MOV2
          INC ROM+1
          INC RAM+1
          DEX
          CPX #$30
          BNE :M2
          JSR SKIP
:M2       CPX #$00
          BNE MOV2
*-------------------------------
          CLI
          LDA #$40      RE-ENABLE THE
          STA NMIEN     INTERRUPTS AND
          RTS           RETURN.
*-------------------------------
* INITIALIZE PAGE ZERO VARIABLES
* FOR ADDRESSES OF RAM AND ROM
* USED FOR INDEXING
*-------------------------------
INIT      LDA # <OSROM
          STA ROM
          LDA # >OSROM
          STA ROM+1
          LDA # <OSRAM
          STA RAM
          LDA # >OSRAM
          STA RAM+1
          LDX #$40
          LDY #$00
          RTS
*-------------------------------
* SET PAGE ZERO VARIABLES TO
* SKIP THE HARDWARE REGISTERS
*-------------------------------
SKIP      LDY #$08
SKIP1     INC ROM+1
          INC RAM+1
          DEX
          DEY
          BNE SKIP1
          RTS
*-------------------------------
* SET THE INIT ADDRESS TO DOS SO
* ALL THE PREVIOUS CODE WILL BE
* EXECUTED AND THE REST OF THE
* PROGRAM WILL LOAD INTO THE TOP
* 16K AND REPLACE THE OLD MATH
* PACKAGE.
*-------------------------------
          *=   $02E2
          .WORD START
*-------------------------------
* THIS JUST RETURNS CONTROL TO
* DOS AFTER THE INTEGER MATH
* PACKAGE HAS BEEN LOADED IN.
*-------------------------------
CONTINUE
          RTS
*-------------------------------
* FOLLOWING ARE THE ACTUAL MATH
* ROUTINES WHICH LOAD RIGHT ON
* TOP OF THE OLD ONES.
*-------------------------------
          *=   $D800
AFP
*-------------------------------
* CONVERTS ASCII TO NUMBER VALUE
*-------------------------------
          JSR SKIP.BLANKS
          LDA #$40
          STA FR0
          LDA #$00
          STA FR0+1
          STA FR0+2
          STA FR0+3
          SEC
          PHP
:0        LDY CIX
          LDA (INBUFF),Y
          CMP #'0       IS IT A DIGIT?
          BCC :2        IF NOT, THEN
          CMP #'9+1     BRANCH TO
          BCS :2        THE RETURN.
          PLP
          CLC
          PHP
          AND #$0F
          PHA           SAVE THE DIGIT.
*-------------------------------
          LDA FR0+2     MULTIPLY THE
          STA FR0+3     NUMBER BY 10
          LDA FR0+1     BEFORE ADDING
          ASL A         IN THE NEW
          ROL FR0+3     DIGIT.
          ASL A
          ROL FR0+3
          CLC
          ADC FR0+1
          STA FR0+1
          LDA FR0+3
          ADC FR0+2
          STA FR0+2
          ASL FR0+1
          ROL FR0+2
*-------------------------------
          PLA
          CLC
          ADC FR0+1
          STA FR0+1
          BCC :B
          INC FR0+2
:B        INC CIX
          JMP :0
:2        PLP
          RTS
*-------------------------------
          *=   $D8E6
FASC
*-------------------------------
* CONVERTS INTEGERS INTO ASCII
* CHARACTER STRINGS.
*-------------------------------
          JSR INTLBF
          LDA FR0+1
          STA FRE+2
          LDA FR0+2
          STA FRE+3
          LDY #$00
:02       LDX #$00
:01       LDA FR0+1
          CMP LO.BYTES,Y
          LDA FR0+2
          SBC HI.BYTES,Y
          BCC NEXT.DIGIT
          LDA FR0+1
          SEC
          SBC LO.BYTES,Y
          STA FR0+1
          LDA FR0+2
          SBC HI.BYTES,Y
          STA FR0+2
          INX
          JMP :01
NEXT.DIGIT
          TXA
          ORA #$30
          STA (INBUFF),Y
          INY
          CPY #$05
          BNE :02
          DEY
          LDA (INBUFF),Y
          ORA #$80
          STA (INBUFF),Y
*-------------------------------
* NOW REMOVE THE LEADING ZEROES
*-------------------------------
          LDY #$00
:03       LDA (INBUFF),Y
          CMP #'0
          BNE :04
          INY
          JMP :03
:04       STY FRE+4
          LDA INBUFF
          SEC
          SBC FRE+4
          STA FRE
          LDA INBUFF+1
          SBC #$00
          STA FRE+1
:05       LDA (INBUFF),Y
          STA (FRE),Y
          INY
          CPY #$05
          BNE :05
          LDA FRE+2
          STA FR0+1
          LDA FRE+3
          STA FR0+2
          RTS
*-------------------------------
* THE FOLLOWING ARE TABLES USED
* IN CONVERTING NUMBERS TO
* ASCII CHARACTER STRINGS.
*-------------------------------
HI.BYTES  .BYTE  >10000
          .BYTE  >1000, >100
          .BYTE  >10, >1
LO.BYTES  .BYTE  <10000
          .BYTE  <1000, <100
          .BYTE  <10, <1
*-------------------------------
          *=   $D9AA
IFP
*-------------------------------
* THIS ROUTINE WILL CONVERT AN
* INTEGER TO FLOATING POINT,
* WHICH IS SIMPLY LOW BYTE-HIGH
* BYTE FORMAT WITH AN EXPONENT.
*-------------------------------
          LDX FR0
          LDY FR0+1
          LDA #$40      ;EXPONENT
          STA FR0
          STX FR0+1
          STY FR0+2
          CLC
          RTS
*-------------------------------
          *=   $D9D2
FPI
*-------------------------------
* THESE ROUTINES WILL CONVERT A
* FLOATING POINT NUMBER BACK TO
* INTEGER FORMAT.
*-------------------------------
          LDA FR0+1
          STA FR0
          LDA FR0+2
          STA FR0+1
          CLC
          RTS
*-------------------------------
          *=   $DA60
FSUB
          JMP FSUB2
*-------------------------------
          *=   $DA66
FADD
*-------------------------------
* THESE ROUTINES WILL ADD AND
* SUBTRACT INTEGERS.
*-------------------------------
          LDA FR0+1
          CLC
          ADC FR1+1
          STA FR0+1
          LDA FR0+2
          ADC FR1+2
          STA FR0+2
          CLC
          RTS
*-------------------------------
FSUB2
*-------------------------------
          .LOCAL
          LDA FR0+1
          SEC
          SBC FR1+1
          STA FR0+1
          LDA FR0+2
          SBC FR1+2
          STA FR0+2
          ASL FR0       THIS SETS THE
          LDA FR0+2     SIGN BIT IN THE
          ASL A         EXPONENT FOR
          ROR FR0       BASIC'S COMPARE.
          LDA FR0+1     IF RESULT
          ORA FR0+2     IS ZERO THEN
          BNE :1        ZERO OUT THE
          STA FR0       EXPONENT.
:1        CLC
          RTS
*-------------------------------
          *=   $DADB
FMUL
*-------------------------------
* THIS ROUTINE WILL MULTIPLY
* THE CONTENTS OF FR0 BY THE
* CONTENTS OF FR1 AND STORE THE
* PRODUCT BACK INTO FR0.
*-------------------------------
          .LOCAL
          LDA #0
          STA FRE+1
          STA FRE
          LDX #16
          BNE :1
:2        CLC
          LDA FRE+1
          ADC FR1+1
          STA FRE+1
          LDA FRE
          ADC FR1+2
          STA FRE
:1        LSR FRE
          ROR FRE+1
          ROR FR0+2
          ROR FR0+1
          DEX
          BMI :3
          BCC :1
          BCS :2
:3        CLC
          RTS
*-------------------------------
          *=   $DB28
FDIV
*-------------------------------
* THIS ROUTINE WILL DIVIDE THE
* CONTENTS OF FR0 BY FR1 AND
* STORE THE RESULT INTO FR0.
*-------------------------------
          .LOCAL
          LDA FR1+1     CANNOT
          ORA FR1+2     DIVIDE
          BEQ :ERROR    BY ZERO
          LDA #0
          STA FRE+1
          STA FRE
          LDY #16
:2        ASL FR0+1
          ROL FR0+2
          ROL FRE+1
          ROL FRE
          SEC
          LDA FRE+1
          SBC FR1+1
          TAX
          LDA FRE
          SBC FR1+2
          BCC :1
          STX FRE+1
          STA FRE
          INC FR0+1
:1        DEY
          BNE :2
          CLC
          RTS
:ERROR
          SEC
          RTS
*-------------------------------
          *=   $DDB9
FLDOR
*-------------------------------
* THIS ROUTINE WILL LOAD FR0
* FROM THE ADDRESS POINTED TO
* BY THE 6502 X,Y REGISTERS.
* OR FLDOP WILL LOAD IT FROM THE
* CURRENT ADDRESS OF FLPTR.
*-------------------------------
```

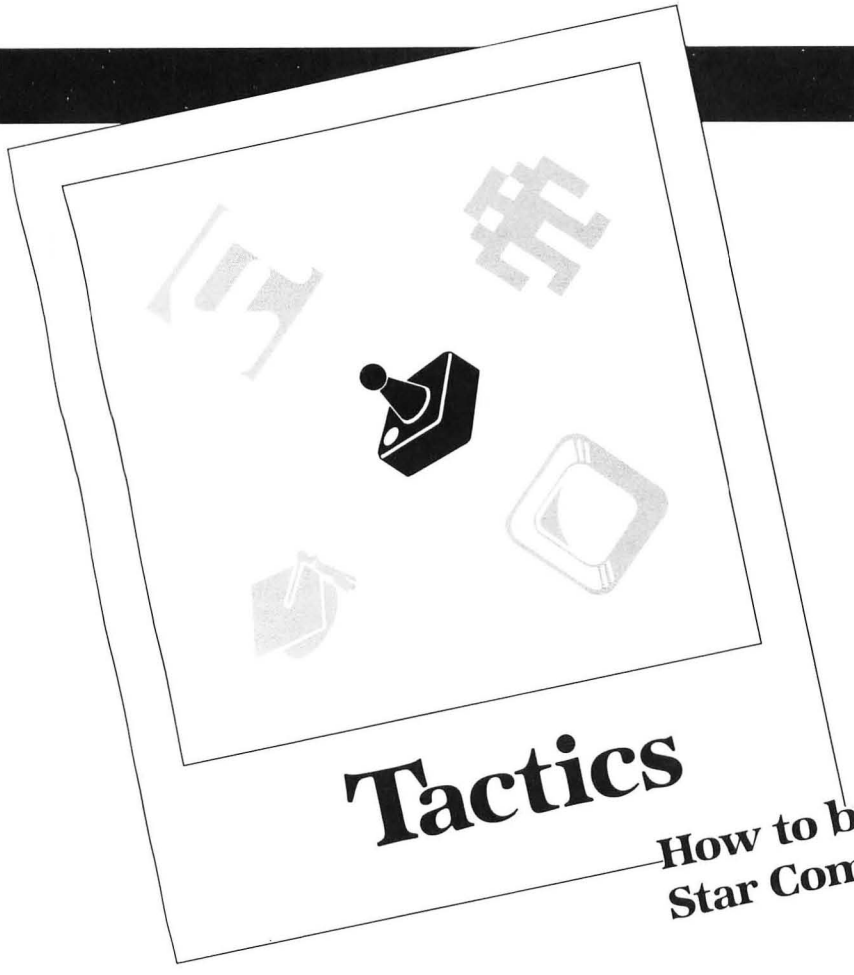
```
          STX FLPTR
          STY FLPTR+1
FLDOP
          LDY #$02
:F1       LDA (FLPTR),Y
          STA FR0,Y
          DEY
          BPL :F1
          RTS
*-------------------------------
          *=  $DD98
FLD1R
*-------------------------------
* THIS ROUTINE WILL LOAD FR1
* FROM THE ADDRESS POINTED TO
* BY THE 6502 X,Y REGISTERS.
* OR FLD1P WILL LOAD IT FROM THE
* CURRENT ADDRESS OF FLPTR.
*-------------------------------
          STX FLPTR
          STY FLPTR+1
FLD1P
          LDY #$02
:F3       LDA (FLPTR),Y
          STA FR1,Y
          DEY
          BPL :F3
          RTS
*-------------------------------
          *=  $DDA7
FSTOR
*-------------------------------
* THIS ROUTINE WILL STORE FR0
* INTO THE ADDRESS POINTED TO
* BY THE 6502 X,Y REGISTERS.
* OR FSTOP WILL STORE IT INTO THE
* CURRENT ADDRESS OF FLPTR.
*-------------------------------
          STX FLPTR
          STY FLPTR+1
FSTOP
          LDY #$02
:F4       LDA FR0,Y
          STA (FLPTR),Y
          DEY
          BPL :F4
          RTS
*-------------------------------
          *=  $DDB6
FMOVE
*-------------------------------
* THIS ROUTINE WILL MOVE THE
* CONTENTS OF FR0 INTO FR1.
*-------------------------------
          LDY #$02
:F5       LDA FR0,Y
          STA FR1,Y
          DEY
          BPL :F5
          RTS
*-------------------------------
          *=  $02E0
          .WORD CONTINUE
```

●

Listing 7.
Assembly listing.

```
*-------------------------------
* SAVE #D:INTBASI2.ASM
* ASM,,#D:INTBASIC.OBJ
*-------------------------------
* by Barry Green
* Out of the Blue Associates
*-------------------------------
* This is a math package designed
* to replace the floating-point
* package in the ATARI 64K XL
* computers. This package is
* entirely integer-based so it
* is much faster. Its internal
* representation of numbers is
* still referred to as Floating-
* point, merely for convenience.
* This representation differs
* from standard 6502 integers in
* that there is an exponent byte
* of $40 added to it. Integers can
* still be used in FPI and IFP
* because these routines restore
* the numbers to proper order.
* This version accepts integers
* in the range of -32768 to 32767.
* Any number entered which is
* greater than 32767 will be
* converted to a negative number.
* These negatives are fully
* usable and have the same value
* as what was entered, they just
* print out as negative.
* For example, if the user entered
* FOR X=30000 to 65535 and then
* LISTed that line, it might say
* FOR X=30000 to -1. It will work
* exactly as you wanted, it just
* prints out as a negative.
* Statements such as POKE 53248,0
* are still perfectly legal, they
* just may look different than
* what you expected.
*-------------------------------
          .OPT NO LIST
FR0       =   $D4
FRE       =   $DA
FR1       =   $E0
INBUFF    =   $F3
CIX       =   $F2
SKIP.BLANKS = $DBA1
INTLBF    =   $DA51
LBUFF     =   $0580
FLPTR     =   $FC
*-------------------------------
          *=  $0600
START
*-------------------------------
* MOVES THE XL OS TO RAM
* THIS CODE IS BASED ON THE
* OFFICIAL RELEASE FROM ATARI INC.
* AND WAS WRITTEN BY THEM.
*-------------------------------
ROM       =   $CB
RAM       =   ROM+2
OSROM     =   $C000
OSRAM     =   $4000
NMIEN     =   $D40E
PORTB     =   $D301
          JSR INIT      SET POINTERS
MOV1      LDA (ROM),Y MOVE THE ROM
          STA (RAM),Y INTO RAM
          INY
          BNE MOV1
          INC ROM+1
          INC RAM+1
          DEX
          CPX #$30
          BNE :M1
          JSR SKIP      SKIP I/O
:M1       CPX #$00
          BNE MOV1
*-------------------------------
          SEI
          LDA #$00      DISABLE THE
          STA NMIEN     INTERRUPTS
          LDA #$FE
          STA PORTB     FLIP ROM OUT
*-------------------------------
          JSR INIT
MOV2      LDA (RAM),Y MOVE THE OS
          STA (ROM),Y BACK INTO THE
          INY           RIGHT ADDRESSES
          BNE MOV2
          INC ROM+1
          INC RAM+1
          DEX
          CPX #$30
          BNE :M2
          JSR SKIP
:M2       CPX #$00
          BNE MOV2
*-------------------------------
          CLI
          LDA #$40      RE-ENABLE THE
          STA NMIEN     INTERRUPTS AND
          RTS           RETURN.
*-------------------------------
* INITIALIZE PAGE ZERO VARIABLES
* FOR ADDRESSES OF RAM AND ROM
* USED FOR INDEXING
*-------------------------------
INIT      LDA # <OSROM
          STA ROM
          LDA # >OSROM
          STA ROM+1
          LDA # <OSRAM
          STA RAM
          LDA # >OSRAM
          STA RAM+1
          LDX #$40
          LDY #$00
          RTS
*-------------------------------
* SET PAGE ZERO VARIABLES TO
* SKIP THE HARDWARE REGISTERS
*-------------------------------
SKIP      LDY #$08
SKIP1     INC ROM+1
          INC RAM+1
          DEX
          DEY
          BNE SKIP1
          RTS
*-------------------------------
* SET THE INIT ADDRESS TO DOS 80
* ALL THE PREVIOUS CODE WILL BE
* EXECUTED AND THE REST OF THE
* PROGRAM WILL LOAD INTO THE TOP
* 16K AND REPLACE THE OLD MATH
* PACKAGE.
*-------------------------------
          *=  $02E2
          .WORD START
*-------------------------------
* THIS JUST RETURNS CONTROL TO
* DOS AFTER THE INTEGER MATH
* PACKAGE HAS BEEN LOADED IN.
*-------------------------------
CONTINUE
          RTS
*-------------------------------
* FOLLOWING ARE THE ACTUAL MATH
* ROUTINES WHICH LOAD RIGHT ON
* TOP OF THE OLD ONES.
*-------------------------------
          *=  $D800
AFP
*-------------------------------
* CONVERTS ASCII TO NUMBER VALUE
*-------------------------------
          JSR SKIP.BLANKS
          LDA #$40
          STA FR0
          LDA #$00
          STA FR0+1
          STA FR0+2
          STA FR0+3
          SEC
          PHP
:0        LDY CIX
          LDA (INBUFF),Y
          CMP #'0       IS IT A DIGIT?
          BCC :2        IF NOT, THEN
          CMP #'9+1     BRANCH TO
          BCS :2        THE RETURN.
          PLP
          CLC
          PHP
          AND #$0F
          PHA           SAVE THE DIGIT.
*-------------------------------
          LDA FR0+2     MULTIPLY THE
          STA FR0+3     NUMBER BY 10
          LDA FR0+1     BEFORE ADDING
          ASL A         IN THE NEW
          ROL FR0+3     DIGIT.
          ASL A
          ROL FR0+3
          CLC
          ADC FR0+1
          STA FR0+1
          LDA FR0+3
          ADC FR0+2
          STA FR0+2
          ASL FR0+1
          ROL FR0+2
*-------------------------------
          PLA
          CLC
          ADC FR0+1
          STA FR0+1
          BCC :B
          INC FR0+2
:B        INC CIX
          JMP :0
:2        PLP
          RTS
*-------------------------------
          *=  $DBE6
FASC
*-------------------------------
* CONVERTS INTEGERS INTO ASCII
* CHARACTER STRINGS.
*-------------------------------
          JSR INTLBF
          LDA FR0+1
          STA FRE+2
          LDA FR0+2
          STA FRE+3
          BPL :00       ;***
          LDA FR0+1     ;***
          EOR #$FF      ;***
          CLC           ;***
          ADC #$01      ;***
          STA FR0+1     ;***
          LDA FR0+2     ;***
          EOR #$FF      ;***
          ADC #$00      ;***
          STA FR0+2     ;***
          LDY #$00      ;***
          LDA #'-       ;***
          STA (INBUFF),Y ;***
          INC INBUFF    ;***
          BNE :00       ;***
          INC INBUFF+1  ;***
:00       LDY #$00
:02       LDX #$00
:01       LDA FR0+1
          CMP LO.BYTES,Y
          LDA FR0+2
          SBC HI.BYTES,Y
          BCC NEXT.DIGIT
          LDA FR0+1
          SEC
          SBC LO.BYTES,Y
          STA FR0+1
          LDA FR0+2
          SBC HI.BYTES,Y
          STA FR0+2
          INX
          JMP :01
NEXT.DIGIT
          TXA
          ORA #$30
          STA (INBUFF),Y
          INY
          CPY #$05
          BNE :02
          DEY
          LDA (INBUFF),Y
          ORA #$80
          STA (INBUFF),Y
*-------------------------------
* NOW REMOVE THE LEADING ZEROES
*-------------------------------
          LDY #$00
:03       LDA (INBUFF),Y
          CMP #'0
          BNE :04
          INY
          JMP :03
:04       STY FRE+4
          LDA INBUFF
          SEC
          SBC FRE+4
          STA FRE
          LDA INBUFF+1
          SBC #$00
          STA FRE+1
:05       LDA (INBUFF),Y
          STA (FRE),Y
          INY
          CPY #$05
          BNE :05
          LDA FRE+2
          STA FR0+1
          LDA FRE+3
          STA FR0+2
          BPL :06       ;***
          LDA INBUFF    ;***
          BNE :07       ;***
          DEC INBUFF+1  ;***
:07       DEC INBUFF    ;***
:06       RTS
*-------------------------------
* THE FOLLOWING ARE TABLES USED
```

```
* IN CONVERTING NUMBERS TO
* ASCII CHARACTER STRINGS.
*--------------------------------
HI.BYTES .BYTE  >10000
         .BYTE  >1000, >100
         .BYTE  >10, >1
LO.BYTES .BYTE  <10000
         .BYTE  <1000, <100
         .BYTE  <10, <1
*--------------------------------
         *=  $D9AA
IFP
*--------------------------------
* THIS ROUTINE WILL CONVERT AN
* INTEGER TO FLOATING POINT,
* WHICH IS SIMPLY LOW BYTE-HIGH
* BYTE FORMAT WITH AN EXPONENT.
*--------------------------------
         LDX FRO
         LDY FRO+1
         LDA #$40      ;EXPONENT
         STA FRO
         STX FRO+1
         STY FRO+2
         CLC
         RTS
*--------------------------------
         *=  $D9D2
FPI
*--------------------------------
* THESE ROUTINES WILL CONVERT A
* FLOATING POINT NUMBER BACK TO
* INTEGER FORMAT.
*--------------------------------
         LDA FRO+1
         STA FRO
         LDA FRO+2
         STA FRO+1
         CLC
         RTS
*--------------------------------
         *=  $DA60
FSUB
         JMP FSUB2
*--------------------------------
         *=  $DA66
FADD
*--------------------------------
* THESE ROUTINES WILL ADD AND
* SUBTRACT INTEGERS.
*--------------------------------
         LDA FRO+1
         CLC
         ADC FR1+1
         STA FRO+1
         LDA FRO+2
         ADC FR1+2
         STA FRO+2
         CLC
         RTS
*--------------------------------
FSUB2
*--------------------------------
         .LOCAL
         LDA FRO+1
         SEC
         SBC FR1+1
         STA FRO+1
         LDA FRO+2
         SBC FR1+2
         STA FRO+2
         ASL FRO       THIS SETS THE
         LDA FRO+2     SIGN BIT IN THE
         ASL A         EXPONENT FOR
         ROR FRO       BASIC'S COMPARE.
         LDA FRO+1     IF RESULT
         ORA FRO+2     IS ZERO THEN
         BNE :1        ZERO OUT THE
         STA FRO       EXPONENT.
:1       CLC
         RTS
*--------------------------------
         *=  $DADB
FMUL
*--------------------------------
* THIS ROUTINE WILL MULTIPLY
* THE CONTENTS OF FRO BY THE
* CONTENTS OF FR1 AND STORE THE
* PRODUCT BACK INTO FRO.
*--------------------------------
         .LOCAL
         LDA #0
         STA FRE+1
         STA FRE
         LDX #16
         BNE :1
:2       CLC
         LDA FRE+1
         ADC FR1+1
         STA FRE+1
         LDA FRE
         ADC FR1+2
         STA FRE
:1       LSR FRE
         ROR FRE+1
         ROR FRO+2
         ROR FRO+1
         DEX
         BMI :3
         BCC :1
         BCS :2
:3       CLC
         RTS
*--------------------------------
         *=  $DB28
FDIV
*--------------------------------
* THIS ROUTINE WILL DIVIDE THE
* CONTENTS OF FRO BY FR1 AND
* STORE THE RESULT INTO FRO.
*--------------------------------
         .LOCAL
         LDA FR1+1
         ORA FR1+2
         BEQ :ERROR
         LDA #0
         STA FRE+1
         STA FRE
         LDY #16
:2       ASL FRO+1
         ROL FRO+2
         ROL FRE+1
         ROL FRE
         SEC
         LDA FRE+1
         SBC FR1+1
         TAX
         LDA FRE
         SBC FR1+2
         BCC :1
         STX FRE+1
         STA FRE
         INC FRO+1
:1       DEY
         BNE :2
         CLC
         RTS
:ERROR
         SEC
         RTS
*--------------------------------
         *=  $DD89
FLDOR
*--------------------------------
* THIS ROUTINE WILL LOAD FRO
* FROM THE ADDRESS POINTED TO
* BY THE 6502 X,Y REGISTERS.
* OR FLDOP WILL LOAD IT FROM THE
* CURRENT ADDRESS OF FLPTR.
*--------------------------------
         STX FLPTR
         STY FLPTR+1
FLDOP
         LDY #$02
:F1      LDA (FLPTR),Y
         STA FRO,Y
         DEY
         BPL :F1
         RTS
*--------------------------------
         *=  $DD98
FLD1R
*--------------------------------
* THIS ROUTINE WILL LOAD FR1
* FROM THE ADDRESS POINTED TO
* BY THE 6502 X,Y REGISTERS.
* OR FLD1P WILL LOAD IT FROM THE
* CURRENT ADDRESS OF FLPTR.
*--------------------------------
         STX FLPTR
         STY FLPTR+1
FLD1P
         LDY #$02
:F3      LDA (FLPTR),Y
         STA FR1,Y
         DEY
         BPL :F3
         RTS
*--------------------------------
         *=  $DDA7
FSTOR
*--------------------------------
* THIS ROUTINE WILL STORE FRO
* INTO THE ADDRESS POINTED TO
* BY THE 6502 X,Y REGISTERS.
* OR FSTOP WILL STORE IT INTO THE
* CURRENT ADDRESS OF FLPTR.
*--------------------------------
         STX FLPTR
         STY FLPTR+1
FSTOP
         LDY #$02
:F4      LDA FRO,Y
         STA (FLPTR),Y
         DEY
         BPL :F4
         RTS
*--------------------------------
         *=  $DDB6
FMOVE
*--------------------------------
* THIS ROUTINE WILL MOVE THE
* CONTENTS OF FRO INTO FR1.
*--------------------------------
         LDY #$02
:F5      LDA FRO,Y
         STA FR1,Y
         DEY
         BPL :F5
         RTS
*--------------------------------
         *=  $02E0
         .WORD CONTINUE
```



Tactics

How to become a *Star Raiders* Star Commander Class 1.

by Dave Pettit

Having played Atari's **Star Raiders** for years, I've learned a few interesting ways to play it faster and more accurately. Some of my strategies are extensions of what the instruction manual says; some are applications of other people's strategies; and still others are unwritten facts of how the game progresses. I hope these ideas will help all players, from Novice to Commander.

The facts here have been grouped by topics, arranged in alphabetical order—except for "Miscellaneous Strategies" and specifics about the Commander Mission, which are placed at the end. Ideas are arranged within a category so that practically anyone can use the beginning suggestions, while more able navigators will see uses for the later concepts. When an idea involving damaged or destroyed equipment is given, it's placed at the section's end.

You can read this article through or use it as a reference. Say you want to find out what you can do to locate a starbase when both your Tracking Computer and Long-Range Scan are damaged. You should look under each section—Starbase, Tracking Computer and Long-Range Scan. You'll probably find just what you need in one.

Aft view.

1. When in aft view, the joystick directions are reversed from those of the front view. An easier way to learn this: the controls are the same as for hyperwarp in PILOT and higher missions (push left and go right; pull back and go down).

2. Don't use the strategy that some take—turning your starship around to get a pursuing enemy. That takes too much time, and you may get hit in the process.

3. If you *must* turn around (say, to pursue a distant enemy behind you), turn to the left or right, up or down, so the horizontal or vertical direction indicators (theta and phi) become larger in absolute value. For example, if you turn so the indicator changes from −350 to 0 and then from 0 to +475 (at which time, the Tracking Computer goes to front view), you'll have wasted a lot of time. It's better to go from −350 to −475 first. Keep turning to 0, once in front view, to center the enemy.

4. When turning around from aft to front view, an enemy can be as much as 40 metrons farther away.

5. If an enemy is pursuing you in aft view and your engines are on, you can slow way down (say, from 6 to 4) and keep firing as they approach. If they don't get hit, they'll probably pass you by. But, as they pass and your screen changes to the front view, you can speed back up to 6 or 7 to match their speed. This keeps them close to you, so you can shoot them when they least expect it. A very effective strategy, this does use a lot of energy (see "Engines," #1, below).

Engines.

1. Don't rely on your engines too much in finding the enemy; if they go out, you're practically stranded. Conserve fuel as much as possible and wait for the enemy to come to you.

2. William Colsher wrote in the November/December 1980 issue of *Compute!* that saving energy is one of the most important ways to increase your rating. He's right. But new players should chase and shoot at the same time, so they can practice aiming. This will keep most of the enemy in close range, where they'll be larger and easier to hit. Speeds of 5 or 6 are recommended here. In time, players will learn when to shoot and when to wait, based on where the enemy is on the screen.

3. Mr. Colsher does not emphasize enough that some enemy ships do not attack you—you have to go after them. The need for this can be determined by centering them in the Attack Computer Display and observing the range indicator. If the range is getting larger or staying constant, you'll have to chase them down with a speed of 6 or 7. If the range is getting smaller, wait them out.

4. Most enemies travel at 0 or 6. If you're chasing one at 6 and the range doesn't change (or if you don't catch them soon), they're playing cat and mouse with you. Increase your speed to 7 if you really want to catch them.

5. Sometimes you notice in the Long-

Range Scan (or with the Tracking Computer) that one enemy is pursuing you from the back, while another is standing still, dead ahead. By pressing a 4 or 5, you can head for the forward enemy and allow the other one to catch you. Then blast the one you see first and the other soon after. That saves a little time and, probably, some energy.

6. Practice moving at speeds of 7, 8 and 9. Try going this fast and shooting meteors. It's tough, but will improve your steering and aiming abilities.

7. Sometimes—at high speeds—it's impossible to turn around and face in another direction with the Tracking Computer on. You must stop all your engines, turn around, then turn on the engines again.

8. If you need to destroy your starbase or an enemy starship quickly, don't hesitate to use a speed of 8 or 9. The loss of energy is small when weighed against the loss of a starbase to Zylon ships.

9. A speed of 9 with damaged engines is the same as a speed of 7 with normal engines. Use this factor to catch a fleeing enemy.

10. If your engines do get destroyed while your enemy is 300 metrons away and not approaching you, you can catch the ship using your Long-Range Scan and hyperwarp. Simply set up the enemy directly in front of you with the Long-Range Scan (see "Long-Range Scan," #6). Then press the *H* key and steer toward that ship, so that it remains in front of you till it's in the first third of the screen in front of your location (you're still in Long-Range Scan). Now press any number key and the *F* key. The enemy should be within visual range. This will cost you only about 100 mergs (units of energy), the same amount you'd use with your engines *working* to take out an enemy at the same distance—but this method is considerably faster. You'll have to experiment with this to get it to work for you.

11. Use the above technique, but, instead of pressing a number key to coast toward the enemy, time the pressing of the number key so that the enemy will pass by you a bit and have to catch up later. This may avoid your getting blasted from the front.

12. If your engines and Long-Range Scan are *both* destroyed, you can use hyperwarp in short bursts to catch an enemy or to get closer to a starbase. Be sure to keep an eye peeled for a passing Zylon starship or starbase. Be careful, or this can waste a lot of time and energy.

13. Destroyed engines operate at a normal speed of about 5 when any key from 5 to 9 is pressed. Keys from 1 to 4 produce speeds just smaller than your normal 1 to 4 speeds.

Galactic Chart.

1. The enemy will move on Star Dates in x.00 and x.50, except 0.50 *and* for 100 centons after surrounding a starbase. It's helpful to know this when you're starting a new game or wiping out the enemy around a starbase—they (and all other Zylon starships) sit and wait for 100 centons, even if the starbase is no longer surrounded!

2. Normally, when beginning a hyperjump, it takes about 8 centons to complete. Thus, the Star Date should not be in the ranges of x.42 to x.49 or x.92 to x.99. However, it's possible to speed up your travel time by remaining with the Galactic Chart on-screen for a few seconds. This can reduce travel time by 1 or 2 centons, but be careful in missions above Novice—you may not be able to recenter the target marker quickly enough to get to the proper sector.

3. Enemy starships line themselves up horizontally and vertically, with a starbase first. They then move in a straight line toward the base to surround it. They seldom move diagonally (see "Galactic Chart," #11).

4. Enemy ships *do* move diagonally when traveling around a starbase.

5. Zylon starship sectors of 1 or 2 enemy ships (patrol groups) usually move every 50 centons. Use this to predict their travels. You decide if you'll have time to destroy a 4-Zylon sector. You might decide to aim for a blind sector if your Sub-Space Radio is out, or if you didn't watch the clock well (refer to "Galactic Chart," #2, above).

6. If the enemy seems to be converging on a starbase on the left and the Zylon starships are on the right, most patrol groups will move toward the base in a horizontal or vertical line. Thus, you can predict the enemy's next sector. Use this information to plan your next move when the Star Date's about to change ("Galactic Chart," #2, above), or while waiting for a distant Zylon starship in your sector (see "Galactic Chart," #8, below; also see #11 for the reason why and the movement of Zylon starships in the other direction).

7. The enemy will most often move toward a group of starbases, rather than a lone one. But that doesn't mean that they *never* go for the lone bases.

8. If you're waiting for the final Zylon ship in a sector to approach and attack, use the Galactic Chart to plan and position your next move. After destroying the Zylon starship—and if the Star Date permits (see "Galactic Chart," #2)—you can hyperjump without looking at the chart again. This can save time and energy.

9. When you've been in a sector for a considerable amount of time, consider updating the Galactic Chart. You can do this quickly by typing *GF*. The fraction of a second that the chart is on will be enough to update it. You won't miss too much action, and you'll be able to avoid problems should your Sub-Space Radio go out.

10. Also, type a quick *GF* when a starbase is first surrounded and you choose to finish clearing the sector you're already in.

11. The Zylon starships in the Galactic Chart are positioned from the left side of the top row. Each sector is placed or left alone, through to the last sector in that row. Each row is positioned in this way, with the sector in the lower right located last. If a series of enemies is traveling toward a starbase on the right, the leftmost Zylon starship sectors will move diagonally. *If the* Zylon starships are clumped to the right, moving toward a starbase on the left, all sectors could move as a group.

12. When you've eliminated most enemy sectors and enemy ships are grouped, use the rook-mate strategy of chess—don't allow any enemy to pass a chosen horizontal or vertical line in the chart. Slowly eliminate the closest enemy first, eventually moving through all enemy sectors.

13. Groups of four enemy starships are slower and don't move often. They're good bets for remaining stationary when your Sub-Space Radio is out, or when you've forgotten to check the Star Date before selecting hyperwarp.

Hyperwarp.

1. When in hyperwarp, the directions say that it's necessary to keep the target marker in the center of the cross hairs. This is true only at the critical moment of entering hyperspace—that is, when the velocity reaches 99 metrons/second. Knowing this will allow you to scratch your nose, make a quick check of the Galactic Chart or do practically whatever you want—and still reach the sector you aimed for.

2. Use as many of your senses as you can. Listen to the sound of your engines at the moment before entering hyperwarp. If you can learn what that volume is, keeping the target marker in the right place at the right time will be easier.

3. Using jerky wiggles of the joystick is the easiest way to steer. Also, better control can be obtained by holding the top of the stick, rather than the middle.

4. Mr. Colsher is generally correct in his rule about not jumping more than four sectors at one time. The cutoff point actually occurs when the hyperwarp energy required changes from 260 to 500 mergs. Use two jumps, rather than a single energy-wasting one, to get to the desired sector.

5. If you must use 260 mergs in a hyperjump, be sure to steer carefully, or you may go off course by enough to use 500 mergs. If in doubt, either set up 250 mergs while on the Galactic Chart, or aim back from the center of the cross hair a little bit (see "Hyperwarp," #8).

6. Mr. Colsher's rule ("Hyperwarp," #4, above) ought to be amended further —don't hyperjump too far *except in an emergency*. The emergency might be a surrounded starbase or a lack of photons or shields. Just don't do it *often* in a game.

7. Be extra careful when hyperjumping to a sector on the edge of the Galactic Chart. A small error in navigation may put you on the wrong side of the galaxy, not to mention causing a huge energy loss.

8. A little experimentation will show that

the book is right: if you position the hyperwarp target marker a little off center, you can hyperjump to a neighboring sector from the one set in the Galactic Chart. This might be helpful if your shields go down, or if the enemy moves just as you press *H*. It can also be used when you know (or can bet) that the enemy will move from where you last saw them. Simply aim the target marker off center by one or more widths of the marker for each sector that you wish to move (see Figure 1). You should be able to move up to four sectors away with only a modicum of experimenting. You could, for example, display the Galactic Chart, find a nearby enemy or starbase, and hyperjump there *without* doing any positioning on the chart. This really speeds up the game. You should try this in Novice level first; it's much easier there.

```
      |                  |     +
+—   ——        ——   ——
      |                  |
      A                  B
```

Figure 1.

If no aim is given, the target marker (+) is positioned off of a normal Galactic Chart aim, or away from your present sector—either by two sectors to the left (A) or by three sectors to the right and one up (B).

9. Not only can't you shoot in hyperwarp, but you can't be shot *at*. At least, you can't be shot at as long as you've reached a minimum speed.

10. When entering an enemy or starbase sector, or when seeking out a distant Zylon ship, use the Attack Computer Display and your own hyperjump momentum to help steer the ship to the desired location. If the target is right or left of center in the display, push the joystick in that direction. If the target is high, pull back (called "nose up"); if low, push forward (called "nose down"). It's possible to steer yourself to within sight range of a starbase more often than not by using this method.

11. Use hyperwarp within a starbase sector to get to your goal faster. This is especially handy with distant starbases and/or destroyed engines. Use your engines for docking maneuvers as needed.

12. In hyperwarp, the range indicator will work for distance to a starbase in your sector, but not for distance to an enemy.

13. During hyperwarp, the Sub-Space Radio doesn't update the Galactic Chart. If a starbase is surrounded because of poor timing, you will have to change course (see "Hyperwarp," #8), do a quick check of the Galactic Chart (see "Pause Key," #2), or cancel the hyperwarp.

14. If your Tracking Computer goes out, steering to another sector in hyperwarp can be difficult. The center of the screen—where the tarket marker is supposed to be—is the point at which no stars appear. You can see this easily in Novice level by starting hyperwarp and turning your Tracking Computer off. The target marker will be positioned correctly.

Long-Range Scan.

1. Another way of centering an enemy, instead of the Attack Computer Display, is the use of the horizontal and vertical displays on the Long-Range Scan screen. By adjusting them both to 0, you'll find the target is straight ahead (see "Long-Range Scan," #6, below).

2. On the Long-Range Scan, little orange rectangles represent the Zylon starships and a "dummy starship." Which is which? The one that disappears and reappears occasionally is the dummy, so go after the other one.

3. The orange rectangles that shoot or move rapidly are the enemy.

4. When in a starbase sector, the rectangle changes to a starbase shape.

5. When you've no better clues as to which rectangle is the dummy (and your Tracking Computer is out) go from *L* to *G* or *F*, and then back to *L*. If one of the rectangles moved drastically and isn't moving much now (or has disappeared), it is probably the dummy.

6. If your Tracking Computer is destroyed, it's still possible for you to get closer to a starbase or the enemy. As soon as you've entered the sector, press *L*. Whatever you're searching for will come into view on the Long-Range Scan screen. Position it in the top half of the screen, directly in front of your ship by pushing the joystick left or right. Then move the stick forward and backward to "stretch out" the target—to get it to its maximum distance from you. (This is the same as being in the center of the Attack Computer Display in the front view.) As soon as the target is close to the center of the Long-Range Scan, go to the front view and dock or shoot, whatever is appropriate.

7. Apparently, it's not possible for both the Tracking Computer and Long-Range Scan to be destroyed together. At least one will be usable to locate an enemy or a starbase.

8. Of course, a destroyed Long-Range Scan won't tell you where the enemy ships are, but it will tell if there are none, one, or (at least) two of them in your sector. Just count the little orange rectangles. One is a dummy; any others are Zylon starships.

Manual Target Selector.

1. In a sector with more than one Zylon ship, don't be tricked, while waiting for one enemy, into ignoring another. Use the *M* key and the range indicator to see if another enemy is approaching— and to find out which one will get to you first.

2. When entering an enemy sector where a large distance must be traveled to catch an enemy, use the Manual Target Selector to see if a second enemy is closer. If less than 400 metrons away, an enemy can usually be caught with hyperwarp momentum (mentioned earlier; "see Long-Range Scan," #6).

3. Sometimes you can cause an enemy to approach you by using the *M* key. It's as if they realize that they're being "scanned," so they decide to attack.

4. When all Zylon starships in a sector are killed, the Manual Target Selector will switch to two different values. Don't be confused and start looking for non-existent enemies.

Pause Key.

1. You can use the pause key (*P*) to temporarily stop the game action and plan an attack strategy. However, some purists may find this a form of cheating.

2. Use this key if you've just entered hyperwarp, then received notice of a surrounded or destroyed starbase. To do this, type *GP* quickly and don't touch the joystick. Determine what your move should be, realizing that the Galactic Chart hasn't been updated since you saw it last (see "Hyperwarp," #13). Plan on using offset navigation of the target marker (see "Hyperwarp," #8), then press the *F* key before moving the stick, so you can start steering as soon as you disengage the pause. Of course, if you decide to cancel hyperwarp, press a number key and move the joystick.

3. If you pause long enough, the enemy's strategy may change. Many times a Zylon ship that won't pursue you when your engines are out *will* pursue you after several minutes on pause. This may only be a coincidence (it doesn't happen every time), but it's been observed after many unplanned interruptions.

Photons—yours and theirs.

1. You can't hit an enemy often by just shooting. You need to steer with the joystick, then fire. It takes a coordinated effort, frustrating many beginning players.

2. Zylon starships can shoot only one photon at you at a time. You, however, can shoot photons two at a time.

3. It's best to shoot in bursts of two. With the photons coming out of alternate tubes, you may forget which one will fire next. By shooting twice, you can guarantee that the tube you want to fire *will*.

4. Many times an enemy is destroyed just after they've shot at you. Don't get caught by that last shot. Either get out of the photon's way or shoot it down, too.

5. The cross hairs in the front and aft views are set for distant shots. The closer the enemy, the lower the ship must be in the view screen for you to hit it directly with a photon. Seldom can an enemy be hit above the horizontal cross hair (but see "Photons," #12). You can check this by firing two shots very quickly and freezing them with the *P* key. You can continue to release and freeze them by alternately moving the joystick a small amount and then pressing *P* again.

6. When an enemy keeps matching you

with photon after photon, only to have them both explode, there are two ways to hit that ship. First, wait for the enemy's photon to come very close to you—but low enough so that it will pass without damage. Then fire away. The photons will pass each other, with yours striking the enemy.

7. The second way to get around this problem is to turn your ship to the left or right, so the other photon tube can be used to hit the enemy ship. It's as if the enemy keeps blocking your right jabs, then gets punched with your left hook!

8. A long, distant shot coming toward you (especially in Command level) can be hard to avoid or destroy. Normal reaction for a high photon is to pull back on the joystick, going nose up. Instead, do the reverse: push forward after shooting your photon. If you time it right, your shot will float up and strike the enemy's photon.

9. The only time a photon of yours will curve up by itself is when the enemy is dead center and very close, called "lock-on" in the manual. Both photons fire in this condition. Don't try to create this condition. Instead, learn how to kill an enemy with single shots when you're ready, rather than waiting for the ship to reach the right position.

10. One time that lock-on is effective and frequent is in combatting an enemy at point-blank range. Usually, each single shot blocks one of the enemy's. When double shots are sent out, one usually blocks an opponent's shot, while the other takes out the starship. Sometimes, when a shot misses the enemy, this process requires three or more pairs of shots.

11. When shooting the enemy at long range, give your photons enough time to reach the Zylon starships before shooting again. With the game only being able to keep track of two photons—one from each photon tube—at a time, you don't want to waste a perfect shot by shooting again. You can see this in the Long-Range Scan by firing twice, waiting a few seconds and firing again. The farthest photon will disappear first.

12. It is even possible to steer a photon after it has been fired! You can prove this by firing a photon and then moving the joystick to the right or left. If you hold the stick this way long enough, you'll see the photon cross to the opposite side of the screen! By causing a nose-down action in front view (joystick forward), you can make a photon go above the horizontal cross hair. With practice, you can direct shots to hit enemy starships at great distances—and on opposite sides of the screen. Using this will save time and energy by destroying the enemy more quickly.

13. Here's how to shoot and steer upon entering a sector. First, use the Attack Computer Display for initial steering (see "Hyperwarp," #10). Then, watch the range to the enemy. When it is less than 200 metrons, shoot two shots. If you can see the enemy, steer one of the shots toward them. But don't waste your time and energy firing ten or twenty times at nothing.

14. Don't try to hit an enemy in Long-Range Scan. You won't be successful often enough to make it worth your while.

15. Shooting at an enemy farther away than 120 metrons may put them into attack mode. They will then come to get you. Try this in front view and in Long-Range Scan, too.

16. If a Zylon starship shoots and is destroyed, but your Tracking Computer changes views, you may need to avoid the enemy photon. To do this, turn away from the photon *hard!* After you're sure the photon has passed you, you may continue your attack on the next starship.

17. If you listen carefully, you may notice a slightly different sound when you fire a photon after your photons have been damaged. The sound has a slightly deeper pitch.

18. If in a heavy battle, where new damage to your ship has just occurred, fire one or two photons to make sure they're still working. Don't wait for the damage report and a Zylon ship to start attacking.

19. If your photons are damaged, it can be difficult to destroy a close Zylon starship on the same side as the damaged photon tube. What you need to do is keep the enemy low on the screen, as you move your ship to position enemies on the other half of the screen. Usually, they'll still be shooting in the same direction as they have been. As soon as they shoot, and when they're right in front of the working photon tube, blast away!

Shields.

1. You're always two shots or less from death: one for your shields and one for you. Be prepared to go into hyperwarp quickly when your shields go out, or you may die trying.

2. If your shields go out, press *H* as quickly as possible. Don't worry about viewing the Galactic Chart. Just get out of there! When you have more time to think, move to a starbase with the help of the chart, and get your shields repaired. (Also, see "Pause Key," #2.)

3. If your shields are destroyed and you're not being blasted by a close enemy, you might want to stay put and clear the sector. When that's been done, or if a more hazardous situation develops, by all means get out fast.

4. If your shields are destroyed and you choose to play more, turn them off. It makes the screen easier to read and stops wasting valuable energy. Getting hit with no shields is the same as getting hit with destroyed shields. Just remember to turn them on when docking is over.

5. If your shields are out, don't use your engines unless you're in front view. A meteor may destroy you.

6. Before leaving a sector with no shields, type *F* and the *H*, rather than the other way around. You may be able to avoid a meteor on your move. If the aft view appears, your forward path should be clear.

Starbases.

1. Games in which all the starbases are grouped together are easier to win than those in which they're spread out. After trying to surround one starbase and failing, the enemy will move to another that is, in this case, close by. Some players may consider this cheating, but it's a good temporary strategy.

2. You need to get close to a starbase to dock, but how close? When you see three windows on each side of the starbase, stop your engines and move the joystick until the *Docking Completed* message appears.

3. It takes 16 centrons to complete repairs after docking. Use this and the time to enter and exit (8 centrons each) to decide when to destroy a surrounded starbase yourself, when to stay docked, or when to attempt a docking. There's also a varying amount of time to locate a starbase and dock with it.

4. Docking too often wastes time and energy. Use the following priority list for decisions on docking: (1) photons destroyed; (2) shields destroyed; (3) Sub-Space Radio destroyed; and (4) other problems. (See "Shields," all paragraphs, and "Sub-Space Radio," #7, for more details and suggestions.

5. While waiting for repairs at a starbase, use the Galactic Chart to plan your next move. Then hyperwarp as soon as docking is completed.

6. There are no meteors in a starbase sector, so turn your shields off when in these sectors to save energy. After docking, remember to turn them back on.

7. To save energy when docking, turn off the Tracking Computer, as there's no need for it once the starbase is in sight. After docking is over, be sure to turn it back on.

8. It's possible to steer your ship with the momentum of hyperwarp directly to your starbase. If your range to the starbase upon entering the sector is 300 to 400 metrons, you can usually do it (see "Hyperwarp," #10). Practice.

9. If your hyperwarp momentum appears to be too fast and the starbase too close to dock, you can add some traveling distance by porpoising. This is done by making your ship go up and down several times very quickly. Do this by pushing forward and back on the joystick, quickly. In Figure 2, you can see that your starship will climb and drive to add the needed distance and avoid passing the starbase.

10. If a starbase is about to be totally surrounded, you can attack early. By entering a sector next to that starbase, you can be destroying the enemy before the starbase is surrounded. You will then have a little less than 100 metrons to destroy the now stationary Zylon sectors.

11. When a starbase is surrounded, go

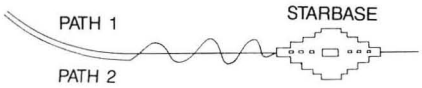


Figure 2.

The side view of your entering the sector from the left demonstrates two paths: Path 1 shows normal entry and passing the starbase; Path 2 shows porpoising action to shorten overall travel distance, to keep the starbase in front of your ship and, possibly, to complete docking maneuvers.

after the groups of three or less Zylon starships. You need to be very skilled and have a lot of time to take out a group of four.

12. If you have just barely cleared a sector around a surrounded starbase and need to dock for repairs, it may be to your advantage to wait for the next Zylon starship movement at star date x.50 or x.00. By your staying there, the Zylon starships can't completely surround that starbase on that move. This will give you at least 150 metrons to hyperwarp, dock, hyperwarp again, and destroy another enemy sector before the starbase could be surrounded again and destroyed.

13. If a starbase has been surrounded for too long and its destruction is inevitable, do it yourself.

14. It's possible for more than one starbase to be surrounded at one time. This can happen when the two starbases are close together and several enemy have converged on the area. To prevent a double loss, destroy the Zylon starships in an intersecting sector as in Figure 3, below.

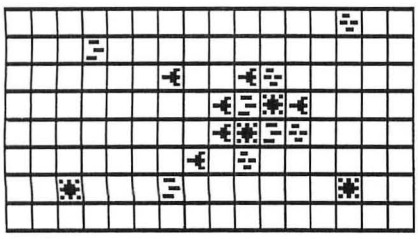


Figure 3.

Two starbases (S) are surrounded. To save them, attack either common sector with three Zylon starships.

15. Suppose your starship is badly damaged: your Long-Range Scanner and Tracking Computer are out, so navigation is difficult. You decide to go to a starbase, but have always had trouble finding them. Don't worry! Most of the time you'll come within visual range of a starbase after a hyperjump.

16. When all starbases have been destroyed, the enemy stop moving. Use your last Galactic Chart as a guide. All will not be lost now if your radio goes out, but you won't make Star Commander this way.

Sub-Space Radio.

1. If a starbase is surrounded or destroyed just before you receive some damage or hyperjump into an enemy sector, the sound (and, sometimes, the word messages) about the starbase will be bypassed for the new message.

2. If your Sub-Space Radio is damaged, it still functions but doesn't update the Galactic Chart. Simply move through the chart with the joystick and watch the number of targets indicator. As long as it's zero, keep searching.

3. Although you can find enemy sectors with a damaged radio by using the targets indicator, a lot of time can be wasted. Also, you don't get the big picture of enemy movement.

4. With a damaged radio, you must be careful. You won't be able to see if any starbase is about to be surrounded or destroyed. Of course, you will be notified by word message and beeps, when one is affected.

5. If a starbase is surrounded while your radio is damaged, you can use the Galactic Chart and still find which one. Watch the targets indicator while searching the sectors around each starbase. If any sector has no enemy, look around another starbase. And if the starbase is adjacent to a second starbase, make sure the second one's not surrounded, as well (see "Starbases," #14). If it is, attack any sector of common enemy Zylon starships, no matter how many there are, or you'll lose one or both starbases.

6. With a damaged radio, you won't know of starbases that have moved since the last chart update. Otherwise, assume that they're as shown on the chart.

7. If your radio gets blasted, go ahead and get some more enemy before docking. But don't wait too long or get too greedy—you may discover that one or more starbases have been surrounded or destroyed while you were fighting!

Tracking Computer.

1. Turning on your Tracking Computer will help in shooting and in locating enemy Zylon starships. Use the crosshairs as a guide in aiming your shots.

2. The instruction book does not recommend the use of the Tracking Computer in Novice level, probably to reduce player confusion and because the enemy won't attack from behind you. When used at this level, however, you can shoot at the enemy in the aft view whenever possible. This can be helpful, as they can't block your shots in this view. (See "Zylon Starships," #2 and #4).

3. The automatic tracking system of the Tracking Computer doesn't use any extra energy. It will change the screen to front or aft view, to show the direction of the enemy who fired last. The only shortcomings occur in a crossfire (see "Zylon Starships," #19, #20, and #21) or when one enemy shoots and gets killed, but their shot still hits after your view switches (see "Photons," #16).

4. You don't need to center an oncoming enemy with the Attack Computer Display. For the most part, Zylon starships are "self-centering" on the attack —they seldom go for your blind sides.

5. Don't look at the Attack Computer Display when the enemy is in visual range; look at the enemy directly. The display should be used when searching out distant enemies (see "Tracking Computer," #6, below).

6. If a distant enemy or your starbase goes off the screen, steer in the direction of the Attack Computer Display. If the image is in the lower left, for example, push the joystick to the front and left. Your ship will start pointing toward its object and, eventually, face the centered and/or visible target.

7. Don't pursue a distant enemy totally through the use of the Attack Computer Display. You may ignore a meteor or a surprise attack by a second Zylon starship.

8. By using the Attack Computer Display with a damaged Tracking Computer, you can still get to a target, but you won't be able to use the number displays at the bottom of the screen. Instead, try to get the target centered in the Attack Computer Display. Then (or even while centering) use your engines to get to the target (see "Engines," #4 and #9). Practice helps.

9. When your Long-Range Scan is destroyed, you can still find an enemy or your starbase with a partly or fully functional Tracking Computer (see "Tracking Computer," #10, below). If the Tracking Computer is working, position the target in front of you (front view screen and a positive distance away). Then position it to the center of the Attack Computer Display and, with the horizontal and vertical indicators on the screen set to about 0, engage your engines (or Hyperwarp as described in "Engines," #12), and steer with the target centered.

10. When your Tracking Computer is destroyed and you're waiting for an enemy to attack, be sure to occasionally flip back and forth between front and aft views, or you might be surprised by another Zylon.

11. If your Tracking Computer is destroyed, it has to be turned on again after being repaired at a starbase. This is the only device that needs action when destroyed and repaired.

Zylon Starships.

1. Know how many Zylon starships are in a sector when you enter and count them as they're killed, so you won't get hit by surprise or exit too soon.

2. If you're playing at Novice level, you don't have to pursue the enemy in the sectors—they'll come to you all of the time.

3. Most of the enemy will come to you in the other missions, too, if you give them

a chance. If the numbers are getting closer to 0 in the range indicator, then sit back and wait.

4. If the enemy were visible on-screen at one time in Novice level (and sometimes in other missions), but can't be seen now, do not move to find them. Stop all movement with the joystick and engines (press *0* to stop engines), and let the enemy come into view. They'll become visible again in either front or aft view, unless you've outrun them.

5. Some enemy will seem to be coming toward you and ready to attack. At a range indicator value of about 150 and 450 metrons, they stop and reverse directions. Now you must pursue them at a speed of 7 or more.

6. When an enemy is centered in the Attack Computer Display, it will be visible in the view screen at about 120 metrons, using the range indicator.

7. When an enemy first appears on the screen, it will show up as a yellow dot (just smaller than a white star) that usually moves against the background of stars. This is most evident when you're not moving, but it can be detected at any speed or time (including Hyperwarp deceleration, even if the screen is flashing red and blue —watch carefully). Many players don't concentrate enough to see this.

8. Before some ships appear on the screen, a meteor is seen. This is like a decoy. Don't attack it; you may be caught by surprise by the Zylons. Instead, just sit tight and get ready to shoot at the correct target.

9. Don't always shoot at the meteors. They can indicate that an enemy's nearby; when a Zylon starship shoots a photon, all meteors disappear.

10. Many of your distant shots can get blocked by an enemy shot, and a cloud of debris hides the enemy. Don't let these fool you. The enemy will stay hidden as long as possible, attacking when (and from where) you least expect it. This cloud can also be created by blasting one enemy, only to have another hide in the dust. If you have the Tracking Computer on, there'll be little doubt of killing the enemy—the Tracking Computer will automatically switch to the opposite view screen if the enemy was blasted and a second enemy is in the other direction.

11. With practice, you can predict a Zylon starship's path before he makes it! Many times they'll move right into your shots after you've made them. For example, if a Zylon starship is hovering for a long time in the top half of the screen, his next move has to be down. By shooting first (before he crosses the middle of the screen), most times you'll destroy him through his own navigation.

12. Some Zylon starships enter high on the screen and shoot before crossing the horizontal line in the crosshairs of the Tracking Computer. The solution? Let them cross that point when off of center so that their shot will miss you. Then, reposition them below the horizontal line and blast away. Or go after the shot first, then the enemy.

13. Basestars can be destroyed at close range, usually with one shot. Getting them into position is hazardous at times, as well as difficult. A conservative way of destroying them is to keep firing and hitting them, even though they're too far away. It's as if their shields weaken with repeated attacks, until they're finally destroyed with what seems the weakest of hits.

14. Another time that lock-on is effective (see "Photons," #9 and #10) is on first approach of a Basestar in Novice through Warrior missions. Their first attack is straight down the center. That will be their last attack, if you wait patiently to time it right. In Commander level, they fire sooner, making it a little more complicated—you may get them or their shot, but seldom both.

15. Basestars can also be positioned for destruction very nicely. Shoot while chasing them at speeds of 6 or 7. It does take some practice to steer while moving at such speeds. Try working up to those speeds and higher by practicing with 4 and 5. (However, see "Engines," #1 and #2.)

16. Another way of blasting a Basestar is by hitting their photon just as they fire it. The combination of both photons exploding so close is too much for their shields. However, this is a strategy of coincidence and luck.

17. Shoot at enemy Basestars at long range, even if there's little chance of killing them. You will at least keep them "in your sights" and also be blocking their shots, when made.

18. Enemy ships have various strategies, including the following:

(a) Pursue you at all cost (see "Zylon Starships," #3 and #4);

(b) Avoid you at all cost (see "Engines," #3 and #4);

(c) Remain stationary and out of range (see "Engines," #3);

(d) Travel back and forth at a distance from you (see "Engines," #3 and #4);

(e) Attack when centered with the Attack Computer Display or when scanned using the *M* key (see "Manual Target Selector," #3);

(f) Sit under your nose at about 15 metrons distance and wait for a sneak attack;

(g) Always attack in front view;

(h) Always attack in the aft view; and

(i) Two enemy in a crossfire (see the next three entries).

19. The Tracking Computer can be disasterous in a crossfire, if you aren't careful. There are several things that you can do to get out of a crossfire. First, turn off the Tracking Computer and concentrate on one enemy. When they've been blasted, turn the Tracking Computer back on and blast the other one.

20. Second, press 8 or 9 and get out of there! After a few seconds, press 0. Sit and wait for them to catch you in aft view—they almost always will—and blast them as they show up.

21. Third, concentrate on one of the pair of enemy, but leave the Tracking Computer on. Avoid getting hit by the other Zylon starship, but don't attack them. Whenever facing the chosen enemy, concentrate on its destruction. The problem in a crossfire is that so much time is wasted in repositioning for each player that it's too late when the enemy's finally in your sights. At that time, the other enemy usually fires, causing the Tracking Computer to change views and mess up your aim.

22. If a mass of enemy is moving toward a distant starbase on the opposite side of the Galactic Chart, you can use one of two strategies. The first is to attack the slowest sectors and gradually destroy all of them.

23. The second is to attack the fastest and forward-most sectors. By always destroying the leaders, you keep the enemy nearer to you and avoid a surrounded starbase. This method works best with either a slow-moving or small group of enemy.

24. If a starbase is surrounded and you're on the other side of the galaxy, you need to get there fast—but efficiently. Using the small-jump method (see "Hyperwarp," #4 and "Zylon Starships," #25) with the shoot-and-fly method (see "Hyperwarp," #10 and "Miscellaneous Strategies," #2), you can reduce (and sometimes eliminate) the enemy in other sectors as you go, and still have time to save your starbase.

25. Use checkerboard-type jumping to move across the galaxy (see Figure 4, below). This will work in destroying isolated sectors of Zylon starships and in hurrying to save a distant starbase (see "Zylon Starships," #24).

26. Sometimes an enemy on one edge of the Galactic Chart moves to the other edge. This is a problem if all of the enemy are on one side of the Galactic Chart—you'll eventually have to travel the length of the galaxy to get them. (Too bad you can't just go over the edge for as little energy as a single sector.) The solution is to get them before they can move. When you have a choice, get the enemy on the edge of the galaxy rather than the enemy one or two sectors in from the edge.

27. If you're having a hard time catching or blasting an enemy, it's possible to get a different enemy (or enemy strategy) by leaving the sector and coming back immediately, or later in the game. Some purists may find this a form of cheating. Use

it at your own discretion, and realize that you will use extra energy to do it.

28. Don't get blasted with less than three sectors of Zylon starships. Sure, no starbases can be destroyed, but you can be! You're always two shots or less away from destruction—one for your shields, and one for you.

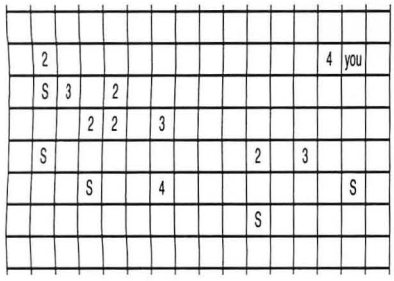


Figure 4.

Using checkerboard jumping, you can get to the other side of the galaxy and clear several sectors, too. Do it in this example by traveling left to right (toward the starbase in the upper left), through the nearby sectors with 4, 3, 2, 4, and 3 Zylon starships.

Miscellaneous Strategies and Variations.

1. Various strategies for destroying all enemy sectors on the Galactic Chart can be used. Generally, start in one area and try to eliminate all sectors. Then, gradually move through other enemy sectors while travelling toward the starbase that will apparently be surrounded. Of course, if the base is surrounded, a more defensive strategy is needed to save it (see "Commander Mission," #1 for a specific application.)

2. Change a 4-enemy sector to a 3- or 2-enemy sector, to help them move together faster if you don't have time to clear the sector. Generally, kill only the one or two Zylon starships that first appear. Don't wait too long for them to get to you.

3. Learn the keyboard positions by feel, rather than by sight. Do this at least for the *F*, *G* and *H* keys, as they're used most often.

4. Learn to steer with the joystick using one hand, so your other hand can work the keyboard. This will help in positioning in the Galactic Chart, Long-Range Scan, and, sometimes, in front and aft views. It shouldn't be necessary to do this for very much of a game—just some parts of it.

5. If the Galactic Chart is poorly arranged or your ship is damaged very quickly, you can always press START. This may be considered a form of cheating, but we all tend to do better with positive feedback and success.

6. If a series of consecutive games is quite hard to win, little change in enemy strategies or destructive resistance will occur upon pressing START or SYSTEM RESET. You may have to turn off the computer for a few seconds and try again that way.

7. If you get tired of regular play, try some variations in game play and goals. Can you complete a Novice game without any shields? Can you earn more than a Lieutenant Class 1 this way? And how many enemy can you destroy in the other levels without dying?

8. Once you've made Star Commander Class 1, you can try for the most games in a row with that rating. Can you triumph in fourteen consecutive games without quitting in the middle or getting destroyed?

9. The game can be made into a 2-player game. One player uses the joystick and calls out commands for the other to carry out on the keyboard. The commands could be "Galactic Chart," followed soon by "Hyperwarp—front view." This is good training for an inexperienced player, who can control the keyboard while watching and learning.

10. Reread the instruction manual. You'll probably learn many more details that you missed in your first reading.

Commander Mission.

1. In Commander level, an effective strategy is to destroy all the four-Zylon-starship sectors you can before a starbase is surrounded. This works well because the other enemy will move more easily to surround the starbases. But four-Zylon-starship sectors move so seldom that they'll almost always be in the same general area where they started. By eliminating them early, you'll make many parts of the galaxy free and clear, and the enemy will be grouped for easy travel from sector to sector on little energy.

2. It's possible to earn Star Commander Class 1 and have a starbase destroyed by you or the enemy. Don't give up after one is lost, but don't plan on the top rank after the loss of two!

3. It's also possible to complete a game without docking. Your energy level can get very low, so be careful. You're almost assured a top ranking this way.

4. It is possible to be destroyed and earn a Star Commander Class 1 rating. Very few enemy are usually left alive, and your shooting during the game was otherwise superb.

5. The last sector is usually the most difficult to clear. It may take several attempts at Commander level just to complete. Sometimes there will only be one ship left to destroy before your own demise.

Conclusion.

If you seem to be in a rut in an advanced mission and can't get any high scores, try an easier mission! By practicing in the lower games, you can improve some of your skills. On returning to the harder levels, you'll probably do better. And don't think that you'll still be as good next month as you are now. You're going to have to warm up or keep practicing to maintain your skills and ratings.

Don't be afraid to experiment. After all, **Star Raiders** is only a game. And, unless you have a winning streak going, it's okay to try some of these ideas and incorporate them into your overall game play. Or, perhaps you want to see if you can improve your aim in aft view, or make some improvements in docking by going back and forth between starbases. If you get blown up or run out of energy, press START and keep flying!

Attention Programmers!

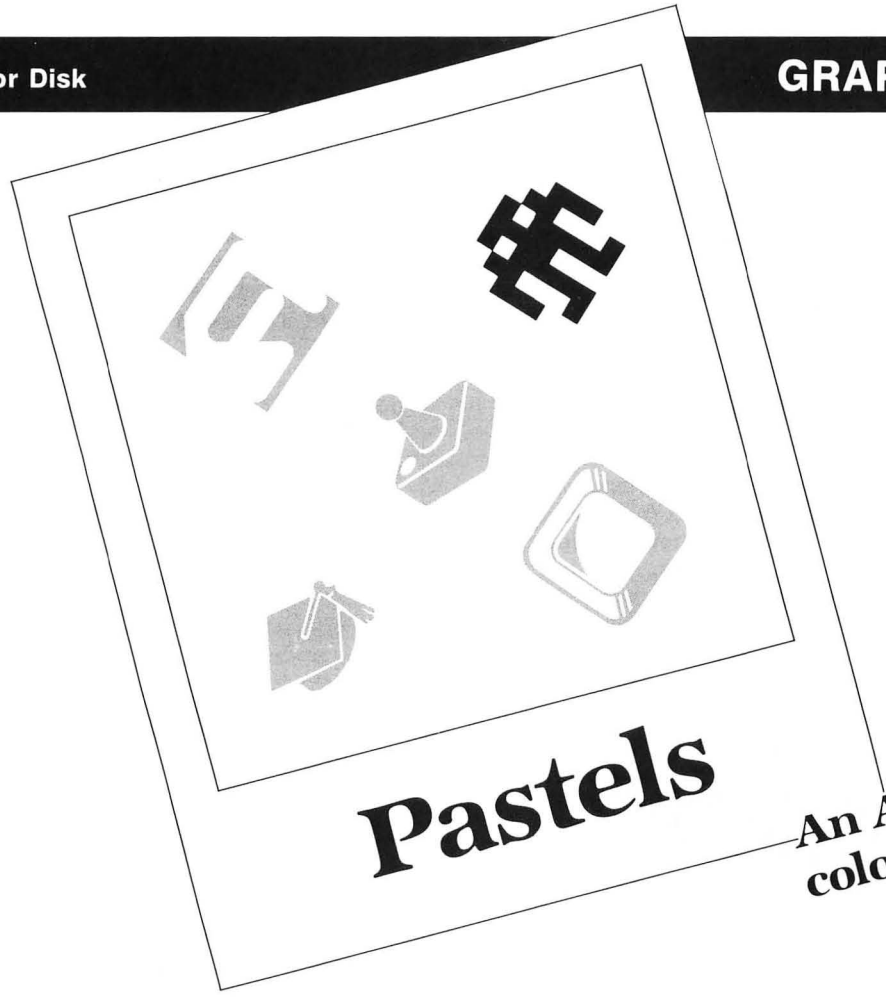
ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your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

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

For those of you who are sincerely interested in the rules and regulations for publication, we've taken this opportunity to print our guidelines for authors. See page 128 of this book for everything you'll need to know.

Send your programs and articles to:
Editor, **ANALOG Computing**
P.O. Box 23, Worcester, MA 01603.

GRAPHICS



Pastels

An Action! color demo.

by David Plotkin

Pastels is fun to look at. It's relaxing, putting fifteen pastel colors on-screen at once, in ever-changing patterns.

As detailed below, several special PROCedures were used to speed up the graphics. Action! has become the language of choice for many serious programmers, being considerably easier than machine language, and outstripping BASIC's performance.

Pastels is written in graphics 11, the 15-color mode available only with the GTIA chip. To understand how the special routines work to speed up the display, you must know something about how colors are displayed in graphics 11. Each byte on the screen is broken up into two halves (or nibbles), with one half containing the lower 4 bits (*0* through *3*), and the other half containing the upper 4 bits (*4* through *7*).

The 4 bits in each nibble can make up a total of sixteen different on/off combinations, thus creating the sixteen colors. Further, since each byte is broken into halves, the first byte on each line corresponds to the screen's x-coordinates *0* and *1*, with the second byte holding *2* and *3*, and so forth.

Byte array *colors* contains sixteen numbers, which correspond to the sixteen bit-patterns available in each nibble, from all bits off (*0*) to all bits on (*255*). Seventeen, for example, is the smallest bit in each nibble (*0* and *4*) on; all others off.

PROC Gr11Init reads the starting address of each screen line into an array of cardinal numbers (CARD), for later reference.

PROC Plot11 actually plots points on the screen much faster than does the system *PLOT*. This is because the first is a specialized routine, which will essentially only work in graphics 11. Three byte arrays are declared, and they're all important.

The first, *tline*, will be equated to the y-element of CARD array *Line*, thus pointing *tline* to the on-screen line we wish to change.

We have *mask* and *mask2* as bitmasks. The first element of *mask* corresponds to all the lower nibble bits being on, and the higher nibble bits off. The second element is just the reverse (all high nibble bits on, all lower nibble bits off).

The bitmask *mask2* just reverses the order of mask's elements. In the equation at the end of this PROCedure, the *tline(x RSH 1)* term determines which byte on the chosen line corresponds to the chosen x-coordinate. Remember: each byte contains two x-coordinates, so it's necessary to divide *x* by *2*, to see which byte to modify.

The *RSH* operation divides by *2* much faster than does the built-in divide routine. The first term—*(= =&mask (x&1))*— takes the byte in question and turns off all bits on the half of the byte to be modified, by ANDing the byte against the mask element.

The element of array *colors* containing the color you

wish is then ANDed against the *mask2* element, to turn off all bits in the color byte in the half of the byte which is *not* being modified.

Finally, these 2 bytes (each with an empty half) are ORed together, to produce the modifed byte.

The balance of the PROCedures don't do anything particularly remarkable, so I won't expand on them. But look over this short demonstration of Action!'s power for yourself.

David Plotkin, with his Master's degree in Chemical Engineering, is a Project Engineer for Chevron U.S.A. He purchased his Atari in 1980 and is interested in programming and game design, as well as word processing.

Listing 1.
Action! listing.

```
;       CHECKSUM DATA
;[4D D7 45 F1 31 EC ]

MODULE; PASTELS by David Plotkin

      ; written in ACTION! from OSS
BYTE ARRAY Colors=[0 17 34 51 68 85
           102 119 136 153 170 187
           204 221 238 255]
CARD ARRAY Line(192)

PROC Gr11Init()
;Initialize Graphics 11
CARD loop,scrn=88
GRAPHICS(11)
FOR loop=0 to 191
DO
  Line(loop)=scrn+40*loop
OD
RETURN

PROC Plot11(BYTE x,y,clr)
;Plot a point using color masks and
;arrays
BYTE ARRAY tline,mask=[15 240],
           mask2=[240 15]
tline=Line(y)
tline(x RSH 1)==&mask(x&1)%
           (Colors(clr)&mask2(x&1))
RETURN


PROC Draw11(BYTE x1,y1,x2,clr)
;Draw a line in Graphics 11
BYTE l1,xx1,xx2
IF x1>x2 then xx2=x1 xx1=x2
   ELSE xx1=x1 xx2=x2
FI
FOR l1=xx1 to xx2
DO
  Plot11(l1,y1,clr)
OD
RETURN

PROC Main()
;The Main driver
BYTE z=[0],i,y=[0],atrct=77
Gr11Init()
DO atrct=0
  FOR i=1 TO 79
  DO z=z+1
     IF z>15 THEN z=1 FI
     Draw11(i,y,79-I,z)
     Draw11(i,190-y,79-i,z)
     y=y+1
     IF y>190 THEN y=0 FI
  OD
  FOR i=1 TO 79
  DO z=z+1
     IF z>15 THEN z=1 FI
     COLOR=z
     PLOT (i,y) DRAWTO (79-i,190-y)
     y=y+1
     IF y>190 THEN y=0 FI
  OD
OD
RETURN
```

CGM

Castell's Graphic Manager: a "GEM" for 8-bit machines.

by David Castell

CGM—Castell's Graphic Manager—is similar to the ST's GEM, in that it acts as an interface between the programmer and the operating system (OS), enabling the programmer to access such features as windows, icons and trackers. One of the many differences is that GEM works with a bit-mapped screen (similar to graphics 8), but **CGM** is designed to work with the standard graphics 0 screen.

Typing it in.

Listing 1 is the BASIC data used to create your copy of **CGM**. See the **M/L Editor** on page 4 for typing instructions. You should create the **CGM** file under the name AUTORUN.SYS.

To load **CGM**, insert the disk containing the AUTORUN.SYS file into drive 1. Turn your computer off and then back on again. After **CGM** loads in, a message indicating it's in memory appears at the top of the screen.

Listings 2, 3, 4 and 5 are examples of BASIC programs using the features of **CGM**. Listing 2 demonstrates the use of windows and overlaying.

Listing 3 is an icon editor. Move the hand tracker within the large rectangle and press the joystick button to turn a blank square white, or vice versa. Two icons, one normal and one reversed, are displayed to the right of the editing square. Press the START key at any time to display data that can be used to create an icon or tracker. When you return to the editor, the editing square will be blank again.

Listing 4 is actually a subroutine that starts at Line 30000. It can be incorporated into any of your own programs that use a graphic 0 screen. This subroutine is actually a mini-DOS that will let you: get a directory, delete files, lock/unlock files, rename files and format disks. This program shows how windows and trackers can be integrated to make menu selection a lot easier.

Listing 5 is an advanced memo pad. The features of **CGM** make using it enjoyable and easy. The icons have been placed in windows, so they can be moved around or removed without disrupting the contents of the screen (memo pad). All these icons were created with the icon editor (Listing 2). The first is a clock. If you select this option, a menu will pop up, giving you the option of setting the clock or displaying it. If you display it, all activity stops so you can see the time. When you're finished with the clock, press the joystick button to go on.

The second icon is a calculator. If you select this, a calculator pops up. It's very simple and performs calculations in the order they're entered, not the order they "should" be in (i.e., multiplication before addition). Just move the hand over the keys and press the joystick button to hit a key.

Most of the keys are self-explanatory. The *X* is the OFF key. When you're finished with the calculator, press OFF to remove it. The *R* is the square root key; the *C* clears the number currently displayed on the calculator's screen; and the *A* is All Clear. This clears the display, operation and memory. The % key is designed to work a special way.

If you enter 5+7%, the answer will be 5.35. It's useful for figuring sales tax. If you enter 7%, the answer will be *0.07*.

The third icon is a disk, for a disk loader routine. When you select this option, a large window is displayed, showing the names of all files on the disk. Move the hand over the one you want to load and press the button. It will be loaded and run automatically. This routine will only load programs that have been saved to disk.

In the fourth window is the word *MEMO* (I honestly couldn't think of an icon to represent this function). If you select this option, a menu will pop up. You may choose to edit, load, save or print the memo. The editing function removes all icons, leaving you the whole screen to edit with normal editing keys.

To exit the editing function, simply press the ESC key. The memo print function is not very advanced. It doesn't support any of Atari's special graphics characters (which is just as well, because most of them are redefined as icons and trackers).

At any time during the operation of the last two programs, you may point the tracker to the top corner of the window you're currently selecting from and press the button. The border of that window will turn to a color. Now, move your tracker to any spot on-screen and press the button again. The window will instantly be moved to this new position, and the border will appear white again.

The MAC/65 source code of **CGM** is available on the disk version of this book.

Windows.

A window is an area of the screen with a border around it, that you can print and input information to and from. **CGM** supports up to five independent windows, which can overlap to maximize space. When you print to a window, it automatically overlaps the other windows.

Creating a window.

When you first create a window, it will appear as a thick white border around a blank area. The contents of the screen underneath the window are stored in memory and restored when the window is removed. To create a window, type in *A = USR(39936,N,X,Y,C,R)* — where: *N* is the number of the window (from 1 to 5); *X* is the column of the top corner of the window; *Y* is the row of the top corner of the window; *C* is the number of columns in the window; and *R* is the number of rows in the window.

Removing a window.

Always be sure to remove a window before you create another with the same number. To remove a window, type *A = USR(39939,N)* — where: *N* is the number of the window you wish to remove.

Moving a window.

At some point, you may want to move a window and its contents to another position on-screen. Instead of removing the window, creating it at another position and reprinting the contents, you can use this special window-moving routine. *A = USR(39942,N,X,Y)* — where: *N* is the number of the window you wish to move; and *X,Y* is the new position of the top corner of the window.

Overlapping.

Creating, removing, or moving a window does not affect the position or contents of other windows, but does affect the order in which they overlap. After executing one of these three commands, the windows will now overlap in the order of their creation, with the first window on the bottom of the stack and the last on the top. The exception is the move function, in which the window moved always appears on top.

Using a window.

After creating the window, you're faced with using it. It's actually very easy: after creating a window, you have a new device *Wn:*, where *n* is the number of the window (1 to 5). As with all other devices, you must use the OPEN command to read or write. *OPEN #aexp,aexp2,0,"Wn:"* — where: *aexp* is IOCB number (1-4) and *aexp2* is a code number to determine input or output operation (4=input, 8=output, and 12=input and output).

The OPEN command sets the window input/output position to the top corner. After input or output you should use the CLOSE command *(CLOSE #aexp)*.

As normal, your input/output commands are:

PRINT #aexp — e.g., *PRINT #1;"OPTION 1"*. This prints *"OPTION 1"* at the current window I/O position.

INPUT #aexp — e.g., *INPUT #1;A$*. This inputs all characters from the current window I/O position to the end of the row.

PUT #aexp — e.g., *PUT #1,65*. This sends character 65 *(A)* to the current window I/O position.

GET #aexp — e.g., *GET #1,A*. This gets the number of the character at the current window I/O position and stores it in the variable A.

Note that PRINT causes the window to be instantly redrawn to show the change in its contents, but this is not the case with the PUT command. With PUT, the window is redrawn when the RETURN character *(155)* is sent to the window.

If you print more rows than are available in a window, the contents will scroll up one line. If you INPUT past the end of the window, you will get an error 136 (End of File). Each line sent to the window should end with a RETURN, because it won't wrap around to the next line without one. For example, if you send a 15-character line to a 5-character wide window, only the first 5 characters are displayed; the rest are ignored. Therefore, if you INPUT that row, only the 5 characters actually displayed will be entered.

If you want to have the contents of a window in a string, so you can send it all with one print statement, you'd run into the problem. There's no way you can put the RETURN character in the middle of the string, without going into complex string manipulation. Here, you can use CTRL-PERIOD instead of RETURN in the string.

```
WIN$="ROW 1♦ROW 2♦ROW 3"
PRINT #1;WIN$
```

CGM keeps track of what row and column within the window the next character will be read from or written to. I've referred to this as the "current window I/O position." Since INPUT reads from the current character to the end of a row, you'll need a way to position this pointer

to the spot you want to read from (or, in the case of PRINT, write to). You're able to do this, and more, through the XIO command. It can be used like the Position X,Y statement in BASIC. The difference is that, in the case of the XIO command, positioning to point 0,0 would be the top corner of the window, not the screen.

There are actually three different XIO functions. All of them change the window I/O pointer, but two perform extra functions.

XIO C,#D,X,Y,"Wn:"—where: *D* is the channel number (1-4).

(1) Position for next I/O to window—where: $C < 100$ and $C \geq 12$; *X*=column of window; *Y*=row of window; and *N*=number of window (1-5).

As an example, *XIO 50,#1,0,0,"W:"* indicates that the next string of characters sent to window 1 (no *n* means 1) will start at the top corner of the window.

(2) Position for next I/O with window and redisplay the contents of window. Normally, the only way to cause an overlapped window to overlap the other windows is to send data to it. Unfortunately, this may cause unwanted scrolling of text in the window. However, this XIO command is similar to the first, except this one will redisplay the contents of window *n*, causing it to overlap the others.

Where: $C \geq 100$ and $C < 200$; *X*=column of window; *Y*=row of window; and *N*=number of window (1-5). So, if window 2 is overlapped by other windows, *XIO 100,#1,0,1,"W2:"* will cause window 2 to overlap other windows. The next I/O with window 2 will start at the second row, first character.

(3) Redisplay contents of a window, position for next I/O with window (see 2, above) and reverse (black print on white square) all of the characters in a desired row—where: $C \geq 200$ and $C < 256$; *X*=column of window (reversing always starts from the beginning of a row, regardless of the value of X); *Y*=row of window; and *N*=number of window (1-5).

XIO 200,#1,1,1,"W2:" — the characters in row 2 of window 2 are reversed. The next I/O with window 2 will start at the second row, second character.

Note that the characters are reversed only on the screen display. Therefore, when the window is redrawn (by PRINT, Create Window, Remove Window, Move Window, XIO 100-199, XIO 200-255), the row is returned to normal. Also, if you INPUT a row that's reversed on-screen, the string input is *not* reversed.

Also, PUT, GET, INPUT, OPEN, CLOSE, XIO 12-99 do not erase the reversed row.

Take the following window as an example:

```
ABC   123
DEF   456
GHI   789
```

(1) *OPEN #1,12,0,"W:"* — Sets up for input/output to window 1. The window I/O position is set to top corner.

(2) *XIO 250,#2,4,1,"W:"* — Reverses the second row, sets window I/O position to column 5, row 2.

(3) *INPUT #1,A$* — Inputs from window I/O position to end of row. A$ now contains *456*.

(4) *XIO 150,#2,0,2,"W:"* — Redraws the window, causing the second row (which was highlighted in the second step) to be returned to normal. The window I/O position is set to the column 1, row 3.

(5) *PRINT #1;"XYZ"* — The "XYZ" prints over top of "GHI".

(6) *PRINT #1* — The top row *ABC 123* scrolls off the top of the window, and the last row of the window is blank.

(7) *CLOSE #1* — You should always close a channel when you're done with it.

Note: XIO commands cannot use a channel already open for I/O. That's why the XIO commands in steps 2 and 4 use channel #2.

Icons.

To **CGM**, an icon is just a picture two characters wide by two characters high. A call to **CGM** will put the data of the four characters of the icon into the RAM character set used by **CGM**, starting at the character you select (ATASCII character value). You should probably choose to put your picture where the Atari special graphics characters normally are (characters 0-31), leaving the letters and numbers alone. However, the tracker uses characters 0-8, so you should also avoid these if you're using the tracker routines.

For example, if you chose character 9, your icon will be made up of characters 9-12. Characters 9 and 10 will be the top half of the icon, and characters 11 and 12 will be the bottom half. Therefore, to print your icon on the screen, print characters 9 and 10 (CTRL-I and J) on one row, and characters 11 and 12 (CTRL-K and L) on the line below.

The data for each character of the icon consists of eight numbers, the binary representations of each row of the character. The data is set up the same way as the icon is drawn—the top two characters, followed by the bottom two characters. If you're familiar with creating character sets, this set-up isn't new. But, even if you don't understand how to set up character data for the character set, you can use the icon editor program (Listing 3) to automatically create data statements (or strings) that can be used with calls requiring icon or tracker data.

To put an icon into the character set, enter *A=USR (39951,N,ADDR)*, where *ADDR* is the address of the icon data. If your icon data was in a string (such as ICON$), *ADDR* would be *ADR(ICON$)*.

N is the number (ATASCII character value) of the first of the four characters whose character data will be replaced by the icon data.

Tracker.

A tracker is actually a movable icon, used mostly as a pointer. You're probably most familiar with the arrow tracker moved by a mouse on both the ST and the Macintosh, or maybe the hand moving around in the "Construction Set" series from Electronic Arts.

The default tracker built into **CGM** is a hand, but you

can change this if you want. The pointer that points to the location of the default tracker can be found at 39962 and 39963. For example, add the following line to the Memo Pad program (Listing 5):

```
85 TRACKHI=INT(ADR(CLOCK$)/256):TRACKL
O=ADR(CLOCK$)-TRACKHI*256:POKE 39962,T
RACKLO:POKE 39963,TRACKHI
```

This causes the clock to be used as the pointer, instead of the hand.

Built-in tracker routine.

The built-in tracker routine works independently of the BASIC program. This routine reads joystick 1 and moves the tracker in the corresponding direction on-screen. To start the tracker, enter *A=USR(39954)*.

After you start the tracker routine, your program really doesn't have to do anything but wait. If you wish to check where the tracker is at any given time, try: *X=PEEK (4)*, then *Y=PEEK (5)*.

If you wish to disable joystick control of the tracker, location 39960 is the tracker mask. *POKE 39960,1* disables joystick control, and *POKE 39960,0* enables control.

If you're finished with the tracker routine and wish to stop it: *A=USR(39957)*. This will stop the routine and remove the tracker. If you wish to put it back, you can just do another *A=USR(39954)*, and it will appear at the same spot it was removed from.

The following program is a sample implementation of the tracker routine:

```
10 GRAPHICS 0:POKE 752,1:? CHR$(125)
20 A=USR(39954)
30 IF STRIG(0) THEN 30
40 A=USR(39957):X=PEEK(4):Y=PEEK(5)
50 COLOR 160:PLOT X,Y
60 GOTO 20
```

This routine will allow you to use the joystick to move the tracker around. Line 30 waits until the joystick button is pressed. When it is, the tracker's removed and a reverse square is placed at the spot where the tracker was pointing. You *must* remove the tracker before altering the screen under it. Otherwise, when the tracker moves again, the screen beneath the tracker will return to the way it was before the tracker was moved over it.

Your own tracker.

As mentioned above, if you wish to use this same tracker routine with your own tracker, just change the pointer at location 39962 and 39963 to point to your tracker.

However, if you wish to create your own tracker routine—or use the tracker for something else (such as a controlled cursor, like the RENAME function of Listing 4, or the SET CLOCK function of Listing 5)—here are a few calls you can use.

To position a tracker: *A=USR(39945,addr,x,y)*, where *addr* is the address of the tracker data (set up in the same manner as icon data); *x* is the horizontal position of the tracker (0 to 318); and *y* is the vertical position of the tracker (0 to 190).

Note that **CGM** has built-in roll-around routines. If the tracker goes off the screen, it appears on the other side.

Also, the tracker automatically removes itself from its old position before locating itself to a new position. Therefore, this is the only call you need to make inside a loop that moves the tracker.

Avoid altering the contents of the screen near the tracker. If characters are printed over the tracker, these will disappear when the tracker is moved.

If you wish to use the built-in hand tracker, just don't include *addr* within the USR call. For example: *A=USR (39942,x,y)* will position the hand at coordinates *x,y*.

You can also use this Move Tracker routine to position the tracker being run by the built-in tracker routine called by *A=USR(39954)* within your program. For example, in the mini-DOS example (Listing 4), when entering the new name of a file to rename, the tracker mask is set (POKE 39960,1) and *A=USR(39942,x,y)* is being used to control the movement of the hand, so it can be used as a cursor. When the name is entered, the tracker mask is cleared (POKE 39960,0), and the joystick once again takes control of the tracker.

If you don't enter any x- or y-coordinates with the *USR(39942)*, the tracker will be printed at the last x,y-coordinates (for example, *A=USR(39942)* will display the icon at the last x,y-coordinates). So, if you need to alter the text below the tracker in your own tracker routine, just remove the tracker, alter the text and use *A=USR(39942)* to return the tracker to where it was before removal.

Removing a tracker.

When you're finished with the tracker, you'll probably want to remove it from the screen. To do so, enter *A=USR(39948)*.

This is not to be confused with *A=USR(39957)* mentioned earlier, which removes the tracker *and* stops the built-in joystick tracker routine.

The only integration between the windows and tracker consists of the tracker "getting out of the way" while the window is updated. However, as mentioned earlier, if you wish to alter the text beneath the tracker, you must first remove the tracker, alter the text and place the tracker back on-screen.

Reading tracker position.

Anytime you use the above tracker routines, locations 4 and 5 will contain the tracker's position on the screen. Location 4 is the column reading (0-39), and location 5 is the row reading (0-23).

Whatever shape your tracker is, if it's to be used as a pointer, its point should be at the top left of the icon, because the built-in roll around and the values in locations 4 and 5 assume that this is the case.

Here's the same application of the tracker routine shown earlier. This time, it doesn't use the built-in tracker routine, but a custom tracker routine for the **Atari Touch Tablet**.

```
10 GRAPHICS 10:POKE 752,1:? CHR$(125)
20 IF PADDLE(0)=228 AND PADDLE(1)=228
THEN 20
30 TX=PADDLE(0)*(320/228):TY=192-PADDL
E(1)*(192/228)
40 A=USR(39945,TX,TY)
50 IF STICK(0)=15 THEN 20
60 A=USR(39948):X=PEEK(4):Y=PEEK(5)
70 COLOR 128:PLOT X,Y
80 GOTO 20
```

To use it with the **Koala Pad**, just delete the *192-* in Line 30.

Special notes.

Location 39961 contains the character (internal character set) with which window borders are drawn. The default is *128*, a solid white square. You can change this by POKE-ing 39961 with any character.

For this very reason, character 123 of the RAM character set used by **CGM** is changed from a "spade" to a solid, colored square, using artifacting. This color will probably appear green or blue, but the color varies between systems. Whichever it is, the reverse of this character (251) will appear as the other color (blue or green).

Since this character is only in the RAM set, you must make sure this is the character set being used. The first tracker or icon routine called automatically selects the RAM set. However, if you want colored borders before you make one of these calls (or if your program doesn't use trackers), you must perform a POKE 756,152 to use the RAM character set.

Another special character is character 255. If you POKE location 39961 with *255*, no border will be placed around the window, causing the contents of the screen around the window (where the border normally is) to be left "as is." Changing location 39961 means that all borders drawn from that point on will appear with the new character. The Window Create, Remove and Move routines will cause the borders of *all* windows to be redrawn with the new border character.

If none of these routines are used, only the windows you PRINT to—or use an XIO (above 100) command on—will be redrawn with the new character. In the mini-DOS and Memo Pad programs, when you press the button with the tracker pointing to the top corner of the window, location 39961 is POKEd with a *123* and an XIO 150 is performed.

This only changes the color of the border of the one window. If the subsequent move window routine was executed before location 39961 was changed back to *128*, all the windows on the screen would be redrawn with a colored border, instead of the usual white one.

It should be noted that this same technique *cannot* be used with the no border (character 255) option. For the no border option to be invoked, it must be placed before a Window Create, Remove or Move command. These three commands cause the contents of the screen behind the borders to be restored and then the new borders are redrawn—or in this case, not drawn.

Whenever a window is created, the contents of the screen behind the window and the contents of the window are stored just below the top of memory pointer (locations 741 and 742). When **CGM** initializes (after it loads, or any time SYSTEM RESET is pressed) these pointers are set to the first free location below **CGM**.

However, whenever a graphics command or a channel is opened to device E: or S:, this pointer is set to just below the display list, which is also the end of **CGM**. This means that—if you use a graphics command or open to E: or S:, then create a window—the contents of the window and the screen behind the window are stored right over **CGM**, causing the computer to lock up.

You'll probably notice that the two sample tracker routines shown earlier use a graphics command. Also, the icon editor opens a channel to S:, so the graphic characters display properly.

These programs get away with this, because they don't use any windows. However, if you ran any of these programs and then ran a program that did use windows, the computer would lock up. To get around this problem, press the SYSTEM RESET key when you run a program with windows. If you wish to use the graphics statement in a program with windows, just reset the top of memory pointer after the graphics statement. For example:

```
10 MEMTOPLO=PEEK(741):MEMTOPHI=PEEK(74
2)
20 GRAPHICS 0 (or OPEN #1,12,0,"E:" or
 OPEN #1,12,0,"S:")
30 POKE 741,MEMTOPLO:POKE 742,MEMTOPHI
```

This will solve any problems you might encounter.

Since there's no real integration between windows and trackers, the task of integrating these two features in a program is yours. However, I've included several all-purpose subroutines that integrate windows and trackers, to perform specific tasks.

The first can be found in the mini-DOS program (Listing 4) at Line 31000, and in the Memo Pad program (Listing 5) at Line 25000. This subroutine is designed for menu selection. Before entering the subroutine, you must create a window, print the different options to the window, start the tracker routine and set the following variables: *N* = number of the window; *X,Y* = top corner of the window; *DX* = Delta X (number of columns); and *DY* = Delta Y (number of rows).

Lines 25030-25050 are the heart of this subroutine. This loop waits for the user to press the joystick button and reverses the window row the tracker is on, or erases this highlight if the tracker is outside the window's border.

Line 25070 checks to see if the button was pressed at the top corner of the window. If it was, Lines 25080-25140, (the Move Window routine) are executed.

When the subroutine is finished, the variables *X* and *Y* will hold the top corner of the window (if the window was moved, your program might need to know), and the variable *CHOICE* will contain the number of the option selected. This result can be used in statements like: *ON CHOICE GOTO OPTION1, OPTION2, OPTION3,* . . . or *ON CHOICE GOSUB OPTION1, OPTION2, OPTION3,* . . .

Another subroutine is the Input String subroutine starting at 30000 in the Memo Pad program. This routine uses the tracker as a cursor to enter a string inside a window. The routine is used to set the clock and enter the filename in the save memo option.

The routine needs IOCB #2 to be opened for input/output *(OPEN #2,12,0,"Wn:")* to the proper window. It also requires that the variables *N, X* and *Y* to be set in the same manner they were in the above subroutine, as well as the variable *LEFT*, which should contain the column of the window the entry should start in.

For example, if the prompt in the window was *File-*

name?, entry should start in column 9, which is the first column to the right of the question mark; therefore, *LEFT* should equal 9.

This subroutine returns the string that was entered in the variable *NAME$*, which should be dimensioned at the start of your program. If you were to enter a number, you would use the VAL command *(NUM=VAL(NAME$))*, to get the number into a numeric variable.

Another subroutine worth mentioning starts at Line 10000 of the Memo Pad program. This routine creates a window the height of the screen, and reads all the names of files on the disk in drive 1, printing them to the window. It then uses the subroutine at Line 25000 to allow the user to select one of the filenames.

It proceeds to get the name into the proper format for disk I/O *(D:FILENAME.EXT)* and returns the final name in FNAME$. This subroutine requires that the strings *FNAME$* and *EXT$* be dimensioned at the beginning of the program.

All these subroutines require certain entry point variables (CL, OP, TRACKER, etc.) See the first couple lines of the Memo Pad program for these variables.

Your first step in learning how to use **CGM** should be to run the four sample programs, so you can see exactly what **CGM** can do. Then look at the listings, to see exactly how we're using **CGM** to do it.

When you start programming with **CGM**, use as many routines from these four samples as possible, as well as creating your own subroutines to incorporate into other programs. You'll never run out of uses for **CGM** in your programs, because it has the ability to make any program user-friendly.

David Castell is currently attending the University of Waterloo in Ontario. Although this is David's first program published in a magazine, he's also written "P.S. Interface" and "The First XLent Word Processor" for the 8-bits and "P.M. Interface" for the ST. All three are available from XLent Software.

The two-letter checksum code preceding the line numbers in Listings 2 through 5 is *not* a part of the BASIC program. For more information, see *BASIC Editor II*, in *ANALOG Computing*'s issue 47.

Listing 1.

```
1000 DATA 255,255,133,142,128,143,104,
216,160,0,104,104,153,109,144,200,9393
1010 DATA 192,5,144,246,32,235,150,32,
73,143,76,251,150,104,216,104,9749
1020 DATA 104,141,109,144,32,235,150,3
2,10,146,76,251,150,216,104,32,7629
1030 DATA 235,150,169,0,141,233,145,32
,162,144,104,104,141,109,144,141,8511
1040 DATA 210,142,32,137,145,104,104,1
53,47,145,104,104,153,48,145,32,5092
1050 DATA 196,144,169,0,141,109,144,32
,137,145,32,212,143,76,251,150,9331
1060 DATA 36,33,54,41,36,0,35,33,51,52
,37,44,44,7,51,0,5526
1070 DATA 167,178,161,176,168,169,163,
128,173,161,174,161,167,165,178,165,36
03
1080 DATA 12,141,23,143,165,13,141,24,
143,160,4,185,219,142,145,88,7666
1090 DATA 200,192,35,144,246,176,3,32,
255,255,169,22,133,12,169,143,8606
1100 DATA 133,13,32,165,150,169,0,160,
4,153,84,148,136,16,250,160,7627
1110 DATA 44,153,55,145,136,16,250,32,
104,148,169,132,141,229,2,169,8619
1120 DATA 142,141,230,2,169,0,141,118,
144,96,173,118,144,201,45,176,8770
1130 DATA 248,32,58,144,32,8,145,32,71
,144,173,229,2,56,229,208,7974
1140 DATA 141,114,144,173,230,2,229,20
9,141,115,144,32,8,145,173,114,8010
1150 DATA 144,56,229,208,141,116,144,1
73,115,144,229,209,141,117,144,173,261
9
1160 DATA 116,144,129,143,124,144,56,2
33,1,141,229,2,173,117,144,233,0
1170 DATA 0,141,230,2,32,34,145,173,19
8,146,240,3,76,226,144,173,9911
1180 DATA 109,144,141,233,145,32,100,1
45,32,162,144,76,196,144,32,58,6085
1190 DATA 144,32,25,144,173,114,144,13
3,210,173,115,144,133,211,174,113,1384
1200 DATA 144,160,0,177,208,145,210,20
0,204,112,144,144,246,32,84,144,830
1210 DATA 32,96,144,202,208,235,76,71,
144,32,58,144,32,25,144,160,5692
1220 DATA 0,173,25,156,201,255,240,48,
145,208,200,204,112,144,144,248,4497
1230 DATA 174,113,144,202,202,32,84,14
4,160,0,173,25,156,145,208,172,9385
1240 DATA 112,144,136,145,208,202,208,
237,32,84,144,160,0,173,25,156,8157
1250 DATA 145,208,200,204,112,144,144,
245,32,71,144,76,192,145,174,111,25
1260 DATA 144,165,88,24,109,110,144,13
3,208,165,89,105,0,133,209,202,9361
1270 DATA 48,25,165,208,24,105,40,133,
208,144,244,230,209,208,240,238,6582
1280 DATA 112,144,238,112,144,238,113,
144,238,113,144,96,206,112,144,206,264
3
1290 DATA 112,144,206,113,144,206,113,
144,96,165,208,24,105,40,133,208,8997
1300 DATA 144,2,230,209,96,165,210,24,
109,112,144,133,210,144,245,230,3488
1310 DATA 211,96,0,0,0,0,0,0,0,0,0,0,3
2,58,144,32,5613
1320 DATA 25,144,125,144,120,145,173,1
14,144,133,210,173,115,144,133,211,207
1
1330 DATA 174,113,144,160,0,177,210,14
5,208,200,204,112,144,144,246,32,2044
1340 DATA 84,144,32,96,144,202,208,235
,76,71,144,160,0,185,55,145,8093
```

```
1350 DATA 205,233,145,240,23,140,1,145
,141,109,144,32,137,145,32,119,6060
1360 DATA 144,172,1,145,32,2,145,204,1
18,144,144,225,96,160,0,185,8484
1370 DATA 55,145,240,23,140,1,145,141,
109,144,32,137,145,32,169,143,6949
1380 DATA 172,1,145,32,2,145,204,118,1
44,144,228,160,0,185,55,145,8268
1390 DATA 240,23,140,1,145,141,109,144
,32,137,145,32,212,143,172,1,6577
1400 DATA 145,32,2,145,204,118,144,144
,228,96,0,152,24,105,9,168,5524
1410 DATA 96,174,113,144,169,0,133,208
,133,209,202,48,243,165,208,24,1267
1420 DATA 109,112,144,133,208,144,243,
230,209,208,239,172,118,144,162,0,2796
1430 DATA 189,109,144,153,55,145,200,2
32,224,9,144,244,140,118,144,96,1068
1440 DATA 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,
0,0,1450
1460 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,16
0,0,173,6468
1470 DATA 116,144,133,208,173,117,144,
133,209,169,32,145,208,230,208,208,477
9
1480 DATA 2,230,121,145,116,146,209,16
5,208,205,114,144,208,239,165,209,5897
1490 DATA 205,115,144,208,232,96,160,0
,204,118,144,176,13,185,55,145,8661
1500 DATA 205,109,144,240,8,32,2,145,2
08,238,160,255,96,162,0,185,269
1510 DATA 55,145,157,109,144,200,232,2
24,9,144,244,96,32,25,144,165,9021
1520 DATA 208,24,105,41,133,208,165,20
9,105,0,133,209,96,32,175,145,8552
1530 DATA 173,116,144,133,210,173,117,
144,133,211,174,113,144,160,0,177,479
1540 DATA 210,32,234,145,145,208,200,2
04,112,144,144,243,32,84,144,32,9313
1550 DATA 96,144,202,208,232,96,0,141,
197,146,41,128,141,8,146,173,8359
1560 DATA 197,146,41,127,201,96,176,12
,201,32,144,5,233,32,76,7,4091
1570 DATA 146,24,105,64,9,0,96,32,137,
145,192,255,240,21,32,119,6961
1580 DATA 144,160,0,185,55,145,205,109
,144,240,9,32,2,145,204,118,7419
1590 DATA 144,144,240,96,173,118,144,5
6,233,9,141,118,144,208,6,32,6695
1600 DATA 57,143,76,191,146,140,198,14
6,204,118,144,240,87,192,0,208,1686
1610 DATA 15,173,118,144,201,9,169,132
,133,210,169,142,133,211,208,15,1102
1620 DATA 185,53,145,56,233,1,133,210,
185,54,145,233,0,133,211,185,935
1630 DATA 62,145,56,233,1,133,208,185,
63,145,233,0,133,209,160,0,8456
1640 DATA 177,208,117,146,112,147,145,
210,198,210,165,210,201,255,208,2,4857
1650 DATA 198,211,198,208,165,208,201,
255,208,2,198,209,205,229,2,208,5023
1660 DATA 225,165,209,205,230,2,208,21
8,172,198,146,204,118,144,176,9,1940
1670 DATA 185,64,145,153,55,145,200,20
8,242,140,198,146,32,57,143,172,858
1680 DATA 118,144,32,160,145,32,80,143
,173,118,144,205,198,146,208,239,3786
1690 DATA 169,0,141,198,146,96,0,0,165
,33,141,109,144,201,6,144,6134
1700 DATA 4,104,104,160,164,96,32,199,
146,32,137,145,192,255,208,3,191
1710 DATA 160,170,96,173,109,144,10,16
8,169,0,153,70,148,153,71,148,7556
1720 DATA 160,1,96,189,65,3,133,33,168
,185,83,148,240,10,169,0,6310
1730 DATA 153,83,148,169,155,160,1,96,
32,140,147,173,83,148,205,113,9094
1740 DATA 144,144,3,160,136,96,160,0,1
77,210,72,238,82,148,173,82,9583
1750 DATA 148,205,112,144,144,15,238,8
3,148,169,0,141,82,148,164,33,7200
1760 DATA 169,1,153,83,148,32,120,147,
104,96,72,189,65,3,133,33,4036
1770 DATA 32,140,147,173,83,148,205,11
3,144,144,3,32,189,147,104,201,9301
1780 DATA 155,240,4,201,96,208,19,32,2
35,150,32,212,143,32,251,150,331
1790 DATA 238,83,148,169,0,141,82,148,
240,15,174,82,148,236,112,144,338
1800 DATA 176,7,113,147,108,148,160,0,
145,210,238,82,148,173,109,144,757
1810 DATA 10,168,173,82,148,153,70,148
,173,83,148,153,71,148,160,1,7597
1820 DATA 96,165,33,141,109,144,10,168
,185,70,148,141,82,148,185,71,8466
1830 DATA 148,141,83,148,32,137,145,17
3,116,144,24,109,82,148,133,210,9031
1840 DATA 173,117,144,105,0,133,211,17
4,83,148,202,48,65,32,96,144,6828
1850 DATA 208,248,173,116,144,133,208,
173,117,144,133,209,160,0,140,82,9851
1860 DATA 148,206,83,148,32,162,147,16
5,208,197,210,208,6,165,209,197,4065
1870 DATA 211,240,17,172,112,144,177,2
08,160,0,145,208,230,208,208,231,5876
1880 DATA 230,209,208,227,169,32,145,2
10,200,204,112,144,144,246,96,32,1860
1890 DATA 199,146,32,137,145,165,42,20
5,112,144,176,3,141,82,148,165,8935
1900 DATA 43,205,113,144,176,52,141,83
,148,165,34,201,100,144,43,32,6352
1910 DATA 235,150,32,212,143,165,34,20
1,200,144,28,32,175,145,174,83,9115
1920 DATA 148,202,48,5,32,84,144,208,2
48,160,0,177,208,73,128,145,9894
1930 DATA 208,200,204,112,144,144,244,
32,251,150,76,120,147,0,0,0,5092
1940 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
212,146,7456
1950 DATA 238,146,241,146,56,147,238,1
46,249,147,76,104,148,160,0,185,702
1960 DATA 26,3,109,148,104,149,240,5,2
00,200,200,208,246,169,87,153,3858
1970 DATA 26,3,169,89,153,27,3,169,148
,153,28,3,96,216,104,170,6923
1980 DATA 41,1,208,14,173,26,156,141,1
34,149,173,27,156,141,145,149,9428
1990 DATA 208,8,104,141,145,149,104,14
1,134,149,224,2,144,13,104,141,7619
2000 DATA 20,151,104,141,19,151,104,10
4,141,18,151,173,18,151,201,192,9380
2010 DATA 144,13,201,240,144,4,169,190
,208,2,169,0,141,18,151,173,8059
2020 DATA 20,151,16,12,169,61,141,19,1
51,169,1,141,20,151,208,21,5370
2030 DATA 201,1,144,17,208,7,173,19,15
1,201,64,144,8,169,0,141,5705
2040 DATA 19,151,141,20,151,173,12,151
,240,3,32,0,149,169,1,141,5065
2050 DATA 23,151,141,12,151,208,23,104
,216,169,0,141,23,151,141,12,5888
2060 DATA 151,173,13,151,133,0,173,14,
151,133,1,76,67,150,169,152,6738
2070 DATA 141,244,2,173,18,151,141,17,
151,173,19,151,133,0,173,20,5270
2080 DATA 151,133,1,162,3,70,1,102,0,7
8,17,151,202,208,246,165,9053
2090 DATA 0,141,122,150,169,40,56,229,
0,141,154,150,174,17,151,142,8588
2100 DATA 126,150,202,48,13,165,0,24,1
05,40,133,0,144,244,230,1,6133
2110 DATA 208,240,165,0,24,101,88,133,
0,141,13,151,165,1,101,89,4162
2120 DATA 133,1,105,149,100,150,141,14
```

```
,151,173,19,151,41,7,141,21,3857
2130 DATA 151,173,18,151,41,7,141,22,1
51,32,133,149,32,206,149,76,6376
2140 DATA 67,150,169,0,141,171,149,24,
105,8,141,177,149,169,0,141,7239
2150 DATA 172,149,105,0,141,178,149,16
0,0,152,153,0,154,200,192,72,9068
2160 DATA 144,248,162,0,172,22,151,189
,255,255,153,0,154,189,255,255,5928
2170 DATA 153,8,154,232,224,24,176,90,
224,8,208,2,162,16,200,152,9115
2180 DATA 41,7,208,227,152,24,105,16,1
68,208,220,172,21,151,136,48,8805
2190 DATA 65,162,0,94,0,154,126,8,154,
126,16,154,232,138,41,7,5170
2200 DATA 208,241,138,24,105,16,170,22
4,72,144,232,176,225,141,15,151,1295
2210 DATA 41,127,133,4,169,0,133,5,160
,3,6,4,38,5,136,208,2252
2220 DATA 249,165,5,24,105,152,133,5,1
73,15,151,48,6,169,0,141,3962
2230 DATA 43,150,96,169,255,208,248,14
2,16,151,140,17,151,32,240,149,725
2240 DATA 160,0,174,66,150,177,4,73,0,
93,0,154,157,0,154,232,6451
2250 DATA 200,192,8,144,240,142,66,150
,174,16,151,172,17,151,96,0,6374
2260 DATA 169,64,141,158,150,160,0,140
,66,150,162,0,177,0,32,130,5179
2270 DATA 150,232,200,192,3,144,245,16
0,40,169,0,141,22,151,177,0,6923
2280 DATA 32,130,101,150,55,151,150,23
2,200,192,43,144,245,160,80,177,2940
2290 DATA 0,32,130,150,232,200,192,83,
144,245,169,0,133,4,169,0,7637
2300 DATA 133,5,96,72,173,23,151,208,7
,104,189,56,151,145,0,96,6126
2310 DATA 104,157,56,151,32,26,150,152
,41,3,201,0,176,4,169,0,3571
2320 DATA 145,0,238,158,150,96,160,0,1
40,12,151,140,24,156,140,18,5862
2330 DATA 151,140,19,151,140,20,151,18
5,0,224,153,0,152,185,0,225,8868
2340 DATA 153,0,153,185,0,226,153,0,15
4,185,0,227,153,0,155,200,9593
2350 DATA 208,229,160,7,169,170,153,21
6,155,136,16,248,169,128,141,25,599
2360 DATA 156,169,24,141,26,156,169,15
1,141,27,156,96,238,24,156,160,9684
2370 DATA 0,140,23,151,173,12,151,240,
242,76,8,149,160,1,140,23,6613
2380 DATA 151,173,12,151,240,3,32,21,1
49,206,24,156,96,0,0,0,1912
2390 DATA 0,0,0,0,0,0,0,0,0,224,112,56
,28,14,15,31,7815
2400 DATA 63,0,0,48,112,96,224,224,224
,63,31,15,0,0,0,0,318
2410 DATA 0,232,216,176,96,192,0,0,0,6
5,151,255,151,104,216,32,8400
2420 DATA 235,150,104,104,32,234,145,3
2,240,149,104,133,1,104,133,0,6372
2430 DATA 160,31,177,0,145,4,136,16,24
9,169,152,141,244,2,32,251,3
2440 DATA 150,76,121,150,104,216,32,23
0,151,162,151,160,118,169,7,76,9366
2450 DATA 92,228,216,173,24,156,208,10
3,238,128,151,169,0,201,2,144,9933
2460 DATA 94,169,0,141,128,151,173,120
,2,201,15,240,82,41,12,201,7282
2470 DATA 4,208,14,173,19,151,24,105,4
,141,19,151,144,3,238,20,4904
2480 DATA 151,173,120,2,41,12,201,8,20
8,14,173,19,151,56,233,4,5542
2490 DATA 141,19,151,176,3,206,20,151,
173,120,2,41,3,201,1,208,5892
2500 DATA 9,173,18,151,24,105,4,141,18
,151,173,120,2,41,3,201,4295
2510 DATA 2,208,9,173,18,151,56,233,4,
141,18,151,32,230,151,76,7472
2520 DATA 98,228,169,0,72,76,132,148,1
04,216,169,7,162,228,160,98,810
2530 DATA 32,92,228,173,12,151,240,3,7
6,0,149,96,0,156,27,156,5352
2540 DATA 76,133,142,76,156,142,76,172
,142,76,132,148,76,254,148,76,398
2550 DATA 65,151,76,104,151,76,236,151
,0,128,24,151,226,2,227,2,7391
2560 DATA 254,142,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,3098
```

●

Listing 2.
BASIC listing.

```
LQ 10 LIST :POSITION 2,15
CA 20 OP=39936:CL=39939
HP 30 A=USR(OP,1,8,10,20,10):GOSUB 200
NB 40 A=USR(OP,2,2,2,10,10):GOSUB 200
UP 50 A=USR(OP,3,5,5,12,12):GOSUB 200
LZ 60 A=USR(OP,4,24,7,7,5):GOSUB 200
JQ 70 A=USR(OP,5,28,0,10,10):GOSUB 200
GT 80 LIST "W:":GOSUB 200
DI 90 LIST "W2:":GOSUB 200
FP 100 LIST "W3:":GOSUB 200
GD 110 LIST "W4:":GOSUB 200
GR 120 LIST "W5:":GOSUB 200
RP 130 GOSUB 210
VP 140 A=USR(CL,3):GOSUB 200
VD 150 A=USR(CL,2):GOSUB 200
WH 160 A=USR(CL,4):GOSUB 200
WX 170 A=USR(CL,5):GOSUB 200
UV 180 A=USR(CL,1):GOSUB 200
OI 190 END
CI 200 FOR N=1 TO 500:NEXT N:RETURN
AP 210 REM 20 RANDOM OVERLAPS
WO 220 DIM N$(3)
PG 230 FOR C=1 TO 20
LN 240 N=INT(RND(0)*5)+1
BS 250 N$="W :":N$(2,2)=CHR$(N+48)
ZL 260 XIO 150,#1,0,0,N$
RM 270 GOSUB 200
EA 280 NEXT C
ZQ 290 RETURN
```

●

Listing 3.
BASIC listing.

```
BD 10 TRACKER=39954:DATTOSET=39951:STPTRA
   CKER=39957:DIM ICON$(32)
MV 20 POKE 752,1:? CHR$(125)
QD 30 FOR N=1 TO 32:ICON$(N,N)=CHR$(0):NE
   XT N
UY 40 GOSUB 130:A=USR(DATTOSET,9,ADR(ICON
   $))
KE 50 A=USR(TRACKER)
XC 60 IF PEEK(53279)=6 THEN 350
ZR 70 IF STRIG(0) THEN 60
UR 80 X=PEEK(4):Y=PEEK(5)
ZY 90 IF X=0 OR X>16 OR Y=0 OR Y>16 THEN
   60
QX 100 GOSUB 200
DF 110 A=USR(DATTOSET,9,ADR(ICON$))
RE 120 GOTO 60
II 130 REM SET UP SCREEN
XB 140 MEM=PEEK(88)+256*PEEK(89)
JG 150 FOR N=MEM TO MEM+17:POKE N,138:POK
   E N+17*40,138:NEXT N
OV 160 FOR N=MEM TO MEM+17*40 STEP 40:POK
```

```
   E N,138:POKE N+17,138:NEXT N
JB 170 POKE MEM+25,73:POKE MEM+26,74:POKE
    MEM+65,75:POKE MEM+66,76
CJ 180 POKE MEM+30,201:POKE MEM+31,202:PO
   KE MEM+70,203:POKE MEM+71,204
ZP 190 RETURN
SM 200 REM PLOT AND UPDATE
ZS 210 A=USR(STPTRACKER)
DV 220 MEM=PEEK(88)+256*PEEK(89)+X+40*Y:Z
   =PEEK(MEM)
SC 230 IF Z=0 THEN POKE MEM,128
EO 240 IF Z=128 THEN POKE MEM,0
CY 250 A=USR(TRACKER)
RK 260 IF X<=8 AND Y<=8 THEN CY=Y:CX=8-X
WI 270 IF X>8 AND Y<=8 THEN CY=Y+8:CX=16-
   X
CN 280 IF X<=8 AND Y>8 THEN CY=Y+8:CX=8-X
GA 290 IF X>8 AND Y>8 THEN CY=Y+16:CX=16-
   X
YQ 300 T=ASC(ICON$(CY,CY))
HJ 310 IF Z=0 THEN T=INT((T+2^CX)+0.5)
SF 320 IF Z=128 THEN T=INT((T-2^CX)+0.5)
YC 330 ICON$(CY,CY)=CHR$(T)
ZH 340 RETURN
DC 350 REM SHOW DATA
EE 360 A=USR(STPTRACKER):OPEN #1,12,0,"S:
   ":POKE 752,1
ZL 370 PRINT #1;"A=USR(39945,ADR(";CHR$(3
   4);ICON$(1,32);CHR$(34);"),X,Y)":?
CQ 380 PRINT "DATA ";:FOR N=1 TO 32:? ASC
   (ICON$(N,N));",";:NEXT N:? CHR$(126)
LR 390 CLOSE #1
IP 400 OPEN #1,4,0,"K:":GET #1,X:CLOSE #1
ET 410 GRAPHICS 0:GOTO 20
```

●

Listing 4.
BASIC listing.

```
LT 10 OP=39936:CL=39939:MOVETRACKER=39945
   :TRACKER=39954:STPTRACKER=39957:TRMASK
   =39960
AZ 20 BORDERCHAR=39961:MOVEWINDOW=39942
PB 30 DIM WIN$(2)
MW 40 GOSUB 30000
AD 50 STOP
ED 30000 X1=0:Y1=0:DX1=11:DY1=6:A=USR(OP,
   1,X1,Y1,DX1,DY1)
XU 30010 OPEN #5,12,0,"W:"
DF 30020 PRINT #5;"DIRECTORY"
WE 30030 PRINT #5;"DELETE"
SS 30040 PRINT #5;"LOCK/UNLOCK"
YJ 30050 PRINT #5;"RENAME"
MT 30060 PRINT #5;"FORMAT"
XA 30070 PRINT #5;"** EXIT **"
BD 30080 MAIN=30110:DIRECTORY=30490:NAME=
   30590:SURE=30750:DIM NAME$(18),REN$(31
   ),T$(10),X$(3)
LF 30090 DIR=30160:LOCK=30220:DEL=30180:F
   RMAT=30440:REN=30260
TG 30100 A=USR(TRACKER)
SB 30110 IF  NOT STRIG(0) THEN 30110
PU 30120 N=1:X=X1:Y=Y1:DX=DX1:DY=DY1:GOSU
   B 31000:X1=X:Y1=Y:INDEX=CHOICE
GC 30130 IF INDEX=6 THEN A=USR(CL,1):A=US
   R(STPTRACKER):RETURN :REM EXIT FROM MI
   NI-DOS
TI 30140 IF INDEX<>5 THEN GOSUB DIRECTORY
   :GOSUB NAME:CLOSE #3
CH 30150 ON INDEX GOTO DIR,DEL,LOCK,REN,F
   RMAT
NZ 30160 REM DIR
TE 30170 GOTO MAIN
JL 30180 REM DEL
XH 30190 GOSUB SURE
DM 30200 XIO 33,#4,0,0,NAME$
SJ 30210 GOTO MAIN
AN 30220 REM LOCK
ZW 30230 IF LCK THEN XIO 36,#4,0,0,NAME$
OL 30240 IF  NOT LCK THEN XIO 35,#4,0,0,N
   AME$
SZ 30250 GOTO MAIN
QE 30260 REM REN
HZ 30265 FOR X=LEN(NAME$) TO 1 STEP -1:IF
    NAME$(X,X)=" " OR NAME$(X,X)=CHR$(155
   ) THEN NEXT X
RO 30270 REN$=NAME$(1,X):A=USR(OP,2,12,0,
   22,1):L=LEN(REN$):POKE TRMASK,1
MB 30280 OPEN #3,12,0,"W2:":OPEN #4,4,0,"
   K:":X=9
OL 30290 PRINT #3;"NEW NAME?"
CY 30300 XIO 100,#2,X,0,"W2:":A=USR(MOVET
   RACKER,(X+13)*8,14)
MG 30310 GET #4,N
UB 30320 IF N<>126 THEN 30360
XD 30330 X=X-1:IF X<9 THEN X=9
JJ 30340 XIO 50,#2,X,0,"W2:":PUT #3,32
YZ 30350 GOTO 30300
BE 30360 PUT #3,N:X=X+1:IF N<>155 THEN 30
   300
HZ 30370 XIO 50,#2,9,0,"W2:"
PK 30380 INPUT #3,NAME$
RR 30390 REN$(L+1)=",":REN$(L+2)=NAME$
YB 30400 POKE TRMASK,0:CLOSE #3:CLOSE #4
XV 30410 XIO 32,#4,0,0,REN$
GD 30420 A=USR(CL,2)
SX 30430 GOTO MAIN
SH 30440 REM FORMAT
BN 30450 IF  NOT STRIG(0) THEN 30450
XE 30460 GOSUB SURE
NV 30470 XIO 254,#4,0,0,"D:"
TR 30480 GOTO MAIN
DT 30490 REM DIRECTORY
KW 30500 A=USR(OP,2,12,0,17,22)
IW 30510 OPEN #4,6,0,"D:*.*"
RJ 30520 OPEN #3,12,0,"W2:"
VD 30530 CNT=0
IP 30540 TRAP 30550:INPUT #4,NAME$:PRINT
   #3;NAME$:CNT=CNT+1:GOTO 30540
SS 30550 CLOSE #4
BA 30560 XIO 20,#4,0,21,"W2:"
HW 30570 PRINT #3;"CANCEL COMMAND "
EU 30580 RETURN
YC 30590 REM NAME
YE 30600 IF INDEX<>1 THEN 30630
CO 30610 IF STRIG(0) THEN 30610
UX 30620 A=USR(CL,2):CLOSE #3:RETURN
IT 30630 N=2:X=12:Y=0:DX=17:DY=22
MO 30640 GOSUB 31000
RT 30650 IF (TY-Y)>=CNT THEN A=USR(CL,2):
   POP :CLOSE #3:GOTO MAIN
WH 30660 XIO 15,#4,0,(TY-Y)-1,"W2:"
PP 30670 INPUT #3,NAME$
HN 30680 LCK=0:IF NAME$(1,1)="*" THEN LCK
   =1
SR 30690 T$="D:":T$(3)=NAME$(3,10):X$=NAM
   E$(11,13)
NQ 30700 FOR N=1 TO 10:IF T$(N,N)=" " THE
   N L=N:N=12
GY 30710 NEXT N:IF N=11 THEN L=11
DH 30720 NAME$=T$:NAME$(L,L)=".":NAME$(L+
   1)=X$:NAME$(L+4)=CHR$(155)
SQ 30730 CLOSE #4
CG 30740 A=USR(CL,2):RETURN
LO 30750 REM SURE
NZ 30760 A=USR(OP,2,12,7,3,2)
CU 30770 OPEN #4,8,0,"W2:"
OV 30780 PRINT #4;"YES"
ET 30790 PRINT #4;"NO"
SH 30800 CLOSE #4
YH 30810 N=2:X=12:Y=7:DX=3:DY=2
MM 30820 GOSUB 31000
GT 30830 A=USR(CL,2)
YG 30840 IF CHOICE=1 THEN RETURN
ST 30850 POP :GOTO MAIN
```

```
GF 31000 REM *** ALL-PURPOSE WINDOW/TRACK
   ER INTEGRATER (INCLUDES WINDOW MOVE OP
   TION) ***
MA 31010 WIN$="W":WIN$(2,2)=STR$(N):OLDY=
   -1
XV 31020 TX=PEEK(4):TY=PEEK(5)
LX 31030 IF TY<>OLDY AND TY>Y AND TY<(Y+D
   Y+1) AND TX>X AND TX<(X+DX+1) THEN XIO
    250,#1,0,TY-(Y+1),WIN$:OLDY=TY
IP 31040 IF OLDY<>-1 AND (TY<=Y OR TY>(Y+
   DY) OR TX<=X OR TX>(X+DX)) THEN XIO 15
   0,#1,0,0,WIN$:OLDY=-1
YO 31050 IF STRIG(0) THEN 31020
YX 31060 IF  NOT STRIG(0) THEN 31060
TW 31070 IF TX<>X OR TY<>Y THEN 31150
IN 31080 POKE BORDERCHAR,123:XIO 150,#1,0
   ,0,WIN$
GL 31090 IF STRIG(0) THEN 31090
RX 31100 IF  NOT STRIG(0) THEN 31100
MR 31110 POKE BORDERCHAR,128
EL 31120 X=PEEK(4):Y=PEEK(5)
YK 31130 A=USR(MOVEWINDOW,N,X,Y)
YD 31140 GOTO 31010
DL 31150 IF TY<=Y OR TY>(Y+DY) OR TX<=X O
   R TX>(X+DX) THEN 31010
PC 31160 CHOICE=TY-Y
EG 31170 RETURN
```

●

Listing 5.
BASIC listing.

```
OB 10 POKE 82,0:? CHR$(125)
MG 20 OP=39936:CL=39939:COPY=39951:BCHAR=
   39961:MOVEWINDOW=39942:STPTRACK=39957:
   TRACKER=39954:MOVETRACKER=39945
UK 30 ERASETRACK=39948
MF 40 DIM CLOCK$(32),CALC$(32),DISK$(32),
   NUM$(8),FUNC$(4):FUNC$="ZX-+"
OO 50 DIM WIN$(2),WIN1$(20),X(4),Y(4),DX(
   4),DY(4),H$(2),M$(2),S$(2),NAME$(18),E
   XT$(4),FNAME$(14):IOCB=848
SP 60 GOSUB 1590
YW 70 WIN1$="▛▜◆▙":WIN1$(6)="▟▄◆▜":WIN1
   $(11)="▛▜◆▙":WIN1$(16)="MEMO  "
SU 80 A=USR(COPY,ASC("▗"),ADR(CLOCK$)):A=
   USR(COPY,ASC("▔"),ADR(CALC$)):A=USR(CO
   PY,ASC("▝"),ADR(DISK$))
GC 90 FOR N=1 TO 4
WB 100 GOSUB 30150
XM 110 A=USR(OP,N,X(N),Y(N),DX(N),DY(N)):
   OPEN #1,8,0,WIN$:T=N*5-4:PRINT #1;WIN1
   $(T,T+4):CLOSE #1
HT 120 NEXT N
CT 130 A=USR(TRACKER)
QZ 140 IF STRIG(0) THEN 140
XH 150 IF  NOT STRIG(0) THEN 150
WE 160 X=PEEK(4):Y=PEEK(5)
XT 170 FOR N=1 TO 4:IF X<>X(N) OR Y<>Y(N)
    THEN 220
LK 180 GOSUB 30150:POKE BCHAR,123:XIO 150
   ,#1,0,0,WIN$
VU 190 IF STRIG(0) THEN 190
SP 200 IF  NOT STRIG(0) THEN 200
SU 210 POKE BCHAR,128:XIO 150,#1,0,0,WIN$
   :X(N)=PEEK(4):Y(N)=PEEK(5):A=USR(MOVEW
   INDOW,N,X(N),Y(N)):POP :GOTO 140
HU 220 NEXT N
OX 230 FOR N=1 TO 4
NF 240 IF Y>Y(N) AND Y<=Y(N)+DY(N) AND X>
   X(N) AND X<=X(N)+DX(N) THEN POP :GOTO
   270
IA 250 NEXT N
NL 260 GOTO 140
FW 270 ON N GOTO 600,880,1560,280
BQ 280 REM MEMO PAD
YP 290 N=5:X=X(4):Y=Y(4):DX=10:DY=4:A=USR
   (OP,N,X,Y,DX,DY)
QT 300 OPEN #1,8,0,"W5:":PRINT #1;"EDIT M
   EMO":PRINT #1;"LOAD MEMO":PRINT #1;"SA
   VE MEMO":PRINT #1;"PRINT MEMO"
OG 310 CLOSE #1:GOSUB 25000
IZ 320 FOR N=1 TO 5:A=USR(CL,N):NEXT N
XL 330 ON CHOICE GOTO 340,380,430,490
PK 340 OPEN #1,4,0,"K:"
KU 350 GET #1,X:IF X=27 THEN 370
CD 360 ? CHR$(X);:GOTO 350
QK 370 CLOSE #1:GOTO 90
AX 380 REM LOAD MEMO
JB 390 A=USR(TRACKER):GOSUB 10000:IF CHOI
   CE>CNT THEN 90
GV 400 TRAP 420:OPEN #1,4,0,FNAME$
WI 410 POKE IOCB+2,7:GOSUB 590
WI 420 TRAP 40000:CLOSE #1:GOTO 90
GY 430 REM SAVE MEMO
GA 440 N=5:X=8:Y=10:A=USR(OP,N,X,Y,24,1):
   OPEN #2,12,0,"W5:":PRINT #2;"Filename?
   ":LEFT=9
AD 450 GOSUB 30000:A=USR(CL,5):CLOSE #2
IS 460 TRAP 480:OPEN #1,8,0,NAME$
ZC 470 POKE IOCB+2,11:GOSUB 590
WU 480 TRAP 40000:CLOSE #1:GOTO 90
OK 490 REM PRINT MEMO
CU 500 DD=PEEK(89)*256+PEEK(88)
JH 510 TRAP 580:OPEN #1,8,0,"P:"
VS 520 FOR N=1 TO 24
BX 530 FOR X=DD TO DD+39:CHAR=PEEK(X):IF
   CHAR>127 THEN CHAR=CHAR-128
EB 540 IF CHAR>=96 THEN 570
LT 550 IF CHAR>=64 THEN CHAR=CHAR-64:GOTO
    570
IP 560 CHAR=CHAR+32
SQ 570 PUT #1,CHAR:NEXT X:DD=DD+40:PUT #1
   ,155:NEXT N
WV 580 TRAP 40000:CLOSE #1:GOTO 90
EC 590 POKE IOCB+4,PEEK(88):POKE IOCB+5,P
   EEK(89):POKE IOCB+9,3:POKE IOCB+8,192:
   A=USR(ADR("hhh▒LV▒"),16):RETURN
YU 600 REM CLOCK
BV 610 N=5:X=X(1):Y=Y(1):DX=13:DY=2
HY 620 A=USR(OP,N,X,Y,DX,DY)
DI 630 OPEN #2,12,0,"W5:"
DW 640 PRINT #2;"SET CLOCK◆DISPLAY CLOCK"
KC 650 GOSUB 25000:A=USR(CL,N)
MU 660 IF CHOICE=2 THEN 760
FT 670 A=USR(OP,N,X,Y,12,1):LEFT=6
WL 680 PRINT #2;"HOURS?":GOSUB 30000:HRS=
   VAL(NAME$):LEFT=8
MY 690 XIO 50,#3,0,0,"W5:"
TJ 700 PRINT #2;"MINUTES?":GOSUB 30000:MI
   N=VAL(NAME$)
MJ 710 XIO 50,#3,0,0,"W5:"
QF 720 PRINT #2;"SECONDS?":GOSUB 30000:SE
   C=VAL(NAME$)
EL 730 POKE 18,0:POKE 19,0:POKE 20,0
MZ 740 A=USR(CL,5)
AC 750 CLOSE #2:GOTO 130
PA 760 REM DISPLAY
FI 770 A=USR(OP,5,X(1),Y(1),8,1)
QL 780 OPEN #1,8,0,"W5:"
LW 790 T=(PEEK(18)*65536+PEEK(19)*256+PEE
   K(20))/60+HRS*3600+MIN*60+SEC
YO 800 H=INT(T/3600):M=INT((T-H*3600)/60)
   :S=INT(T-(H*3600+M*60))
JY 810 H$=STR$(H):IF H<10 THEN H$="0":H$(
   2)=STR$(H)
MR 820 M$=STR$(M):IF M<10 THEN M$="0":M$(
   2)=STR$(M)
VD 830 S$=STR$(S):IF S<10 THEN S$="0":S$(
   2)=STR$(S)
MQ 840 XIO 50,#3,0,0,"W5:":PRINT #1;H$;":
   ";M$;":";S$
AV 850 IF STRIG(0) THEN 790
FZ 860 IF  NOT STRIG(0) THEN 860
XD 870 CLOSE #1:CLOSE #2:A=USR(CL,5):GOTO
```

```
   130
DN 880 REM CALCULATOR
CR 890 X=X(2):Y=Y(2):SUM=0:FUNC=0
XU 900 A=USR(OP,5,X,Y,10,11)
DH 910 OPEN #2,12,0,"W5:"
JT 920 GOSUB 10150:PRINT #2:PRINT #2;"
   "
ND 930 PRINT #2;"X MR M- M+"
JH 940 PRINT #2:PRINT #2;"% 7 8 9 / "
JB 950 PRINT #2:PRINT #2;"R 4 5 6 * "
PH 960 PRINT #2:PRINT #2;"C 1 2 3 - "
RH 970 PRINT #2:PRINT #2;"A 0 . = + "
CF 980 CNT=1
CW 990 IF STRIG(0) THEN 990
LP 1000 IF  NOT STRIG(0) THEN 1000
MA 1010 TX=PEEK(4):TY=PEEK(5)
GN 1020 IF TX<>X OR TY<>Y THEN 1070
PA 1030 POKE BCHAR,123:XIO 150,#1,0,0,"W5
   :"
QO 1040 IF STRIG(0) THEN 1040
RY 1050 IF  NOT STRIG(0) THEN 1050
PN 1060 POKE BCHAR,128:X=PEEK(4):Y=PEEK(5
   ):A=USR(MOVEWINDOW,5,X,Y):GOTO 990
WJ 1070 IF TX<=X OR TX>X+10 OR TY<Y+3 OR
   TY>Y+11 THEN 990
EM 1080 CX=TX-X-1:CY=TY-Y-3
OJ 1090 IF (CY AND INT(CX/2)<>CX/2) OR IN
   T(CY/2)<>CY/2 THEN 990
KN 1100 IF CNT=2 AND NUM$="." THEN NUM$(2
   )="0"
DL 1110 IF CY=0 THEN 1220
IX 1120 NX=(CX-2)/2:NY=(CY-2)/2
VH 1130 IF NX=0 AND NY=3 THEN NUM=0:GOTO
   1180
HR 1140 IF NX=1 AND NY=3 THEN NUM$(CNT,CN
   T)=".":GOTO 1190
BV 1150 IF NX<0 OR NX>2 OR NY<0 OR NY>2 T
   HEN 1300
LP 1160 NUM=7-3*NY+NX
HM 1170 IF CNT=9 THEN 990
CQ 1180 NUM$(CNT,CNT)=STR$(NUM)
AY 1190 IF CNT=1 THEN GOSUB 10150:NUM$(2)
   =""
EX 1200 XIO 50,#1,1,0,"W5:":PRINT #2;NUM$
JV 1210 CNT=CNT+1:GOTO 990
OR 1220 REM MEMORY
NX 1230 IF CX=0 THEN A=USR(CL,5):CLOSE #2
   :GOTO 140
AO 1240 IF CX=2 OR CX=3 THEN GOSUB 10150:
   PRINT #2;MEM:NUM$=STR$(MEM):GOTO 980
BY 1250 IF NUM$="" THEN 990
CK 1260 IF CX=5 OR CX=6 THEN MEM=MEM-VAL(
   NUM$):GOTO 1290
DP 1270 IF CX=8 OR CX=9 THEN MEM=MEM+VAL(
   NUM$):GOTO 1290
VI 1280 GOTO 990
NU 1290 FUNC=0:XIO 50,#1,0,0,"W5:":PRINT
   #2;"X":GOSUB 10150:GOTO 980
PU 1300 C=(CY-2)/2+1
DD 1310 IF CX=0 AND C>2 THEN ON C-2 GOTO
   1490,1510
BR 1320 IF NUM$="" THEN 990
WX 1330 IF FUNC=0 THEN SUM=VAL(NUM$)
YK 1340 IF CX=0 THEN ON C GOTO 1430,1460
DR 1350 IF FUNC THEN ON FUNC GOSUB 1390,1
   400,1410,1420:GOSUB 10150:PRINT #2;SUM
DL 1360 IF CY=8 AND CX=6 THEN FUNC=0:NUM$
   =STR$(SUM):GOTO 980
HT 1370 FUNC=C
SL 1380 GOSUB 10150:PRINT #2;SUM:GOSUB 15
   30:GOTO 980
GM 1390 SUM=SUM/VAL(NUM$):RETURN
DA 1400 SUM=SUM*VAL(NUM$):RETURN
EQ 1410 SUM=SUM-VAL(NUM$):RETURN
DT 1420 SUM=SUM+VAL(NUM$):RETURN
YT 1430 REM PERCENT
VW 1440 IF FUNC=0 THEN SUM=VAL(NUM$)/100:
   GOTO 1480
HZ 1450 NUM=SUM*(VAL(NUM$)/100):CY=8:CX=6
   :NUM$=STR$(NUM):GOTO 1350
YE 1460 REM SQUARE ROOT
SU 1470 SUM=SQR(VAL(NUM$))
ZS 1480 FUNC=0:NUM$=STR$(SUM):GOTO 1380
ES 1490 REM CLEAR
YG 1500 GOSUB 10150:GOSUB 1530:NUM$="":GO
   TO 980
LI 1510 REM ALL CLEAR
ZS 1520 GOSUB 10150:XIO 50,#1,0,0,"W5:":P
   RINT #2;" ":FUNC=0:SUM=0:MEM=0:NUM$=""
   :GOTO 980
IG 1530 REM DISPLAY OPERATION
TQ 1540 IF FUNC THEN XIO 50,#1,9,0,"W5:":
   PRINT #2;FUNC$(FUNC,FUNC)
RV 1550 XIO 50,#1,1,0,"W5:":RETURN
XQ 1560 REM DISK LOADER
UR 1570 FOR N=1 TO 4:A=USR(CL,N):NEXT N:G
   OSUB 10000:IF CHOICE>CNT THEN 90
DF 1580 RUN FNAME$
NC 1590 REM GETDATA
CW 1600 CLOCK$="▗▟▝▚▟cc[illegible]0"▝▝/[illegible]♦♦▟▗▟0♥♥"
   "0▖0▟♥♥"
UM 1610 CALC$="?p♦▝♦1♦1▝p8▟▝▚▟▝▚▟▝▚♦1♦1♦1♦?▟▝▚
   ▟▝▚▟▝▚▟▝▚▟▝▚"
CP 1620 DISK$="♥♥▝▝▝▝▝▝|♥♥[illegible]|<<▝▝<<<<♥[illegible]
   [illegible]|♥"
OA 1630 FOR N=1 TO 4:READ X,Y,DX,DY:X(N)=
   X:Y(N)=Y:DX(N)=DX:DY(N)=DY:NEXT N
AX 1640 RETURN
ZP 1650 REM WINDOW DATA
PL 1660 DATA 0,0,2,2,10,0,2,2,20,0,2,2,30
   ,0,4,1
WM 10000 REM DISK SELECTION
DU 10010 X=13:Y=0:DX=11:DY=22:N=5
EZ 10020 A=USR(OP,N,X,Y,DX,DY)
HA 10030 OPEN #1,6,0,"D:*.*"
US 10040 OPEN #2,12,0,"W5:":CNT=0
VO 10050 INPUT #1,NAME$:IF NAME$(2,2)=" "
   THEN PRINT #2;NAME$(3,13):CNT=CNT+1:G
   OTO 10050
QP 10060 CLOSE #1
NX 10070 GOSUB 25000
KN 10080 XIO 50,#1,0,CHOICE-1,"W5:":INPUT
   #2,NAME$:CLOSE #2
AV 10090 EXT$=".":EXT$(2)=NAME$(9)
OT 10100 FOR N=1 TO 8
DE 10110 IF NAME$(N,N)=" " THEN 10130
IC 10120 NEXT N
TX 10130 NAME$(N)=EXT$:FNAME$="D:":FNAME$
   (3)=NAME$
DJ 10140 CLOSE #2:A=USR(CL,5):RETURN
CN 10150 REM CLEAR DISPLAY
BG 10160 XIO 50,#1,1,0,"W5:":PRINT #2;"
   "
JN 10170 XIO 50,#1,1,0,"W5:":RETURN
GM 25000 REM *** ALL-PURPOSE WINDOW/TRACK
   ER INTEGRATER (INCLUDES WINDOW MOVE OP
   TION) ***
CB 25010 GOSUB 30150:OLDY=-1
YC 25020 TX=PEEK(4):TY=PEEK(5)
ME 25030 IF TY<>OLDY AND TY>Y AND TY<(Y+D
   Y+1) AND TX>X AND TX<(X+DX+1) THEN XIO
   250,#1,0,TY-(Y+1),WIN$:OLDY=TY
IW 25040 IF OLDY<>-1 AND (TY<=Y OR TY>(Y+
   DY) OR TX<=X OR TX>(X+DX)) THEN XIO 15
   0,#1,0,0,WIN$:OLDY=-1
BT 25050 IF STRIG(0) THEN 25020
CR 25060 IF  NOT STRIG(0) THEN 25060
XT 25070 IF TX<>X OR TY<>Y THEN 25150
EB 25080 POKE BCHAR,123:XIO 150,#1,0,0,WI
   N$
JQ 25090 IF STRIG(0) THEN 25090
VR 25100 IF  NOT STRIG(0) THEN 25100
MS 25110 POKE BCHAR,128
ES 25120 X=PEEK(4):Y=PEEK(5)
YR 25130 A=USR(MOVEWINDOW,N,X,Y)
ZY 25140 GOTO 25010
```

```
KI 25150 IF TY<=Y OR TY>(Y+DY) OR TX<=X O
   R TX>(X+DX) THEN 25010
MI 25160 A=USR(STPTRACK)
PN 25170 CHOICE=TY-Y
ER 25180 RETURN
TI 30000 REM INPUT STRING (NEEDS: LEFT,N,
   OPENED IOCB#2,X,Y)
PT 30010 GOSUB 30170
XM 30020 C=LEFT:OPEN #1,4,0,"K:"
LP 30030 XIO 100,#3,C,0,WIN$:A=USR(MOVETR
   ACKER,((C+X+1)*8),(Y*8+14))
KF 30040 GET #1,CHAR
MR 30050 IF CHAR<>126 THEN 30090
DD 30060 C=C-1:IF C<LEFT THEN C=LEFT
BY 30070 XIO 50,#3,C,0,WIN$:PUT #2,32
ZF 30080 GOTO 30030
JK 30090 IF C=4 THEN 30030
ZT 30100 PUT #2,CHAR:C=C+1:IF CHAR<>155 T
   HEN 30030
ZK 30110 XIO 50,#3,LEFT,0,WIN$
HW 30120 INPUT #2,NAME$:PRINT #2
FN 30130 CLOSE #1:A=USR(ERASETRACK)
DS 30140 RETURN
BZ 30150 REM FIND NAME
IT 30160 WIN$="W":WIN$(2,2)=STR$(N)
EE 30170 RETURN
```

•

TUTORIALS

Display List Mod

Intermixing graphics modes on your screen.

by Mark Andrews

Your Atari computer has a large selection of text and graphics modes, and it isn't difficult to switch from one mode to another in the middle of a program. But using more than one graphics mode on the same screen at the same time—well, that's a little harder. To mix graphics modes on a screen display, it's necessary to understand a programming technique called display-list modification. And that's our topic.

In a type-and-run program listed at the end of this article, I'll demonstrate how to create a screen display that includes three different modes: graphics 0, graphics 1 and graphics 2. There'll be one line of text in each mode, and each line will be displayed in a different color. The result will be a good-looking title screen that you can use with any homemade BASIC or assembly language program. Once you understand the principles used to design the display, you can create many kinds of mixed-mode screens.

The program used for this demonstration was written in assembly language on a MAC/65 assembler-editor package from OSS. It's a type-and-run program I've named HELLO. If you own a MAC/65 assembler, you can type, assemble and run the program as written. If you own another assembler, you may have to make some modifications in the program. And, if all of this talk about assemblers and assembly language is a complete mystery to you, you can learn assembly language by reading my book, *Atari Roots: A Guide to Atari Assembly Language*, published by Datamost in 1983.

Your Atari's graphics modes.

Before I list the HELLO program and explain how it works, here's some background information on how your Atari generates its screen display.

When you turn on your Atari, it automatically goes into a screen mode called graphics 0—a standard 40-column text mode. But if you type the statement *GRAPHICS 1*, or include it in a BASIC program, your computer will switch to a special text mode that displays "fat" characters—characters twice as wide as normal text. The command *GRAPHICS 2* will give you giant characters, twice as high *and* twice as wide as ordinary characters. And there are several other graphics instructions you can use to create high-resolution graphics displays.

That's an extraordinarily powerful set of graphics modes. And, if you know how to program in assembly language, you can make it even more powerful. With assembly, you can mix your Atari's graphics modes in any combination you like. You can print normal characters, fat characters, giant characters and even high-resolution graphics—all on the same screen. Then, you can add fine-scrolling to any part of the screen you want, for an even more eye-catching display!

Along with their many graphics modes, Atari computers also have some other graphics-generating capabilities that

are quite sophisticated. In computers less advanced than your Atari, one block of RAM is usually dedicated to screen memory. Within that block of memory—often known as "screen memory"—there's usually one memory location for each text character on the screen. When a certain text character is to be printed in a particular screen location, a code number representing that character is placed in the memory register that corresponds to its screen location. A character which equates to whatever code number was used then appears in the desired location on-screen.

Atari graphics are a bit more sophisticated than that—and just a bit more complicated, too. Your Atari uses two special chips to generate its graphics display: one called an ANTIC chip and one called a CTIA/GTIA chip. (The early Ataris were built with a CTIA chip; newer models use a GTIA.)

The CTIA/GTIA is a nonprogrammable chip that controls colors and performs various other functions. But your computer's other graphics chip, the ANTIC, is a real microprocessor. It has its own miniature instruction set, and its operations can be controlled with a special kind of program called a "display list." So, to create graphics using the ANTIC chip, you have to know how to use the ANTIC chip's instruction set and how to write display-list programs for the ANTIC microprocessor. And, to understand how ANTIC works, it's necessary to know some fundamental facts about the operation of a video display.

Scan lines and mode lines.

The picture on a TV screen is made up of tiny horizontal lines—262, to be exact. Each of these is called a "scan line."

These scan lines are produced by an electron gun behind your TV monitor's picture tube. This electronic pistol fires electrons at the TV picture tube in what's known as a "raster scan" pattern—a zigzag pattern that begins at the upper left-hand corner of the screen and ends in the bottom right-hand corner.

There are 262 horizontal scan lines on a video tube, and the whole 262-line display is replaced by a completely new display sixty times each second. Between each of these lightning-fast scenery changes, there's an extremely brief interval—called a "vertical blank" period—in which the whole screen goes blank.

Dot-matrix characters.

Look closely at a computer-generated text display on a TV screen, and you may be able to see that each character on the screen is made up of tiny dots. If you could look closely enough at the screen text graphics generated by your Atari—while your computer is in its normal 40-column by 24-line text mode—you'd be able to see that each letter is made up of sixty-four dots, arranged in a matrix eight dots wide and eight dots high.

Because of a picture-tube design technique called "overscan," however, not all of the 262 scan lines available for a TV picture appear on-screen; some fall off the edges and are never seen. So computer programs that generate video displays don't usually make use of all of those lines. Your Atari, for example, uses only 192 of the 262 scan lines available.

Atari BASIC supports four text modes, each of which produces letters of a different size. But, no matter what text mode you're in, and no matter how large the letters on your screen are, each line of *text* in an Atari display is always called a "mode line." In your Atari's normal 40-column by 24-line text mode—the mode referred to in Atari BASIC as "graphics 0"—each letter in a mode line is eight dots high, and each of those dots equates to one scan line.

In BASIC's graphics 0 mode, therefore, one mode line is equal to eight scan lines.

There are two other text modes in Atari BASIC—graphics 1, in which the characters on-screen are the same height as graphics 0 characters but twice as wide; and graphics 2, in which the characters are twice as high *and* twice as wide as standard graphics 0 characters. When your computer is in its graphics 1 mode, each mode line is made up of eight scan lines—the same number of scan lines used in a mode line in graphics 0. When your Atari is in its graphics 2 mode, however, each mode line equates to sixteen scan lines.

Antic mode 3.

There's another text mode, called "ANTIC mode 3," that's not supported by BASIC. In ANTIC mode 3, each mode line is made up of ten scan lines. You can find out more about ANTIC mode 3 by reading the Atari programmer's manual *De Re Atari*, or by consulting the Atari 400/800 Technical Reference Notes published by Atari.

In addition to their four text modes, Atari computers have numerous graphics modes—either ten or thirteen of them, depend on what kind of graphics hardware came installed in your model. (The number of graphics modes offered by Atari computers varies, with which chip is included, CTIA or GTIA).

In the non-text graphics modes, the number of scan lines per mode line can range from one (in high-resolution graphics) to eight (in low-resolution). The number of colors available also differs from graphics mode to graphics mode.

Table 1 shows the graphics modes available to Atari programmers. You may notice that there are differences between the ANTIC and the BASIC designations of these modes, and that ANTIC supports more modes than Atari BASIC does. And this table doesn't include the special modes available to owners of GTIA chips, since programs using those modes won't work properly on all Atari computers. If you want to use them anyway, you can find out how in *De Re Atari*.

Customizing your Atari's screen display.

Two steps are needed to custom design an Atari screen display. First, you have a special kind of program called a "display list." Then you have to tell your computer how to use the display list you've designed.

A display list is made up of a series of 1-byte instructions that can be placed almost anywhere in your computer's available RAM. Anytime you want to see what a display list looks like, you can find one by using your as-

Table 1.

Atari Text and Graphics Modes			
ANTIC mode	BASIC mode	Scan lines per mode line	No. of colors
2	0	8	2
3	None	10	2
4	None	8	4
5	None	16	4
6	1	8	5
7	2	16	5
8	3	8	4
9	4	4	2
A	5	4	4
B	6	2	2
C	None	1	2
D	7	2	4
E	None	1	4
F	8	1	2

sembler's debugging utility to peek into your computer's memory.

When you turn on your computer, it automatically goes into its graphics 0 mode, and the address of the display list it uses to generate that mode is always stored in two locations—specifically, memory addresses $230 and $231. Memory register $230 always holds the low byte of the starting address of your computer's display list, and memory register $231 always holds the high byte of the display list's starting address. So, once you know the contents of memory registers $230 and $231, you'll be able to locate the display list your computer's currently using.

Once you locate your computer's graphics 0 display list, you'll find that it looks something like this:

```
70 70 70 42 20 7C 02 02
02 02 02 02 02 02 02 02
02 02 02 02 02 02 02 02
02 02 02 02 02 41 E0 7B
```

As you can see, a display list is a pretty strange-looking program. Let's examine it, byte by byte, right now:

```
BYTES 1 - 3
$70 $70 $70
```

Each byte in a display list has a specific meaning to the ANTIC chip. Within each byte, each nybble—that is, each hexadecimal digit—also has a specific meaning. For example, this display begins with three bytes that hold the same hexadecimal value: *$70*. In the programming language of the ANTIC chip, the value *$70* tells ANTIC to display one blank mode line—which in BASIC graphics 0, equates to eight blank scan lines.

This, as it turns out, is the standard way to start a display list for a graphics 0 display. Because of the overscan characteristic of a TV screen, it's standard practice to kick off a graphics 0 display with three blank mode lines—or, in ANTIC language, with three *$70*s. That will pull the beginning of your graphics 0 display down to the top of your TV's picture tube, where you can be pretty sure your complete display will be visible on-screen.

```
BYTES 4 - 6
$42 $20 $7C
```

After three blank mode lines have been displayed, we get to the first actual display byte on our sample display list: the hexadecimal number *$42*. In ANTIC language, the value *$42* is what's known as a "load memory scan" (LMS) command. After all necessary blank lines have been taken care of, the first display byte in a display list is always a load memory scan command, and an LMS command is always a 3-byte instruction. In the display list we're examining, the load memory scan instruction is made up of the 3 bytes *$42, $20* and *$7C*.

The first nybble in this instruction—the digit 4—alerts ANTIC that this is an LMS instruction.

The second nybble in the LMS instruction—the digit 2—tells ANTIC to display an ANTIC mode 2 line. Consult the table on graphics modes presented a few paragraphs back, and you'll see that, in ANTIC language, mode 2 is the same as BASIC mode 0.

The next 2 bytes of the LMS command—the bytes *$20* and *$7C*—provide ANTIC with the address at which screen memory will begin. ANTIC interprets these 2 bytes low-byte first, in standard 6502 fashion. When ANTIC encounters the LMS instruction *$42 $20 $7C*, therefore, the first byte displayed on your Atari's video screen will be whatever byte is stored in memory location $7C20.

When you write a display list, you can put your screen memory in just about any convenient, available block of RAM. And you can fill that RAM up with whatever you like—codes that equate to text, display screens drawn with the help of a graphics program, or character graphics created with a graphics-generator program. Once you have a display created, you can tell your display list where to find it, by placing its starting address in the 2 bytes that follow your display list's LMS command.

```
BYTES 7 - 29
The byte $02, repeated 22 times
```

As explained above, the first LMS command in a display list tells ANTIC two things: the address at which screen memory begins, and the graphics mode to use to display the first mode line of text or data that will be found starting at that address.

After ANTIC has been presented with this information, it must be told what graphics mode to use to generate each subsequent mode line that will displayed on-screen.

In the display list we're examining, every mode line on the screen is an ANTIC mode 2 line. Therefore, the next twenty-three instructions in this display list are all the same; each will tell ANTIC that the next line on the screen will be an ANTIC mode 2 line.

What would happen, you may ask, if all these instructions were *not* the same? Well, if they weren't, then more than one graphics mode could be displayed on-screen simultaneously. Text of various sizes could be displayed on the same screen, and text and graphics modes could be intermixed as desired. This is a very powerful—and quite unusual—capability of Atari computers. You'll get a chance to see exactly how it works before you finish this article.

```
BYTES 30 - 32
$41 $E0 $7B
```

Every display list must end with a 3-byte command called a JVB (jump on vertical blank) instruction. The first byte in a JVB instruction is always the value *$41*. The next 2 bytes always combine to form a jump address. The destination of the jump is always the beginning of the display list in which the jump is contained.

As it happens, the display list we're looking at starts at memory address *$7BE0*. So that's the address that follows (low byte first) the JVB instruction *$41*.

When ANTIC encounters the JVB instruction *$41* in a display list, it jumps back to the beginning of the display list, waits for the next vertical blank period between raster scan displays, then jumps to the address that follows the JVB instruction. And, since this address is the address of the beginning of the display list, what the JVB instruction really does is run the display list again.

Running a display list.

As I've pointed out, a display list can be placed in almost any convenient spot available in your computer's memory. Screen memory can be placed just about anywhere in RAM, too. Once you've created a display list and a block of data to be used as screen memory, all you have to do to put your custom-designed display on your TV screen is write a simple program that tells your computer's operating system (OS) where your display list is.

To direct your computer to your custom display list, you simply store new values into a pair of OS memory locations known as "shadow" locations. Shadow addresses are used often in Atari programming, so I might as well explain what they are right now.

In your computer's memory, there are some very useful hardware registers not normally accessed by user-written programs. But sixty times per second, the data in each of these memory locations is updated. During this updating process, the value stored in each register is replaced by data that's been stored in a corresponding shadow register. And shadow registers are in user-accessible RAM. So, by changing the value in a shadow register, you can also change the value of its corresponding hardware register. For most intents and purposes, therefore, a shadow register works just about like any other OS register situated in RAM.

Three shadow addresses that are often used in display-list programs are *$22F*, *$230* and *$231*. Address *$22F* is an Atari OS memory location called SDMCTL (Shadow Direct Memory Access Control). Addresses *$230* and *$231* are OS locations called SDLSTH (Shadow Display List Pointer - Low) and SDLSTL (Shadow Display List Pointer - High).

To write a program that will put a custom display list on your Atari's screen, all you have to do is follow these three steps:

(1) Turn your computer's ANTIC chip off by storing a *0* in $22F (SDMCTL).

(2) Store the starting address of your custom display list in $230 and $231 (SDLSTL and SDLSTH).

(3) Turn your computer's ANTIC chip on again by storing the value *$22* in $22F (SDMCTL).

Doing it.

Now that you know *how* to do all of this, we're ready for action. The following program, together with the article you've just read, should provide you with all of the information you'll need to start designing your own customized display lists and creating your own mixed-mode screen displays.

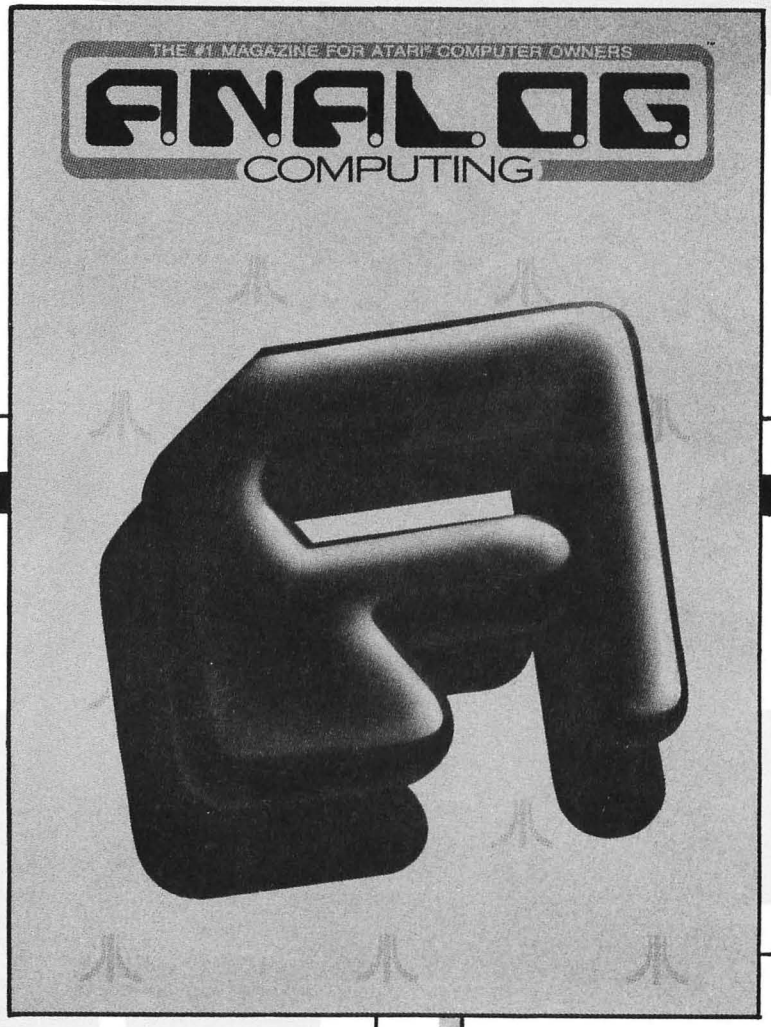
Mark Andrews is the author of Atari Roots *(Datamost: 1984), the top-selling book on Atari assembly language programming. He is also a frequent contributor to many computer magazines. This is the first article he's written for* ***ANALOG Computing****.*

Listing 1.
Assembly listing.

```
;
;'HELLO': MIXED MODE SCREEN DISPLAY
;
          .OPT OBJ
          *=  $3000
          JMP INIT
;
SDMCTL    =   $022F
;
SDLSTL    =   $0230
SDLSTH    =   $0231
;
COLOR0    =   $02C4    ;OS COLOR REGS
COLOR1    =   $02C5
COLOR2    =   $02C6
COLOR3    =   $02C7
COLOR4    =   $02C8
;
;DISPLAY LIST DATA
;
START
;
LINE1     .SBYTE " FROM "
          .SBYTE "A.N.A.L.O.G.:"
LINE2     .SBYTE "   a title "
          .SBYTE "screen  "
LINE3     .SBYTE "           "
          .SBYTE " By (Your Name)"
          .SBYTE "             "
LINE4     .SBYTE "   PLEASE "
          .SBYTE "STAND BY  "
;
;DISPLAY LIST
;
HLST
;3 BLANK LINES
          .BYTE $70,$70,$70
;MORE BLANK LINES
          .BYTE $70,$70,$70,$70,$70
;LMS, ANTIC MODE 6 (BASIC MODE 2)
          .BYTE $46
;TEXT LINE: "FROM A.N.A.L.O.G.:"
          .WORD LINE1
;MORE BLANK LINES
          .BYTE $70,$70,$70,$70
;LMS, ANTIC MODE 7
          .BYTE $47
;TEXT LINE: "A title screen"
          .WORD LINE2
;ANOTHER BLANK LINE
```

```
          .BYTE $70
;LMS, ANTIC MODE 2 (GRAPHICS 0)
          .BYTE $42
;TEXT LINE: "By [Your Name]"
          .WORD LINE3
; BLANK LINES
          .BYTE $70,$70,$70,$70
;LMS, ANTIC MODE 6
          .BYTE $46
;TEXT LINE: "PLEASE STAND BY"
          .WORD LINE4
;5 BLANK LINES
          .BYTE $70,$70,$70,$70,$70
;JVB INSTRUCTION
          .BYTE $41
;ADDRESS OF DISPLAY LIST
          .WORD HLST
;
;RUN PROGRAM
;
;SWITCHING COLOR REGISTERS
;FOR NICELY COLORED DISPLAY
INIT
          LDA COLOR3
          STA COLOR1
          LDA COLOR4
          STA COLOR2
;NOW WE'LL RUN THE PROGRAM
          LDA #0
;TURN OFF ANTIC WHILE WE STORE
;OUR NEW LIST'S ADDRESS IN THE
; OS DISPLAY LIST POINTER.
          STA SDMCTL
          LDA #HLST&255
          STA SDLSTL
          LDA #HLST/256
          STA SDLSTH
;TURN ANTIC BACK ON
          LDA #$22
          STA SDMCTL
;
FINI
          RTS
```

●



Check
Us Out

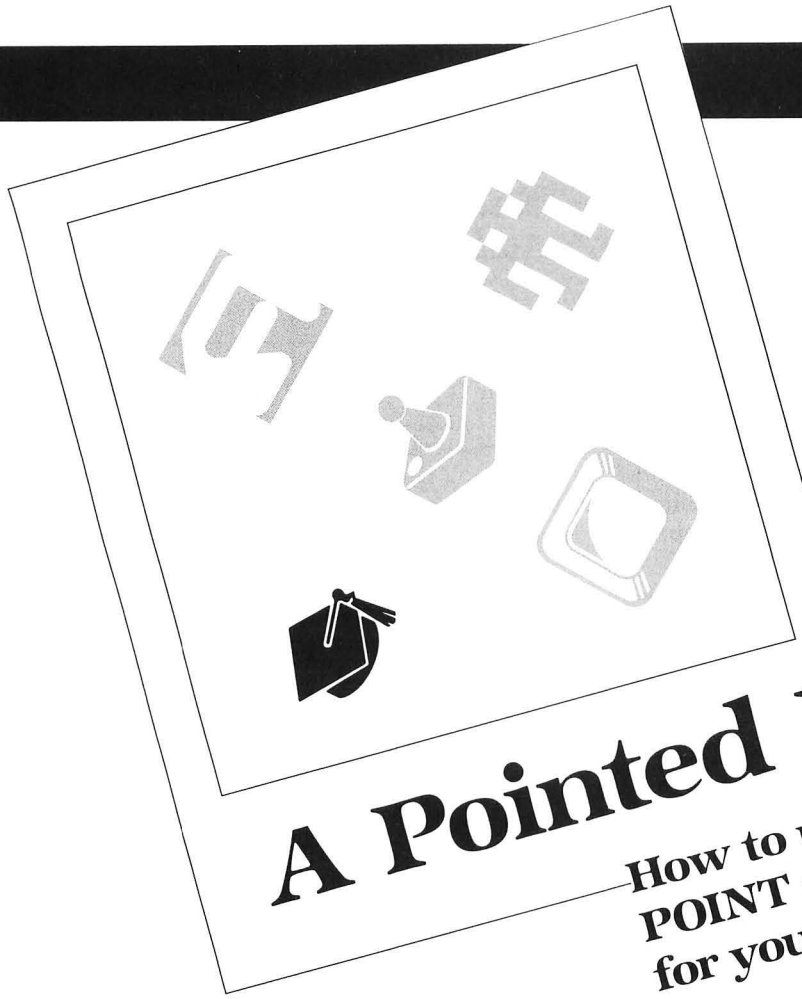
WANT TO SUBSCRIBE?

CALL TOLL FREE 1-800-345-8112

IN PENNSYLVANIA 1-800-662-2444

OR USE THE POSTAGE-PAID CARDS IN THE BACK OF THIS BOOK

For the #1 magazine for Atari computer owners!



A Pointed Note

How to use your Atari's POINT and NOTE functions for your own databases.

by Barbara Donovan

Have you ever wanted a tailor-made file management program that will allow you to choose exactly the information you want to include—*and* how you want to manipulate it? Have I got a program for you!

The inspiration for this came while I was on a diet. Because I'm a bit of a health food nut, I wanted to know how many calories and nutrients various recipes would have. This seemed like an excellent application for a computer. After all, they're supposed to be good at numbers—multiplying and dividing, and stuff like that. I realized I needed a file with the information I wanted, which could be accessed selectively.

Sounds simple, doesn't it? Well, I had a database program on which I could have set up records for each item needed. But that meant, every time I wanted to add up a recipe, I would have to go through *all* those records and, by editing them, put a key to indicate which to include. I would also have to change the multiple for each item desired. Doing it with pencil and paper would be faster.

After a little thought, I decided what was needed was a way to choose which items I wanted to include in a particular calculation (in computerese, that's an "indexed-sequential" file). This put me off my stride a little. I had read that the technique was really complicated. Guess what? It's not so bad.

Using the Atari's NOTE and POINT functions, the program became relatively simple to write. **Pointed Note** is written in a top-down way (this just means everything possible is in subroutines), so it could be debugged more easily. Now, if you don't care how many calories things have, don't despair. **Pointed Note** will work for any type of data you wish to store, recall, organize and/or manipulate—calories are just one example. Also, if you save the various subroutines by LISTing them, you can easily build another program without a lot of retyping.

There are a few things you should know about databases. Picture a file cabinet with several drawers. That's the "database." Each drawer has a bunch of file folders relating to the same sort of thing (like bills to be paid). That's a "file." Each file folder has a sheet of paper with information about a particular member of the file (like the electric bill). That's a "record." And, finally, each sheet of paper has various entries on it (like your account number). Those are "fields." Fields are made up of alphanumeric characters.

So, what I want to set up is a file of foods, containing records for each food with fields of pertinent information. For the sake of this discussion, we'll consider the fields to be: (1) the name of the food; (2) and (3) the number of base units (i.e., *1 cup*—to be treated as *1* and *cup*—two fields); and (4) the calories. Please note that the number and type of fields can easily be modified for any application.

This is a simple matter. I decided the name of the food could be a maximum of twenty-five characters, number of units a maximum of three characters, the unit itself fifteen characters, and the calories three characters. Why is it important to decide on this information in advance? Two reasons: you need to know how large to dimension the variables; and each record must take up exactly the same amount of disk space.

That last needs some explanation. When your computer writes data to the disk, it puts one character after another until it comes to an end-of-line character (EOL), which it also puts on the disk to demarcate the end of that field. When your computer inputs data from the disk, it reads each characater until it comes to the EOL, then stops. Therefore, if we want to be able to make any later changes to our data, each field must always be printed with exactly the right number of characters—or the updates and previous data may become confused with each other in the overwriting.

In order to make sure of field length, we must pad any items less than the set field length with blanks. This is done in the PADDATA and PADITEM subroutines, before the data is written to disk.

Now, getting back to that indexed-sequential stuff, we want to be able to find any record, read its information, manipulate that information and get results. However, we *don't* want to have to read through every record to find the one we want. In our sample program this wouldn't be a big deal, but if each record had, in addition to calories, Vitamin A, Vitamin B-1, Vitamin E, Calcium, Sodium. . . well, you get the idea. It would take forever to go through *all* of them every time. With this program, we only have to go through all of them when adding new records.

When we add new records, we'll find out at which sector and character on disk the record begins and go directly to that location. This is done by having a separate file (another drawer in our cabinet), containing a record with only the name of the item, sector and character. Neat, huh?

Basically, we need two functions to accomplish this. NOTE tells us where on disk the read/write head is located. The form is:

NOTE #(channel no.),SECT,CHAR

Channel number refers to the line you've opened to the disk drive, as in: *OPEN #1,4,0,"D:DATA.FIL"*. *SECT* and *CHAR* are variable names which will contain, respectively, the sector number and character position of the drive head. Each sector has 128 character positions.

POINT tells the read/write head to move to a specified position and start operations there. The form is:

POINT #(channel no.),SECT,CHAR

As you see, it's similar to the NOTE statement. The sector and character positions must be given as variables. (Also, keep in mind that you can only POINT to a position in a file which exists and has been OPENed).

The OPEN statement allows you to communicate directly with peripheral devices, such as disk drives, cassettes, printers, keyboard, etc. Each device has a letter specification. We're only concerned with the disk drive, which is indicated by "D: (the colon is necessary), and must be followed by an existing filename (with one exception, discussed below).

In examining the OPEN statement, we're most concerned with the second specification (i.e., the *#4* in *OPEN #1,4,0,"D:DATA.FIL"*). There are four modes of communication available when OPENing to the disk drive: (1) input (mode 4) allows you to read *only* the data in your file; (2) output (mode 8) allows you to write *only* to the data file—when OPENing in this mode, the drive head will be at the beginning of the file; (3) append (mode 9) allows you to add data to the end of the file—the drive head upon OPENing will point to the end of the file and automatically allot another 128 characters, minimum, of disk space to that file; and (4) update (mode 12), which allows both reading and writing to the file—upon OPENing, the drive head will be at the beginning of the file.

Mode 8 is the only way to create a file. If mode 8 is specified (write only), DOS will open a file with the name specified and write data to it, if desired. Modes 8 and 12 will write over and destroy any previous data.

Now we get to the simple part. All we have to do to add data to our file is specify mode 9 (append), give the data to the computer, and have it added to the end of our file. Then, to find out where it is on the disk so that we can index it, we OPEN for a read (mode 4), NOTE the position of the head, read a record, write the name of the record and its position to the index, NOTE the position of the head again, read the next record, etc. . .until the end of the file.

Now, when we want the info back, all we have to do is search our shorter index file, find the location of the item we're looking for, and have the read/write head POINT there and start inputting.

Let's look at the program—a lot easier than trying to explain all this. The beginning merely dimensions the string variables needed, fills BLANK$ with spaces and assigns line numbers to variables. This way, when I call a subroutine, I have an idea what it *does* (instead of *seeing* a meaningless number).

And, if I renumber a subroutine, I just change one variable to point to that subroutine from any part of the program. Also, Line 82 identifies an end character, so the computer knows when we're finished adding, updating, or using data.

Lines 100-260 are the main menu.

Lines 300-391 form the routine to find a particular record, show you the basic unit, ask for a multiple, and print the number of calories for that food item. As you can see, most all it does is call on subroutines. This simplifies writing the program since the same procedures are used in the other main routines. For example, ITEMIN, merely asks for the name of the food you want, has it padded with spaces in the PADITEM subroutine so it's the required twenty-five characters long, and returns.

Data is padded by converting all nonstring variables to string variables, checking variable lengths, and adding blanks if necessary to fill the allotted space.

FINDITEM locates the item in the index file and reports back. Next, the data is read—RDDATA subroutine—by

POINTing to the correct position on the disk file and reading the information.

The UNITIN subroutine gets the multiple (i.e., you're using two apples in a recipe and the base unit is *1*).

Then FIGCAL and CALPRNT, as their names imply, figure out the calories, print them out, and return.

Once the last item is indicated by a CTRL-E entry, the program calls subroutine TOTPRNT, to print out the total number of calories for all items and quantities specified, then returns and waits for you to hit RETURN, to go back to the main menu.

Lines 400-450 are the main routine to add new records to the file. Notice that this opens a channel to mode 9 (append). This routine calls subroutines already found, such as ITEMIN.

WRITEDATA prints on disk all the data fields (i.e., name, number, unit and calories) and waits at the end of the file for any more additions.

If CTRL-E is entered, signalling no more additions, the channel to the data file is closed, and the program calls the INDEX subroutine. It OPENs two channels, one to read the data file and NOTE the position of each record, and one to write a new index file (mode 8) over any old one still there. When finished, it returns to the main menu.

Lines 500-570 are the update main routine, to easily alter a record (*old record:Banana,1,medium,101/new record:Banana,1,large,116*). This is the reason we've been padding our data fields. Even though a new field may have more or less characters than the old, it won't change the disk position of the record.

The only new subroutines are: NWDATA, which gets the changes you want to make; and WRITNW, which then POINTs to the beginning of the old record and writes over it with the new data.

Finally, the TRAP statements send the program to ERROR1. This is an easy way to handle end-of-file. When the end is reached and the computer's asked to read more, an error occurs, and program execution halts. By using an error handler routine to check which error had occurred, the program may continue—even though an item is not found. Control is returned to the correct portion of the program by assigning each main routine a number contained in the variable TEST.

All that's left is to create the files to be used. This is done in immediate mode (when the screen says *READY*). Just key *OPEN #1,8,0,"D:DATA.FIL"* and press RETURN. Then CLOSE #1 and RETURN. The same should be done to create the index file: *OPEN #1,8,0,"D:INDEX.FIL"*. To start your file, run the program and hit *2* to enter the ADD routine. Now, key the information asked for, and it will be written to the data file and index file, appropriately.

Three useful expansions of this type of program would be a directory, a multiple file routine and a hard-copy subroutine.

The directory could print out either all names of the records or all the records in their entirety, by accessing the index file or the data file, respectively, and sequentially listing their contents.

A multiple file would be used to remember a certain combination of records, such as a recipe. By putting in the name of a food item and its unit multiple, another file can be added to, with records containing: (1) recipe name, (2) number of items, (3) list of sector/character locations with a HOWMANY unit multiplier. By setting up a FOR/NEXT loop based on field (2), you could quickly POINT to each item included and figure the multiple for that particular recipe.

You'll notice that I've used arrays for SECT and CHAR to facilitate this type of application. The arrays will contain sector/character positions for each item. When the FOR/NEXT loop is entered, the loop index is used as the array index, and the correct data will be written or read for the assigned number of items.

Simplest of all, to obtain hard-copy, insert a question at the beginning of a routine—such as, *Do you want a print out?* Then, based on the INPUT answer, a variable (i.e., *PO$*) is set. If the answer is yes, a hard-copy subroutine is called from a main routine.

As I mentioned, this program is easily adapted to other uses: inventories, student test data, bills paid and payable, etc. I would be interested in hearing of other applications.

Barbara Donovan, a native New Yorker, lives in Virginia with her writer-husband and their three children. She is now taking courses and plans to seek a Ph.D. in Computer Science in the near future. She's been a computer hobbyist since 1979 (starting on a TRS-80, which was destroyed in a fire) and, since 1983, has been a loyal Atari owner.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
AE 10 REM ******************************
MF 11 REM *    FILE MANAGER            *
MB 12 REM *       BY                   *
OK 13 REM *    BARBARA DONOVAN         *
AM 14 REM ******************************
BI 15 REM
IM 17 REM ***** DIMENSION AND *********
RJ 18 REM *** INITIALIZE VARIABLES ****
NR 20 DIM ITEM$(25),UNIT$(15),NAME$(25),C
   $(3),LASTITEM$(25),SECT$(3),CHAR$(3),N
   UM$(3),CAL$(3),BLANK$(25),TEMP$(3)
ME 25 DIM CHAR(15),SECT(15)
PZ 26 BLANK$=" ":BLANK$(25)=BLANK$:BLANK$
   (2)=BLANK$
ZB 30 ADDITEM=400:CALPRNT=1350:CLEARSCR=1
   400:COUNTCAL=300:ERROR1=1500:FIGCAL=13
   00:FINDITEM=1050:INDEX=1600
WV 63 PADITEM=1950:PADSECT=1980:PADDATA=2
   040:ITEMIN=1000:IUPDATE=500:MENU=100:N
   WDATA=1800:RDDATA=1150:TOTPRNT=1450
HM 74 UNITIN=1250:WRITEDATA=1700:WRITNW=1
   850
WZ 82 LASTITEM$(1,1)="┐":LASTITEM$(2,25)=
   BLANK$
CW 97 REM ********* MAIN MENU **********
QM 100 PRINT "{"
TZ 110 POSITION 2,2:PRINT "TO":POSITION 2
   0,2:PRINT "PRESS:"
```

```
PA 120 POSITION 2,4:PRINT "COUNT CALORIES
   ":POSITION 20,4:PRINT "1 <RETURN>"
JZ 130 POSITION 2,6:PRINT "ADD AN ITEM":P
   OSITION 20,6:PRINT "2 <RETURN>"
QZ 140 POSITION 2,8:PRINT "UPDATE ITEM":P
   OSITION 20,8:PRINT "3 <RETURN>"
BA 180 TRAP 270:REM /ONLY TAKE NUMBERS/
NS 190 POSITION 2,22:PRINT "WHICH";:INPUT
    C
RN 200 ON C GOTO COUNTCAL,ADDITEM,IUPDATE
UE 260 GOTO 190:REM /NO.BETWEEN 1-6 ONLY
ZF 297 REM ****** CALORIES FOR 1 *******
DJ 298 REM ****** OR MORE ITEMS ********
LC 300 PRINT "{":TOTAL=0:P=8:TEST=310:I=1
DZ 305 TRAP ERROR1
DM 310 GOSUB ITEMIN
YJ 320 IF ITEM$=LASTITEM$ THEN 390
VB 330 GOSUB FINDITEM
RN 340 GOSUB RDDATA
LW 350 GOSUB UNITIN
NH 360 GOSUB FIGCAL
FW 370 GOSUB CALPRNT
HJ 380 GOSUB CLEARSCR:GOTO 310
XR 390 GOSUB TOTPRNT
XG 391 POSITION 2,23:PRINT "PRESS RETURN
   FOR MENU";:INPUT C$:GOTO MENU
BV 397 REM ****** ADD NEW ITEM *******
OX 400 PRINT "{":TEST=410:TRAP ERROR1
YO 402 OPEN #1,9,0,"D:DATA.FIL"
DN 410 GOSUB ITEMIN
GF 420 IF ITEM$=LASTITEM$ THEN CLOSE #1:G
   OSUB INDEX:GOTO MENU
YD 430 GOSUB WRITEDATA
ID 450 GOSUB CLEARSCR:GOTO 410
ZP 497 REM ***** UPDATE DATA FILE *****
QQ 500 PRINT "{"
BH 510 TEST=510:TRAP ERROR1
DQ 520 GOSUB ITEMIN
AA 525 IF ITEM$=LASTITEM$ THEN GOTO MENU
VD 530 GOSUB FINDITEM
RP 540 GOSUB RDDATA
YT 550 GOSUB NWDATA
XD 560 GOSUB WRITNW
JH 570 GOSUB CLEARSCR:GOTO 510
CW 997 REM ****** ENTER ITEM/NAME ******
VS 1000 POSITION 2,2
WN 1010 PRINT "(<CTRL>'E'<RET>=STOP)";
WR 1020 PRINT "ENTER ITEM:":INPUT ITEM$:L
   I=LEN(ITEM$)
NM 1025 GOSUB PADITEM
AI 1030 RETURN
UP 1047 REM **** FIND ITEM LOCATION ****
DG 1050 OPEN #1,4,0,"D:INDEX.FIL"
QW 1060 TRAP ERROR1
DB 1070 INPUT #1;NAME$
KY 1080 INPUT #1;SECT$:SECT(I)=VAL(SECT$)
RD 1090 INPUT #1;CHAR$:CHAR(I)=VAL(CHAR$)
IL 1100 IF NAME$=ITEM$ THEN CLOSE #1:RETU
   RN
QG 1110 GOTO 1070
HY 1147 REM *** READ IN DATA FOR ITEM ***
GQ 1150 OPEN #1,4,0,"D:DATA.FIL"
UZ 1160 POINT #1,SECT(I),CHAR(I)
DD 1170 INPUT #1;NAME$
SK 1180 INPUT #1;NUM$
XE 1190 INPUT #1;UNIT$
YG 1200 INPUT #1;CAL$
MQ 1210 CLOSE #1
AJ 1220 RETURN
JZ 1247 REM *** GET NUMBER OF UNITS ****
QG 1250 POSITION 2,4:PRINT "UNIT:";VAL(NU
   M$);" ";UNIT$
NC 1260 POSITION 2,6:PRINT "HOW MANY UNIT
   S";:INPUT HOWMANY
AY 1270 RETURN
LF 1297 REM ** FIGURE NUMBER CALORIES **
WS 1300 ICAL=VAL(CAL$)
OO 1305 ICAL=ICAL*HOWMANY
RV 1310 TOTAL=TOTAL+ICAL
AL 1320 RETURN
PS 1347 REM * PRINT NUMBER OF CALORIES *
YA 1350 RP=2
BM 1360 POSITION RP,P:PRINT ITEM$(1,LI):R
   P=RP+LI+2
CB 1365 NUM=VAL(NUM$)
XI 1370 POSITION RP,P:PRINT NUM*HOWMANY;"
    ";UNIT$
FE 1380 POSITION 32,P:PRINT ICAL
VU 1390 P=P+1:RETURN
AA 1397 REM ***** CLEAR INPUT AREA *****
LH 1400 POSITION 2,2:PRINT "
                                  "
SO 1405 POSITION 2,3:PRINT "
                                  "
NG 1410 POSITION 2,4:PRINT "
                                  "
SX 1412 POSITION 2,5:PRINT "
                                  "
AA 1415 POSITION 2,6:PRINT "
                                  "
UZ 1417 POSITION 2,7:PRINT "
                                  "
AN 1420 RETURN
WO 1447 REM *** PRINT TOTAL CALORIES ***
HN 1450 POSITION 2,21:PRINT "TOTAL CALORI
   ES:",TOTAL
AZ 1460 RETURN
GP 1497 REM ***** ERROR HANDLER #1 *****
EU 1500 ERR=PEEK(195):CLOSE #1:POP
RZ 1505 GOSUB CLEARSCR
JT 1510 IF ERR=136 THEN POSITION 2,2:PRIN
   T "ITEM NOT FOUND":GOTO 1530
QP 1511 IF ERR=8 AND TEST=310 THEN POSITI
   ON 2,2:PRINT "NUMERIC INPUT ONLY":GOTO
    1530
MY 1512 IF ERR=8 AND TEST=410 OR TEST=510
    THEN POSITION 2,2:PRINT "SEPARATE NUM
   BER AND UNIT BY A COMMA":GOTO 1530
EN 1520 POSITION 2,2:PRINT "UNEXPECTED ER
   ROR #";ERR
XK 1530 POSITION 2,3:PRINT "ANYKEY=CONTIN
   UE"
YD 1535 POSITION 2,4:PRINT "<CTRL>'E'=MEN
   U"
TC 1540 OPEN #2,4,0,"K:":GET #2,C:CLOSE #
   2
OS 1550 IF C=5 THEN GOTO MENU
BL 1560 GOSUB CLEARSCR:GOTO TEST
OP 1597 REM *** RECORD INDEX LISTING ***
UH 1600 TRAP 1675
NU 1605 GOSUB PADSECT
FK 1610 OPEN #1,8,0,"D:INDEX.FIL"
HD 1620 OPEN #2,4,0,"D:DATA.FIL"
AA 1630 NOTE #2,SECT,CHAR
EI 1635 INPUT #2;NAME$
SV 1640 INPUT #2;NUM$
YG 1645 INPUT #2;UNIT$
ZQ 1650 INPUT #2;CAL$
CE 1655 PRINT #1;NAME$
KK 1660 PRINT #1;SECT
BC 1665 PRINT #1;CHAR
SC 1670 GOTO 1630
TN 1675 IF PEEK(195)=136 THEN CLOSE #1:CL
   OSE #2
BJ 1680 RETURN
ZD 1697 REM *** RECORD DATA FOR ITEM ***
NH 1700 POSITION 2,3:PRINT "NUMBER,UNIT:"
   ;:INPUT NUM,UNIT$:NUM$=STR$(NUM)
OB 1710 POSITION 2,4:PRINT "CAL:";:INPUT
   CAL:CAL$=STR$(CAL)
KG 1720 PRINT #1;ITEM$
AY 1725 GOSUB PADDATA
QK 1730 PRINT #1;NUM$
VE 1740 PRINT #1;UNIT$
XI 1750 PRINT #1;CAL$
AQ 1755 NOTE #1,SECT,CHAR
```

```
BF 1760 RETURN
BB 1797 REM *** GET NEW DATA FOR ITEM ***
QF 1800 POSITION 2,4:PRINT "NUMBER,UNIT=
   ";VAL(NUM$);" ";UNIT$
OS 1810 POSITION 2,5:PRINT "CHANGE TO:";:
   INPUT NUM,UNIT$:NUM$=STR$(NUM)
ZA 1820 POSITION 2,6:PRINT "CALORIES= ";V
   AL(CAL$)
VV 1825 POSITION 2,7:PRINT "CHANGE TO: ";
   :INPUT CAL:CAL$=STR$(CAL)
AY 1830 RETURN
JE 1847 REM **** WRITE REVISED DATA ****
QD 1850 OPEN #1,12,0,"D:DATA.FIL"
VN 1860 POINT #1,SECT(I),CHAR(I)
KX 1870 PRINT #1;ITEM$
BP 1875 GOSUB PADDATA
RB 1880 PRINT #1;NUM$
VV 1890 PRINT #1;UNIT$
WX 1900 PRINT #1;CAL$
DW 1910 CLOSE #1:RETURN
DX 1947 REM ***** PAD ITEM FIELD *****
RB 1950 LI=LEN(ITEM$)
QK 1955 ITEM$(LI+1,25)=BLANK$
BJ 1960 RETURN
DB 1977 REM *** PAD SECT/CHAR FIELDS ***
RR 1980 LTH=LEN(SECT$)
ZR 1985 IF LTH=2 THEN SECT$(2,3)=SECT$:SE
   CT$(1,1)="0"
QW 1990 IF LTH=1 THEN SECT$(3,3)=SECT$:SE
   CT$(1,2)="00"
JD 1995 LTH=LEN(CHAR$)
QZ 2000 IF LTH=2 THEN CHAR$(2,3)=CHAR$:CH
   AR$(1,1)="0"
JP 2005 IF LTH=1 THEN CHAR$(3,3)=CHAR$:CH
   AR$(1,2)="00"
AD 2010 RETURN
UW 2037 REM ***** PAD DATA FIELDS ******
CX 2040 LU=LEN(UNIT$)
ZU 2045 UNIT$(LU+1,15)=BLANK$
XM 2050 LTH=LEN(NUM$)
RM 2055 IF VAL(NUM$)<1 THEN GOTO 2070
AF 2060 IF LTH=2 THEN NUM$(2,3)=NUM$:NUM$
   (1,1)="0"
ND 2065 IF LTH=1 THEN NUM$(3,3)=NUM$:NUM$
   (1,2)="00"
FQ 2070 LTH=LEN(CAL$)
FC 2075 TEMP$=CAL$:CAL$="000"
KV 2080 IF LTH=2 THEN CAL$(2,3)=TEMP$
MB 2085 IF LTH=1 THEN CAL$(3,3)=TEMP$
BT 2090 CAL$=TEMP$
BV 2095 RETURN
```

●

ULTIMATE STORAGE

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

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

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

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

Call Toll Free 1-800-972-5858 Charge orders only, minimum $15.00

Enclosed is my check or money order in the amount of $_____.

Please send me: ____ **ANALOG Computing** files ____ **ANALOG Computing** binders.

PLEASE PRINT.

Name: _____

Address: (No P.O. Boxes) _____

City: _____ State: _____ Zip Code: _____

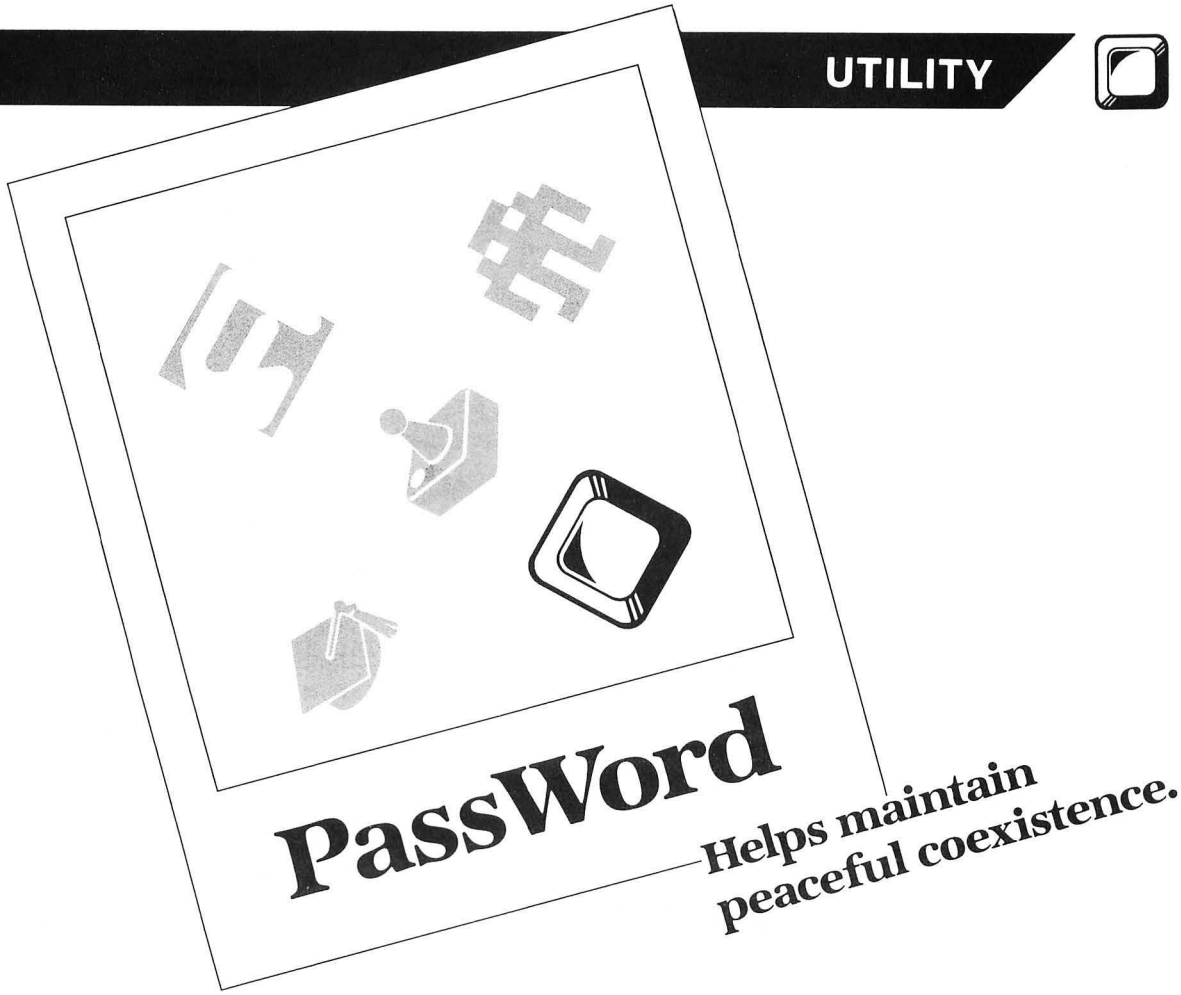
Send your order to:

Jesse Jones Industries

DEPT. ACOM, 499 East Erie Ave., Philadelphia, PA 19134

VISA PA residents, add 6% sales tax *Satisfaction guaranteed or money refunded.* MasterCard

UTILITIES



PassWord

Helps maintain peaceful coexistence.

by Jim Ehninger

A month ago, a friend of mine was having problems with his disks. His little brother was playing his games without permission; his sister was reading his **AtariWriter** files, and his dad replaced one of his BASIC files with a program that said, "Happy April Fool's Day." He came to me for help.

I created a short AUTORUN.SYS file that required a certian keypress before you could run DUP.SYS. But they could still get in by booting another disk, then reading the directory from there. "Give me time," I said. A week later, **PassWord** was created.

So what will it do?

Type in Listing 1 (the BASIC listing). Do *not* type in the assembly language source code; this is for advanced programmers to look over. Save the program. Insert a blank disk (or a disk that may be formatted) and execute the program, using *item P* at the first prompt. The current DOS in memory will be written to the disk, then the program will ask you for a password. A hint: use a short, one-word password that's easy to type in, easy to remember, and isn't too obvious.

After the program is through, it will reboot and let you try out your new, protected disk. I won't guarantee it will work for people like Tom Hudson or Kyle Peacock, but it will probably keep your family and friends out.

Any questions?

The following are some questions you may have about **PassWord**. First, how does it work? The program works by moving the disk directory (normally sector 361) to a different location. When you try to boot another disk and read the directory, you don't see a list of the files on that disk. Therefore, we have the protection we want.

Can someone else boot their **PassWord** disk and "get into" my disk? No, I thought of that, too. On almost all disks the directory is in a different place. The odds of two **PassWord** programs being the same are about 1 in 256.

Sounds good. How can I transfer DOS files to my **PassWord** disk? There are two methods I've found: (1) use the COPY+ 4.0 program supplied in Listing 1; or (2) type *POKE 1955,89:POKE 1956,228* to toggle DOS access, load the file, then *POKE 1955,0:POKE 1956,4* to return to **PassWord** access, and save the program.

Great! You've just given away the secret. No, they still have to boot up your disk before those POKEs will work. And please be careful when using these POKEs.

COPY+ 4.0.

The subroutine at the end of Listing 1 is a utility that enables you to copy any DOS file to your **PassWord** disk, and vice versa. *You must boot your **PassWord** disk up to run this program*. Otherwise, your computer will lock up and take a short trip to the Twilight Zone.

The COPY+ 4.0 program (yes, there were four versions)

and the **PassWord** program should be saved as one program. It's best to copy this program onto your **PassWord** disk if you're going to be doing a lot of copying (say, using it as a programming disk). Please specify this at the prompt.

If you have any questions about **PassWord** or COPY+ 4.0, or if you make up some new utilities for **PassWord**, write to the **Reader Comment** section. Users with 300-baud modems can leave me a message on StarGate Earth Bulletin Board System: (801) 272-1518, 10 p.m. to 7 a.m., seven days a week, ATASCII and full duplex. Have fun with your **PassWord**.

Jim Ehninger has owned an Atari 800 since 1982. Jim enjoys telecommunications and programming 6502 machine language and BASIC. He enjoys more cerebral games and is a remote SYSOP of Wally World BBS, 801-255-9345.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
AB 10 REM ** P/W - BY: JIM EHNINGER **
GU 20 REM *** FOR ANALOG COMPUTING ***
IG 30 OPEN #2,12,0,"E:"
SD 40 GRAPHICS 0:SETCOLOR 2,0,0
TB 50 OPEN #3,4,0,"K:":DIM DOS$(128),PW$(
   30),D$(1000),DSKINV$(5),B$(128),A$(255
   ),NO$(128),P$(2),ME$(10)
YD 55 ? "(P)/W program, or (G)o to COPY+
   4.0 :";:INPUT #2,P$:IF P$="G" THEN GOS
   UB 1000:END
EX 60 K1=PEEK(58404)+1:K2=PEEK(58405):P1=
   PEEK(58374)+1:P2=PEEK(58375):P$=CHR$(P
   1):P$(2)=CHR$(P2)
MT 70 FOR A=1 TO 128:NO$(A)=CHR$(0):NEXT
   A
CT 80 ? "P/W DISK BOOT MAKER - BY JIM EHN
   INGER"
HR 90 ? "(c) 1985 ANALOG COMPUTING↓"
AS 100 ? "INSERT A DISK INTO DRIVE 1.↓"
DP 110 ? "WARNING-THIS WILL ERASE ALL CON
   TENTS"
GW 120 ? "ON THAT DISK!↓"
QX 130 ? "PRESS RETURN:";
KP 140 GET #3,D:IF D<>155 THEN 140
DZ 150 ? "↑FORMATTING DISK...";
UP 160 XIO 254,#1,0,0,"D:"
PE 170 ? "↑CREATING SPACE...";
YG 180 OPEN #1,8,0,"D:SPACE.PW"
DL 190 FOR A=1 TO 380:PUT #1,0:NEXT A:CLO
   SE #1
RS 200 ? "↑WRITING DOS.SYS...";
HX 210 OPEN #1,8,0,"D:DOS.SYS":CLOSE #1:?
CP 211 ? "WOULD YOU LIKE OTHER FILES ON T
   HE":? "THE DISK BESIDES DOS? (i.e. DUP
   .SYS,":? "AUTORUN.SYS, etc.) <Y/N>";
WS 212 INPUT #2,PW$:IF PW$="N" THEN 220
SN 213 SGE=1:GOSUB 1000:SGE=0
VW 220 ? "↑WHAT WOULD YOU LIKE YOUR PASSW
   ORD":? "TO BE?":? "20 CHARACTERS MAX."
WF 230 ? "NO CONTROL CHARACTERS OR INVERS
   E!↓"
EP 240 ? "PASSWORD:";:INPUT #2,PW$
JQ 250 IF LEN(PW$)>20 THEN ? "20 CHARACTE
   RS MAX!":GOTO 240
OH 260 FOR A=1 TO LEN(PW$)
PO 270 IF ASC(PW$(A,A))>31 AND ASC(PW$(A,
   A))<125 THEN NEXT A:GOTO 290
WE 280 ? "NO CONTROL CHARACTERS OR INVERS
   E!!!":GOTO 240
MI 290 ? "YOUR PASSWORD IS:";PW$:? "IS TH
   IS CORRECT? ";:INPUT #2,A$:IF A$(1,1)<
   >"Y" THEN 220
AW 300 ? "↖PLEASE STAND BY - INITIALIZING
    DATA..":RESTORE :SE=0
AX 310 FOR A=1 TO 442
JN 320 READ D:SE=SE+D:D$(A)=CHR$(D):NEXT
   A:FOR A=1 TO 70:D$(442+A)=CHR$(0):NEXT
    A
VT 330 D$(14,15)=P$:D$(191,192)=P$:D$(206
   ,207)=P$:D$(225,226)=P$:D$(259,260)=P$
   :D$(313,314)=P$:D$(352,353)=P$
PX 340 D$(158,158)=CHR$(K1):D$(159,159)=C
   HR$(K2)
XE 350 IF SE<>42148 THEN ? "ERROR IN DATA
    STATEMENTS.":END
PM 360 FOR A=1 TO LEN(PW$):D$(229+A,229+A
   )=CHR$(ASC(PW$(A,A))*2):NEXT A
EV 370 N5=INT(RND(1)*245)+8:D$(427,427)=C
   HR$(N5)
TO 380 ? "↓WRITING PROGRAM..."
FU 390 DSK=768:SE=4:TT=1:RW=87
FL 400 DSKINV$="h SD♦"
PR 410 POKE DSK+1,1
VU 420 B$=D$(TT,TT+127):GOSUB 440:TT=TT+1
   28:SE=SE+1:IF SE<8 THEN 420
QB 430 GOTO 660
ZB 440 AD=ADR(B$):POKE DSK+2,RW
MZ 450 HIGH=INT(AD/256):LOW=AD-(HIGH*256)
OJ 460 POKE DSK+4,LOW:POKE DSK+5,HIGH
SX 470 SHI=INT(SE/256):SLO=SE-(SHI*256)
ST 480 POKE DSK+10,SLO:POKE DSK+11,SHI
UO 490 A=USR(ADR(DSKINV$)):IF PEEK(DSK+3)
   <>1 THEN ? "↖STATUS ERROR- ";PEEK(DSK+
   3):GOTO 440
ZB 500 RETURN
JT 510 DATA 162,0,142,198,2,189,162,8,240
   ,21,134,255,32,164,246,162,16,160,0,13
   6,208,253,202,208,248,166,255
FM 520 DATA 232,76,133,8,76,23,9,125,80,4
   7,87,32,45,32,40,99,41,32,49,57,56,53,
   32,65,78,65,76,79,71,32,67,79
TD 530 DATA 77,80,85,84,73,78,71,155,80,8
   2,79,71,82,65,77,32,66,89,58,32,74,73,
   77,32,69,72,78,73,78,71,69,82
BO 540 DATA 155,80,65,83,83,87,79,82,68,5
   8,30,30,30,30,30,30,30,30,30,255,255,2
   55,255,255,157,157,157,157,157
HR 550 DATA 157,157,157,157,157,29,29,29,
   29,29,29,29,29,29,29,31,31,31,31,31,31
   ,31,31,31,31,31,31,31,31,0,0
JV 560 DATA 169,0,141,22,9,32,226,246,201
   ,155,240,86,201,126,240,46,24,201,32,1
   44,240,201,125,176,236,141,21
CA 570 DATA 9,24,173,22,9,201,20,144,8,16
   9,253,32,164,246,76,28,9,173,21,9,174,
   22,9,157,2,1,32,164,246,238,22
GK 580 DATA 9,76,28,9,173,22,9,240,194,20
   6,22,9,169,126,32,164,246,76,28,9,32,3
   2,32,32,32,32,32,32,32
GL 590 DATA 32,32,32,32,32,32,32,32,32,32
   ,32,169,155,174,22,9,157,2,1,32,164,24
   6,162,0,24,126,101,9,232
VL 600 DATA 224,21,208,247,162,0,189,101,
   9,221,2,1,208,4,232,76,145,9,189,101,9
   ,201,16,208,10,189,2,1,201,155
QA 610 DATA 208,3,76,213,9,162,0,189,192,
   9,240,249,134,255,32,164,246,166,255,2
   32,76,176,9,155,253,126,80,65
HL 620 DATA 83,83,87,79,82,68,32,68,69,78
   ,73,69,68,33,155,0,162,0,189,231,9,240
   ,28,134,255,32,164,246,166,255
SI 630 DATA 232,76,215,9,155,66,111,111,1
```

```
   16,105,110,103,32,68,79,83,46,46,46,15
   5,0,169,4,141,164,7,169,0,141
ME 640 DATA 163,7,162,0,189,18,10,157,0,4
   ,232,224,40,208,245,76,20,7,24,173,10,
   3,201,105,144,28,201,112,176
GN 650 DATA 24,173,11,3,201,1,208,17,24,1
   73,10,3,105,0,141,10,3,173,11,3,105,0,
   141,11,3,32,89,228,96
WE 660 SE=361:RW=82:GOSUB 440:LET DOS$=B$
   :RW=87
RV 670 B$=NO$:SE=361:GOSUB 440
NB 680 FOR J=361+NS TO 368+NS:SE=J:B$=NO$
   :GOSUB 440
GY 690 NEXT J
AG 700 SE=361+SN:B$=DOS$:GOSUB 440
EM 710 B$=NO$:FOR A=1 TO 128 STEP 16:B$(A
   ,A)="B":NEXT A
SY 720 B$(6,16)="This Disk  ":B$(22,32)="
   has been   ":B$(38,48)="PROTECTED  ":B
   $(54,64)="By: P/W!!!  "
FI 730 B$(70,80)="──────────":B$(86,96)=
   "(c) 1985   ":B$(102,112)="By: ANALOG
    ":B$(118)="COMPUTING  "
ND 740 SE=361:GOSUB 440
KY 750 B$(6,16)="├──────────┤":B$(22,32)="
   Program by:":B$(38,48)="J. Ehninger":B
   $(54,64)="          "
VJ 760 B$(70,80)="  PLEASE   ":B$(86,96)=
   "  REBOOT   ":B$(102,128)="
   ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
NT 770 SE=362:GOSUB 440
KI 780 B$=DOS$:B$(1,1)="▽":SE=361+NS:GOSU
   B 440
LV 790 SE=1:RW=82:GOSUB 440:RW=87:B$(2,2)
   =CHR$(7):B$(8,9)="▽/":GOSUB 440:GOSUB
   830
HZ 800 ? "OK!  YOU ARE ALL READY!  REMEMB
   ER":? "YOUR PASSWORD: ";PW$:?
ZP 810 ? "PRESS RETURN TO BOOT P/W DISK:"
   ;:GET #3,D:IF D<>155 THEN 810
RW 820 ANALOG=USR(58487)
UZ 830 SE=360:RW=82:GOSUB 440
AD 840 A=56+(NS/8):ME$="♥♥":B$(A,A+1)=ME$
BY 850 B$(4,4)=CHR$(ASC(B$(4,4))-16)
XZ 860 SE=360:RW=87:GOSUB 440:RETURN
XU 1000 REM ** P/W COPY+ VERSION 4.0 **
W5 1010 REM * BY JIM EHNINGER  4/7/85 *
RX 1020 D1=PEEK(1955):D2=PEEK(1956)
WO 1040 GRAPHICS 0:SETCOLOR 2,0,0:DIM BUF
   FER$(FRE(0)-500),WOW$(30),FN$(30),KITE
   $(10)
XZ 1050 ? "KP/W COPY+ 4.0 - BY JIM EHNING
   ER":TRAP 1290
CV 1060 ? "(c) 1985 ANALOG COMPUTING":?
JR 1070 ? "}A. DOS to P/W"
OM 1080 ? "}B. P/W to DOS"
AO 1085 ? "}C. EXIT TO P/W":?
KZ 1090 ? "}SELECT:";:INPUT #2,KITE$
KC 1100 IF KITE$<"A" OR KITE$>"C" THEN ?
   "}A,B, OR C!":GOTO 1090
XQ 1110 ? "}FILE:";:INPUT #2,WOW$:FN$="D1
   :":FN$(4)=WOW$
RJ 1120 ? "INSERT SOURCE DISK, HIT RETURN
   ":INPUT #2,WOW$
QB 1125 IF SGE THEN 1150
WR 1130 IF KITE$="A" THEN POKE 1955,89:PO
   KE 1956,228
AZ 1140 IF KITE$="B" THEN POKE 1955,0:POK
   E 1956,4
NX 1150 OPEN #1,4,0,FN$
XX 1160 A$(255)=" ":BUFFER$=""
DH 1170 TRAP 1175:XIO 7,#1,4,0,A$:BUFFER$
   (LEN(BUFFER$)+1)=A$:GOTO 1170
EE 1175 IF PEEK(195)=5 THEN ? "PROGRAM TO
   O LARGE.":END
UH 1180 IF PEEK(195)<>136 THEN 1290
RC 1190 TRAP 1290:IF PEEK(856) THEN BUFFE
   R$(LEN(BUFFER$)+1)=A$(1,PEEK(856))
MN 1200 CLOSE #1
CT 1210 ? "INSERT DESTINATION, HIT RETURN
   ":INPUT #2,WOW$
PZ 1215 IF SGE THEN 1240
XG 1220 IF KITE$="B" THEN POKE 1955,89:PO
   KE 1956,228
AI 1230 IF KITE$="A" THEN POKE 1955,0:POK
   E 1956,4
QA 1240 OPEN #1,8,0,FN$
LK 1250 ? #1;BUFFER$;:CLOSE #1
VN 1260 ? "COPY+ COMPLETE!":? "(A)nother
   copy or (E)xit:";:INPUT #2,WOW$:IF WOW
   $(1,1)="A" THEN 1050
IR 1270 POKE 1955,D1:POKE 1956,D2
UZ 1280 TRAP 40000:RETURN
TW 1290 ? "COPY+ ERROR- ";PEEK(195):END
```

●

Listing 2.
Assembly listing.

```
0100  .OPT NOLIST
0110  *=$880
0120 ;
0130 ; +--------------------+
0140 ; |P/W ASSEMBLY LISTING|
0150 ; |   By: Jim Ehninger |
0160 ; |USES ASM/EDITOR (tm)|
0170 ; +--------------------+
0180 ;
0190 ; OS EQUATES!
0200 ;
0210 GETKEY  =$F6E2
0220 COLOR2  =$02C6
0230 PRINTCHR=$F6A4
0240 DOSVEC  =$07A2
0250 CHARS   =$0102
0260 BOOTDOS =$0714
0270 ;
0280  LDX #$00        ;Get rid of
0290  STX COLOR2      ;that color!
0300 PRINT            ;Read the
0310  LDA DATA,X      ;welcome
0320  BEQ STARGATE    ;message, and
0330  STX $FF         ;print it to
0340  JSR PRINTCHR    ;the screen.
0350  LDX #$10        ;put in a delay
0360 WHX              ;so it looks
0370  LDY #$00        ;neat.
0380 WHY              ;Isn't this
0390  DEY             ;just like
0400  BNE WHY         ;WarGames?
0410  DEX             ;Are we through
0420  BNE WHX         ;dilly-dallying
0430  LDX $FF         ;around?
0440  INX             ;check...
0450  JMP PRINT       ;no, keep going
0460 STARGATE         ;YES!  lets get
0470  JMP OTAY        ;started...
0480 DATA             ;Data for MSG
0490  .BYTE 125,"P/W - "
0500  .BYTE "(c) 1985 ANALOG "
0510  .BYTE "COMPUTING",155
0520  .BYTE "PROGRAM BY: "
0530  .BYTE "JIM EHNINGER",155
0540  .BYTE "PASSWORD:"
0550  .BYTE "                    "
0560  .BYTE "                    "
0570 TEMP BRK         ;Temp location
0580 NUMS BRK         ;# of chars.
0590 ;                ;
0600 OTAY             ;Put zip into
0610  LDA #$00        ;the number
0620  STA NUMS        ;count location
0630 JIMCO            ;
```

```
0640  JSR GETKEY     ;Get a key.
0650  CMP #$9B       ;is it RETURN?
0660  BEQ HEDONE     ;YES-Check P/W
0670  CMP #$7E       ;is it BACK S?
0680  BEQ GETDOWN    ;YES-Decrement!
0690  CLC            ;Clear the way!
0700  CMP #$20       ;is it < 32?
0710  BCC JIMCO      ;YES!  Branch!
0720  CMP #$7D       ;is it >124?
0730  BCS JIMCO      ;YES!  Leave!
0740  STA TEMP       ;It's OK.
0750  CLC            ;Kill the FLAG!
0760  LDA NUMS       ;Get the amount
0770  CMP #$14       ;is it over 20?
0780  BCC LESS20     ;if not, branch
0790  LDA #$FD       ;YES!  Scream
0800  JSR PRINTCHR   ;at him! CTRL-2
0810  JMP JIMCO      ;Another key...
0820 LESS20          ;It is ok, add
0830  LDA TEMP       ;the key onto
0840  LDX NUMS       ;the list of
0850  STA CHARS,X    ;keys already
0860  JSR PRINTCHR   ;entered, print
0870  INC NUMS       ;it, and INC!
0880  JMP JIMCO      ;get another...
0890 GETDOWN         ;Pressed BACK S
0900  LDA NUMS       ;Is he < 0?
0910  BEQ JIMCO      ;YES! get key..
0920  DEC NUMS       ;Down baby!
0930  LDA #$7E       ;Erase the
0940  JSR PRINTCHR   ;mistake.
0950  JMP JIMCO      ;get key......
0960 HISPW           ;His Password!
0970  .BYTE "                    "
0980 HEDONE          ;Pressed RETURN
0990  LDA #$9B       ;Erase all the
1000  LDX NUMS       ;old entries
1010  STA CHARS,X    ;he wanted
1020  JSR PRINTCHR   ;erased.
1030  LDX #$00       ;Decode the
1040 CHANGE          ;message so we
1050  CLC            ;can decipher
1060  ROR HISPW,X    ;what is trying
1070  INX            ;to say!
1080  CPX #$15       ;We through?
1090  BNE CHANGE     ;No-keep going!
1100  LDX #$00       ;Now let's see
1110 OUTTA           ;if the man
1120  LDA HISPW,X    ;at the keys
1130  CMP CHARS,X    ;won the prize!
1140  BNE LATER      ;if not, later.
1150  INX            ;keep going...
1160  JMP OUTTA      ;Check it!
1170 LATER           ;Let's see
1180  LDA HISPW,X    ;where we ended
1190  CMP #$FF       ;up.  Is it FF?
1200  BNE YELL       ;NO!  A FAKE!
1210  LDA CHARS,X    ;mabye so..
1220  CMP #$9B       ;The RETURN!
1230  BNE YELL       ;Impersonator!
1240  JMP ITSHIM     ;YES! Welcome!
1250 YELL            ;Allright, so
1260  LDX #$00       ;you are the
1270 SCREAM          ;intruder?
1280  LDA BALLOUT,X  ;Scream at him!
1290  BEQ YELL       ;Shoot him at
1300  STX $FF        ;dawn!  Attack
1310  JSR PRINTCHR   ;men! Two ARMS,
1320  LDX $FF        ;Two Legs!
1330  INX            ;SCREAM AT HIM!
1340  JMP SCREAM     ;Don't let him
1350 BALLOUT         ;escape!!!
1360  .BYTE 155,253,126
1370  .BYTE "PASSWORD "
1380  .BYTE "DENIED!",155,0
1390 ITSHIM          ;Hey guy!
1400  LDX #$00       ;Let's insure
1410 HELLO           ;him by telling
1420  LDA WELCOME,X  ;him we are
1430  BEQ BOOTDOS    ;booting up the
1440  STX $FF        ;Disk.
1450  JSR PRINTCHR   ;He is waiting
1460  LDX $FF        ;for the
1470  INX            ;READY prompt..
1480  JMP HELLO      ;Keep printing!
1490 WELCOME         ;The data:
1500  .BYTE 155,"Booting "
1510  .BYTE "DOS...",155,0
1520 BOOTDOS         ;Change DOS to
1530  LDA #$00       ;jump to our
1540  STA DOSVEC+1   ;routine every
1550  LDA #$04       ;time!
1560  STA DOSVEC     ;And we boot!
1570  JMP BOOTDOS    ;L8R days...
1580                 ;Modems call:
1590  END.           ;(801) 272-1518
```

●

Dump1020

Dump graphics screens to your 1020 plotter.

by Donald E. Glover

When I purchased my Atari 1020 printer/plotter, I was disappointed that no program was provided to plot on paper screens drawn in the standard Atari graphics modes. The **Dump1020** gives you such a screen dump routine (DUMP1020), written in BASIC. This article includes instructions for the program and examples of its use with programs published in **ANALOG Computing**.

Program logic.

(1) The graphics mode is determined by a variable set by the user.

(2) The user is given the option of just outlining (fast) or completely filling (slower) pixels not set to background color (color 0).

(3) An appropriate frame is drawn.

(4) Each pixel of each row on the screen is tested for a non-background color. If a non-background color is found, the pixel is outlined or filled on paper, depending on the decision made in (2), above. Note that the screen is completely scanned once for each color. This is more efficient than scanning the screen once and checking for all three colors, because of the length of time required to change pens.

A BASIC screen dump.

After entering **Dump1020** (Listing 1), it should be LISTed to disk or tape. It can then be merged with a main program by using the BASIC command, *ENTER*. **Dump1020**, by starting at Line 32000 and using variable names which begin with ZZ, is designed to be merged into most BASIC programs without conflict. (**Dump1020** can be easily renumbered by a renumber utility, or the RENUM command in BASIC XL, if the main program has line numbers within its range.)

The picture on the TV screen is plotted on paper by calling the plotting subroutine (GOSUB 32013) from the appropriate part of the main program. Before you make this call, the variable ZZFILL must be set to *1* if you desire to fill the pixels, or *0* if you do not. Next ZZMODE should be set to a legal BASIC screen mode (*3-8, 19-24, 15* or *31*). Note that mode 15(7+) is supported, even on non-XL computers (see the example below).

Finally, the initialization subroutine (GOSUB 32087) must be called prior to the plotting subroutine. This call should be made at the beginning of the main program. If the initialization routine is called at some other point, it may move BASIC arrays which the main program assumes are fixed. This could cause problems, if the main program is to continue running after the plot has finished.

Examples.

The BASIC version of **Dump1020** is demonstrated using the program **Space Assault**, found in issue 13. After you've loaded **Space Assault**, add the following lines:

```
1050 GOSUB 32087
1395 ZZMODE=7+16:ZZFILL=1:GOSUB 32013
```

Now, merge **Dump1020** with the **Space Assault** program by using the ENTER command, then run the program. When the joystick trigger is pressed to shoot an enemy ship, the screen (including the "fission beam") will be fro-

zen and dumped to the 1020 plotter. The player shapes will, of course, not be plotted.

The program from the article **Graphics 7+ Handler** (from issue 11 of **ANALOG Computing** is used to demonstrate **Dump1020** in graphics mode 15. After loading that program, add the following lines:

```
10 GOSUB 32087
410 IF PEEK(764)<>255 THEN ZZFILL=1:ZZ
MODE=15+16:GOSUB 32013:END
```

Again, merge **Dump1020** with this program and run it. When you wish to plot the display on paper, hit the SPACE BAR.

Speeding things up.

BASIC is slow. For example, it will take four to five minutes before the first pixel is plotted for **Space Assault**. However, if the main program doesn't use Lines 0 through 8 (or if Lines 0-8 are just comments and can be deleted), the following modifications to **Dump1020** will make the plotting subroutine run much faster.

(1) Renumber Lines 32024 to 32030 of the plotting subroutine to 1 through 7.

(2) Change the new Line 2 to:

```
2 IF IF ZZMODE=15 THEN GOSUB 32073
:GOTO 4
```

(3) Add the following lines:

```
0 GOTO (first line of main program
-1000 for SPACE ASSAULT)
8 RETURN
32024 GOSUB 1
```

(4) Delete Lines 32025 through 32030.

The program, as modified here, works in a way identical to the original. However, the modification moves the most often executed inner loop of **Dump1020**'s three nested FOR loops to the beginning, which greatly speeds up execution. (With BASIC XL from OSS, the above modifications are unnecessary. Simply run the program in the FAST mode.)

Program enhancements.

These routines can be added to any program using standard graphics modes, but beware of programs that start with a standard display list, then change it. An interesting modification would have to be **Dump1020** switch graphics modes and screen memory locations as dictated by the commands in such nonstandard display lists.

Also, if more pen colors were available, the program could easily be modified to work in graphics modes 9, 10 and 11. The program would have to stop every four colors, to allow the pens to be changed.

Finally, **Dump1020** could be expanded to work in graphics modes 1 and 2. This would require using the data in screen memory as pointers into the character memory. With these modifications, the program could be turned into a generalized plotting routine, which would plot virtually any display created on the Atari.

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
UK 32000 REM ****************************
XV 32001 REM *                          *
GT 32002 REM *        DUMP1020          *
HY 32003 REM *    BY DONALD GLOVER      *
YK 32004 REM *                          *
VJ 32005 REM ****************************
FH 32006 REM ZZFILL=1--FILL PIXELS
QA 32007 REM ZZFILL=0--NO FILL
NE 32008 REM ZZMODE=3,4,5,6,7,8,15 OR THE
   SE MODES+16
PE 32009 REM MUST INITIALIZE BEFORE CALLI
   NG MAIN PROGRAM USING GOSUB 32082
QT 32010 REM ZZO,ZZN AND ZZS ARE THE SAME
    AS OLDCOLOR,NEWCOLOR AND SCREENCOLOR
   IN ACTION! PROGRAM
ON 32011 REM ALL OTHER VARIABLES ARE THE
   SAME AS THE ACTION! VARIABLES PREFIXED
    BY ZZ
DZ 32012 REM EXCEPT FOR ZZA-ZZF,ZZYINCYPO
   S AND ZZXINCXPOS WHICH WERE NEEDED TO
   HOLD INTERMEDIATE CALCULATIONS
EB 32013 REM ******MAIN PROGRAM--CALL AS
   A SUBROUTINE ***************
KA 32014 GOSUB 32105:REM SET UP SCREEN PA
   RAMETERS
PC 32015 GOSUB 32080:REM INITIALIZE PLOTT
   ER
MP 32016 IF ZZMODE=15 THEN GOSUB 32118:RE
   M GRAPHICS 15(7+) ONLY
IE 32017 GOSUB 32113:REM DRAW FRAME
HJ 32018 FOR ZZS=1 TO ZZNUMCOLORS(ZZMODE)
   -1:REM FOR ALL COLORS
XF 32019 IF ZZMODE=15 THEN POKE 89,INT(ZZ
   SCREENADD/256):POKE 88,ZZSCREENADD-256
   *PEEK(89):REM GRAPHICS 15(7+) ONLY
WS 32020 PRINT #2;"C";ZZS:REM PICK PROPER
    COLOR PEN
FK 32021 FOR ZZYPOS=0 TO ZZYMAX:REM FOR A
   LL ROWS
GT 32022 IF ZZMODE=15 THEN GOSUB 32077:GO
   TO 32024:REM GRAPHICS 15(7+)
RV 32023 LOCATE 0,ZZYPOS,ZZO
AS 32024 FOR ZZXPOS=0 TO ZZXMAX-1:REM FOR
    ALL COLUMNS EXCEPT LAST
HG 32025 IF ZZMODE=15 THEN GOSUB 32073:GO
   TO 32027:REM GRAPHICS 15(7+)
KP 32026 LOCATE ZZXPOS,ZZYPOS,ZZN
PG 32027 IF ZZN=ZZS THEN GOSUB 32041:REM
   THIS IS CURRENT COLOR
DB 32028 IF (ZZN<>ZZO) AND (ZZO=ZZS) THEN
    GOSUB 32046:REM WE NEED TO DRAW A BOX
YT 32029 ZZO=ZZN
VI 32030 NEXT ZZXPOS
TG 32031 REM NOW DO LAST COLUMN WHICH IS
   SPECIAL
DG 32032 IF ZZMODE=15 THEN GOSUB 32073:GO
   TO 32034:REM GRAPHICS 15(7+)
CE 32033 LOCATE ZZXMAX,ZZYPOS,ZZN
OV 32034 IF ZZN=ZZS THEN GOSUB 32041:REM
   THIS IS CURRENT COLOR
DE 32035 IF ZZO=ZZS OR ZZN=ZZS THEN GOSUB
    32046:REM WE NEED TO DRAW A BOX
YI 32036 ZZO=ZZN
XF 32037 NEXT ZZYPOS
BD 32038 NEXT ZZS
XN 32039 IF ZZMODE=15 THEN POKE 89,INT(ZZ
   SCREENADD/256):POKE 88,ZZSCREENADD-256
   *PEEK(89):REM GRAPHICS 15(7+) ONLY
```

```
DT 32040 RETURN
RA 32041 REM *********SUBROUTINE EXECUTED
    IF PIXEL COLOR IS CURRENT COLOR******
    **
IV 32042 ZZXINCXPOS=ZZXINC*ZZXPOS:ZZYINCY
    POS=ZZYINC*ZZYPOS
MY 32043 IF ZZXPOS=0 THEN PRINT #2;"M";0,
    ",";-ZZYINCYPOS:ZZXSTART=0
GH 32044 IF ZZO<>ZZS THEN PRINT #2;"M";ZZ
    XINCXPOS;",";-ZZYINCYPOS:ZZXSTART=ZZXI
    NC*ZZXPOS
ES 32045 RETURN
NP 32046 REM ***********SUBROUTINE TO DRA
    W BOX**************
MS 32047 ZZXINCXPOS=ZZXINC*ZZXPOS:ZZYINCY
    POS=ZZYINC*ZZYPOS:ZZA=(ZZN=ZZS):ZZB=(Z
    ZXPOS=ZZXMAX)
NW 32048 IF ZZB AND ZZA THEN GOTO 32053
LK 32049 REM NOT LAST COLUMN SO DRAW NORM
    AL BOX
YC 32050 PRINT #2;"D";ZZXINCXPOS;",";-ZZY
    INCYPOS
BM 32051 PRINT #2;"D";ZZXINCXPOS;",";-ZZY
    INCYPOS-ZZYLENGTH
FD 32052 GOTO 32056
QP 32053 REM LAST COLUMN SO DRAW SPECIAL
    BOX
UF 32054 PRINT #2;"D";ZZXINCXPOS+ZZXLENGT
    H+1;",";-ZZYINCYPOS
MF 32055 PRINT #2;"D";ZZXINCXPOS+ZZXLENGT
    H+1;",";-ZZYINCYPOS-ZZYLENGTH
EO 32056 PRINT #2;"D";ZZXSTART;",";-ZZYIN
    CYPOS-ZZYLENGTH
II 32057 PRINT #2;"D";ZZXSTART;",";-ZZYIN
    CYPOS
AQ 32058 IF ZZFILL=1 THEN GOSUB 32062:REM
     FILL PIXEL IF FLAG SET
OP 32059 PRINT #2;"M";ZZXSTART;",";-ZZYIN
    CYPOS
EB 32060 RETURN
RS 32061 REM *********SUBROUTINE TO FILL
    PIXEL**********
YJ 32062 FOR ZZLINE=0 TO ZZXINC
OS 32063 PRINT #2;"D";ZZXSTART;",";-ZZYIN
    CYPOS-ZZLINE
VR 32064 IF ZZB AND ZZA THEN GOTO 32068
KU 32065 REM NOT LAST COLUMN
UM 32066 PRINT #2;"D";ZZXINCXPOS;",";-ZZY
    INCYPOS-ZZLINE
DS 32067 GOTO 32070
TH 32068 REM LAST COLUMN
WW 32069 PRINT #2;"D";ZZXINCXPOS+ZZXLENGT
    H+1;",";-ZZYINCYPOS-ZZLINE
BR 32070 NEXT ZZLINE
EK 32071 RETURN
DE 32072 REM *********SPECIAL LOCATE ROUT
    INE FOR GRAPHICS 15(7+)***********
YO 32073 IF ZZYPOS=ZZCHANGE THEN POKE 88,
    ZZLOMOD:POKE 89,ZZHIMOD
XI 32074 LOCATE ZZXPOS,ZZYPOS-ZZCHANGE*(Z
    ZYPOS>=ZZCHANGE),ZZN
FE 32075 RETURN
OT 32076 REM *****SPECIAL LOCATE ROUTINE
    FOR GRAPHICS 15(7+)--ZZXPOS=0 ONLY****
ZI 32077 IF ZZYPOS=ZZCHANGE THEN POKE 88,
    ZZLOMOD:POKE 89,ZZHIMOD
LF 32078 LOCATE 0,ZZYPOS-ZZCHANGE*(ZZYPOS
    >=ZZCHANGE),ZZO
FY 32079 RETURN
MQ 32080 REM ********SUBROUTINE TO INITIA
    LIZE PLOTTER IN GRAPHICS MODE********
UH 32081 CLOSE #2:REM JUST TO MAKE SURE
SG 32082 OPEN #2,8,0,"P:":REM OPEN CHANNE
    L TO 1020
YK 32083 PRINT #2;CHR$(27);CHR$(7):REM PR
    INTER IN GRAPHICS MODE
FN 32084 PRINT #2;"H":PRINT #2;"I":REM HO
    ME AND INIT PLOTTER
KC 32085 PRINT #2;"M";0;",";0:REM MOVE PE
    N TO 0,0
FN 32086 RETURN
NC 32087 REM *************SUBROUTINE TO I
    NITIALIZE ARRAYS*****************
ER 32088 DIM ZZMODEXMAX(15),ZZMODEYMAX(15
    ),ZZMODEYINC(15),ZZMODEXINC(15),ZZNUMC
    OLORS(15)
OI 32089 DIM ZZADDFORFULL(15)
PC 32090 REM FILL ARRAYS
UG 32091 RESTORE 32098
XS 32092 FOR ZZTMODE=0 TO 15
KP 32093 READ ZZA,ZZB,ZZC,ZZD,ZZE,ZZF
YG 32094 ZZMODEXMAX(ZZTMODE)=ZZA:ZZMODEYI
    NC(ZZTMODE)=ZZB
DV 32095 ZZMODEYMAX(ZZTMODE)=ZZC:ZZMODEXI
    NC(ZZTMODE)=ZZD
OY 32096 ZZNUMCOLORS(ZZTMODE)=ZZE:ZZADDFO
    RFULL(ZZTMODE)=ZZF
FR 32097 NEXT ZZTMODE
WT 32098 REM DATA FOR ARRAYS
SK 32099 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0
WA 32100 DATA 39,12,19,12,4,4,79,6,39,6,2
    ,8,79,6,39,6,4,8,159,3,79,3,2,16
IH 32101 DATA 159,3,79,3,4,16,319,1,159,1
    ,2,32
RI 32102 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
    ,0,0,0
GB 32103 DATA 159,2,159,3,4,32
OZ 32104 RESTORE :RETURN
EM 32105 REM **********SUBROUTINE TO SET
    UP SCREEN PARAMETERS**************
TM 32106 IF ZZMODE>15 THEN ZZMODE=ZZMODE-
    16:ZZFULL=1:REM CHECK FOR FULL SCREEN
    MODE
LY 32107 ZZXMAX=ZZMODEXMAX(ZZMODE)
QL 32108 ZZYMAX=ZZMODEYMAX(ZZMODE)+ZZFULL
    *ZZADDFORFULL(ZZMODE)
MI 32109 ZZCHANGE=(ZZYMAX+1)/2:REM USED T
    O FIND MIDDLE OF SCREEN FOR GRAPHICS 7
    +
JH 32110 ZZXINC=ZZMODEXINC(ZZMODE):ZZYINC
    =ZZMODEYINC(ZZMODE)
MR 32111 ZZXLENGTH=ZZXINC-1:ZZYLENGTH=ZZY
    INC-1:REM LENGTH OF BOX SIDES
DU 32112 RETURN
DB 32113 REM ***********SUBROUTINE TO DRA
    W FRAME *************
FK 32114 PRINT #2;"C";0:REM BLACK PEN FOR
     FRAME
RF 32115 PRINT #2;"M";0;",";0:PRINT #2;"D
    ";ZZXMAX*ZZXINC+ZZXINC;",";0
KZ 32116 PRINT #2;"D";ZZXMAX*ZZXINC+ZZXIN
    C;",";-ZZYMAX*ZZYINC-ZZYINC:PRINT #2;"
    D";0;",";-ZZYMAX*ZZYINC-ZZYINC
LX 32117 PRINT #2;"D";0;",";0:RETURN
KJ 32118 REM *********SUBROUTINE USED TO
    FOOL BASIC LOCATE COMMAND FOR GRAPHICS
     15(7+) *************
YQ 32119 ZZSCREENADD=PEEK(88)+256*PEEK(89
    )
RS 32120 ZZMODADD=ZZSCREENADD+ZZCHANGE*40
SB 32121 ZZHIMOD=INT(ZZMODADD/256)
BK 32122 ZZLOMOD=ZZMODADD-ZZHIMOD*256
ZS 32123 IF ZZMODE=15 THEN POKE 87,7:REM
    FOOL BASIC
EI 32124 RETURN
```

ANALOG Computing Writers' Guidelines

Make sure your submission gets the attention it deserves.

Many of the following suggestions are applicable to all computer magazines. They assist us in the typesetting accuracy of your submission and in the speed of publication. **ANALOG Computing**, a monthly magazine, publishes new articles, programs and reviews concerning only Atari home computers and their related hardware and software. We have published many first-time authors, so by following these guidelines, you may soon find your article and byline in the pages of **ANALOG Computing**.

1. The upper left-hand corner of the first page should contain your name, address, telephone number and the date of your submission. *Important:* when you submit an article to us, you must indicate whether or not it is a simultaneous submission. A simultaneous submission is a photocopied manuscript submitted to more than one magazine at a time. Many magazines do not appreciate this practice (we are among them) and view any photocopied manuscripts warily. We do accept manuscripts text printed on a word processor. Your article should also be submitted on disk, along with any programs which the article requires.

2. The title of the article should be underlined, starting halfway down the first page.

3. If your article is a product review, please include the following information, in lieu of a title: the product's official name; the product's author (if available); the company producing and/or distributing it; the company's address and phone; memory or hardware requirements; and suggested retail price.

4. The following pages should be typed normally (double spaced), except that, in the upper right-hand corner, the title of the article should be prominent, along with your last name and the page number (e.g., *Disk/Jones/3*).

5. If your article has program listings of between five and twenty lines, you may include them within the text. Longer programs should be included with your article, but it is not essential. However, it *is* imperative that we have a copy of the program on disk. The disk should be labeled with the author's name and the title of the article or program.

6. It is much easier for our readers to type in your program if you use CHR$(X) values instead of cursor manipulators to format your output. In some cases, it may be necessary to include special control characters to create special displays. In these cases, control characters are allowable.

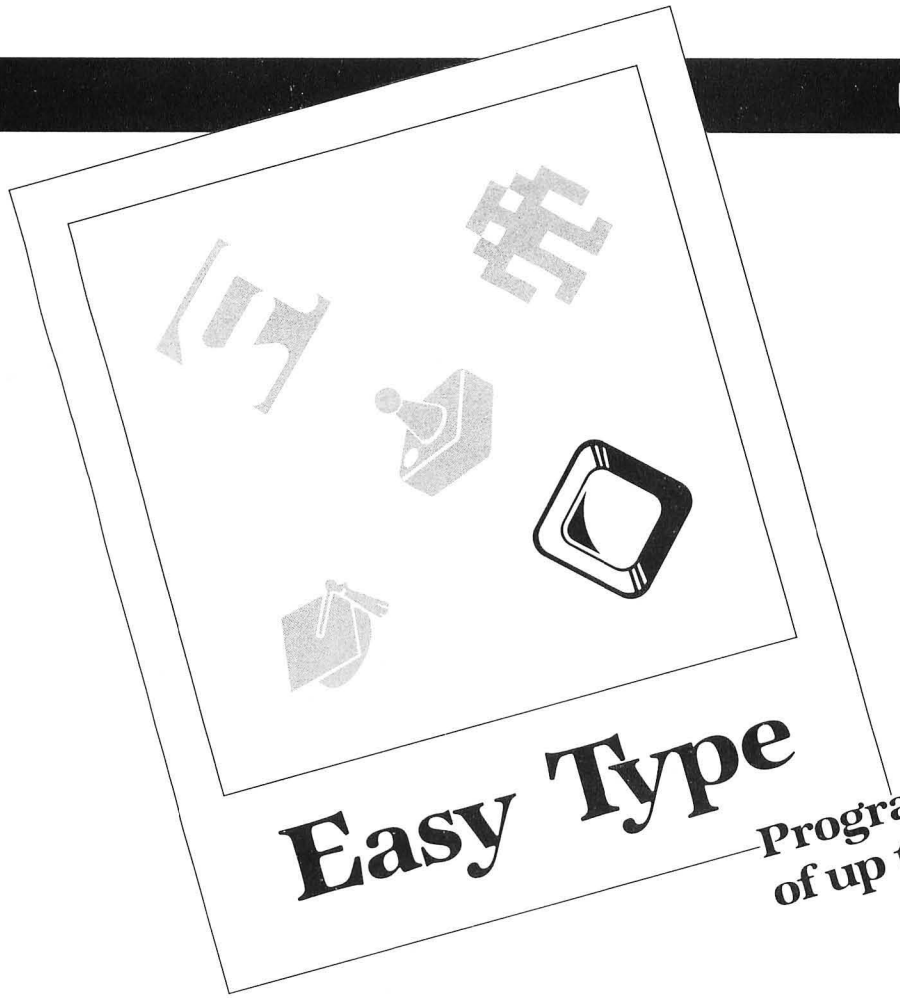
7. The printers used for **ANALOG**'s program listings will accept all Atari control, escape and inverse video characters. BASIC programs containing machine language subroutines in string variables should use DATA statements to contain the machine language numeric values. Authors should avoid using the assignment of a string variable to a complex machine language string literal. For example, *(ML$="mj+7")* could confuse readers. Authors should provide commented assembly source code listings for any machine language subroutines used in their programs. Any machine language game programs should be located in the lowest possible amount of memory.

8. Standard manuscript format—rules such as double spacing, one-inch margins all around the text, standard typing paper and typing only on one side of the paper—should be followed when submitting an article or review to **ANALOG Computing**. The pages of your submission should be paper clipped together, *not* stapled.

9. The best way to write for us is by studying previous issues. For instance, reviews of hardware and software should list the information requested in paragraph 3. Your article should be written in continuity with **ANALOG**'s style—the acronym BASIC is always all caps, as are keys like RETURN and BREAK, while names of other languages are spelled in various ways (Pascal, FORTH, assembly).

10. **ANALOG Computing** pays between $25 and $390 for published articles. The standard rate of pay is $65 per typeset page, up to six pages total (not including space taken for advertisements, art and photos). Articles over that length will be paid the flat maximum fee. If we do determine that an article may be over eight magazine pages, it may be split into two parts or sent back to the author for editing.

Send all articles, reviews or program submissions to: **ANALOG Computing Magazine,** Submissions Department, P.O. Box 23, Worcester, MA 01603.



Easy Type

Programmable keys of up to 17 characters

by Gary Heitz

Some time ago, I was typing in a magazine program. Upon checking my typing, I found several mistakes. There had been many variables used which were quite long and cryptic. Most of the typing errors occurred on these variables.

I decided I needed a program that would allow for programmable keys, to help me accurately type these listings. **Easy Type** is just that—with a few extras.

Easy Type has nine programmable keys and four programmed keys. The programmable keys are available to you, to hold a string of any characters you desire, up to 17 characters in length.

When typing, you can access the string you want by pressing the ESCAPE key and a number. The correct number is displayed on-screen, along with the contents of the string. For example, if you programmed key *1* to contain *X=USR(THR32DFP)*, then hit ESC *1*, **Easy Type** would print your string on the screen at the current cursor position. This only takes two keystrokes, and you don't have to worry about your accuracy.

Because the key number to press is displayed on-screen along with the string, all you have to remember is to type the ESC key first. If you forget, don't go hunting for this article. The screen reminds you that you should hit the ESC key and then the number.

As mentioned above, there are four programmed keys. They are ESC-0 (zero), ESC-A, ESC-I and ESC-D. The instructions for each of these are also shown on the screen.

Pressing ESC and *0* will clear the screen and display **Easy Type**'s menu.

In the menu, choice number one is "Start DATA statements." The word *DATA* will be printed after each succeeding line number entered if you select this option. You'll be asked which type of data you're going to enter: decimal, hexadecimal, or other. If you answer *decimal*, what you type will be checked by a machine language subroutine. It will see that you didn't accidentally type a letter, or any other key not appropriate to a decimal data statement. The editing keys will still function properly.

If you choose hexadecimal, the subroutine will make sure you're typing only characters acceptable to a hexadecimal data statement.

Choosing "other" will result in no checking. The word *DATA* will simply be added after each new line number.

Option number two is "Stop DATA statements." It does just that. The word *DATA* will no longer be printed after each new line number.

The third choice is "Start/Alter line numbering." If you pick this option, **Easy Type** will ask for the starting line number and the increment used between lines. Type these in, and all succeeding lines will be automatically numbered for you.

Option number four is "Stop line numbering." Choosing this will cancel the automatic line-numbering feature.

Choice number five is "Make a programmable key." You'll be asked to input the characters you desire for the next available programmable key. You don't have to enter all nine programmable keys at once. You may come back later and enter more.

Option number six is "Save program." By using this feature, you can save the entire program onto tape or disk without having to BREAK away from the program.

Menu selection number seven is "Exit menu." Use this option to leave the menu and continue typing your program.

This leaves us with three more programmed keys: ESC-A, ESC-I and ESC-D.

Let's say you type in a line and hit RETURN. You then see that you made a syntax error, or you want to change or copy part of that line. Press ESC and *A*, and the last line will be displayed. Alter the line and/or line number, then hit RETURN.

The next programmed key is ESC-I. Some magazine programs are numbered in an orderly fashion, with even increments between line numbers. Many aren't. If you've chosen menu item three (automatic line numbering) and need to skip some line numbers, type ESC-I and the next line number will appear.

In the same vein, if your increment is ten, and the program you're entering has a line number whose last digit is a five, use ESC-D. Your line number will be decremented. Hit the BACKSPACE a couple of times, hit the number 5, and be on your way.

That's about all there is. I hope you find **Easy Type** not only easy, but also a time-saving aid to accuracy. There may be several things you'd like to see added to **Easy Type**. Please alter it to your needs. The program is made to work for you—not you for it.

---

*Gary Heitz bought his first Atari computer in 1982. Through **ANALOG Computing**, he has learned to program in BASIC and assembly language.*

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 of *ANALOG Computing*.

Listing 1.
BASIC listing.

```
WK 32000 CLR :C0=0:DIM D$(15),P$(180),TEM
   P$(17),L(11),F$(15):P$="MENU":P$(5)=CH
   R$(155):FOR I=C0 TO 9
NM 32010 L(I)=C0:NEXT I:L(1)=5:D$=CHR$(15
   6):D$(15)=D$:D$(2)=D$:GRAPHICS C0:POKE
    559,C0:POKE 710,178:POKE 712,178
LP 32020 RESTORE 32591:FOR I=1 TO 133:REA
   D IN:POKE 1535+I,IN:NEXT I:IN=32190:ME
   NU=32370:GOTO 32080
AF 32030 REM *** CKTMP ***
GD 32040 INPUT TEMP$:IF TEMP$="MENU" THEN
    POP :PKEYS=PKEYS-1:GOTO MENU
MO 32050 P$(LEN(P$)+1)=TEMP$:P$(LEN(P$)+1
   )=CHR$(155):L(PKEYS+1)=LEN(P$):RETURN
PL 32060 PKEYS=PKEYS+1:IF PKEYS>9 THEN PK
   EYS=9:GOTO MENU
NF 32070 ? "ESC-";PKEYS;" ";:GOSUB 32030:
   GOTO 32060
KY 32080 REM *** CLEAR ***
KZ 32090 ? CHR$(125):POKE 82,C0:POSITION
   C0,5:? "■";CHR$(27);CHR$(156);" Progra
   mmed Keys ";CHR$(27);CHR$(156);
JB 32100 ? "      Use ESC number ":POSITIO
   N C0,C0
CA 32110 TRAP 32140:X=PKEYS:IF PKEYS>4 TH
   EN X=4
NX 32120 FOR I=C0 TO X:? I;" ";P$(L(I)+1,
   L(I+1)-1):NEXT I
YM 32130 IF PKEYS>X THEN POKE 82,20:POSIT
   ION 20,C0:FOR I=5 TO PKEYS:? I;" ";P$(
   L(I)+1,L(I+1)-1):NEXT I
FD 32140 TRAP 40000:POKE 82,2:X=USR(1590)
   :LINE=PEEK(1021)+PEEK(1022)*256
BS 32150 CLOSE #1:OPEN #1,4,C0,"K":POSITI
   ON 2,6:LIST LINE:POSITION C0,11:L=C0
SK 32160 ? "■";CHR$(27);CHR$(156);" Last
   Line ";CHR$(27);CHR$(156);"       To al
   ter, use ESC A "
BZ 32170 POSITION 2,12:? :IF NUM THEN ? S
   TART;" ";:START=START+INC
NH 32180 IF TYPE AND NUM THEN ? "DATA ";
QS 32190 REM *** IN ***
AD 32200 POKE 559,34:GET #1,KEY:IF KEY=27
    THEN 32290
PK 32210 IF TYPE AND NUM=C0 AND L=C0 AND
   KEY=32 THEN ? " DATA ";:L=1:GOTO IN
OM 32220 IF KEY=155 THEN POKE 559,C0:GOTO
    32260
AC 32230 IF KEY=28 AND PEEK(84)<14 THEN G
   OTO IN
RJ 32240 IF TYPE=1 OR TYPE=2 THEN 32580
JM 32250 ? CHR$(KEY);:GOTO IN
SO 32260 REM *** SCREEN ***
CC 32270 POSITION 2,20:? "CONT":POSITION
   C0,11:POKE 842,13:STOP
EG 32280 POKE 842,12:POSITION C0,7:? D$:G
   OTO 32140
JI 32290 REM *** ESCAPE ***
DZ 32300 GET #1,KEY:IF KEY=27 THEN ? CHR$
   (27);CHR$(27);:GOTO IN
GX 32310 IF KEY=48 THEN START=START-INC:G
   OTO MENU
RE 32320 IF KEY=73 THEN 32170
SJ 32330 IF KEY=68 THEN START=START-INC-I
   NC:GOTO 32170
RO 32340 IF KEY=65 THEN POSITION 2,12:LIS
   T LINE:POSITION 2,12:? :START=LINE+INC
   :GOTO IN
HU 32350 IF KEY<49 OR KEY>PKEYS+57 THEN ?
    CHR$(27);:GOTO 32250
YL 32360 TRAP IN:KEY=KEY-48:? P$(L(KEY)+1
   ,L(KEY+1)-1);:TRAP 40000:GOTO IN
JC 32370 REM *** MENU ***
HQ 32380 POKE 752,1:POKE 201,7:? CHR$(125
   );:IF NUM THEN POSITION C0,C0:? " Use:
   ESC I to Increment the line number";
QA 32390 IF NUM THEN ?"     ESC D to Decr
   ement the line number"
SW 32400 POSITION 17,2:? " MENU ":? :? ,"
   1 Start DATA statements"
JC 32410 ? :? ,"2 Stop DATA statements":?
    :? ,"3 Start/Alter line numbering":?
   :? ,"4 Stop line numbering"
IG 32420 ? :? ,"5 Make a programmable key
   ":? :? ,"6 Save program":? :? ,"7 Exit
    menu":POSITION 14,18
ZM 32430 ? "Your choice?":? :GET #1,KEY:K
   EY=KEY-48:IF KEY<>1 AND KEY<>2 AND KEY
   <>4 THEN POKE 752,C0
NW 32440 IF KEY=1 THEN 32540
JD 32450 IF KEY=2 THEN TYPE=C0
RE 32460 IF KEY=3 THEN ? "Enter START,INC
   REMENT";:TRAP 32460:INPUT START,INC:TR
   AP 40000:NUM=1
```

```
HE 32470 IF KEY=4 THEN NUM=C0
VF 32480 IF KEY=5 THEN ? CHR$(125):? :? "
      Type MENU to go to the MENU.":? :
   GOTO 32060
KJ 32490 IF KEY=6 THEN GOSUB 32520:CLOSE
   #1:SAVE F$:OPEN #1,4,C0,"K"
HK 32500 IF KEY=7 THEN POKE 559,C0:GOTO 3
   2080
BP 32510 GOTO MENU
SQ 32520 IF LEN(F$) THEN RETURN
GP 32530 POSITION 2,21:? "D#:Filename.Ext
   ";:INPUT F$:RETURN
RH 32540 REM **** DATA ****
ZP 32550 POSITION 2,20:? "Is the data in:
   ":? "1 Decimal","2 Hexadecimal   3 Oth
   er"
CD 32560 GET #1,KEY:TYPE=KEY-48:IF TYPE<1
    OR TYPE>3 THEN ? CHR$(253):GOTO 32550
QH 32570 H=18+(TYPE=2)*6:GOTO MENU
AR 32580 X=USR(1536,KEY,H):IF PEEK(204) T
   HEN 32250
QJ 32590 ? CHR$(253);:GOTO IN
RM 32591 DATA 104,104,104,133,203,104,104
   ,168,169,0,133,204,185,29,6,197,203,24
   0,5,136,208,246,240,4,169,255,133
FN 32592 DATA 204,96,0,28,29,30,31,126,25
   4,255,44,48,49,50,51,52,53,54,55,56,57
   ,65,66,67,68,69,70
CH 32593 DATA 169,0,170,141,0,4,141,1,4,1
   41,253,3,141,254,3,165,136,133,203,165
   ,137,133,204,160,1,177,203
CM 32594 DATA 201,125,240,46,160,0,177,20
   3,141,253,3,200,177,203,141,254,3,238,
   0,4,173,0,4,208,3,238,1
XV 32595 DATA 4,200,177,203,141,255,3,165
   ,203,24,109,255,3,133,203,144,208,230,
   204,224,0,240,202,104,96
```

Get the *Extra* on disk!

A special offer for *Extra* owners. . .

2 DISKS for only $24.95

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

Ready to *run*, on two double-sided disks.

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

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

Send for it now!

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

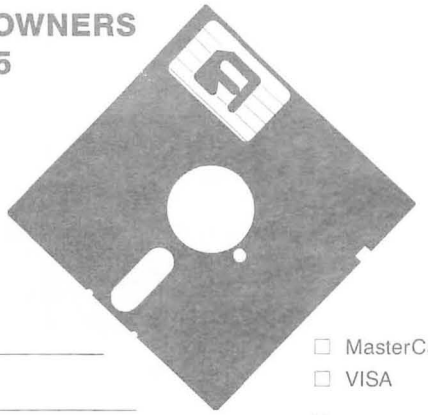
ANALOG COMPUTING

P.O. BOX 23, WORCESTER, MA 01603
(617) 892-9230

Get the Extra on disk!

A SPECIAL OFFER FOR *EXTRA* OWNERS
2 DISKS — ONLY $24.95

All the programs from **An Atari 8-Bit Extra from ANALOG Computing**, ready to run, on two double-sided disks.



Yes! Please send me the disk version.

Name _____

Street _____

City, State _____ Zip Code _____

☐ MasterCard
☐ VISA

No. _____

Exp. date _____

☐ Payment enclosed. ☐ Bill my Master Card or VISA (right). Sign _____

MAGAZINE SUBSCRIPTIONS

ANALOG ONLY	ST-LOG ONLY	ANALOG and ST-LOG
☐ 1 yr. (12 iss.) $28.00	☐ 1 yr. (12 iss.) $28.00	☐ 1 yr. (12 iss. each) . . . $42.00
☐ 2 yrs. (24 iss.) $52.00	☐ 2 yrs. (24 iss.) $52.00	☐ 2 yrs. (24 iss. each) . . $78.00
☐ 3 yrs. (36 iss.) $76.00	☐ 3 yrs. (36 iss.) $76.00	☐ 3 yrs. (36 iss. each) . $114.00
CANADIAN & MEXICAN RATES (First Class)		
☐ 1 yr. (12 iss.) $36.00	☐ 1 yr. (12 iss.) $36.00	☐ 1 yr. (12 iss. each) . . . $54.00
☐ 2 yrs. (24 iss.) $68.00	☐ 2 yrs. (24 iss.) $68.00	☐ 2 yrs. (24 iss. each) . $102.00
☐ 3 yrs. (36 iss.) $99.00	☐ 3 yrs. (36 iss.) $99.00	☐ 3 yrs. (36 iss. each) . $149.00
FOREIGN RATES (Surface)		
☐ 1 yr. (12 iss.) $39.00	☐ 1 yr. (12 iss.) $39.00	☐ 1 yr. (12 iss.) $59.00
☐ 2 yrs. (24 iss.) $72.00	☐ 2 yrs. (24 iss.) $72.00	☐ 2 yrs. (24 iss. each) . $112.00
☐ 3 yrs. (36 iss.) $104.00	☐ 3 yrs. (36 iss.) $104.00	☐ 3 yrs. (36 iss. each) . $165.00

☐ Check # _____ ☐ Bill me.
☐ Money Order # _____

☐ MasterCard ☐ VISA
Exp. date _____
Card No. _____

Name _____
Address _____
City _____ State _____ Zip _____

PLEASE ALLOW 4 to 6 WEEKS FOR DELIVERY OF FIRST ISSUE.

DISK/MAGAZINE SUBSCRIPTIONS

ANALOG DISK	ST-LOG DISK	ANALOG and ST-LOG
☐ ½ yr. (6 iss.) $59.00	☐ ½ yr. (6 iss.) $59.00	☐ ½ yr. (6 iss. each) . . . $95.00
☐ 1 yr. (12 iss.) $105.00	☐ 1 yr. (12 iss.) $105.00	☐ 1 yr. (12 iss. each) . . $159.00
CANADIAN & MEXICAN RATES (First Class)		
☐ ½ yr. (6 iss.) $69.00	☐ ½ yr. (6 iss.) $69.00	☐ ½ yr. (6 iss. each) . . $109.00
☐ 1 yr. (12 iss.) $119.00	☐ 1 yr. (12 iss.) $119.00	☐ 1 yr. (12 iss. each) . . $179.00
FOREIGN RATES (Surface)		
☐ ½ yr. (6 iss.) $74.00	☐ ½ yr. (6 iss.) $74.00	☐ ½ yr. (6 iss. each) . . $114.00
☐ 1 yr. (12 iss.) $126.00	☐ 1 yr. (12 iss.) $126.00	☐ 1 yr. (12 iss.) $189.00

☐ Check # _____ ☐ Bill me.
☐ Money Order # _____

☐ MasterCard ☐ VISA
Exp. date _____
Card No. _____

Name _____
Address _____
City _____ State _____ Zip _____

PLEASE ALLOW 4 to 6 WEEKS FOR DELIVERY OF FIRST ISSUE.

PLACE STAMP HERE
THE POST OFFICE WILL NOT DELIVER MAIL WITHOUT POSTAGE

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

P.O. Box 23
Worcester, MA 01603

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 31 WORCESTER, MA

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

P.O. Box 625
Holmes, PA 19043

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 31 WORCESTER, MA

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

P.O. Box 625
Holmes, PA 19043

When it comes to flying fighter planes

IT'S THE MAN— NOT THE MACHINE

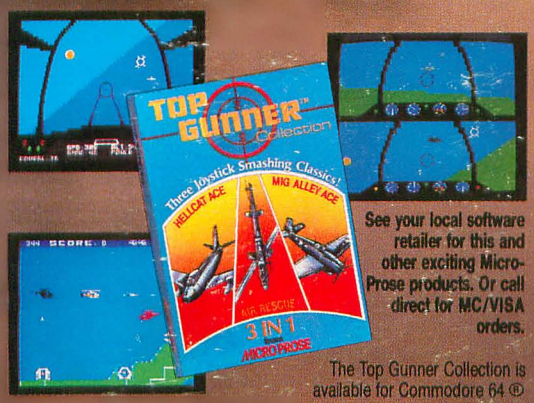
that makes the difference

TOP GUNNER™ Collection

MIG Alley Ace: Thrilling head-to-head Sabre Jet aerial dog fighting includes single player flying mode, as well as two player competition.

Hellcat Ace: Exciting 3-dimensional aerial combat over the Pacific in World War II. For one to four players.

Air Rescue: Sensational assault chopper rescue raids for daring pilots in solo or team configuration. For one to eight players.



See your local software retailer for this and other exciting MicroProse products. Or call direct for MC/VISA orders.

The Top Gunner Collection is available for Commodore 64 ® and Atari XL/XE computers at a suggested retail of only $24.95.

Do you have what it takes?

- Guts
- Stamina
- Intense concentration
- Fast reflexes
- Willingness to take risks
- Good eye-hand coordination

If so, the Top Gunner Collection prepares you for the basics in flight training and combat tactics:

- Fly solo or in team configuration
- Maneuver in 3-D space
- "See and avoid" techniques
- Advanced aerobatics: loops, rolls, G's, split S's
- Outfly the enemy

"You'll experience the wind in your face and the intense dog fighting action of some of the world's most honored and respected combat aviators: the original stick-and-rudder fighter pilots!"

MAJOR BILL STEALEY,
U.S.A.F. Reserve
President, MicroProse
(over 3,000 flying hours)

MICROPROSE
SIMULATION • SOFTWARE

120 Lakefront Drive, Hunt Valley, MD 21030 (301) 667-1151

Screen shots from Commodore 64

ISBN #0-914177-01-X