The #1 Magazine For Atari Computer Owners

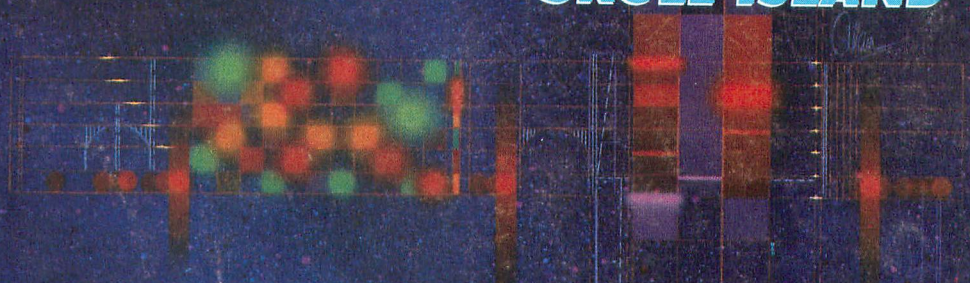
ANALOG COMPUTING™

OCTOBER 1989
ISSUE 77

USA $3.95
CANADA $4.95

SPECIAL GAMES ISSUE!

DOUBLE SIX
TX CRUNCHER
SKULL ISLAND



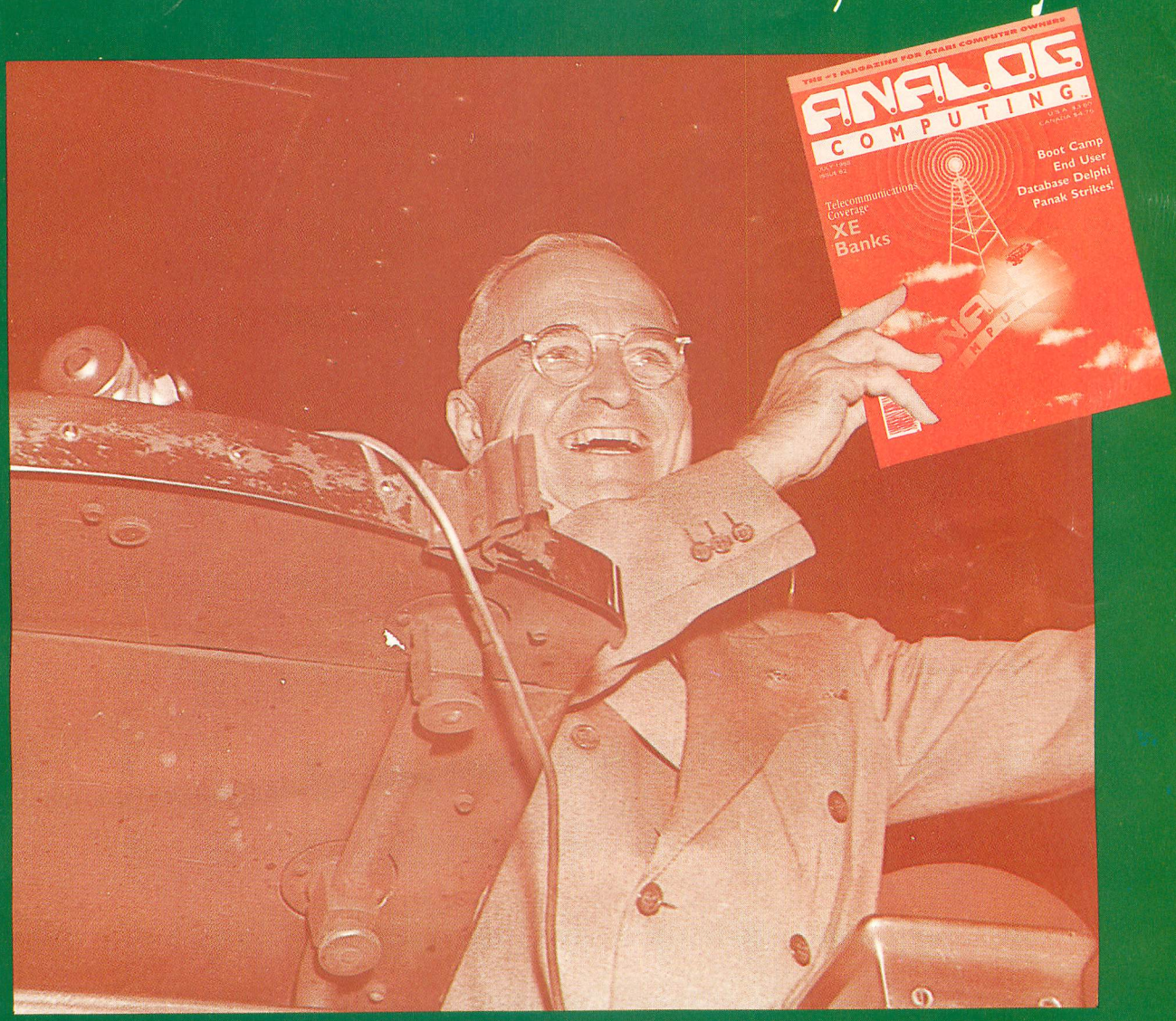
REVIEWS:

Astronauts
L.A. Swat
Panther

A VISIT WITH STAR TREK®: THE NEXT GENERATION

0 74369 50075 4

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



Two Historic Facts:

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

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

1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE!

EDITORIAL

BY CLAYTON WALNUM

As most of you know, this is the last issue of this magazine in its current form. As of next month, ANALOG Computing will be merged with ST-LOG to form a comprehensive Atari-specific publication. If you've read the publisher's letter in the previous issue, you know our reason for the merge: The U.S. Atari market is not large enough to support two Atari-specific magazines from the same publisher. Specifically, advertising, which provides an important portion of every magazine's earnings, is an at all-time low.

The publisher's letter also stated that this month we would give you more details about the new magazine. That task has fallen to me (lucky guy).

The new ANALOG Computing will be much larger than the magazine you're now holding in your hands. It will contain 132 pages, 48 of which will be in full color. A magazine of this size will give us plenty of space to cover the Atari market in full, while still providing the types of features and columns you've come to expect.

Although we'll still be offering monthly disks, both 8-bit and ST, we've decided not to provide the disk version on the newsstands. We feel that having two versions of the same magazine will be confusing to both buyers and retailers. If you're interested in obtaining the disk each month, we urge you to subscribe. Those who don't wish to subscribe will be able to order the disks by mail. We will be offering a service that will get disks out to you immediately upon the receipt of your order. In addition, we hope to be able to lower the disk price.

Little otherwise is going to change. Essentially, the merging of the magazines will give you more for your money. We will be providing complete Atari coverage in a much larger format for the same price.

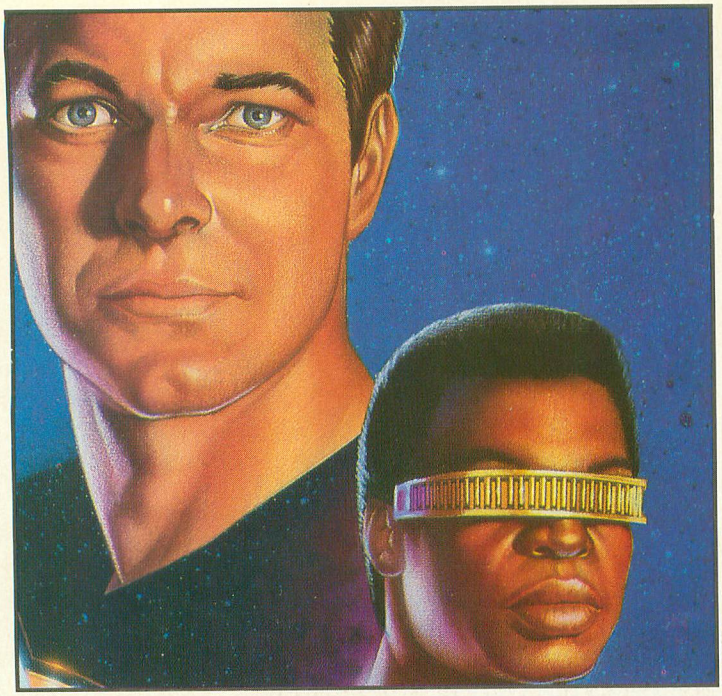
As usual, we would like to hear from you. Your input is important to us. If you have any ideas, let us know. If there's something we can do to make the new magazine better suit your needs, drop us a line. We'll give serious consideration to all your comments, and even share some of them in "Reader Comment."

As always, we at ANALOG Computing are looking forward to serving you, our readership, for many years to come.

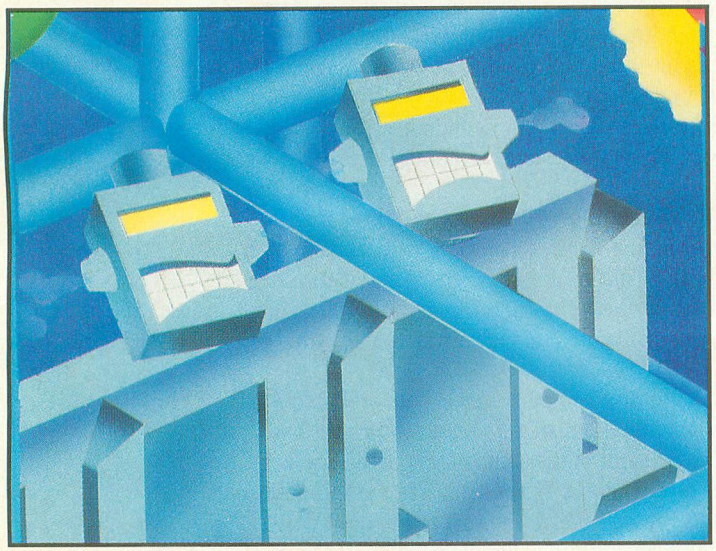
Please send all correspondence regarding this editorial to: ANALOG Computing, *P.O. Box 1413-M.O., Manchester, CT 06040.*

CONT

FEATURES



ANALOG Zooms Into the 24th Century 10



TX Cruncher 52

Skull Island 58

ENTS

OCTOBER 1989
ISSUE 77

REVIEWS

COLUMNS

DEPARTMENTS

ANALOG COMPUTING STAFF

Where to Write

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

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

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

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

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

Advertising Sales

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

Permissions

Subscriptions

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

Authors

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

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

JE Publishers Representative

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

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

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1989 L.F.P., Inc. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. All rights reserved on entire contents; nothing may be reproduced in whole or in part without written permission from the publisher. U.S. subscription: $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscription: Add $10 per year. Single copy $3.50 (add $1 for postage). Change of address: Six weeks advance notice, and both old and new addresses are needed. POSTMASTER: Send change of address to **ANALOG Computing** Magazine, P.O. Box 16927, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

THE ERROR MANUAL

by Matthew J. W. Ratcliff

The most frustrating aspect of Atari BASIC programming has got to be dealing with the language's arcane error codes. Atari BASIC is a marvel, considering all its functionality squeezed into a mere 8K of ROM, but reasonable explanations for error numbers simply couldn't fit.

As Kevin Pate explained in "Accessing Atari XL Hidden Memory" (ANALOG Computing, June 1989), XL, XE and XEGS computers have a full 8K of RAM loitering under the built-in BASIC ROM. It would be nice to stash part of our BASIC reference manual in there—the section with all the error-code explanations. This can streamline the BASIC debugging process, eliminating the need to flip through the dog-eared pages of the BASIC manual, looking for the error description and possible cause.

The Error Manual presented here loads a file called ERROR.MAN into memory under Atari BASIC and stashes an index routine into page six (1536-1791). The indexer hooks into the keyboard handler ("K:" device) and is activated with the Control-Escape "hot key." Any time your program bombs, press the hot key to see a description of the error at the top of the display. It even works while the program is running.

Typing it in

Listing 1 is the program that will create your copy of *Error Manual*. Type it in, checking your work with *BASIC Editor II* (found elsewhere in this issue), then save it to disk. When this program is run, a file named AUTORUN.SYS will be written to the disk in Drive 1. This file is the *Error Manual* program.

Once you've created the main program from Listing 1, type in Listing 2, also checking your work with *BASIC Editor II*. After saving a copy of this program to disk, run it. A file called ERROR.MAN will be written to the disk in Drive 1. This file contains all the error descriptions and is needed by *Error Manual* if it is to run properly.

Using the Program

When run, *Error Manual* first asks which drive the ERROR.MAN file is on. Press Return to accept the default Drive 1, or enter the drive number.

You may add any error descriptions you wish to the ERROR.MAN file, including status messages that may be used to provide information about a program while it is running. This is done by modifying Listing 2 and rerunning it to create a new ERROR.MAN file. Each data statement must begin with the error number and be followed by a text description of the error code, separated by a comma. The explanation of the error cannot have any embedded commas, a limitation of the Atari BASIC READ command. Use semicolons or dashes for punctuation.

Although the errors are defined in numerical order, they may be in any sequence desired because of the search algorithm used by the indexer. You can define custom error codes and explanations for unused error values of one through 255. A special message is defined for error code 255 in Line 5620, as an example. Note that the data statement for error number zero in Line 5630 must always be last, since it is used as an end-of-manual marker by the index routine.

Each description must be 38 characters or less. This is enough to provide a useful report of each error code and possible remedy. For example: 169: DISK DIR FULL;XIO#254=format.

An error 169 occurs when all of the directory entries of the disk have been used (64 files with Atari DOS). An XIO #254 command can be issued from BASIC to format a new disk. As you can see, this brief note can save you a lot of frustration; an error description requires only one line of the display with this approach. In some of the errors, part of the description begins with a question mark to indicate a possible cause of the error. Potential solutions are recommended in some messages.

The error-code report is always posted at the top line of the screen. If a graphics mode is enabled with a text window, the error code is displayed on the top line of the window. *Error Manual* does not attempt to describe an error if the cursor is on the top line (which could mangle your program, should you press return on top of an error-message line), or a graphics mode where no text window is currently enabled. If you suspect an error has occurred, break from the program and press Control-Escape again.

Atari BASIC does not always save the error codes in memory location 195, where *Error Manual* looks for an error to interpret. Add the following two lines of code to your programs to ensure that BASIC always updates location 195 for you:

```
0 TRAP 32000
32000 STOP
```

The TRAP signals Atari BASIC to save the error number for you at memory location 195. Without a TRAP, this location is not updated. You can POKE 195 with the error in question and press Control-Escape for a description.

As an example of custom error codes, suppose you have a program that processes huge amounts of data and you wish to keep tabs on its progress. The following status messages could be defined:

```
5611 DATA 175,Reading RAW data
5612 DATA 176,Performing Calculations
5613 DATA 177,Writing formatted data
```

After creating a new ERROR.MAN file with these changes, you can then read the operating status of your program without disrupting its computational flow. (No time is

wasted printing to the screen or checking for user status requests by your data processing program.) The read, calculate and write sections of code would use a single POKE to location 195 with a 175, 176 or 177. Atari BASIC saves error codes in location 195 after having processed an error TRAP. Your poking values here will have no adverse effects on the execution of the program, but will allow *Error Manual* to provide reports while the program is running.

Error Manual works with just about any Atari-compatible DOS except SpartaDOS X. Atari BASIC must be on when the program is first run. If it is disabled because you held the Option key at power-up or an external cartridge is installed, *Error Manual* detects it, issues a warning message and exits gracefully.

I did a bit of testing with the Atari Assembler Editor (Asm/Ed) cartridge, and *Error Manual* will work with it. However, Asm/Ed does not keep assembly error codes in the same location as BASIC. Adventurous programmers may wish to eliminate the external-cartridge test code and expand *Error Manual* for use with the Assembler Editor. I use MAC/65 almost exclusively, however. When *Error Manual* is allowed to bank switch ROM and BASIC RAM with MAC/65 installed, nasty things happen. It works a couple of times and then crashes the system.

A great deal of code—8K total—could be loaded into the RAM under BASIC. A short USR routine or small handler in page four or page six of memory could allow access to a massive amount of additional computing power. Since this technique works properly only while using built-in Atari BASIC, it is logical to use this space to enhance the language. *Error Manual* serves as a good example. *The Atari BASIC Source Book* from Compute! Books, by Wilkinson, O'Brien and Laughton, is an excellent tour guide as you strive to augment BASIC's power.

There is much more memory available under the operating system. However, DOS XL, Atari DOS XE, SpartaDOS and others like to use this extra RAM to give you more BASIC programming space. The 130XE memory banks and XL memory expansions can be used for such extensions too, but they are most commonly used for RAM disks. That 8K under Atari BASIC goes to waste more often than any other segment of your computer's RAM. Use the techniques of the *Error Manual* to get the most out of your machine.

```
WN 0 REM ******************************
LH 1 REM *        ERROR MANUAL        *
WJ 2 REM *         LISTING 1          *
EF 3 REM *     by Matthew Ratcliff    *
ZD 4 REM *                            *
IN 5 REM *      COPYRIGHT 1989        *
PM 6 REM *    BY ANALOG COMPUTING     *
WU 7 REM ******************************
NN 8 REM
RR 10 DIM A$(1)
RT 20 ? "PLACE DISK IN DRIVE 1,":? "THEN
   PRESS RETURN":INPUT A$
BH 30 OPEN #1,8,0,"D:AUTORUN.SYS"
QL 40 READ A:IF A<>-1 THEN PUT #1,A:GOTO
   40
UZ 50 ? "ALL DONE!":END
UA 1000 DATA 255,255,0,52,251,52,69,114,1
   14,111,114,32,77,97,110,117
IM 1010 DATA 97,108,44,32,98,121,32,77,97
   ,116,42,82,97,116,155,40
II 1020 DATA 99,41,32,49,57,56,57,44,32,6
   5,110,97,108,111,103,32
IU 1030 DATA 67,111,109,112,117,116,105,1
   10,103,155,155,27,66,65,83,73
DV 1040 DATA 67,32,105,115,32,78,79,84,32
   ,79,78,33,253,155,155,73
FY 1050 DATA 116,32,77,85,83,84,32,98,101
   ,32,111,110,32,116,111,155
LZ 1060 DATA 114,117,110,32,69,82,82,79,8
   2,32,77,97,110,117,97,108
RA 1070 DATA 155,27,67,97,110,110,111,116
   ,32,114,117,110,32,69,82,82
GP 1080 DATA 79,82,32,77,97,110,117,97,10
   8,253,155,119,104,101,110,32
TM 1090 DATA 97,110,32,69,88,84,69,82,78,
   65,76,67,65,82,84,82
KH 1100 DATA 73,68,71,69,155,105,115,32,1
   12,114,101,115,101,110,116,33
HL 1110 DATA 155,27,73,110,115,116,97,108
   ,108,97,116,105,111,110,32,111
BB 1120 DATA 102,155,69,82,82,79,82,32,77
   ,97,110,117,97,108,155,67
FO 1130 DATA 79,77,80,76,69,84,69,33,155,
   67,111,110,116,114,111,108
SY 1140 DATA 45,69,83,67,65,80,69,32,116,
   111,32,101,110,97,98,108
FJ 1150 DATA 101,46,155,68,111,32,78,79,8
   4,32,117,115,101,32,66,65
JB 1160 DATA 83,73,252,52,247,53,67,32,79
   ,70,70,32,105,110,155,83
KD 1170 DATA 112,97,114,116,97,68,79,83,4
   4,32,111,114,32,121,111,117
MG 1180 DATA 32,119,105,108,108,32,67,82,
   65,83,72,155,27,65,66,79
TT 1190 DATA 82,84,32,69,114,114,111,114,
   32,77,97,110,117,97,108,32
GU 1200 DATA 105,110,115,116,97,108,108,9
   7,116,105,111,110,155,27,69,82
RY 1210 DATA 82,79,82,46,77,65,78,32,110,
   111,116,32,102,111,117,110
MO 1220 DATA 100,155,27,67,97,110,39,116,
   32,111,112,101,110,32,100,101
XH 1230 DATA 115,116,105,110,97,116,105,1
   11,110,155,27,80,114,101,115,115
SY 1240 DATA 32,69,83,67,65,80,69,32,116,
   119,105,99,101,32,97,110
PG 1250 DATA 100,155,82,69,84,85,82,78,32
   ,116,111,32,97,98,111,114
GS 1260 DATA 116,46,155,155,87,104,97,116
   ,32,100,114,105,118,101,32,105
RP 1270 DATA 115,32,69,82,82,79,82,46,77,
   65,78,32,111,110,32,63
AJ 1280 DATA 32,91,49,93,32,27,85,110,101
   ,120,112,101,99,116,101,100
QQ 1290 DATA 32,101,114,114,111,114,32,11
   1,110,32,111,112,101,110,155,111
VR 1300 DATA 102,32,69,82,82,79,82,32,77,
   97,110,117,97,108,32,102
```

PROGRAM LISTINGS

```
GK 1310 DATA 105,108,101,46,155,27,85,110
   ,101,120,112,101,99,116,101,100
NP 1320 DATA 32,101,248,53,243,54,114,114
   ,111,114,32,111,110,32,82,69
ZX 1330 DATA 65,68,155,27,111,102,32,69,8
   2,82,79,82,32,77,97,110
PM 1340 DATA 117,97,108,32,102,105,108,10
   1,46,155,27,49,32,32,32,32
TQ 1350 DATA 32,32,32,32,32,32,68,49,58,6
   9,82,82,79,82,46,77
MH 1360 DATA 65,78,155,27,169,0,32,69,55,
   169,0,162,52,32,113,55
JV 1370 DATA 173,250,3,240,8,169,108,162,
   52,32,113,55,96,173,1,211
LH 1380 DATA 41,2,240,8,169,54,162,52,32,
   113,55,96,169,49,141,41
DE 1390 DATA 54,169,113,162,53,32,113,55,
   169,54,162,29,160,10,32,44
CA 1400 DATA 55,173,29,54,201,27,208,8,16
   9,35,162,53,32,113,55,96
FN 1410 DATA 201,155,240,11,201,49,144,21
   2,201,57,176,208,141,41,54,162
HP 1420 DATA 16,169,12,157,66,3,32,86,228
   ,162,16,169,3,157,66,3
IV 1430 DATA 169,4,157,74,3,169,40,157,68
   ,3,169,54,157,69,3,32
EX 1440 DATA 86,228,152,16,10,169,68,162,
   53,32,113,55,76,59,54,173
GV 1450 DATA 1,211,9,2,141,1,211,162,16,1
   69,7,157,66,3,169,0
QY 1460 DATA 157,68,3,169,160,157,69,3,16
   9,254,157,72,3,169,31,157
YK 1470 DATA 73,3,32,86,228,192,136,240,1
   8,169,236,162,53,32,113,55
PA 1480 DATA 162,16,244,54,156,55,169,12,
   157,66,3,32,86,228,96,162
WV 1490 DATA 16,169,12,157,66,3,32,86,228
   ,165,12,141,1,6,165,13
FC 1500 DATA 141,2,6,169,0,133,12,169,6,1
   33,13,32,3,6,169,172
CC 1510 DATA 162,52,32,113,55,173,1,211,4
   1,253,141,1,211,96,142,68
ZA 1520 DATA 3,141,69,3,140,72,3,162,0,14
   2,73,3,169,5,141,66
OB 1530 DATA 3,76,86,228,83,58,0,72,162,9
   6,169,12,157,66,3,32
KX 1540 DATA 86,228,162,96,169,3,157,66,3
   ,169,66,157,68,3,169,55
PY 1550 DATA 157,69,3,104,157,75,3,41,240
   ,73,16,9,12,157,74,3
VA 1560 DATA 76,86,228,141,68,3,142,69,3,
   133,224,134,225,160,0,140
KA 1570 DATA 73,3,177,224,201,27,240,10,2
   00,208,247,238,73,3,230,225
YW 1580 DATA 208,240,140,72,3,169,11,141,
   66,3,162,0,76,86,228,0
ZA 1590 DATA 6,233,6,32,255,255,120,173,8
   ,2,141,82,6,173,9,2
QU 1600 DATA 141,83,6,169,72,141,8,2,169,
   6,141,9,2,88,96,48
VO 1610 DATA 58,65,79,75,44,69,114,114,11
   1,114,32,77,97,110,117,97
QQ 1620 DATA 108,44,98,121,32,77,97,116,4
   2,82,97,116,155,63,58,69
DJ 1630 DATA 114,114,111,114,32,117,110,1
   00,101,102,155,72,173,9,210,201
WI 1640 DATA 156,240,4,104,76,255,255,72,
   138,72,152,72,173,1,211,9
SU 1650 DATA 2,141,1,211,165,84,240,4,165
   ,87,240,3,76,218,6,165
XF 1660 DATA 195,240,57,169,0,133,208,169
   ,160,133,209,160,0,177,208,240
VN 1670 DATA 24,197,195,240,31,230,208,20
   8,2,230,209,177,208,201,155,208
IZ 1680 DATA 244,230,208,208,232,230,209,
   208,228,169,58,133,208,169,6,133
VA 1690 DATA 209,76,177,6,230,208,208,12,
   230,209,208,8,169,28,133,208
OE 1700 DATA 169,6,133,209,160,0,177,208,
   201,155,240,20,201,32,208,4
KE 1710 DATA 169,0,240,7,201,91,176,3,56,
   233,32,145,88,200,208,230
XH 1720 DATA 192,39,240,9,169,0,145,88,20
   0,192,39,208,249,173,1,211
PB 1730 DATA 41,253,141,1,211,104,168,104
   ,170,104,104,104,64,224,2,225,2,54,54
FE 1740 DATA -1
```

LISTING 2: BASIC

```
WN 0 REM ******************************
LH 1 REM *        ERROR MANUAL        *
XI 2 REM *         LISTING 2          *
IZ 3 REM *     by MATTHEW RATCLIFF    *
ZD 4 REM *                            *
IN 5 REM *      COPYRIGHT 1989        *
PM 6 REM *    BY ANALOG COMPUTING     *
WU 7 REM ******************************
NN 8 REM
HY 10 GRAPHICS 0:DIM E$(38),BL$(38):REM M
   AX ERROR LINE = 38 BYTES
XT 15 BL$(1)=" ":BL$(38)=" ":BL$(2)=BL$
GM 20 ? "Ready to make ERROR Manual":? "P
   ress RETURN ";:INPUT E$:POKE 752,1:?
PB 30 RESTORE :TRAP 40:OPEN #1,8,0,"D:ERR
   OR.MAN":GOTO 50
YE 40 ? "CAN'T OPEN 'D:ERROR.MAN'":? "ERR
   OR ";PEEK(195);" LINE ";PEEK(186)+256*
   PEEK(187):END
YM 50 READ E$:ERNO=VAL(E$):IF ERNO=-1 THE
   N GOTO 100
MG 60 READ E$
DC 70 POSITION 2,5:? "Working on error #
   ";ERNO;"   ";:POSITION 2,6:? BL$;
QY 75 POSITION 2,6:? " :";E$;
IZ 80 PUT #1,ERNO:? #1;ERNO;":";E$
TA 90 GOTO 50
PI 100 POSITION 2,10:? "ERROR Manual comp
   lete, wrapup"
KZ 110 CLOSE #1
BM 120 POKE 752,0:? "Done!"
UT 5000 REM ERROR  MANUAL File DATA
XJ 5020 DATA 2,NOT enuf RAM for next LINE
    or DIM
ZF 5030 DATA 3,VALUE ERR; # out of expect
   ed range
KV 5040 DATA 4,TOO Many Variables; 128 MA
   X
ET 5050 DATA 5,STRING Length Error; bad i
   ndex
PT 5060 DATA 6,OUT of DATA; READ past end
CL 5070 DATA 7,NUMBR>32767 for LINE # or
   'INT'
DS 5080 DATA 8,INPUT stmnt Error; # expec
   ted
LB 5090 DATA 9,DIM Error; too BIG or re-D
   IM
CC 5100 DATA 10,ARGUMENT Stack Overflow
IL 5110 DATA 11,FLOATING Point ERR;# too
   big/small
EI 5120 DATA 12,LINE # referenced NOT FOU
   ND
KY 5130 DATA 13,NEXT with NO previous FOR
BI 5140 DATA 14,LINE Too Long; MAX length
   =120
RG 5150 DATA 15,GOSUB or FOR Line Deleted
WL 5160 DATA 16,RETURN found with NO GOSU
   B first
FJ 5170 DATA 17,'ERROR-' LINE found durin
   g RUN
HT 5180 DATA 18,BAD String Char for VAL f
   unc
```

(continued on page 50)

The Complete ATARI

12 ISSUES ONLY $28

INTRODUCING ANALOG'S NEW EXTENDED FORMAT!

128 FACT-FILLED PAGES BRINGING YOU THE LATEST FROM *BOTH THE WORLD'S OF 8-BIT AND ST!* FINALLY, ONE PUBLICATION GIVING YOU THE COMPLETE ATARI COVERAGE YOU NEED!

ANALOG COMPUTING

P.O. Box 16927
North Hollywood
CA 91615

☐ **12 Issues $28** CJWYY

GET THE NEW ANALOG COMPUTING ON DISK!

☐ **12 ISSUES OF ANALOG COMPUTING WITH 8-BIT DISK—ONLY $79!** (CDJXY)

☐ **12 ISSUES OF ANALOG COMPUTING WITH ST DISK—ONLY $79!** (CDJXW)

NAME _____

ADDRESS _____

CITY _____ *STATE* _____ *ZIP* _____

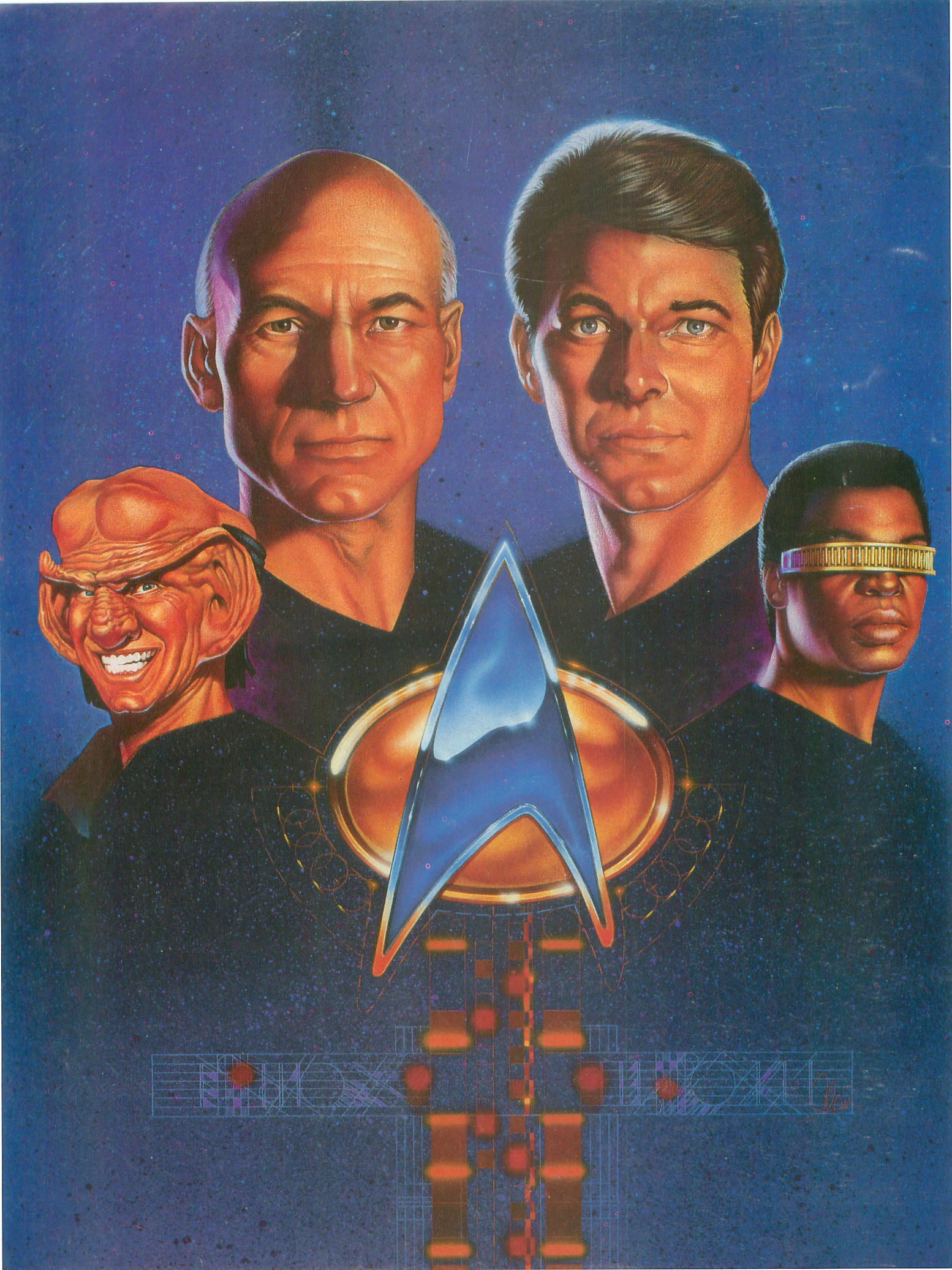
☐ *PAYMENT ENCLOSED* ☐ *BILL ME*

CHARGE MY ☐ *VISA* ☐ *MC*

_____ *EXP* _____

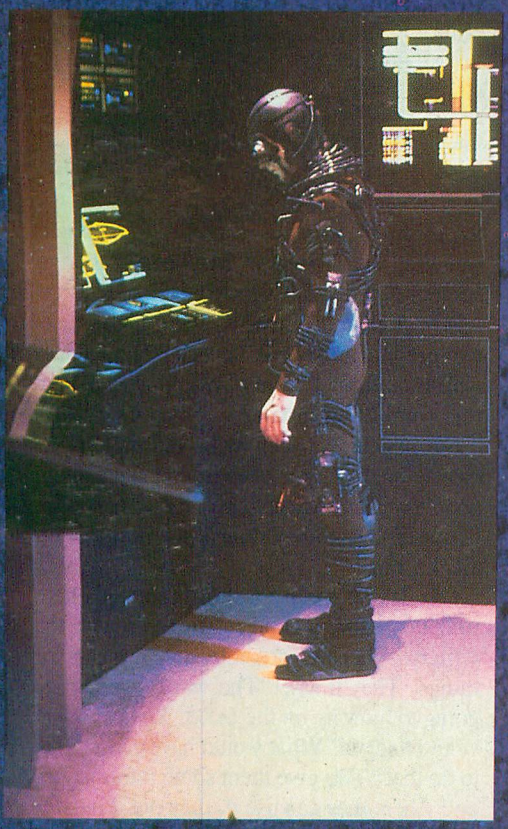
SIGNATURE _____

MONEY BACK ON UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED! FOREIGN—ADD $10. MAKE CHECKS PAYABLE TO LFP, INC. YOUR FIRST ISSUE WILL ARRIVE IN 6 TO 8 WEEKS. WATCH FOR IT! OFFER EXPIRES JANUARY 31, 1990



BY FRANK COHEN

ANALOG ZOOMS INTO THE 24TH CENTURY ON THE SET OF STAR TREK



"For I dipt into the future, far as human eye could see, saw the vision of the world and all the wonder that would be." So wrote Alfred, Lord Tennyson about the coming era of wonder. This was the 1800s, an era in which Tennyson saw the light of a new hope and prosperity. The industrial age was decades away, but technological change was inevitable.

We are now approaching another wave of wonder and awe. The fin de siécle is bringing computers into a new age of computer affinity. The days when computers were viewed as a threat are slipping away, yielding to a view of them as companions to be regarded more as helpful tools than as unknown chemical compounds.

Television has been the best reflection of the public's opinion of computers and technology. In the 50s, Commander Corey fought alien space ships equipped with highly technological-looking devices. The 60s saw crazed mainframe computers threatening to destroy the world in the *Outer Limits* television series. In the 70s we watched secret agent Max Smart defend the country with the aid of computers and, occasionally, an android. The 80s have brought computers to our homes and bombarded us with a hundred channels of cable television featuring such computer-age characters as Max Headroom.

The accuracy of many television programs has been startling. Interstellar space flight, artificial hibernation, cybernetics and artificial intelligence seem less like science fiction today partly because television has covered these futuristic issues. Television currently offers an unusual mix of new ideas and programs, with *Star Trek: The Next Generation* leading the pack. Once a week the starship *Enterprise* journeys into unknown parts of the galaxy to discover new cultures and life forms.

The Next Generation builds on the original *Star Trek* series, which premiered in the late 1960s. The story ideas and visual effects made *Star Trek* different from other science-fiction television shows. Science-fiction writers were called upon to create a world that was believable, an extrapolation of the period's technology three hundred years into the future.

Twenty years after the original show was cancelled, *The Next Generation* unites a new

cast to cover morality and ethics issues once a week. The audience gets a glimpse at what the world of the future looks like, and the results are impressive.

In two seasons of new episodes, we have seen the crew of the *Enterprise* struggle with computer viruses, computer pirates and advanced technology. Many of the scripts deal with today's problems, projected into the *Enterprise* scenario. Written by futurists, the scripts are believable.

Recently, ANALOG Computing was given a chance to discuss the world of *Star Trek* with two of the artists who bring the show's technology alive every week: Mike Okuda, graphic designer, and Rick Sternbach, senior illustrator. Although their principal responsibility is to make the sets, computer displays and other visual effects come to life, Okuda and Sternbach have a clear futuristic vision of technology and computers.

ANALOG: *Star Trek*'s new *Enterprise* is currently the public's most visible computer terminal of the future. What went into the design of the computer interfaces of the *Enterprise*?

Okuda: What we wanted to project was the idea that all the [computer] displays were software definable. For example, when the ship is in warp drive, the displays show different information than when the ship is orbiting a planet. The same display acts and reacts depending on the situation or context of usage. We very deliberately didn't push the technology of the computers we use. We couldn't afford to see a great deal of control reconfiguration on camera.

ANALOG: When you talk about not being able to afford something, I take it you mean budgetwise for filming the effect.

Okuda: Yes. Some sets have working computers built into them. For the most part, if we need a display screen to do a specific thing, we can rig it to do that particular thing. It might be a computer panel turning on or on object lighting up. But for the most part you can really see the technology working, not because of the action on a panel, but because of the way the actor works with it.

ANALOG: Are the voice-activated computer panels designed to look different from the others?

Okuda: Not really, because, theoretically, each panel is a general-purpose terminal that just happens to be currently configured for a specific task. Some of them just lend themselves to being more voice-aware than others.

Sternbach: Theoretically, you could walk up to any of these panels and fly the vehicle. If you have the correct training, as a pilot, for example, you could walk up to any of these black panels and say, "Reconfigure for navigation." And just go.

ANALOG: There are a number of portable computing devices that the crew walks around with from time to time. Are these just calculators, or are they something more?

Sternbach: You could call them Pocket Cray computers. They are separate and portable. When you see an actor walking around with one of these small grey Personal Access Display Devices (PADDs), it is like a baby panel. You could be walking down a corridor or sitting in a lounge while working with your PADD; you don't always have to be standing in front of a panel. Even though the image is fairly small, you have all the resolution of a larger panel. You can pop from menu to menu.

Okuda: If you have enough memory. [laughs]

Sternbach: Whenever you see a panel [portable or not], they pretty much all do the same thing.

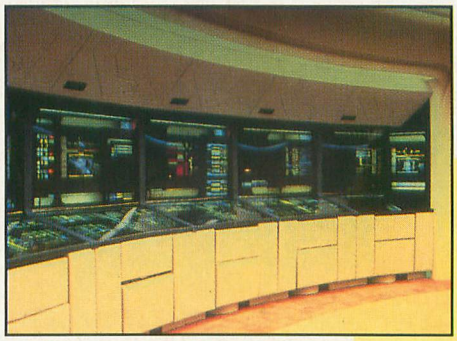
ANALOG: Have you given any consideration to the type of operating system that would make it all work?

Okuda: The operating system is the Library Computer Access Retrieval System, the version is 40273.

ANALOG: The *Enterprise* appears to be the flagship of the fleet. Is the operating system available off the ship to the common person?

Sternbach: Oh, gosh no. This was produced by Star Fleet Research and Development.

Okuda: That's true; however, one would assume that eventually this standard technology would absolutely be available to the average person. But you should remember that the average person wouldn't need all the add-on modules for warp [drive] field regulation or other ship maintenance.

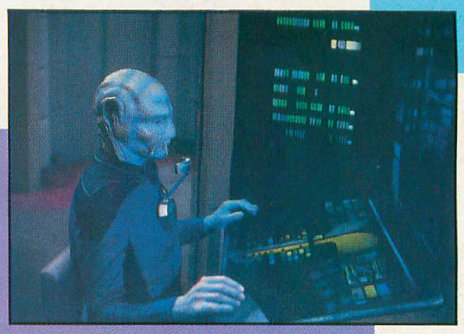


ANALOG: Do you envision the *Star Trek* world to be like *1984*, with Big Brother computer looking over your shoulder?

Okuda: I very much believe in the power of the personal computer to get people working and productive. I try to project, in subtle little ways, that even though the technology is advanced, it is accessible by everyone. For example, one of the scripts had a crew member looking at a personal document. In the upper-left corner of the screen I drew in red letters, "Personal Information, Restricted Access." The idea was that even though computers are everywhere, the people who run the computers respect the privacy of the individual.

Sternbach: The downside to all this widespread use of computers can be seen today. Look at some companies monitoring telephone operators to determine their productivity. It stifles creativity and puts a lot of pressure on the workers. We have almost the opposite here in the future. You're free to design things [with the computer] that will be useful and exciting. Using computers as tools lets you build an exciting future.

One of the reasons Mike and I have been



slaving over this show is that *Star Trek* did for us what the show did for us originally 27 years ago. It was a spark. We thought, "Wow, maybe we can do that out in space!" We try to prompt that kind of response from people watching the show now. Maybe we can give someone else that little kick of imagination.

ANALOG: It is really refreshing to hear two graphic artists talk in such technical and well-thought-out terms. Would you expect this from other artists?

Sternbach: Between Mike and myself, we have enough science background and enough people we can contact to give the writers what they need. And they do ask for information all the time.

Okuda: We are in a fortunate situation here in that we have a good relationship with the writers. Whenever possible, they'll ask us for things. They'll say, "The warp engines are going to blow up on the *U.S.S. Yamato*. What would happen? What would have caused that to do that?" We give them some information, and where they can use it, they put it into the story.

Sternbach: Another part of this is that we didn't come from the traditional Hollywood background. I mean, we didn't just come off a detective show. We came from science to *Star Trek*.

ANALOG: The mission of the *Enterprise* seems somewhat military in nature. Has that affected your design of the technology?

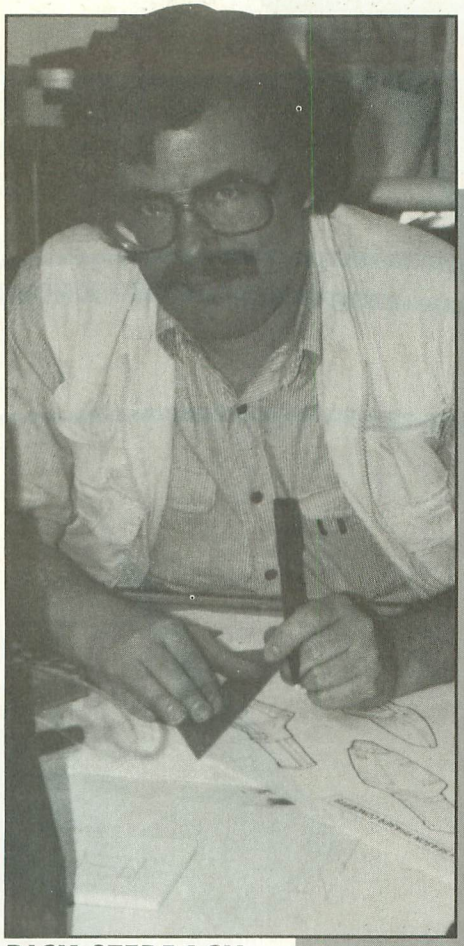
Okuda: The typical crew member is hired onboard because he is responsive, resourceful and technically able. We're not trying to say that in the 24th century everyone has to learn to use a computer. You won't find messages saying "Don't use DELETE **." There is a difference between the crew members on the *Enterprise* and everyone else. The crew members are creative, highly skilled professionals who have been sent off to do a bunch of assignments. The buttons on our panels are tools; they are not mind-controlling directors.

Sternbach: The people who contribute to the running of the ship are not being judged on their productivity—for example, how many buttons they push in a day. It's based on whether they can do their jobs.

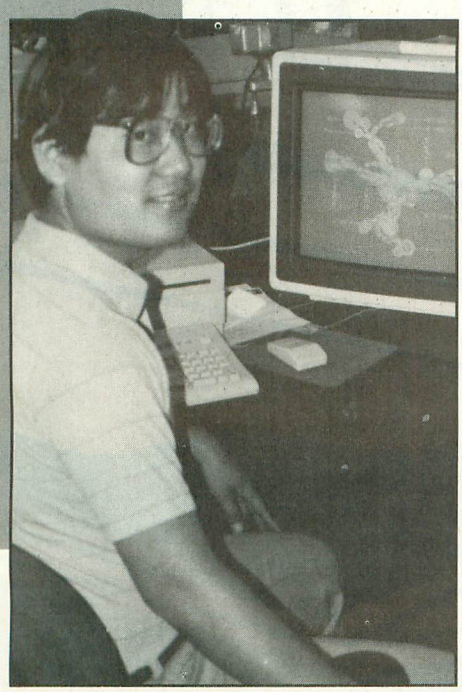
ANALOG: What about situations other than their decision-making, on which the crew members could be judged. For example, in consumer situations, what happens when they want to buy something? Are there any other uses for the panels, other than maintaining the ship?

Sternbach: I don't know. Do they have money in the 24th century?

Okuda: There is presumably some sort of tracking of resources. But [the people of the 24th century] are by no means as success-oriented as we are today.



RICK STERBACH



MIKE OKUDA

ANALOG: Can the computers be thought-driven?

Okuda: We've abandoned the idea of that kind of personal control. Some of the alien cultures the *Enterprise* encounters will have mind-directed devices, but the *Enterprise* panels are manually oriented.

Sternbach: That will surely happen in the future, but today it's awfully hard to get that across to the audience.

Okuda: On television, that would translate into a lot of voiceovers. At this point, many of our ideas are constrained to the television program. For example, we have a lot more buttons than I really think we would have this far into the future. Our first concept of a control panel had 12 buttons on it, which didn't go over very well. You have to consider what a panel looks like to the audience. Twelve buttons are probably all you might need, but it doesn't look interesting visually.

ANALOG: For that matter, doesn't it seem like having 1,000 people on a space ship is unnecessary? If you only need 12 buttons on a panel, why 1,000 crew members on the *Enterprise*?

Sternbach: These thousand-plus people have a reason for being here; they're not all here to run the ship. They are here to run scientific experiments, some are en route to somewhere else, and there are people of various disciplines.

Imagine packing all the support people necessary to run the Space Shuttle into the Shuttle craft—all the people back in Houston supporting the ship. In Houston you have science-support people and engineers. Beyond the orbiter itself, you really do need more than five people to make it work. The *Enterprise* uses everyone it needs to support the ship, but it brings all of them along for the ride.

ANALOG: Back to the idea of buttons, some new Buick cars come with a touch-sensitive video screen to set the radio, air-conditioning and trip indicators. They use a CRT coated with touch-sensitive materials. It's crude, but it exists today. Where do you see this technology going?

Sternbach: We started with that kind of idea, only took it into the future and made it better.

Okuda: You've got your basic buttons. These each control a function or process. You also have organizational controls that split the panel into areas. We also have these small joystick panels. We are specifically not supposed to use them very often. [As a user] I shouldn't have to aim a pointer at deck 27 to get information from the display. I should be able to touch the screen or ask it for information. But sometimes you might want to do something out of the ordinary that requires this sort of control.

Sternbach: What happens if your user isn't humanoid? In your Ammonia helmet, you might not be able to talk to the panel.

ANALOG: All of the panels seem to have accompanying sounds. When an actor hits a button, you hear a small tone. Was this something you developed?

Okuda: All of the sounds were created by the post-production company. Many times you can take something that looks exciting, but by adding those little beeps, it brings it to life much more than if we had spent money to rig a control to work with an actor.

Sternbach: I think [the sounds] are great. There have been times when an actor goes to push a button and it doesn't work and you hear a "wrong" sound. The idea that a button can reject its user is great.

Okuda: There is something more than just a touch matrix designed into the material of the panel. In addition to the touch matrix and the display matrix, there are programmable transponders built in there. When you hit a

(continued on page 26)

KEEPING YOUR ATARI BUSY

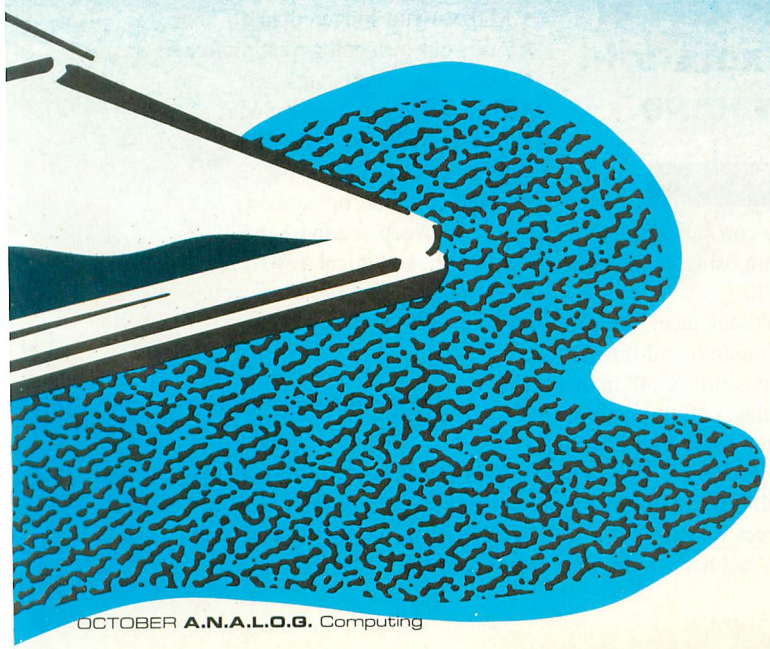
by Reid Brockway

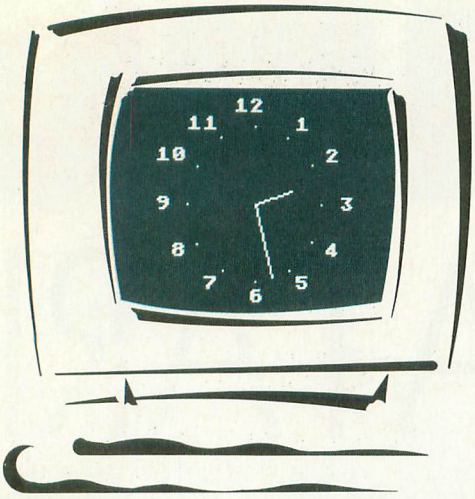
What do you do with your Atari when you're not using it to compute? Do you turn it off? Do you leave it on, announcing patiently to the world that it is "READY"? Do you load up a game and leave it endlessly playing itself in the demo mode? Well, here's something more practical for your Atari to do when it has time on its hands: Turn it into a clock.

"Right," you say. "Just what I need, another digital clock."

Guess again. This is an *analog* clock, with hands and tick-tocks and everything. Actually, it only *looks* like an analog clock; you can't get much more digital than a digital computer. So I guess you'd have to say it's a digital analog clock.

Besides being a fun novelty, this program illustrates several useful techniques you may wish to use in other programs. (See "What Makes it Tick?" below.) And there are several ways you can enhance the program if you want to. Plus, this clock has a special feature: a metric version. If you select the metric option when you start your clock, you'll get a metric readout.





Building and Running Your Clock

To make your digital analog clock, simply type in the BASIC program (and don't forget to check your typing with BASIC Editor II), save it, then type RUN. The program will ask you for the current time in hours and minutes, separated by a comma. (I know, a colon is proper. I got lazy.) It will take anything, even negative time. The clock then winds and displays itself and starts keeping time, synchronized to the power line just like any appliance clock. Its ticks and tocks are exactly one half-second apart, and it updates its hands every ten seconds. The color-cycling Attract mode is disabled, so you can leave your clock running if you want to, although it would be a good idea to turn the brightness down on your TV if you do. (One of the suggested enhancements is to make the colors rotate through various pleasing combinations—your own Attract mode.)

What Makes it Tick?

The program incorporates several useful techniques you may want to use in other programs. These include generating characters in a graphics mode, measuring time accurately, calculating and plotting positions around a circle and generating sounds with controlled attack, peak and decay times.

Line 120 reads a machine-language routine into page-six RAM. This routine is used to generate the ticks, tocks and winding sound and is an example program straight out of Appendix 9 of the Atari Assembler/Editor manual. It is called with the BASIC USR function as follows:

X = USR(1536,F,A,P,D)

where F controls the frequency of the note you want, A is the attack time, P is the plateau time (loudest portion) and D is the decay time. If you have the manual you can study the details, or you can simply plug the routine into your program using the contents of Lines 120 and 690-710.

FINALLY, THERE'S THE TIMEKEEPING "WORKS" OF OUR CLOCK. THE ATARI HAS THREE ADDRESSES—18, 19 AND 20—WHICH ARE REGISTERS COLLECTIVELY REFERRED TO AS THE "REAL-TIME CLOCK." THEY COUNT 1/60TH-SECOND TV FRAMES, WHICH ARE SYNCHRONIZED TO THE 60-HERTZ AC POWER IN YOUR WALLS. THESE REGISTERS ARE ZEROED BY THE PROGRAM WHEN THE CLOCK IS STARTED (LINE 330). THEN THE COUNT THEY ACCUMULATE IS USED TO UPDATE A VARIABLE REPRESENTING TIME, CALLED (OF ALL THINGS) TIME. ACTUALLY, TIME IS KEPT IN DEGREES OF ROTATION OF THE BIG HAND.

Lines 240-270 draw a clock face. The program runs in Graphics mode 6 and the center of the face is at X,Y position 83,48 in that mode. The two formulas compute points around a circle at a radius R for angles theta in degrees measured clockwise (how appropriate!) from vertical. Note that numbers on a clock are 30 degrees apart.

Since Graphics 6 is a nontext mode, displaying the numerals is a bit tricky. The subroutine that does this, Lines 600-670, is a little complicated but very handy. With a few changes it can be used in other modes as well. It finds where the Display List is located, and, from that, the location of screen RAM. Then it draws the characters of string MSG$ where you want them by copying the shape-defining information from Atari's standard character set into screen RAM. MSG$, and the desired XY coordinates of the leftmost character are supplied by the calling routine. (See Line 300.)

Finally, there's the timekeeping "works" of our clock. The Atari has three addresses—18, 19 AND 20—which are registers collectively referred to as the "real-time clock." They count 1/60th-second TV frames, which are synchronized to the 60-Hertz AC power in your walls. These registers are zeroed by the program when the clock is started (Line 300). Then the count they accumulate is used to update a variable representing time, called (of all things) TIME. Actually, TIME is kept in degrees of rotation of the big hand.

Note that when the real-time clocks register reach their maximum value, 65535, they reset to zero. This takes a little over 18 hours. Currently, when this happens our clock stops. One of the suggested enhancements is to overcome this limitation.

TIME is used to compute new positions of the hands. This is done every ten seconds (Lines 410 and 390), based on another variable, SEC, which actually is a count of seconds. SEC is also used to time the ticks and tocks. SEC is updated every half-second when 30 more counts of the real-time clock accumulate.

Those are the key design features of our clock. If you would like to add some features of your own, here are some suggestions:

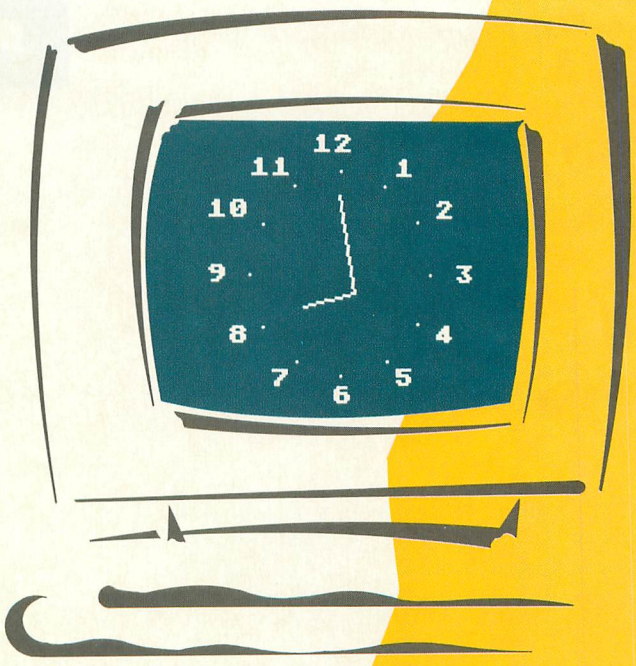
- Make it run longer than 18 hours.
- Add your own color-rotating Attract mode.
- Add a true digital readout.
- Turn it into an alarm clock.
- Add graphics to make it a mantle clock, cuckoo clock, etc.
- Make it chime or cuckoo.
- Add a sweep second-hand.

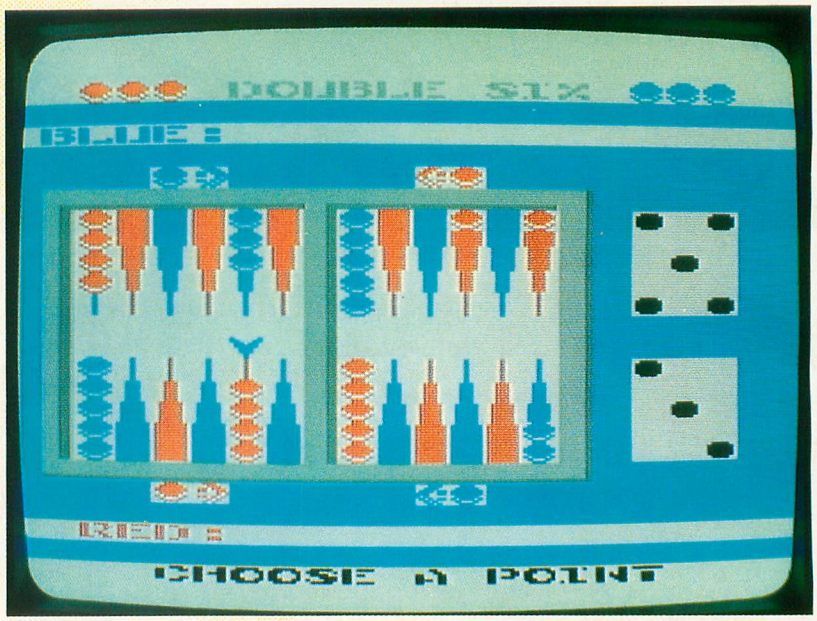
After all, you might as well keep your computer as busy as possible when it has time on its hands.

Reid Brockway is a systems and software engineer for Intermetrics, Inc., where he designs real-time software for aircraft and space applications.

LISTING 1: BASIC

```
HY 10 REM ******************************
HT 11 REM *     DIGITAL ANALOG CLOCK   *
AG 12 REM *       by Reid Brockway     *
ZW 13 REM *                            *
VT 14 REM *       COPYRIGHT 1989       *
VD 15 REM *     BY ANALOG COMPUTING    *
IK 16 REM ******************************
GG 20 DIM A$(1):INCR=30:DATLN=730:DIGITS=
   12
VY 30 ? "K":? :? "   DIGITAL CLOCK":? :?
   RUN
CS 40 ? "Tell me what time it is.":?
EM 50 ? "( hours , minutes )":?
OF 60 TRAP 50:INPUT HRS,MIN
ZP 70 ? :? "Do you want a 'metric' clock"
   ;
LS 80 INPUT A$:IF A$<>"Y" THEN 100
PO 90 INCR=36:DATLN=740:DIGITS=10
YH 100 TRAP 40000:GRAPHICS 2+16:POSITION
   1,4:? #6;"JUST A MOMENT":POSITION 1,6:
   ? #6;"WHILE I WIND MYSELF"
LX 110 REM Read in sound subroutine
JP 120 FOR I=0 TO 73:READ X:POKE 1536+I,X
   :NEXT I
SI 130 REM Make winding sound
XC 140 FOR I=1 TO 5:FOR J=1 TO 20
OR 150 A=USR(1536,30-J/5,1,1,1)
GE 160 NEXT J:FOR W=1 TO 200:NEXT W:NEXT
   I
LO 170 REM Initialization
GA 180 TIME=HRS*360+MIN*6-1
TB 190 DIM MSG$(2)
HU 200 GRAPHICS 6+16
WR 210 SETCOLOR 4,2,0:COLOR 1
RE 220 R=35:SEC=0:NXTHLF=30:LITTLE=20:BIG
   =34:HANDSFL=0:TICK=0:DEG
JN 230 REM Draw clock face
YK 240 FOR THETA=0 TO 360 STEP INCR
BJ 250 X=83+R*SIN(THETA)
BC 260 Y=48-R*COS(THETA)
QK 270 PLOT X,Y:NEXT THETA
FB 280 REM Add numerals to face
YT 290 RESTORE DATLN
NW 300 FOR N=1 TO DIGITS:MSG$=STR$(N):REA
   D X,Y:GOSUB 600:NEXT N
JG 310 GOSUB 430:GOSUB 480:REM Add hands
FT 320 REM Zero TV frame count register
EH 330 POKE 18,0:POKE 19,0:POKE 20,0
ZK 340 REM Main clock routine
LT 350 IF (PEEK(18)*65536+PEEK(19)*256+PE
   EK(20))>=NXTHLF THEN 370
VX 360 POKE 77,0:GOTO 350
TB 370 NXTHLF=NXTHLF+30
IT 380 A=USR(1536,20*(TICK+1),1,3,1)
PT 390 IF HANDSFL<>0 THEN ON TICK+1 GOSUB
    430,480
DQ 400 SEC=SEC+0.5:TICK= NOT TICK
OQ 410 IF SEC=10 THEN HANDSFL=1:SEC=0
OK 420 GOTO 350
NH 430 REM Subr. to update little hand
XN 440 T=TIME:GOSUB 550:XOLD=X:YOLD=Y
LB 450 T=TIME+1:GOSUB 550:XNEW=X:YNEW=Y
MR 460 GOSUB 570:RETURN
LN 470 REM Subroutine to update big hand
WJ 480 T=TIME:GOSUB 530:XOLD=X:YOLD=Y
JT 490 T=TIME+1:GOSUB 530:XNEW=X:YNEW=Y
VO 500 GOSUB 570
DB 510 TIME=TIME+1:HANDSFL=0:RETURN
SO 520 REM Subrs to compute hand posns.
HW 530 X=83+BIG*SIN(T):Y=48-BIG*COS(T)
ZJ 540 RETURN
HT 550 X=83+LITTLE*SIN(T/12):Y=48-LITTLE*
   COS(T/12):RETURN
RJ 560 REM Subroutine to redraw hand
IL 570 COLOR 0:PLOT 83,48:DRAWTO XOLD,YOL
   D:COLOR 1:PLOT 83,48:DRAWTO XNEW,YNEW
ZR 580 RETURN
MB 590 REM Subr. to display text in Gr.6
RP 600 DLIST=PEEK(560)+PEEK(561)*256
HK 610 SCRAM=PEEK(DLIST+4)+PEEK(DLIST+5)*
   256
WK 620 FOR I=1 TO LEN(MSG$)
XM 630 SHAPE=57344+8*(ASC(MSG$(I,I))-32)
CH 640 STARTLOC=SCRAM+X+20*Y+I-1
PG 650 FOR J=0 TO 7
FL 660 POKE STARTLOC+20*J,PEEK(SHAPE+J)
AO 670 NEXT J:NEXT I:RETURN
LH 680 REM Sound subroutine code
DV 690 DATA 104,104,104,141,0,210,104,104
   ,133,204,104,104,133,205,104,104,133,2
   06,169,160,141,1,210,166,204,32,65,6
HR 700 DATA 24,105,1,201,176,208,241,169,
   14,166,205,32,65,6,56,233,1,208,246,16
   9,175,141,1,210,166,206,32,65,6,56
UV 710 DATA 233,1,201,159,208,241,96,160,
   19,136,208,253,202,208,248,96,256
JM 720 REM Numerals position data
WJ 730 DATA 13,8,15,22,16,44,15,65,13,80,
   10,86,7,80,5,65,4,44,4,22,6,8,9,0
WU 740 DATA 14,13,16,33,16,57,14,78,10,88
   ,6,78,4,57,4,33,6,13,9,0
```





DOUBLE

by Pierre Roberge

Every computer owner has in his software collection at least one computerized "thinking game." The Atari 8-bit owner is no exception. Whether it be chess, checkers, reversi, battleship or connect four, there's surely one of these games present on your disks. There is, however, a game that's been overlooked: backgammon. But now *Double Six* is here.

Typing it in

Type Listing 1, check it with BASIC Editor II (found elsewhere in this issue) and save it. Now type NEW and type Listing 2, checking it with BASIC Editor II and saving it when it's been entered correctly. When RUN, Listing 2 will create all the missing lines from the main program that contain the control characters used in machine-language subroutines. These lines will be listed to a file called D:DOUBLE6.CHR.

Now LOAD "D:DOUBLE6.TMP", ENTER "D:DOUBLE6. CHR" and save the complete program as DOUBLE6.BAS. You are now ready to go!

Playing the Game

Double Six is a game for two players in which Joystick One moves the red men and Joystick Two moves the blue men. Each player has 15 men set on "points" around the game board. The red men move counterclockwise from the top-right section of the board to the bottom-right section (red's inner table); the blue men move clockwise from the bottom-right section to the top-right section (blue's inner table). Note the red and blue arrows that indicate the way around. The object of the game is to remove all your men from the game board by the roll of the dice. More on that later.

First you press the joystick button in order to determine who goes first. The dice will stop rolling. The blue player has the top die, the red player has the bottom one. If both players roll the same number, they must try again until one of them has rolled the higher number. That player then goes first, using the numbers shown on the two dice. The players then play in turn using the two dice. The pointer in the center of the board turns red or blue depending on which player's turn it is.

SIX

Use your joystick to place the pointer on the man you want to move, use it again to choose a die. Your man will move according to the number shown on the die. You can move the same man with the two dice as long as the points designated by each die are open, or you can play each of the two numbers with different men.

When a die has been used, an "X" will appear in front of it to indicate you cannot use it again. A player must use both numbers of each roll whenever possible. If he can use only one number, he must, if possible, use the larger. When you cannot move at all, press "P" to pass the play to your opponent.

If you roll doubles (the same number on both dice), you move according to the numbers shown on one die four times. You can move the same man all four moves, or any other combination of men you choose.

Any point on the playing board on which two or more men of the same player sit is called a blocked point. A blue man cannot land on a point blocked by red men, and vice versa. On a real board there is no limit to the number of men one player may have on a point. But since this is a "graphic" board, there is a limit of ten men per point.

Since men of opposite colors cannot occupy the same point, when a man lands on a point containing only one man of the other color, he removes the opponent's piece and takes his place. The removed man is placed on the "bar" (the middle strip that separates the board). Once one of your men has been placed on the bar, you must "enter" it into your opponent's inner table before you may move any of your other men. Entering is accomplished by moving the man on the bar to the point indicated by either of the two dice, as long as that point is not blocked. If you cannot enter because both points indicated are blocked, the turn passes to the other player. A "shutout" occurs when each point in your opponent's inner table is covered by at least two men.

You cannot start removing your men from the board until all 15 of them are in your inner table. You may then remove your men from points corresponding to the numbers rolled or you may move your men within your inner table according to the dice. You must use your entire roll, if possible. This means that if you roll a five but have no men on your fifth point (counted from the right), you must take a man from your sixth point and advance him to the first one. You cannot remove a man if the point he is on is lower than the number you've chosen.

There are a few options available during the game: <P>ass, <Q>uit and <S>ave Game. These options are self-explanatory.

Programming Notes

Double Six is written in BASIC but uses machine-language strings to speed up the initializing procedure. The game is ready to play almost instantly after typing RUN. I used the wide playfield to draw the game on a GRAPHICS 1 screen, using the unseen parts of the playfield to store information about the game. That permitted me to greatly simplify the logic of the game. Also, the program contains plenty of REM statements for those of you who want to know how it ticks.

***Pierre Roberge** is a mechanical engineer who works for a company based in Quebec City. He's been programming his 130XE for more than two years.*

LISTING 1: BASIC

```
WN 0 REM ********************************
BZ 1 REM *         DOUBLE SIX           *
JQ 2 REM *          LISTING 1           *
NZ 3 REM *      by Pierre Roberge       *
ZD 4 REM *                              *
IN 5 REM *        COPYRIGHT 1989        *
PM 6 REM *      BY ANALOG COMPUTING     *
WU 7 REM ********************************
NN 8 REM
SZ 10 GOSUB 1000:GET #N2,A:IF A<>79 THEN
   164
VP 11 REM ** LOAD GAME **
JB 12 SL=N4:RW=7:SL$="./!S":GOSUB 152:IF
   A<>89 THEN 164
UT 13 REM ** MAIN LOOP **
AK 14 W=N64*TURN:CS=32+W:CD=11+W:CSO=96-W
   :CDO=75-W:PT=28+W:POKE SC+X+Y,PT+N2*(Y
   =N288)
YX 16 IF OK THEN GOSUB 112
NO 17 REM ** CHOOSE POINT **
FT 18 TXT$="♥♥♥H(//3%♥!♥0/).4♥♥♥":GOSUB 1
   26
ZD 20 POKE 77,N0:IF PEEK(764)<>N255 THEN
   128
ZD 22 ST=PEEK(632+TURN):XA=X:YA=Y:X=X+(ST
   =7)*(X<N15)-(ST=11)*(X>N3):Y=Y+N24*((S
   T=13)*(Y=N264)-(ST=N14)*(Y=N288))
IM 24 IF X<>XA OR Y<>YA THEN POKE SC+XA+Y
   A,N194*(XA=N9)
GJ 26 CHA=PT+N2*(Y=N288):POKE SC+X+Y,CHA:
   IF PEEK(644+TURN) THEN 20
XB 28 SOUND N0,N16,N10,N14:P1=SC+N48:X1=X
   :Y1=Y:I=I*N1:SOUND N0,N0,N0,N0
RP 29 REM ** CHECK FOR ILLEGAL MOVE **
PH 30 IF (PEEK(SC+153+N264*TURN)<>N194 AN
   D X<>N9) OR (X=N9 AND Y=N288-N24*TURN)
    THEN 86
WE 31 REM ** FIND TOP MAN ON PILE **
PC 32 DP=N24*((Y=N288)-(Y=N264)):PA=SC+X+
   Y+DP*N5:P=PEEK(PA):PB=(P=CD)
OS 34 E=(INT(PA/N2)=PA/N2):C=ABS((E=N0)*(
   Y=N264)+E*(Y=N288)-(X>N9)):IF P<>CS AN
   D P<>CD THEN 86
ID 36 PA=PA-DP:P=PEEK(PA):IF P=CS-PB*N21
   THEN 36
WX 38 PA=PA+DP:P=PEEK(PA):FOR I=N1 TO N4:
   FOR J=N1 TO 20:POKE PA,P:NEXT J:FOR J=
   N1 TO 20:POKE PA,N0:NEXT J:NEXT I
WE 39 REM ** ERASE MAN FROM POINT **
OX 40 P1=PA:CH=P:POKE PA,(PEEK(PA-(PA-(IN
   T((PA-SC)/N24))*N24-SC)-C)*(X<>N9)+N19
   4*(X=N9))*(PB=N0)+CS*PB
TQ 41 REM ** DRAW MAN ON POINTER **
GW 42 POKE SC+X+Y,CS:TXT$="♥♥♥♥H(//3%♥!♥S
   )%♥♥♥♥":GOSUB 126
AB 43 REM ** CHOOSE A DIE **
PP 44 ST=PEEK(632+TURN):ZA=Z:Z=Z+N24*((ST
   =13)*(Z=N282)-(ST=N14)*(Z=N306))*(INT(
   MAX/N2)=MAX/N2)
ZH 46 IF Z<>ZA THEN POKE SC+ZA,N126:POKE
   SC+ZA+N1,N126
TI 48 POKE SC+Z,66:POKE SC+Z+N1,PT+N2*(Z=
   N306):IF PEEK(644+TURN) THEN 44
QG 50 SOUND N0,N16,N10,N14:D=D1*(Z=N282)+
   D2*(Z=N306)+N1:T=(TURN=N0)-TURN
HY 51 REM ** MOVE MAN IN INNER TABLE **
IS 52 SOUND N0,N0,N0,N0:IF X+D>N15 AND Y=
   N288-N24*TURN THEN GOSUB 104
VO 53 REM ** MOVE MAN ON BAR **
WO 54 IF X=N9 THEN X=N16-D:D=N1:GOTO 58
TV 55 REM ** WALK MAN ON BOARD **
JL 56 A=(X=N3 AND Y=N264+N24*TURN):DX=((Y
   =N288)-(Y=N264))*T* NOT A:DY=N24*T*A:X
   A=X:YA=Y:X=X+DX:X=X+DX*(X=N9):Y=Y+DY
YX 58 POKE SC+XA+YA,N194*(XA=N9):POKE SC+
   X+Y,CS:SOUND N0,N0,N0,N8:SOUND N0,N0,N
   0,N0:FOR I=N1 TO 50:NEXT I
BR 60 D=D-N1:IF D THEN 56
RT 61 REM ** CHECK FOR REMOVAL OF MAN **
AD 62 IF X=N16 THEN PA=PAH:GOTO 72
KM 63 REM ** PUT MAN ON POINT **
JX 64 DP=N24*((Y=N288)-(Y=N264)):PA=SC+X+
   Y+N4*DP:P=PEEK(PA):IF P=CSO OR P=CDO T
   HEN 86
IH 66 A=N0:PA=PA+DP:P=PEEK(PA):PB=(P=CD):
   PC= NOT (P=N2 OR P=66):IF P=CSO THEN G
   OSUB 94
AN 68 A=A+N1:PA=PA-DP*PC:P=PEEK(PA):IF P=
   CS-PB*N21 AND A<N5 THEN 68
JJ 70 IF A=N5 THEN PA=SC+X+Y+N5*DP:IF PB
   THEN 86
AC 72 POKE SC+X+Y,N194*(X=N16):FOR I=N1 T
   O N4:FOR J=N1 TO N10:POKE PA,N0:NEXT J
   :FOR J=N1 TO N10
WM 74 POKE PA,CS-N21*(PB OR A=N5)*(X<>N16
   ):NEXT J:NEXT I:POKE SC+Z+N1,184:ZB=Z:
   Z=Z+N24*((Z=N282)-(Z=N306))
YQ 75 REM ** PUT AN X ON USED DIE **
OG 76 IF INT(MAX/N2)<>MAX/N2 THEN POKE SC
   +Z,N126:POKE SC+Z+N1,N126:POKE SC+ZB,N
   126:POKE SC+ZB+N1,N126
```

```
OX 77 REM ** CHECK FOR WINNER **
BT 78 IF X=N16 THEN X=X1:IF PAH>PBH+N14 T
   HEN 136
EK 79 REM ** MEN IN INNER TABLE **
IX 80 IF X>N9 AND X1<N9 AND Y=N288-N24*TU
   RN THEN POKE SC+N24*TURN,PEEK(SC+N24*T
   URN)+N1
QS 82 MAX=MAX-N1:IF MAX THEN 18
IH 83 REM ** END OF MAIN LOOP **
FM 84 TURN= NOT TURN:POKE SC+23,TURN:OK=N
   1:GOTO 14
DJ 85 REM ** ILLEGAL MOVE ROUTINE **
KS 86 SOUND N0,N24,N6,N10:POKE SC+X+Y,N19
   4*(X=N9):POKE P1,CH:POKE SC+X1+Y1,CHA:
   POKE SC+Z,N126:POKE SC+Z+N1,N126
PL 88 FOR I=N1 TO N24:NEXT I:X=X1:Y=Y1:SO
   UND N0,N0,N0,N0
BL 90 TXT$="♥♥[illegible]♥[illegible]♥♥":GOSUB 1
   26:FOR I=N1 TO N194:NEXT I:GOTO 18
CT 91 REM ** QUIT ROUTINE **
RC 92 POKE 53277,N0:POKE 53265,N0:POKE 55
   9,34:POKE 106,PEEK(740):END
XT 93 REM ** BAR ROUTINE **
HL 94 A=N0:DPB=N24*(TURN-(TURN=N0)):PAB=S
   C+417-N264*TURN:PP=PEEK(PAB):PBB=(PP=C
   DO):PCB= NOT (PP=N194)
TM 96 A=A+N1:PAB=PAB+DPB*PCB:PP=PEEK(PAB)
   :IF PP=CSO-PBB*N21 AND A<N5 THEN 96
WF 98 IF A=N5 THEN PAB=SC+417-N264*TURN:I
   F PBB THEN POP :GOTO 86
CX 100 IF X>N9 AND Y=N264+N24*TURN THEN P
   OKE SC+ NOT TURN*N24,PEEK(SC+ NOT TURN
   *N24)-N1
ZB 102 POKE PA,N0:POKE PAB,CSO-N21*(PBB O
   R A=N5):PC=N0:RETURN
XN 103 REM ** REMOVAL ROUTINE **
DX 104 PAH=SC+510-456*TURN:PBH=PAH
AO 106 PAH=PAH+N1:IF PEEK(PAH)=CS THEN 10
   6
VN 108 IF PEEK(SC+N24*TURN)<N15 OR X+D>N1
   6 THEN POP :GOTO 86
YZ 110 RETURN
TT 111 REM ** ROLLING DICE ROUTINE **
ST 112 TXT$="[illegible]♥[illegible]♥[illegible]♥[illegible]":GOSUB
   126
JO 114 D1=INT(RND(N0)*N6):D2=INT(RND(N0)*
   N6):FOR I=N0 TO N2:A=USR(ADR(MOVE$),AD
   R(D$)+D1*N9+I*N3,SC+162+I*N48,N3)
BP 116 A=USR(ADR(MOVE$),ADR(D$)+D2*N9+I*N
   3,SC+330+I*N48,N3):NEXT I:SOUND N0,150
   ,N2,N8:SOUND N0,N0,N0,N0
GF 118 IF PEEK(644+TURN) THEN 114
KU 120 SOUND N0,N16,N10,N14:MAX=N2+N2*(D1
   =D2):POKE 764,N255:SOUND N0,N0,N0,N0
UW 122 IF PEEK(644+TURN)=N0 THEN 122
ZN 124 RETURN
PZ 125 REM ** PRINT TEXT ROUTINE **
WQ 126 A=USR(ADR(MOVE$),ADR(TXT$),SC+554,
   20):RETURN
DK 127 REM ** QUIT **
FK 128 GET #N2,A:IF A=81 THEN 92
BR 129 REM ** SAVE GAME **
NX 130 IF A=83 AND INT(MAX/N2)=MAX/N2 THE
   N SL=N8:RW=11:SL$="♥[illegible]":GOSUB 152:GOT
   O 18
XD 131 REM ** PASS **
NS 132 IF A=80 THEN SOUND N0,N14,22,N10:P
   OKE SC+ZB,N126:POKE SC+ZB+N1,N126:GOTO
    84
QE 134 GOTO 20
BL 135 REM ** WINNER ROUTINE **
ZF 136 W=SC+514-456*TURN:W$="        ":IF T
   URN THEN W$="♦♦♦♦♦♦"
JP 138 FOR I=PAH TO PAH-N14 STEP -N1:POKE
    I,N0:SOUND N0,I,N10,N10:POKE I+456*(2
   *TURN-N1),N0
YS 140 FOR J=N1 TO N16:NEXT J:NEXT I:SOUN
   D N0,N0,N0,N0
LK 142 FOR I=N1 TO N6:A=USR(ADR(MOVE$),AD
   R(W$),W,N6):FOR J=N1 TO N16:NEXT J:A=U
   SR(ADR(MOVE$),ADR("winner"),W,N6)
MI 144 FOR J=N1 TO N24:NEXT J:NEXT I
AR 146 TXT$="♥→→♥[illegible]♥[illegible]♥[illegible]♥[illegible]♥":GOSUB
   126
KY 148 GET #N2,A:IF A=89 THEN GOSUB 1172:
   OK=N0:GOTO 164
TK 150 GOTO 92
GW 151 REM ** LOAD/SAVE ROUTINE **
HX 152 TXT$="    ♥♥♥♥♥[illegible]♥[illegible]♥♥♥♥":TXT$(N
   5,N8)=SL$:GOSUB 126:TRAP 158:CLOSE #N1
   :OPEN #N1,SL,N0,"D:DOUBLE6.SCR"
TC 154 POKE SC+X+Y,N0:POKE 850,RW:POKE 85
   2,PEEK(88):POKE 853,PEEK(89):POKE 856,
   N64:POKE 857,N2
LU 156 A=USR(ADR("hhh[illegible]LV[illegible]"),N16):TURN=PEE
   K(SC+23):CLOSE #N1:OK=N0:RETURN
RV 158 SOUND N0,N64,N10,N10:TXT$="[illegible]♥[illegible]♥[illegible]♥":GOSUB 126:I=I^N1:SOUND
   N0,N0,N0,N0
LR 160 GET #N2,A:IF A=89 THEN 152
ZP 162 RETURN
SW 164 GOSUB 112
UM 165 REM ** WHO GOES FIRST ROUTINE **
HS 166 TXT$="♥♥♥♥♥♥♥♥♥♥[illegible]♥♥♥♥":IF D1=
   D2 THEN TXT$="♥♥♥[illegible]♥[illegible]♥[illegible]♥♥♥♥"
VG 168 IF D1<D2 THEN TXT$(5,8)="♥2%$":TUR
   N=N0:POKE SC+X+Y,28+N2*(Y=N288)
XX 170 IF D1>D2 THEN TXT$(5,8)="blue":TUR
   N=N1:POKE SC+X+Y,92+N2*(Y=N288)
OI 172 GOSUB 126:FOR I=N0 TO N194:NEXT I:
   IF D1=D2 THEN 164
EM 174 POKE SC+N282,N126:POKE SC+N282+N1,
   126:POKE SC+N306,N126:POKE SC+N306+N1,
   126:GOTO 14
RD 999 REM ** INITIALIZE ROUTINE **
IM 1010 N1=1:N2=2:N3=3:N4=4:N5=5:N6=6:N8=
   8:N9=9:N10=10:N14=14:N15=15:N16=16:N21
   =21:N24=24:N48=48:N64=64:N126=126
TJ 1012 N194=194:N255=255:N264=264:N282=2
   82:N288=288:N306=306:X=N3:Y=N264:Z=N28
   2
LH 1014 DIM TXT$(20),SL$(N4),W$(N6):CLOSE
    #N2:OPEN #N2,N4,N0,"K"
JP 1015 REM ** MOVE CHARACTER SET **
YC 1016 RAMTOP=PEEK(106)-N8:CHSET=RAMTOP*
   256:A=USR(ADR(MOVE$),57344,CHSET,1024)
FW 1018 POKE 106,RAMTOP-N1:GRAPHICS 17:PO
   KE 559,N0:POKE 756,CHSET/256
PH 1019 REM ** CHANGE CHARACTER SET **
OG 1020 POKE N16,N64:POKE 53774,N64:A=USR
   (ADR(TRANSF$),1022,63,CHSET+N8)
YM 1022 DATA 0,102,102,102,102,0,102,0
DZ 1024 DATA 254,254,254,254,254,254,254,
   254
SR 1026 DATA 254,254,254,254,254,254,124,
   124
HB 1028 DATA 124,124,124,124,124,124,124,
   56
HA 1030 DATA 56,56,56,56,56,56,56,16
LU 1032 DATA 16,16,16,16,16,16,16,16
NW 1034 DATA 0,28,28,4,8,0,0,0
CP 1036 DATA 124,254,254,254,254,254,254,
   254
CB 1038 DATA 56,124,124,124,124,124,124,1
   24
KJ 1040 DATA 16,56,56,56,56,56,56,56
YW 1042 DATA 56,124,254,124,186,68,186,68
XI 1044 DATA 0,0,0,0,0,28,28,56
ID 1046 DATA 0,0,0,126,126,0,0,0
MP 1048 DATA 0,0,0,0,0,28,28,0
XS 1050 DATA 0,6,14,24,48,224,192,0
HT 1052 DATA 0,124,198,202,210,226,124,0
RG 1054 DATA 0,52,116,52,52,52,52,0
FD 1056 DATA 0,124,206,28,56,0,254,0
JS 1058 DATA 0,254,0,28,14,198,124,0
BV 1060 DATA 0,6,54,102,198,246,6,0
JA 1062 DATA 0,254,0,252,14,206,124,0
SW 1064 DATA 0,28,64,220,206,206,92,0
WX 1066 DATA 0,254,0,28,56,112,112,0
QQ 1068 DATA 0,60,198,56,198,198,60,0
RJ 1070 DATA 0,116,230,230,118,4,112,0
AS 1072 DATA 0,0,28,28,0,28,28,0
QX 1074 DATA 0,0,28,28,0,28,28,56
HS 1076 DATA 16,16,56,56,124,108,198,130
XZ 1078 DATA 0,0,126,126,0,126,0,0
HF 1080 DATA 130,198,108,124,56,56,16,16
YH 1082 DATA 0,124,230,206,28,56,0,56
QR 1084 DATA 56,124,254,254,124,186,68,56
HI 1086 DATA 0,24,12,198,198,246,198,0
BS 1088 DATA 0,220,198,220,198,198,220,0
FY 1090 DATA 0,28,198,192,192,198,28,0
WU 1092 DATA 0,216,198,198,198,198,216,0
RT 1094 DATA 0,222,192,216,192,192,222,0
ZM 1096 DATA 0,222,192,192,216,192,192,0
QW 1098 DATA 0,30,192,192,206,194,28,0
BY 1100 DATA 0,198,198,222,198,198,198,0
DQ 1102 DATA 0,244,48,48,48,48,188,0
KS 1104 DATA 0,14,6,6,6,198,116,0
XV 1106 DATA 0,204,216,192,216,204,198,0
ZM 1108 DATA 0,192,192,192,192,194,222,0
TN 1110 DATA 0,194,214,222,202,194,194,0
FJ 1112 DATA 0,198,198,214,222,206,198,0
MC 1114 DATA 0,124,230,194,194,230,124,0
BY 1116 DATA 0,220,198,198,220,192,192,0
CL 1118 DATA 0,28,198,198,198,204,54,0
OL 1120 DATA 0,220,198,196,216,204,230,0
GY 1122 DATA 0,124,226,120,30,142,124,0
VN 1124 DATA 0,254,0,56,56,56,56,0
```

```
XO 1126 DATA 0,198,198,198,198,198,222,0
BQ 1128 DATA 0,198,198,198,198,12,24,0
QV 1130 DATA 0,194,194,202,222,214,194,0
GG 1132 DATA 0,198,238,24,48,238,198,0
NQ 1134 DATA 0,198,198,28,24,24,24,0
MF 1136 DATA 0,254,0,56,112,0,254,0
BC 1138 DATA 28,56,118,230,160,86,40,20
UF 1140 DATA 0,192,224,48,24,14,6,0
VN 1142 DATA 56,28,110,103,5,106,20,40
MP 1144 DATA 255,255,255,255,255,255,255,
   255
DT 1146 DATA 0,60,126,126,126,126,60,0
KE 1147 REM ** TURN ON P/M GRAPHICS **
DC 1148 PMBASE=(RAMTOP+N4)*256
QJ 1150 POKE 54279,RAMTOP+N4
FZ 1152 POKE 53277,N3:POKE 623,N8
QS 1153 REM ** CLEAR P/M MEMORY **
QO 1154 A=USR(ADR(CLEAR$),PMBASE,1024)
PZ 1155 REM ** CREATE 3-D BORDER **
RO 1156 FOR I=N0 TO N48:POKE PMBASE+423+I
   ,63:NEXT I
PE 1158 POKE PMBASE+551,127:POKE PMBASE+6
   03,N255:POKE PMBASE+679,252:POKE PMBAS
   E+731,N255
MI 1160 POKE PMBASE+807,N255:POKE PMBASE+
   859,N255:POKE PMBASE+935,240:POKE PMBA
   SE+987,252
GU 1162 FOR I=N0 TO N2:POKE PMBASE+420+I,
   N48:POKE PMBASE+472+I,N48:NEXT I:POKE
   PMBASE+475,N48
PZ 1164 A=USR(ADR(TRANSF$),1166,N1,53248)
BU 1166 DATA 48,80,111,143,52,109,166,0,3
   ,3,3,3,0
WR 1167 REM ** COLORS **
NQ 1168 A=USR(ADR(TRANSF$),1170,N1,704)
HQ 1170 DATA 6,6,6,6,70,134,0,8,10
VI 1171 REM ** DRAW SCREEN **
IN 1172 SC=PEEK(88)+PEEK(89)*256:A=USR(AD
   R(TRANSF$),1174,N24,SC):POKE 559,47:RE
   TURN
KN 1174 DATA 5,0,0,32,32,32,0,228,239,245
   ,226,236,229,0,243,233,248,0,96,96,96,
   0,0,0
KY 1176 DATA 5,126,126,126,126,126,126,12
   6,126,126,126,126,126,126,126,126,126,
   126,126,126,126,126,126,126
ZA 1178 DATA 0,0,98,108,117,101,90,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0
RB 1180 DATA 126,126,126,126,126,126,126,
   126,126,126,126,126,126,126,126,126,12
   6,126,126,126,126,126,126,126
FB 1182 DATA 126,126,126,126,66,96,125,12
   6,126,126,126,126,59,32,126,126,126,12
   6,126,126,126,126,126,126
TB 1184 DATA 126,126,254,254,254,254,254,
   254,254,254,254,254,254,254,254,254,25
   4,126,126,126,126,126,126,66
YN 1186 DATA 2,126,194,32,2,66,2,96,2,194
   ,96,2,66,2,66,32,254,126,191,0,191,126
   ,126,67
CF 1188 DATA 3,126,194,32,3,67,3,96,3,194
   ,96,3,67,3,67,32,254,126,0,0,0,126,126
   ,68
BE 1190 DATA 4,126,194,32,4,68,4,96,4,194
   ,96,4,68,4,68,4,254,126,191,0,191,126,
   126,69
NW 1192 DATA 5,126,194,32,5,69,5,69,5,194
   ,96,5,69,5,69,5,254,126,0,0,0,126,126,
   70
HW 1194 DATA 6,126,194,32,6,70,6,70,6,194
   ,96,6,70,6,70,6,254,126,191,0,191,126,
   126,126
JB 1196 DATA 126,126,194,0,0,0,0,0,0,194,
   0,0,0,0,0,0,254,126,66,92,126,126,126,
   126
TX 1198 DATA 126,126,194,0,0,0,0,0,0,194,
   0,0,0,0,0,0,254,126,66,30,126,126,126,
   70
LE 1200 DATA 6,126,194,96,70,6,70,6,70,19
   4,32,70,6,70,6,70,254,126,191,0,191,12
   6,126,74
OK 1202 DATA 10,126,194,96,74,10,74,10,74
   ,194,32,74,10,74,10,74,254,126,0,0,0,1
   26,126,73
ZT 1204 DATA 9,126,194,96,73,9,73,32,73,1
   94,32,73,9,73,9,73,254,126,191,0,191,1
   26,126,72
GA 1206 DATA 8,126,194,96,72,8,72,32,72,1
   94,32,72,8,72,8,96,254,126,0,0,0,126,1
   26,66
RN 1208 DATA 2,126,194,96,66,2,66,32,66,1
   94,32,66,2,66,2,96,254,126,191,0,191,1
   26,126,126
RZ 1210 DATA 126,126,254,254,254,254,254,
   254,254,254,254,254,254,254,254,254,25
   4,126,126,126,126,126,126,126
VO 1212 DATA 126,126,126,126,66,32,61,126
   ,126,126,126,126,123,96,126,126,126,12
   6,126,126,126,126,126,126,126
QY 1214 DATA 126,126,126,126,126,126,126,
   126,126,126,126,126,126,126,126,126,12
   6,126,126,126,126,126,126,126
ZF 1216 DATA 0,0,0,50,37,36,26,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0
RO 1218 DATA 126,126,126,126,126,126,126,
   126,126,126,126,126,126,126,126,126,12
   6,126,126,126,126,126,126,126
LD 1220 DATA 0,0,0,0,167,161,173,165,154,
   0,46,165,183,0,175,178,0,47,172,164,0,
   0,0,0
```

LISTING 2: BASIC

```
WN 0 REM ******************************
BZ 1 REM *        DOUBLE SIX         *
KO 2 REM *        LISTING 2          *
NZ 3 REM *     by Pierre Roberge     *
ZD 4 REM *                           *
IN 5 REM *      COPYRIGHT 1989       *
PM 6 REM *    BY ANALOG COMPUTING    *
WU 7 REM ******************************
NN 8 REM
WY 10 CLOSE #1:OPEN #1,8,0,"D:DOUBLE6.CHR
   ":FOR I=1 TO 397:READ A:PUT #1,A:NEXT
   I:CLOSE #1
EB 100 DATA 49,48,48,48,32,68,73,77,32,84
   ,82,65,78,83,70,36
OC 102 DATA 40,49,49,53,41,58,84,82,65,78
   ,83,70,36,40,49,44
DC 104 DATA 53,48,41,61,34,104,104,133,20
   9,104,133,208,104,104,133,205
KN 106 DATA 104,133,213,104,133,212,165,1
   36,133,203,165,137,133,204,160,2
BM 108 DATA 165,203,24,113,203,133,203,14
   4,2,230,204,160,1,177,203,217
DI 110 DATA 208,0,144,234,208,5,136,34,15
   5,49,48,48,50,32,84,82
EK 112 DATA 65,78,83,70,36,40,53,49,44,49
   ,49,53,41,61,34,192
ZF 114 DATA 255,208,242,160,4,169,0,133,2
   14,200,177,203,201,71,176,23
ZF 116 DATA 201,44,240,21,233,48,133,215,
   165,214,10,133,214,10,10,101
GD 118 DATA 214,101,215,133,214,144,226,1
   60,0,132,207,160,0,165,214,145
SZ 120 DATA 212,230,212,208,2,230,213,164
   ,207,208,202,198,205,208,167,96
GF 122 DATA 34,155,49,48,48,52,32,68,73,7
   7,32,77,79,86,69,36
DJ 124 DATA 40,52,51,41,58,77,79,86,69,36
   ,61,34,104,104,133,211
HY 126 DATA 104,133,210,104,133,213,104,1
   33,212,104,133,214,104,170,208,2
AU 128 DATA 198,214,160,0,177,210,145,212
   ,200,208,4,230,211,230,213,202
LC 130 DATA 208,242,198,214,16,238,96,34,
   155,49,48,48,54,32,68,73
BA 132 DATA 77,32,67,76,69,65,82,36,40,52
   ,50,41,58,67,76,69
QM 134 DATA 65,82,36,61,34,104,104,133,20
   4,104,133,203,104,133,206,104
EH 136 DATA 133,205,166,206,160,0,169,0,1
   45,203,136,208,251,230,204,202
EB 138 DATA 48,6,208,244,164,205,208,240,
   198,204,160,0,145,203,96,34
BT 140 DATA 155,49,48,48,56,32,68,73,77,3
   2,68,36,40,53,52,41
IL 142 DATA 58,68,36,61,34,0,0,0,0,191,0,
   0,0,0,191,0
VD 144 DATA 0,0,0,0,0,0,191,191,0,0,0,191
   ,0,0,0,191
MJ 146 DATA 191,0,191,0,0,0,191,0,191,191
   ,0,191,0,191,0,191
CG 148 DATA 0,191,191,0,191,191,0,191,191
   ,0,191,34,155
```

BASIC TR

by Clayton Walnum

So far, all the programs we've written have run from top to bottom, processing one instruction after another until they get to the last line of code. It is, of course, impossible to write full-sized programs without some way of controlling program flow. Therefore, we need to be able to construct loops so we can easily perform repetitive processing. We also need to give our programs decision-making abilities. In this installment, we'll take a look at these two ways of controlling a program's order of execution.

What Is a Loop, Anyway?

Many times when writing a program, we come across a process that must be performed repeatedly. It's inefficient to write the same instruction over and over when we could just tell the computer, "Hey, you! Do this ten times!" A loop is a program construct that allows us to say just that to the computer. It gives us a way to send the program back to the same point over and over until certain criteria are met.

For example, let's say we want to get three numbers from the user. We could do it this way:

```
10 DIM NUMBERS(3)
20 PRINT "ENTER THREE NUMBERS:"
30 INPUT A:NUMBERS(1)=A
40 INPUT A:NUMBERS(2)=A
50 INPUT A:NUMBERS(3)=A
```

But by using a loop, we can get all three numbers with only a single INPUT statement. Here's the program above written using a simple loop:

```
10 DIM NUMBERS(3)
20 X=0
30 X=X+1:IF X=4 THEN GOTO 60
40 INPUT A:NUMBERS(X)=A
50 GOTO 30
60 END
```

Do you see the loop? Here we're using the variable X to both count the number of times we've passed through the INPUT statement and also as an index into the array NUMBERS. But before I describe how this program works, there are a few new things you need to understand.

Those of you who are used to seeing algebraic equations may find the statement X=X+1 in Line 30 a little confusing. Just remember that, in BASIC, a statement like this is solved from right to left. In other words, first we add one to X, then give that value back to X. All we're doing is simply adding one to X.

In Line 30 there's also an IF. . .THEN statement, which allows us to control program execution based on the values of variables. If the expression immediately following the IF is true (in this case, X=4), the instructions following the THEN (in this case, GOTO 60) are performed. If the expression is false, program execution drops down to the next line after the IF. . .THEN statement. Anything on the same line following the THEN will be ignored.

And that brings us to the GOTO statement. Common sense has probably already told you what it does, but just for record let me state it here: The GOTO statement lets us send program execution to any program line we wish, simply by placing the line number after the GOTO. In other words, the statement GOTO 60 will cause the program to jump to Line 60 and continue execution from there.

Now that we're familiar with these new statements, let's look at the program flow for the above example. We'll list the lines in the

AINING

order the running program encounters them. Note that for some reason, Atari BASIC doesn't allow us to INPUT a value directly into an array. First we have to get the value into a regular numeric variable, then set the array element equal to that variable.

Line 10: Dimension the array NUMBERS.

Line 20: Set X equal to zero.

Line 30: Add one to X, making it one. Is X=4? No, so drop down to the next line.

Line 40: Get a value for NUMBERS(1) (remember: X=1) from the keyboard.

Line 50: Go back to Line 30.

Line 30: Add one to X, making it two. Is X=4? No.

Line 40: Get a value for NUMBERS(2) (now X=2) from the keyboard.

Line 50: Go back to Line 30.

Line 30: Add one to X, making it three. Is X=4? No.

Line 40: Get a value for NUMBERS(3) from the keyboard.

Line 50: Go back to Line 30.

Line 30: Add one to X, making it four. Is X=4? Yes! Jump to Line 60.

Line 60: End program.

Now, I'll admit that the second program, the one using the loop, is actually longer and harder to understand than the "non-loop" version. But what if we wanted to get ten numbers from the keyboard? Or 50? Or 100? The fact is, we can perform the loop any number of times without enlarging the program. Here's the same program, changed to get 1,000 numbers from the keyboard:

```
10 DIM NUMBERS(1000)
20 X=0
30 X=X+1:IF X=1001 THEN GOTO 60
40 INPUT A:NUMBERS(X)=A
50 GOTO 30
60 END
```

If we had written this program using the "simple" method, without a loop, we would have needed 1,000 lines of INPUT statements! Now you can see the real power of a loop. We can simplify the above program even further. BASIC has a special construction known as the FOR...NEXT loop, which lets us set up a loop without having to do all the housekeeping; that is, the program itself will keep track of the value of X, automatically incrementing it as needed and breaking out of the loop at the proper time. Here's our program again, written using a FOR...NEXT loop:

IF WE HAD WRITTEN THIS PROGRAM USING THE "SIMPLE" METHOD, WITHOUT A LOOP, WE WOULD HAVE NEEDED 1,000 LINES OF INPUT STATEMENTS! NOW YOU CAN SEE THE REAL POWER OF A LOOP.

```
10 DIM NUMBERS(1000)
20 FOR X=1 TO 1000
30 INPUT A:NUMBERS(X)=A
40 NEXT X
50 END
```

The first time the program gets to Line 20, it sets X to one. It then drops down to Line 30, where it gets a value for NUMBERS(1). At Line 40, X is incremented (increased by one), and the program jumps back to Line 20, where X is tested to see if it's larger than 1,000. If it's not, the program again drops down to Line 30 to get a value for NUMBERS(2). At Line 40, X is again incremented, and program execution jumps back to Line 20. The value of X is still less than 1,000, so we go through the loop again. Eventually, X will become 1,001. The value will be tested at Line 20, and because X is now out of the range of our loop, the loop is ended. Program execution automatically continues at the statement immediately following the NEXT X, or in this case, at Line 50, where the program ends.

We don't need to construct our FOR...NEXT loops exactly as shown in the first example. We could have put the entire program on one line. It would still operate properly:

```
10 DIM NUMBERS(1000):FOR X=1
TO 1000:INPUT A:NUMBERS(X)=A:
NEXT X:END
```

This is hard to read, though. I'd write the above program like this:

```
10 DIM NUMBERS(1000)
20 FOR X=1 TO 1000:INPUT A:NU
MBERS(X)=A:NEXT X
30 END
```

By putting the loop all on one line, it can execute much faster.

When using a FOR...NEXT loop, we don't have to "count by ones." By using the STEP keyword, we can make a loop count by whatever interval we want:

```
10 FOR X=0 TO 10 STEP 2
20 PRINT X
30 NEXT X
```

This program will give us this output:

```
0
2
4
6
8
10
```

We can even have a loop that counts backwards:

```
10 FOR X=10 TO 0 STEP -2
20 PRINT X
30 NEXT X
```

The output from the above program would be:

```
10
8
6
4
2
0
```

Conclusion

Now that we know how to handle loops and IF...THEN statements, we have most of the tools we need to write fairly sophisticated programs. Still, BASIC is a rich language and there's a lot of territory left to cover. Next time, we'll learn about simple disk access and write a program that'll review everything we've learned so far. See you then.

Clayton Walnum is the executive editor of ST-LOG *and* ANALOG Computing, *as well as the associate editor of* Videogames & Computer Entertainment.

PACKAGE DEALS FROM SAN JOSE COMPUTER

RAINBIRD SIX-PACK $49.95

YOU SAVE OVER $175

SUPER ATARI ST SOFTWARE PACKAGE

TRACKER -	ACTION / STRATEGY / 3D GRAPHICS	$44.95 REG.
THE GUILD OF THIEVES -	GRAPHIC TEXT ADVENTURE	$34.95 REG.
THE PAWN -	GRAPHIC TEXT ADVENTURE	$34.95 REG.
THE SENTRY -	STRATEGIC / CHESSLIKE	$35.95 REG.
GOLDEN PATH -	GRAPHICS ADVENTURE (NO TYPING!!)	$34.95 REG.
STARGLIDER -	ACTION SHOOT-EM UP W/ FAST 3D GRAPHICS	$39.95 REG.

1200XL PACKAGE $99.95

» COMPUTER «» PRINTER «» PLOTTER «
» 1200XL «» 1025 «» 1020 «

INCLUDES:

- 64K COMPUTER
- 1020 PRINTER PLOTTER
- 1025 DOT MATRIX PRINTER (80 COL.)
- PAC-MAN CARTRIDGE
- BASIC WITH TUTOR SET

ALL ITEMS ARE RECONDITIONED EXCEPT 1020 PRINTER

1200XL W/PLOTTER $59

» COMPUTER «» PLOTTER «
» 1200XL «» 1020 «

INCLUDES:

- 64K COMPUTER
- 1020 PRINTER PLOTTER
- BASIC CARTRIDGE
- PAC-MAN CARTRIDGE

ALL ITEMS ARE NEW EXCEPT 1200XL COMPUTER

DRIVE SPECIAL $99*

A SAVINGS OF $30

» 810 «

INCLUDES:

- 810 DISK DRIVE (RECON)
- DOS 2.5 W/MANUAL
- POWER SUPPLY AND CABLES

* SPECIAL PRICE AVAILABLE ONLY WHEN PURCHASED WITH ABOVE SYSTEMS. ONE PER SYSTEM ONLY!

RAINBIRD TITLES ST

Knight Orc	$9.95
Tracker	$9.95
Advanced Art Studio	$9.95
Guild of Thieves	$9.95
The Pawn	$9.95
Golden Path	$7.95
Universal Military Simulator	$14.95
Carrier Command	$29.95
Jinxter	$9.95
Sentry	$9.95
StarGlider	$9.95

ALL ITEMS REG. $39.95 - $49.95

ST SOFTWARE TITLES AVAILABLE

SOFTWARE PAK#1 $29.95

INCLUDES:

- SLIME
- CHICKEN
- BASIC
- GORF
- JOURNEY TO THE PLANETS
- TURMOIL
- PAC-MAN
- INV. TO PROGRAMING
- DONKEY KONG
- BANDITS
- CLAIM JUMPER
- DELUXE INVADERS

ASTRA 2001 DUAL DRIVE $199

DOUBLE DENSITY
DUAL DISK DRIVE
(2 DRIVES IN ONE CASE)

ATARIWRITER 80

- 80 COLUMN EDITING
- EDITABLE PRINTER DRIVER
- 30,000 WORD DICTIONARY
- MAIL MERGE
- ON 5¼ DISKETTE

$49

ATARI XEP 80

- CONNECTS TO ANY COMPOSITE MONITOR
- 80 COL. DISPLAY
- PRINTER INTERFACE

$79

ATARI 1200XL PACKAGE

W/ 1025 PRINTER & 1020 PLOTTER

$99

OR THE WHOLE THING!!

- ATARIWRITER 80
- ATARI XEP 80
- 1200XL PRINTER SYSTEM

$149

1020 COLOR PRINTER PLOTTER

COMPLETE WITH:
- 2 PEN SETS
- 1 ROLL PAPER
- POWER SUPPLY & CABLE

$14.95

BRAND NEW

EXTRA PEN SETS — COLOR $3.98, BLACK $.89

1200XL 64K COMPUTER

$49 RECONDITIONED

256 COLORS, XEGS COMPATIBLE, FREE PAC-MAN

$69 FACTORY NEW!!

1027 PRINTERS 1025

LETTER QLTY **$79**

1027 IS FACTORY NEW! FOR YOUR 8-BIT 1025 IS RECONDITIONED

DOT MATRIX **$69**

HARDWARE & ACCESSORIES

850 PRINTER INTERFACE	$79.95
1010 PROGRAM RECORDER	$29.95
TWO JOYSTICK SET	$9.95
SPACE AGE JOYSTICK	$14.95
WICO COMMAND JOYSTICK	$24.95
WICO THE BOSS JOYSTICK	$19.95
ATARI TRACKBALLS	$9.95
ATARI SX212 MODEM	$89.95
AVATEX 1200HC MODEM	$89.95
DOS 2.5 W/ MANUAL	$4.95
600XL (NO Transformer)	$19.95
400,800,850,1200XL Transformer	$14.95
XL/XE Transformer	$19.95

DISK DRIVES FOR 800/XL/XE

COMPLETE WITH: POWER SUPPLY, I/O CABLE, DOS 2.5 W/ MANUAL

ATARI 810 $129.00
ATARI 1050 $169.00
RECONDITIONED

THE BEST LIGHT GUN

FOR YOUR 800/XL/XE

$35

OR THE PACKAGE **$109**
- LIGHT GUN
- CRIME BUSTERS
- CROSSBOW
- BARNYARD BLASTER

BOOKS - BOOKS

101 PROGRAMING TIPS & TRICKS	$2.50
INSIDE ATARI BASIC	$2.50
DE RE ATARI	$14.95

CARTRIDGES FOR THE 800/XL/XE

BASIC CARTRIDGE	$4.95					STAR RAIDERS II	$19.95
BASIC TUTOR (2 BOOKS)	$4.95	ET	$4.95	ROBOTRON	$19.95	DAVID'S MIDNIGHT MAGIC	$19.95
QIX	$4.95	FACEMAKER	$4.95	TENNIS	$19.95	ARCHON	$19.95
TURMOIL	$4.95	MATH ENCOUNTER	$7.95	FINAL LEGACY	$19.95	KARATEKA	$19.95
PAC-MAN (no box)	$4.95	DANCE FANTASY	$8.95	MARIO BROS	$19.95	CHOPLIFTER	$19.95
DONKEY KONG (no box)	$4.95	LOGIC LEVELS	$8.95	DONKEY KONG JR	$19.95	GATO	$24.95
GORF (400,800)	$4.95	MEMORY MANOR	$8.95	JUNGLE HUNT	$19.95	ACE OF ACES	$24.95
CHICKEN	$4.95	LINKING LOGIC	$8.95	MOON PATROL	$19.95	LODE RUNNER	$24.95
SLIME (400,800)	$4.95	DELTA DRAWING	$9.95	BATTLEZONE	$19.95	BARNYARD BLASTER (LG)	$24.95
CLAIM JUMPER	$4.95	HEY DIDDLE DIDDLE	$9.95	FOOD FIGHT	$19.95	DARK CHAMBERS	$29.95
DELUXE INVADERS	$4.95	GRANDMA'S HOUSE	$9.95	HARDBALL	$19.95	AIRBALL	$29.95
JOURNEY TO THE PLANETS	$4.95	FRACTION FEVER	$9.95	FIGHT NIGHT	$19.95	SUMMER GAMES	$29.95
STAR RAIDERS	$4.95	ALPHABET ZOO	$9.95	ONE ON ONE BASKETBALL	$19.95	CROSSBOW (LG)	$29.95
MISSLE COMMAND	$4.95	ALF	$9.95	DESERT FALCON	$19.95	EAGLES NEST	$29.95
GALAXIAN	$4.95	DIG DUG	$14.95	NECROMANCER	$19.95	CRIME BUSTERS (LG)	$29.95
DEFENDER	$4.95	MILLIPEDE	$14.95	RESCUE ON FRACTALUS	$19.95	MICROFILER (database)	$39.95
		SKY WRITER	$14.95	BALLBLAZER	$19.95		
		FOOTBALL	$14.95	BLUE MAX	$19.95		

DISK SOFTWARE FOR THE 800/XL/XE

DAVID'S MIDNIGHT MAGIC	$4.95	COMMBAT	$4.95	ADVENTURELAND	$4.95	ALIEN AMBUSH	$4.95
REPTON	$4.95	PREPPIE I	$4.95	PIRATE ADVENTURE	$4.95	DISPATCH RIDER	$9.95
BANDITS (48K 400,800)	$4.95	PREPPIE II	$4.95	SECRET MISSION	$4.95	SILICON DREAMS	$9.95
CLAIM JUMPER	$4.95	THE COUNT	$4.95	VOODOO CASTLE	$4.95	VISICALC	$29.95
TIME WISE	$4.95	FREAKY FACTORY	$4.95	STRANGE ODYSSEY	$4.95	BOOKKEEPER W/ num keypad	$29.95
CROSSCHECK	$4.95	LASER HAWK	$4.95	REPTON	$4.95	HOME ACCOUTANT	$29.95
MISSION ASTEROID	$4.95	CRYSTAL RAIDERS	$4.95	HULK	$4.95		

1010 PROGRAM RECORDER

SAVE AND LOAD YOU OWN PROGRAMS

$29.95 RECONDITIONED

LIGHT SPEED C DEVELOPMENT KIT

$35.95

SAN JOSE COMPUTER

THE ATARI STORE

640 BLOSSOM HILL RD. SAN JOSE, CA 95037

STORE (408) 224-8575 • BBS (408) 224-9052 • FAX (408) 224-8574

FAX US YOUR ORDER!

FOR FASTER SERVICE PLEASE INCLUDE:
- NAME
- BILL TO ADDRESS
- SHIP TO ADDRESS
- PHONE #
- ITEM(S) YOU WISH TO PURCHASE

VISA / MASTERCARD ORDERS ONLY

CARD ADDRESS MUST MATCH BILL TO ADDRESS

(408) 224-8574

SHIPPING: ADD $5.00 TO ALL ORDERS. AIR AND INTERNATIONAL SHIPPING EXTRA. THAT'S IT.

WARRANTY: 90 DAY WARRANTY ON ALL ITEMS. TAX: CALIFORNIA RESIDENTS ADD 7% SALES TAX.

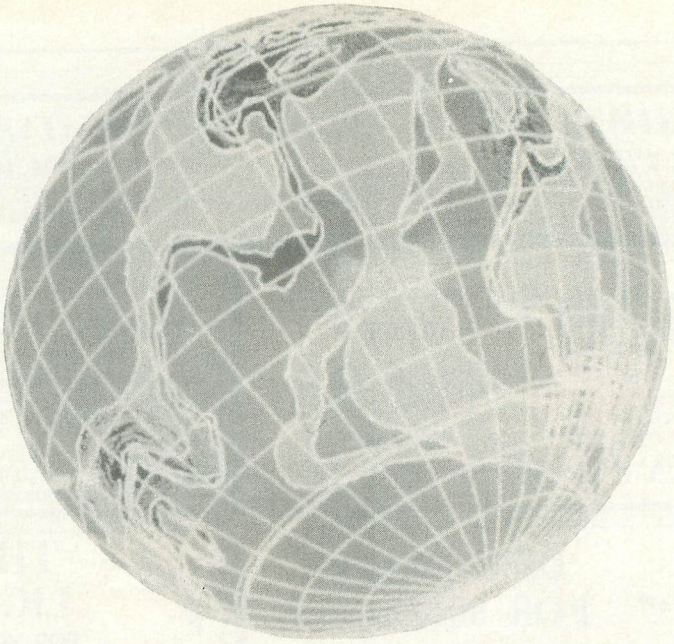
PREPAYMENT: USE VISA, MASTERCARD, MONEY ORDER, CASHIER'S CHECK OR PERSONAL CHECK. PERSONAL CHECK MUST CLEAR PRIOR TO SHIPMENT. C.O.D.: CASH, CASHIER'S CHECK OR M.O. ONLY.

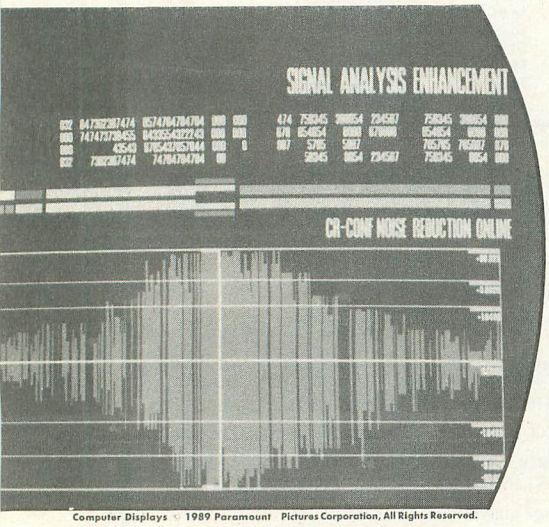
Prices subject to change without notice.

Ad produced on an ATARI ST using CALAMUS and printed on an ATARI SLM804 PostScript compatible laser printer.

CIRCLE # 102 ON READER'S SERVICE CARD

In two seasons of new episodes, we have seen the crew of the *Enterprise* struggle with computer viruses, computer pirates and advanced technology. Many of the scripts deal with today's problems projected into the *Enterprise* scenario. Written by futurists, the scripts are believable.





part of the panel, you feel the reaction of the control. There is a depth dimensionality to the panel.

Sternbach: The idea is part of a nanotechnology. We take a piece of ¼-inch Plexiglas and lay it into the set to create a panel. If this were the world of the 24th century, we would be using a material that has 50 or more layers of materials, each layer performing a different task. For us to create this material today would cost an arm and a leg. But in the 24th century, this stuff comes off a roll!

ANALOG: Can these panels project three-dimensional displays?

Okuda: They can; however, the style of the operating system tends not to use this capability. Budget limitations on the show have made us keep the displays two-dimensional; however, the panels could perform multidimensionally. An example of this operation might be a crew member looking at a projection of the harmonic subspace distortions of the ship.

Sternbach: It could also be a two-dimensional display of a three-dimensional object. You have to remember that these panels are also used by nonhumans. They might not have depth perception and could not use the dimensional display because their interoccular dimensions are too close for depth.

Okuda: There is another part to this. If we are designing for the 24th century, we really don't know how a lot of three-dimensional imagery will appear when a user is looking into it. For example, if I had designed a little three-dimensional ball coming out of each button, I would be most concerned with how easily that control works. It might be that the panel would become too cluttered, or it might get in the way of another panel's use.

ANALOG: One of the shows featured a computer virus attacking the *Enterprise* computer. What did you think about that episode?

Okuda: I thought it was exciting, but the level of computer technology was not quite as advanced as it will really be that far into the future. The idea of an information-based weapon of that kind was pretty good. I thought the computer would be a little more protected than it appeared.

Sternbach: That's really a problem of the time constraints we have working on a show like this. [The writers] might want to polish a script for a couple of months, but they just don't have that kind of time.

Time hasn't hindered these artists' creativity. Okuda and Sternbach have implanted their vision of computer/human symbiosis into the new series by *The Next Generation* portraying computers and technology in a fundamentally different way than did its predecessors: Okuda and Sternbach have added a friendly and open image to the touch panels aboard the *Enterprise*.

If television is a reflection of the public perception, the technology on display in *Star Trek: The Next Generation* could well be right around the corner. After all, it was less than 50 years ago that television brought us to the moon and Mars. By making it visual and responsive, the *Star Trek* vision may have brought us that much closer to new worlds and wonders.

Frank Cohen is a programmer, author, graphic designer and music hobbyist. You may contact Frank directly on CompuServe (76004,1573) and GEnie (FRANK.COHEN), or by writing to P.O. Box 14628, Long Beach, CA 90803-1208.

BIT NEWS

Atarifest

The Washington Area Atari Computer Enthusiasts has announced the fifth annual Washington, D.C.-area Atarifest, to be held at the Fairfax High School in Fairfax, Virginia on Saturday and Sunday, October 7 and 8. The show will feature seminars, demonstrations and exhibits designed to show how the Atari computers can be used in business and the home. Vendors interested in displaying their products should contact Johnna Ogden at (703) 450-3992. People wishing to obtain general information should call John Barnes at (301) 652-0667.

Quiet Those Printers ▶

The Silencer Mat is now available from JTB Communications. The mats come in a variety of sizes and are designed to reduce office noise produced by typewriters, computers and printers. They also protect desktop surfaces from scratches. The Silencer Mat is available in charcoal-gray and features acoustical-foam lamination.

Also available is the Mighty Mat series, which comes in blue, silver and red. Its sponge rubber construction grips the desk surface, reducing vibration.

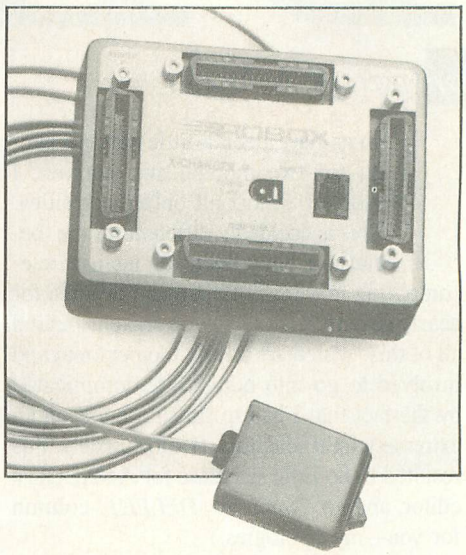
JTB Communications
222 W. Adams Street, Suite 589
Chicago, IL 60606
(312) 263-3063

NEW DESKTOP MATS FROM COMPUTER COVERUP
The SILENCER is one of two new series of mats for personal computers, typewriters and printers introduced by Computer Coverup, Inc.

Peripheral Switch Box ▶

The Robox X-Changer data switch allows two computers to electronically switch between two different peripherals, such as serial or parallel printers, plotters and modems. Switching can be accomplished using a remote switch or by software. The X-Changer comes in two models, one for parallel devices and one for serial devices, the former featuring 36-pin Centronics connectors, the latter featuring DB-25 connectors. The unit lists for $129.95.

Support Systems International Corp.
150 South Second Street, Dept. CC
Richmond, CA 94804
(415) 234-9090



Address Book

Twenty-Fifth Century has announced *Mailing List Plus* for the 8-bit Ataris. Names and addresses are typed on-screen just as they will appear on the labels, and two keys allow the user to move through the address database, making corrections wherever needed. Names and addresses can be categorized, allowing the user to group entries based on a keyword. This feature makes it possible to create labels for a specific subset of the address database.

Entries can be searched on several fields, including first name, last name, city or zip code. *Mailing List Plus* will also print standard labels one or two across and provides the ability for the user to adjust spacing to accommodate just about any label size. *Mailing List Plus* sells for $14.95, including labels.

Twenty-Fifth Century
234 Fifth Avenue, Suite 301
New York, NY 10001
(516) 932-5330

Frog Software

UltraBASIC has just released a group of four new programs for the Atari 8-bits. *Superfrogs* is a seven-game arcade pack that allows over 10,000 game variations. *Superfrogs Funspeller* features six one- or two-player educational games designed to help improve spelling skills. In *Tank Math*, players can try their hands at various math problems, including practicing Roman numerals. In the event of an incorrect answer, the program will work out the problem step by step. *Track Stack 2.0* will transfer up to 15 unprotected machine-language programs to a Track Stack disk, from which they can be loaded with the press of a single key.

UltraBASIC, Inc.
10 East 10th Street
Bloomsburg, PA 17815
(717) 784-4545

DATA BASE DELPHI

by Michael A. Banks

hings have been a little strange lately; for two months this summer, I avoided almost all online activities. This is decidedly different for me because I have been an intense telecomputing maven, buff, fan and devotee for nearly seven years. The main reasons behind all of this, which are a little too personal and involved to go into here, were complicated by the fact that I had to finish a book on an extremely tight schedule. (Unfortunately, this resulted in no little suspense for ANALOG's editor, and no *Database DELPHI* column for you—my apologies.)

Thus, I couldn't afford the luxury of chatting with friends in conference on DELPHI. Nor could I dribble away precious minutes browsing DELPHI's excellent travel service; forget planning a vacation—I was working to pay for last year's.

I had no time to check out the databases in the Atari SIG, and I couldn't waste time swapping bad puns and outrageous gossip via E-mail. No cruising SIG Forums for interesting (or stupid) message threads either. (I didn't even read Forum messages to *me*, and delayed reading some E-mail for up as long as three weeks.)

During one particularly intense 16-day period, I didn't log on to any online services or BBSs.

Although I was too busy to think about it, I missed DELPHI. But it wasn't so much the habit of signing on and checking mail and such that I missed; it was the people. E-mail, messages, conferences—all the "people" activities on DELPHI were what I missed the most.

When I finally started signing on to DELPHI regularly again, it was a strange feeling—not unlike visiting, for the first time in ten years, the town where you grew up.

Things had changed. A lot.

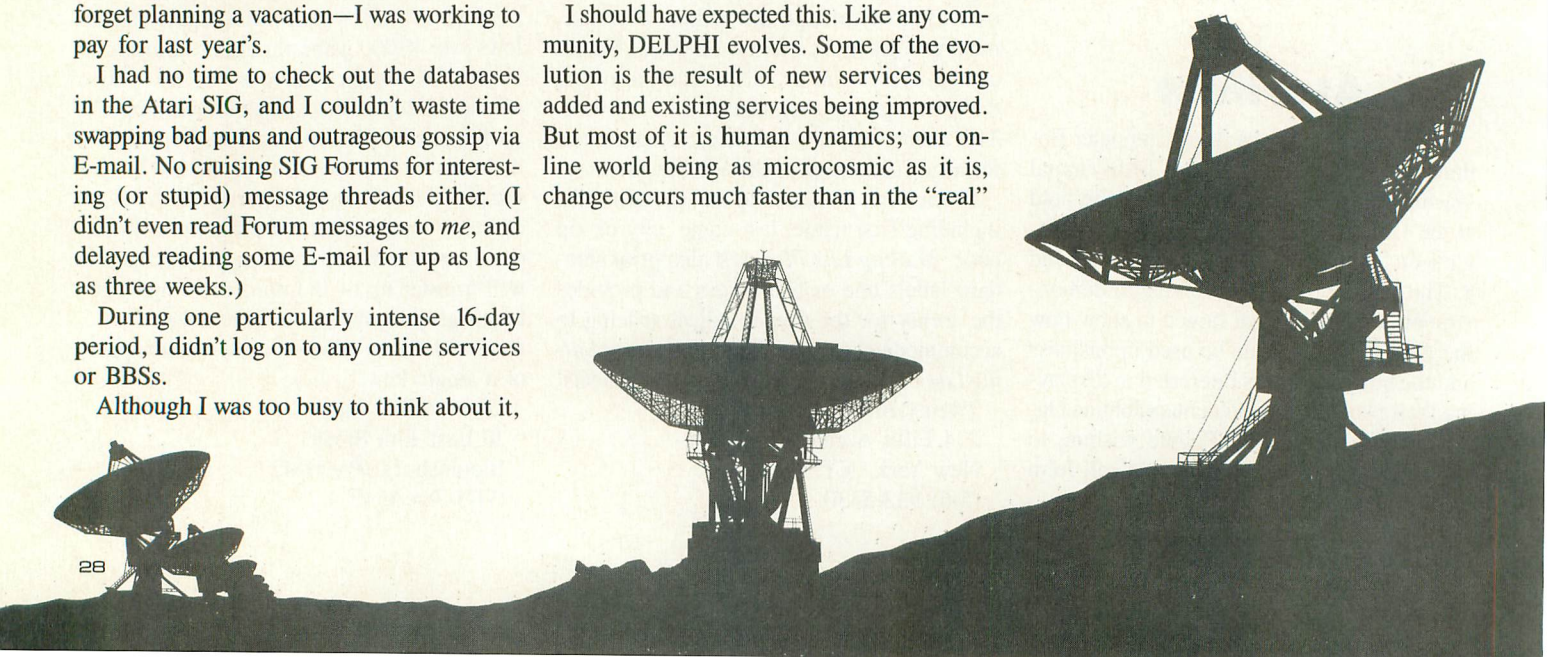
I found new files galore in the Atari SIG databases and encountered lots of new DELPHI members—many of whom were already old hands. The sheer quantity of new messages in the Atari SIG Forum alone left me babbling in the dust—there were over 500 of them!

I should have expected this. Like any community, DELPHI evolves. Some of the evolution is the result of new services being added and existing services being improved. But most of it is human dynamics; our online world being as microcosmic as it is, change occurs much faster than in the "real" world—with the result that the changes in less than two months were almost staggering. When you can assimilate the changes one or two at a time on a week-to-week basis, it's not that noticeable. But being hit with two months' worth of changes all at once threw me off a little.

All of this says something about the vitality imparted DELPHI and its SIGs by its designers and members.

It was an interesting sabbatical, though one I'm not likely to take again soon—just catching up on E-mail and Forum messages took three hours.

Despite (or perhaps because of) my odd sabbatical, I've a few new things to report. So. . . .



Catching Up

So, how did I catch up on what I'd been missing? Well, after wading through my E-mail, I hit the SIGs. I've already mentioned the wealth of new Forum messages in the Atari SIG (there are some pretty good threads on gaming going on, by the way). What I didn't mention was how I dealt with them.

I started to scan all new messages by headers using the DIR NEW command, but found it overwhelming (imagine trying to "scan" 20-odd screens of message headers!) So I narrowed them down by using the DIR NEW TOPIC command. For example, I typed DIR NEW TOPIC TELE to display the headers for new messages stored under the Telecommunications topic. That made things a bit easier—there were only a couple of screens of headers, so I could scan for interesting subjects, which I duly noted and used to further trim the list with DIR NEW SUBJ <subject>.

I did the same thing with other topics of interest, in most cases just using READ NEW SUBJ <subject> after I noted the interesting messages. (This proved to be easier than using DIR NEW SUBJ <subject> because I didn't have to try to remember or write down message numbers.)

When I'd finished going through the Forum, I typed 99999 to read the very last message in the Forum (some had been added while I was catching up!), just for the heck of it.

As usual, I checked out the preview of the upcoming issue of ANALOG Computing, which is always in the Recent Arrivals database (type DA REC to access it, then press Return to see a directory of the newest items). I also scanned the programs from the current issue (DA CUR, for the Current Issue database).

If you find yourself offline for a couple weeks or more, try these "catch-up" techniques—they work!

New News

DELPHI has replaced the Associated Press newswire with United Press International (UPI) news and features. UPI offers a wide selection of interesting features and columns, in addition to the news coverage typical of newswire services.

To check out the new UPI news service, type GO NEWS UPI at the Atari SIG menu. You'll see this menu:

```
UPI News Menu:

Newsbrief                  Sports
Human Interest Stories     Entertainment
National News              Weather
International News          Exit
Business & Finance

UPI>(Please select an item):
```

Select an item, and you'll see a numbered list of news stories and features; to read a story, type its number.

Other items on the News menu—including Accu-Weather—remain unchanged.

Se Habla Español?

The DELPHI/Regional menu now boasts a fourth selection; in addition to versions of DELPHI for Argentina, Boston and Kansas City, there's now a DELPHI/Miami selection, which is a Spanish-language regional operation in Florida. The entire service is in Spanish and was developed by SISCOTEL, the company that developed DELPHI/Argentina.

To take a quick trip to Miami, type GO DEL MIAMI at the Atari SIG menu. (It helps to be able to speak Spanish, of course! *Bienvenido*!)

Hot Tip: Meet the Pros in the SF SIG!

Do you read science fiction and fantasy? Want to meet some of the foremost authors in the field? Check into DELPHI's Science Fiction/Fantasy SIG (GO GR SF), where you'll find fascinating conversations among fans and writers in the Forum. You can also meet such popular writers as Pat Cadigan, Jack Chalker, Nebula-winner George Alec Effinger, Mike Resnick, Joel Rosenberg and others "in person" during the SF SIG's weekly conference, which is held on Wednesday nights between 9:30 p.m. and 11:00 p.m., EST.

Poker Tournaments

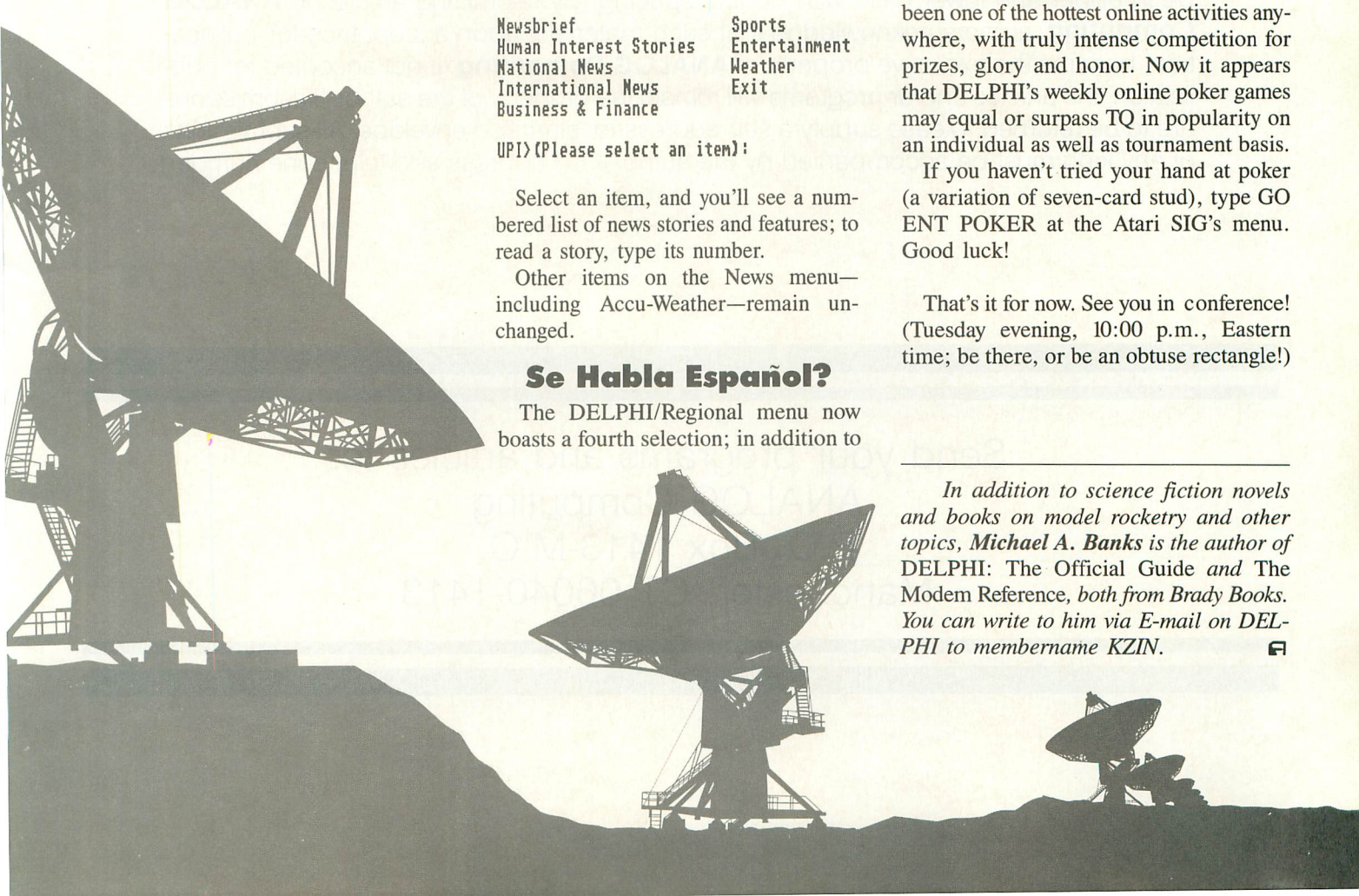
DELPHI's Trivia Quest has for months been one of the hottest online activities anywhere, with truly intense competition for prizes, glory and honor. Now it appears that DELPHI's weekly online poker games may equal or surpass TQ in popularity on an individual as well as tournament basis.

If you haven't tried your hand at poker (a variation of seven-card stud), type GO ENT POKER at the Atari SIG's menu. Good luck!

That's it for now. See you in conference! (Tuesday evening, 10:00 p.m., Eastern time; be there, or be an obtuse rectangle!)

---

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



Attention Programmers!

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

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

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

DISK LISTING

THE ANALOG #77 DISKETTE CONTAINS 13 MAGAZINE FILES. THEY ARE LISTED BELOW:

SIDE 1:

FILENAME.EXT	LANG.	LOAD	ARTICLE NAME
DOUBLE6 .BAS	BASIC	LOAD	DOUBLE SIX
ERRMAN .OBJ	ML	(#4)	ERROR MANUAL
ERROR .MAN			ERROR MANUAL DATA
TXCRUNCH.BAS	BASIC	LOAD	TX CRUNCHER
CLOCK .BAS	BASIC	LOAD	KEEPING BUSY
SKULL .BAS	BASIC	LOAD	SKULL ISLAND
STRING1			SKULL ISLAND DATA
STRING2			SKULL ISLAND DATA
FASTMOVE.BAS	BASIC	LOAD	FAST MOVE DEMO

SIDE 2:

FILENAME.EXT	LANG.	LOAD	ARTICLE NAME
ERRMAN .M65	MAC/65	LOAD	ERROR MANUAL SOURCE
FASTMOVE.M65	MAC/65	LOAD	FAST MOVE SOURCE
MLEDITOR.BAS	BASIC	LOAD	M/L EDITOR
EDITORII.LST	BASIC	ENTER	BASIC EDITOR II

TO LOAD YOUR ANALOG DISK

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

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

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

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

LOADING NOTES

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

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

BO CA

BY TOM

If you solved last issue's multi-byte math problems, give yourself a pat on the back. Successful completion of these programming puzzles indicates that you're well on your way to becoming proficient in 6502 assembly language.

Whether you solved the problems or not, take a look at the following possible solutions. There are many ways to solve any programming problem, and these examples may show you a different approach.

```
10  *=$600
20  SED                ;DECIMAL MODE
30  LDA OLDBAL         ;GET LOW BYTE
40  SEC                ;FIRST SUBTRACT
50  SBC WITHD          ;SUBTRACT LOW
60  STA NEWBAL         ;STORE RESULT
70  LDA OLDBAL+1       ;GET MED BYTE
80  SBC WITHD+1        ;SUBTRACT MED
90  STA NEWBAL+1       ;STORE RESULT
0100  LDA OLDBAL+2     ;GET HI BYTE
0110  SBC #0           ;SUBTRACT DUMMY
0120  STA NEWBAL+2     ;STORE RESULT
0130  BRK              ;ALL DONE!
0140 OLDBAL .BYTE $73,$86,$10
0150 WITHD  .BYTE $85,$42
0160 NEWBAL *=*+3
0170  .END
```

The above shows the solution to the first problem given last month. You were asked to subtract the two-byte BCD variable WITHD from the three-byte variable OLDBAL, placing the result in the three-byte variable NEWBAL; OLDBAL = 108673 and WITHD = 4285.

As you can see, both OLDBAL and WITHD are defined using the .BYTE directive. Standard data-storage formats are used, so the values are defined from low-order to high-order. That is, 108673 is defined as .BYTE $73,$86,$10. The variable NEWBAL is simply set up as *=*+3, reserving three bytes for the result of the operation.

The program itself uses the usual multi-byte subtract structure for the first two subtract operations. The third subtract uses a "dummy" value of zero for the third byte of WITHD, since it is one byte shorter than OLDBAL. This ensures that any borrows from lower-order bytes will be processed properly.

Try executing this program on your computer. After it is finished, examine the three-byte NEWBAL to be sure it contains 104388 (108673-4285). NEWBAL is located at memory location $0622-0624. If you display these locations, you will see something like this:

```
0622 88 43 10
```

You will note that the number 104388 contained in NEWBAL is stored in low-order to high-order format, just like OLDBAL and WITHD.

IF THE INC OPERATION IS PERFORMED ON A BYTE CONTAINING $FF, THE BYTE'S VALUE WILL "WRAP AROUND" TO ZERO.

Solution Two

The second problem I assigned last month asked you to subtract each byte of the ten-byte TABLE2 from the corresponding byte of TABLE1, placing the results in the ten-byte TABLE3. The initial values for TABLE1 and TABLE2 are:

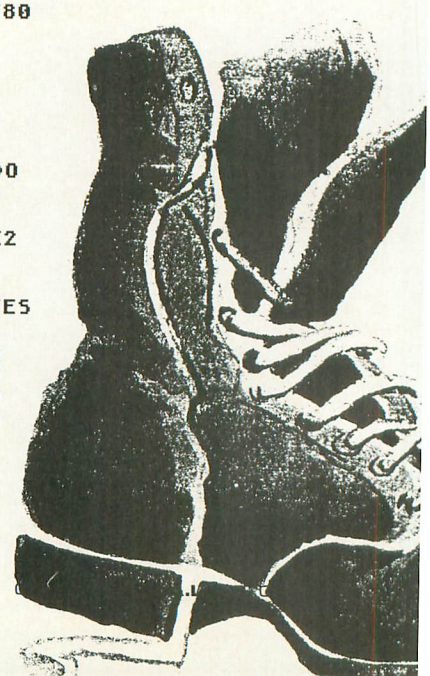
```
TABLE1 .BYTE $10,$18,$40,$86,$9A
       .BYTE $A0,$BC,$C0,$F0,$F8
TABLE2 .BYTE $00,$08,$14,$2F,$9A
       .BYTE $90,$0B,$22,$65,$78
```

If done properly, TABLE3 should contain the following values when the program is finished:

```
$10,$10,$2C,$57,$00,$10,$B1,$9E,$8B,$80
```

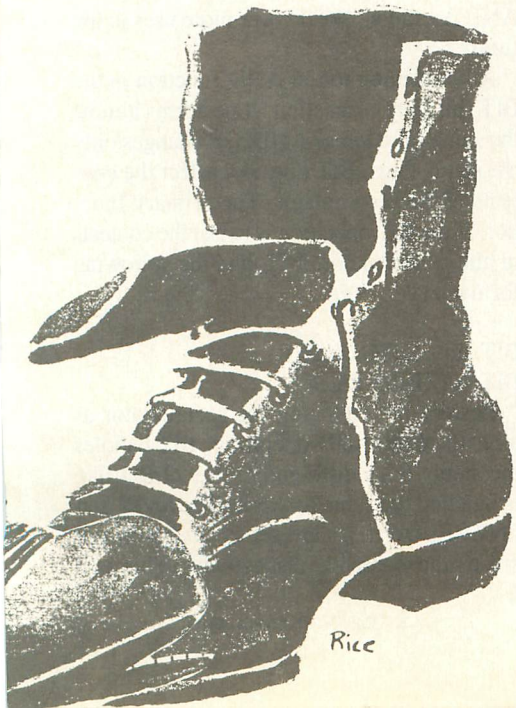
A possible solution to this problem is shown here:

```
10          *=$600
20          CLD            ;BINARY MODE
30          LDX #9         ;10 BYTES TO DO
40 SUBLP    LDA TABLE1,X   ;GET BYTE 1
50          SEC            ;SINGLE BYTE!
60          SBC TABLE2,X   ;SUBTRACT BYTE2
70          STA TABLE3,X   ;AND STORE IT
80          DEX            ;NEXT BYTE
90          BPL SUBLP      ;DO ALL 10 BYTES
0100         BRK           ;ALL DONE!
0110 TABLE1 .BYTE $10,$18,$40,$86,$9A
0120        .BYTE $A0,$BC,$C0,$F0,$F8
0130 TABLE2 .BYTE $00,$08,$14,$2F,$9A
0140        .BYTE $90,$0B,$22,$65,$78
0150 TABLE3 *=*+10
0160       .END
```



OT
MP

HUDSON



As you can see, this problem can be solved by simply indexing through all ten bytes of the tables in the loop SUBLP. Within this loop, the X register points to the desired byte of each table. Each time the loop is executed, the byte from TABLE2 is subtracted from the corresponding byte of TABLE1, and the result is placed in the proper location in TABLE3. Note that each subtract is preceded by the SEC (set carry) instruction, so that the subtracts will be treated as single-byte operations.

Ups and Downs

There are two handy instructions we haven't covered yet that can sometimes be considered math instructions. These are INC (increment memory by one) and DEC (decrement memory by one).

INC n (ZERO PAGE)
INC nn (ABSOLUTE)
INC n,X (ZERO PAGE INDEXED X)
INC nn,X (INDEXED X)

DEC n (ZERO PAGE)
DEC nn (ABSOLUTE)
DEC n,X (ZERO PAGE INDEXED X)
DEC nn,X (INDEXED X)

The INC instruction simply adds one to the value contained in the memory byte referenced and places the result back into the memory location. The accumulator is not affected, but the Sign and Zero flags reflect the result of the operation. Here is an example of the INC operation:

```
10        *=$0600
20        LDA #5          ;5 IN ACCUMULATOR
30        STA VALUE       ;AND IN VALUE
40        INC VALUE       ;VALUE = 6
50        INC VALUE       ;VALUE = 7
60        INC VALUE       ;VALUE = 8
70        BRK             ;ALL DONE!
80 VALUE  *=*+1
90        .END
```

This program will place the value five in the accumulator and the location labeled VALUE. It then increments VALUE three times. When finished, the accumulator will still contain five, but VALUE will contain eight.

If the INC operation is performed on a byte containing $FF, the byte's value will "wrap around" to zero. Note that this instruction is not a true math instruction because the carry resulting from the byte wraparound is not shown in the status flags.

The DEC instruction is similar to the INC instruction, but operates in reverse. Instead of adding one to the value of the byte, DEC subtracts one. Here is an example of the use of the DEC instruction:

```
10        *=$600
20        CLD             ;BINARY MODE
30        LDA #5          ;SET COUNTER...
40        STA COUNT       ;TO 5
50        LDA #7          ;SET TO ADDVAL
60        STA ADDVAL      ;TO 7
70 LOOP   LDA ADDVAL      ;GET ADDVAL
80        CLC             ;SINGLE-BYTE ADD
90        ADC ADDVAL      ;ADD TO ITSELF
0100        STA ADDVAL    ;SAVE RESULT
0110        DEC COUNT     ;HIT ZERO YET?
0120        BNE LOOP      ;NO! LOOP BACK
0130        BRK           ;ALL DONE!
0140 ADDVAL *=*+1
0150 COUNT  *=*+1
0160        .END
```

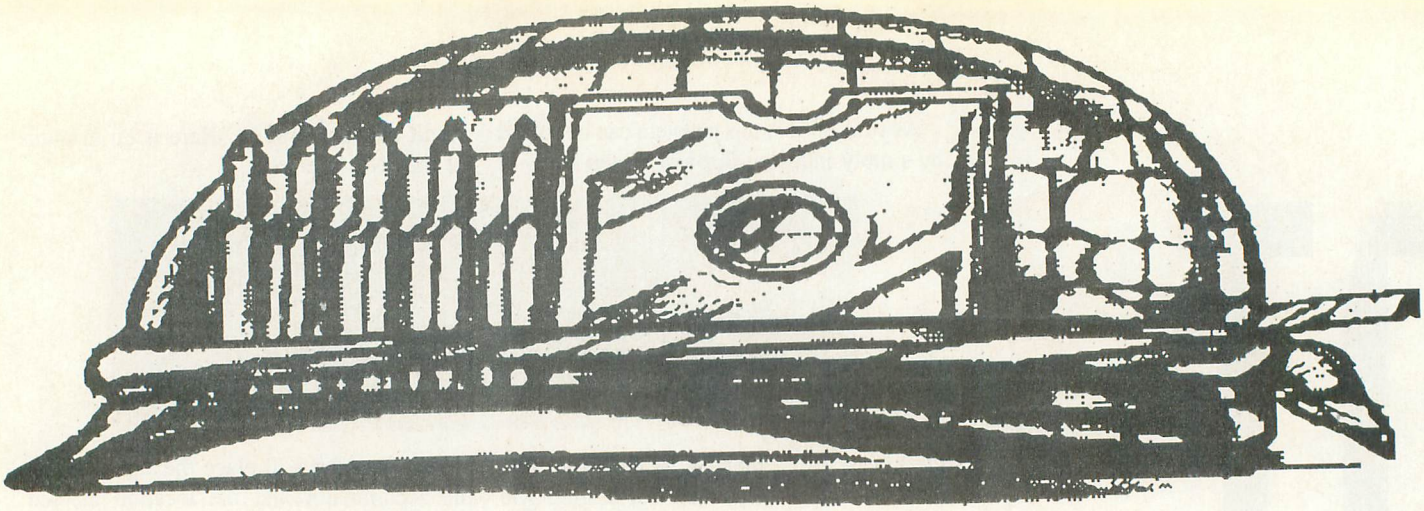
Here we're using the variable COUNT as a simple counter to control the addition of ADDVAL. We will add ADDVAL to itself five times. When finished, ADDVAL will be multiplied by 32. Let's walk through this example.

Line 20 clears the decimal mode so that we'll be working in binary mode.

Lines 30-40 initialize COUNT to five.

Lines 50-60 initialize ADDVAL to seven. When complete, this program will multiply seven by 32, with a result of 224 ($E0) in the accumulator.

Lines 70-100 add ADDVAL to itself, placing the result back in ADDVAL. This has the effect of multiplying ADDVAL by two each



WHEN YOU GET DEEPER INTO ASSEMBLY LANGUAGE, YOU'LL NEED TO MANIPULATE BYTES IN WAYS THAT BASIC CAN'T.

time it is done.

Line 110 decrements COUNT by one. When COUNT reaches zero, the Zero flag will be set. This will be our signal to stop.

Line 120 checks the Zero flag to see if all five multiplies have been done. If the Zero flag is *not* set, the program will branch (BNE) back to the label LOOP.

Line 130 breaks the program when all five iterations of the loop are complete.

Lines 140-150 define the one-byte storage areas ADDVAL and COUNT.

As you can see, the INC and DEC instructions can come in handy when you need a counter or want to add or subtract without affecting the accumulator. We have used the X and Y registers to perform counter functions, but if these registers are in use, you can always set up a byte and use the INC and DEC instructions instead.

Bit-Flipping

When you get deeper into assembly language, you'll need to manipulate bytes in ways that BASIC can't. Now we'll look at four instructions that allow a wide variety of ways to manipulate and test the contents of the accumulator. These instructions are AND, BIT, ORA and EOR.

```
     BYTE 1:  0 1 1 0 1 0 1 1
AND  BYTE 2:  1 0 1 1 0 0 0 1
              ---------------
     RESULT:  0 0 1 0 0 0 0 1
```

The above shows how the AND function works. As you can see, two bytes are used as inputs to the function. The corresponding bits of these two bytes are examined. If the bit of the first byte is one AND the bit of the second byte is one, the result for that bit will be one. Otherwise, that bit of the result will be set to zero. This process is repeated for all eight bits.

In 6502 assembly language, the AND function has the following eight formats:

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

In each of these formats, the accumulator is ANDed with the memory byte indicated in the operand. The result of the AND function is placed in the accumulator. The Sign and Zero flags are set according to the result.

The AND function is most often used to mask off certain bits of the accumulator or test bits to see if they are on.

Let's say you want to get a random number that does not exceed seven. You could use the code:

```
GETRND LDA RANDOM
       CMP #8
       BCS GETRND
```

This code gets a random number and checks to see if it is greater than seven. If it is, the program loops back to GETRND and tries again. This routine works, but it may need to try several times before it gets a good value.

We can perform the same function easily with the AND instruction. By using the AND instruction, only one try is necessary. It even takes less memory than the previous example. The code is:

```
LDA RANDOM
AND #07
```

This code masks the contents of the accumulator with the value seven. Below are three possible outcomes of the procedure. As you can see, none of them exceeds seven.

```
     BYTE:  1 0 0 1 1 1 0 1
AND  MASK:  0 0 0 0 0 1 1 1
            ---------------
   RESULT:  0 0 0 0 0 1 0 1 = 5

     BYTE:  1 1 1 1 0 1 1 1
AND  MASK:  0 0 0 0 0 1 1 1
            ---------------
   RESULT:  0 0 0 0 0 1 1 1 = 7

     BYTE:  0 0 0 1 0 0 0 0
AND  MASK:  0 0 0 0 0 1 1 1
            ---------------
   RESULT:  0 0 0 0 0 0 0 0 = 0
```

This is just one example of the use of the AND operation. We'll cover more uses in the future.

A companion to the AND function is the BIT (bit test) instruction. It performs almost the same function as AND, but changes only the status flags. BIT does *not* affect the contents of the accumulator. The primary function of the BIT operation is to test the contents of the accumulator. BIT has the following formats:

BIT nn (ABSOLUTE)
BIT n (ZERO PAGE)

Besides not changing the accumulator as a result of the AND operation, BIT handles the status flags differently. The ZERO flag is handled the same as AND. The SIGN and OVERFLOW flags are set to bits seven and six of the operand, respectively. This is a

strange twist, and I've not yet encountered a situation where I've used this odd flag setting. The following code shows a typical use of the BIT instruction.

```
        LDA BYTE
        BIT TESTBT
        BNE BITON
          .
          .
          .
BITON     .
          .
          .
          .
BYTE    *=*+1
TESTBT  .BYTE $01
        .END
```

This code uses the bit mask TESTBT to see if the one bit of the memory location labeled BYTE is set. The value contained in BYTE is placed in the accumulator, then the BIT instruction is executed. Since TESTBT is the location used by the BIT operand, the accumulator is set, and the result of the BIT operation will be a not-equal condition. In this case, the BNE instruction would cause the program to branch to the location BITON. Otherwise, the program would fall through to the code after the BNE instruction.

I personally don't use BIT instructions much. Unfortunately, the designers of the 6502 didn't allow for an immediate format of this instruction. As a result, you must set up all the masks you use somewhere in memory, making the operation a bit more cumbersome.

This OR That

Another bit-manipulating instruction used fairly often is the ORA (OR accumulator) operation. The formats of this instruction are:

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

Unlike the AND operator, which only sets the result bit when both input bits are one, the OR operator sets the result bit when either input bit is one. The following example shows how the OR function works:

```
    BYTE 1:  1 0 1 1 0 1 1 0
 OR BYTE 2:  0 1 0 1 0 0 1 0
             ---------------
    RESULT:  1 1 1 1 0 1 1 0
```

ANOTHER BIT-MANIPULATING INSTRUCTION USED FAIRLY OFTEN IS THE ORA (OR ACCUMULATOR) OPERATION.

As you can see, the OR operation sets the result bit if either bit one OR bit two is set. If both bits are off, the result bit will also be off. Like the AND operation, the ORA operation affects only the Sign and Zero flags.

The OR operation is used to turn on specific bits in a byte, most often in graphics handlers. The following code demonstrates how the OR instruction works.

```
    *=$600
    LDA #$4C        ;$4C IN ACCUM.
    ORA #$11        ;OR WITH $11
    ORA OR3         ;OR WITH $80
    BRK             ;ALL DONE!
OR3 .BYTE $80
    .END
```

Line 20 loads the accumulator with $4C (01001100 binary).

Line 30 ORs the accumulator with the contents of the memory location OR3. Since OR3 is defined as $80, the accumulator will be OR'd with 10000000 binary. After this instruction is executed, the accumulator will contain $DD (11011101 binary).

Line 50 stops the execution of the program. At this point you can see that the accumulator contains $DD.

An ANALOG Exclusive

The last accumulator manipulation instruction we're going to look at this time is EOR (exclusive-OR). This instruction works like OR except that when both input bits are set, the result bit will be turned off. The following example shows how EOR works:

```
    BYTE 1:  1 0 1 1 0 0 1 1
EOR BYTE 2:  1 0 0 1 1 0 1 0
             ---------------
    RESULT:  0 0 1 0 1 0 0 1
```

The EOR instruction is commonly used in graphics routines and for flipping the setting of bits in program flags. Let's see how the EOR instruction lets us flip bits. The following example shows the EOR function flipping all the bits of a byte to the opposite binary settings:

```
    BYTE 1:  1 0 1 1 0 0 0 1
EOR BYTE 2:  1 1 1 1 1 1 1 1
             ---------------
    RESULT:  0 1 0 0 1 1 1 0
```

No matter what the contents of byte one, if it is exclusive-OR'd with $FF (binary 11111111), the result of the operation will be the mirror-image of the first byte. The 6502 code necessary for this operation is:

```
LDA #$B1
EOR #$FF
```

What if we only want to flip a certain bit? The following example shows the flipping of only the four bit of byte one:

```
    BYTE 1:  1 0 1 1 0 0 0 1
EOR BYTE 2:  0 0 0 0 0 1 0 0
             ---------------
    RESULT:  1 0 1 1 0 1 0 1
```

As you can see, the bit has been flipped to a one. The equivalent 6502 code for this example is:

```
LDA #$B1
EOR #$04
```

The EOR operation is easy to use. All you need to do is determine which bits you want to flip and exclusive-OR the accumulator with the appropriate byte. Like the AND and ORA operation codes, EOR sets the Sign and Zero flags according to the result of the operation.

Problem Time.

Here are some good bit-manipulation problems for you to solve for next month.

In each of the following problems, you are given bit patterns before and after a bit-manipulation operation. You must determine (1) the operation (AND, ORA, EOR) and (2) the second bit pattern used to obtain the result. Some problems have two possible answers. These are indicated with a (2) to the right of the problem. If you've read carefully, these should be a snap to solve.

BYTE 1	OPN	BYTE 2	RESULT	ANS
01000011			01000001	(2)
11001011			10100010	
11110000			01000000	(2)
01010101			11111111	(2)
11001000			01111100	
11111111			11110001	(2)
00100100			10111000	
01000111			00010010	

Until next time, try developing some problems of your own. It's a good idea to try some addressing modes other than the ones used in this column. Next month we'll find out how to do simple multiplication and division! □

FAST MOVE

by John W. Little

To see *Fast Move* in action, type in Listing 1, save and run it. You'll see four players, two cars and two arrows. after the period described on the screen, the arrows will move by themselves, turning to face the directions in which they are moving.

This is a machine-language routine that will move players smoothly at speeds I can't even imagine a use for. The speed is completely variable, so it should be able to accommodate just about any need. An important benefit of the speed capability of this routine is its ability to move all four players, quickly and smoothly, at the same time. The movement routine is completely machine language and runs during the Vertical Blank Interrupt; only the setup is in BASIC. This allows those unfamiliar with assembly language to use it, and also allows changing of parameters while the BASIC program is running. It is compatible with all the 8-bit Ataris and is completely relocatable, except for the pointer tables in page six and three bytes in page zero.

Using *Fast Move*, the BASIC programmer can:

1) Move different players at different speeds;

2) Change player speed, while the program is running, from the keyboard or from within the BASIC program;

3) Choose horizontal or vertical (or both) wraparound or have the players stop at user-defined screen limits;

4) Change players' shapes to match the directions in which they are moving;

5) Combine players zero and one or two and three to make multicolored players;

6) Control player movement by poking memory locations with "joystick-type" values instead of using joysticks;

7) Choose single- or double-line resolution.

The first time you use Player/Missile (P/M) graphics, it seems like a complicated procedure; so many variables to define, so many things to remember, not to mention the fact that you first have to create the players and then put the value of each byte of the player into the program. After you go through all that, the result is usually disappointing as your creation jerks slowly across the screen.

The fact is, P/M graphics are complex, but with a good reference (there are numerous good books and magazine articles on the subject) and a memory map, the average BASIC programmer can make them work. *Fast Move* will not save you from having to know all that, because you have to tell *Fast Move* everything it needs to know in order to move your players just the way you want them to move. The big difference is, using *Fast Move*, your BASIC program will have a degree of control over your players that is possible only from machine language. *Fast Move* looks complex because it is versatile, but the results will more than make up for the effort required as your players move smoothly across the screen, just the way you pictured when you decided to try P/M graphics.

However, you don't have to know anything to run the demo program; so before we get into the nitty-gritty of using *Fast Move*, I'll describe it.

Running the Demo Program

To see *Fast Move* in action, type in Listing 1, save and run it. You'll see four players, two cars and two arrows. After the period described on the screen, the arrows will move by themselves, turning to face the directions in which they are moving. The cars will respond to joysticks zero and one. The speeds of the arrows have been set in the BASIC program, but the cars' speeds can be changed by keys one through five on the keyboard.

Players are normally of a single color, but when two differently colored players are combined to form a single player, any area of overlap can be a third color. To see the multicolored players, with some really terrible sound effects thrown in, stop the demo by hitting the system reset key. Delete Lines 360-470. This removes the shape-changing capability. A little familiarity with the program will show that changing the shape of combined players would be rather complex, although certainly possible. Now add these lines:

```
195 POKE 1612,1: POKE 1613,1
225 POKE 1614,PEEK(1545)-PEEK
(1544):POKE 1615,PEEK(1547)-P
EEK(1546)
508 IF PEEK(632)<>15 THEN SOU
ND 1,30,10,8:GOTO 510
509 SOUND 1,0,0,0
510 POKE 1603,INT(RND(0)*15):
IF PEEK(1603)=14 THEN SOUND 0
,2,6,15:FOR I=0 TO 50:NEXT I:
SOUND 0,0,0,0: GOTO 508
511 IF PEEK(1603)=10 THEN SOU
ND 0,12,6,15:FOR I=0 TO 50:NE
XT I: SOUND 0,0,0,0:GOTO 508
512 IF PEEK(1603)=11 THEN SOU
ND 0,22,6,15:FOR I=0 TO 50:NE
XT I:SOUND 0,0,0,0:GOTO 508
513 IF PEEK(1603)=9 THEN SOUN
D 0,32,6,15:FOR I=0 TO 50:NEX
T I:SOUND 0,0,0,0:GOTO 508
514 IF PEEK(1603)=13 THEN SOU
ND 0,42,6,15:FOR I=0 TO 50:NE
XT I:SOUND 0,0,0,0:GOTO 508
515 IF PEEK(1603)=5 THEN SOUN
D 0,52,6,15:FOR I=0 TO 50:NEX
T I:SOUND 0,0,0,0:GOTO 508
516 IF PEEK(1603)=7 THEN SOUN
D 0,62,6,15:FOR I=0 TO 50:NEX
T I:SOUND 0,0,0,0:GOTO 508
517 IF PEEK(1603)=6 THEN SOUN
D 0,72,6,15:FOR I=0 TO 50:NEX
T I:SOUND 0,0,0,0:GOTO 508
518 FOR I=0 TO 25:NEXT I:GOTO
 508
```

Now change these lines:

```
170 POKE 623,33
210 POKE 1544,95:POKE 1545,95
:POKE 1546,135:POKE 1547,140:
FOR I=0 to 3:POKE 53248+I,PEE
K(1544+I):NEXT I
220 POKE 1556,15:POKE 1557,15
:POKE 1558,9:POKE 1559,9
230 POKE 1610,1:POKE 1606,0:P
OKE 1607,0:POKE 1608,2:POKE 1
609,2
280 FOR A=P0+48 TO P0+48+14:R
EAD B:POKE A,B:NEXT A
290 FOR A=P1+48 TO P1+48+14:R
EAD B:POKE A,B:NEXT A
520 DATA 0,231,231,231,126,36
,255,255,255,255,255,0,0,0,0
530 DATA 0,0,0,0,0,0,0,255,25
5,255,153,153,153,153,129
540 DATA 0,135,135,255,65,127
,64,64,64
550 DATA 0,225,225,255,130,25
4,2,2,2
```

After making these changes, run the program again. The players will begin their movement more quickly now because the data for all the different shapes doesn't have to load this time. The players are no longer cars and arrows: one is orange, green and blue, the other is red, white and green. The second one will move around by itself and emit terrible noises, while the first will respond to stick zero and emit a high tone while moving. To see how the third color is dependent on player priority, change Line 170 back to POKE 623,1 and run again. Or, to see what the individual players looked like before they were combined, delete Line 195. The players will break in half, one vertically, one horizontally.

So far, the players have been in single-line resolution. To see the combined players in double-line resolution, change the following lines:

```
175 POKE 559,34+8
200 POKE 1550,15:POKE 1551,11
0:POKE 1548,50:POKE 1549,200
240 POKE 1605,1:POKE 1554,PEE
K(1550):POKE 1555,PEEK(1551)
270 P0=PMBASE+512:P1=P0+128:
P2=P1+128:P3=P2+128
```

Run the program now, and you'll see that the players have stretched to twice their original height. At this size, they can travel the screen vertically in no time. Slower speeds and shorter players seem to be in order when using double-line resolution.

Using *Fast Move*

Probably the best way to explain the use of *Fast Move* is to go through the demo program, line by line. Each feature will be explained as we come to the program line that sets up that feature.

But first let me explain the basic premise behind the shape-changing ability: Instead of creating one shape for a player and putting its data into the P/M area, you define several shapes (such as the arrows that point in eight different directions) and store the data at predefined locations outside the P/M area. To display the player, you copy its initial shape into the P/M area. To change the shape, you simply copy the new shape into the P/M area, right on top of the old one. *Fast Move* handles the copying for you; you control it with POKES in BASIC.

Line 80 creates a new RAMTOP and sets aside 16 pages of protected memory above it. This is enough space for the P/M area, the ML routine (*Fast Move* itself), if you want to put it there (the demo puts it in a string), data for all the player shapes and 800 bytes of unused memory, which must be left directly above the new RAMTOP. (In some situations, the OS will scroll screen data into this area and overwrite anything that happens to be there.)

Line 90 sets the graphics mode, which forces the OS to set up a new display list under the new RAMTOP. For this demo, any graphics mode will work.

Line 100 tells the OS where to find PMBASE. For single-line resolution P/M, PMBASE has to be an address that is divisible by 2048. Our reserved area contains two such addresses, RAMTOP and RAMTOP+2048. Since the 800 bytes directly above RAMTOP are subject to clearing by the OS, we choose RAMTOP+2048. And, since the contents of RAMTOP and PMBASE must be expressed in "pages" of 256 bytes each for the OS, PMBASE is first defined as RAMTOP+8 pages and this is poked into 54279. However, our program requires PMBASE to be an actual address, so it is redefined by multiplying it by 256.

Lines 110 and 120 define the data that make up the string in Line 130. Lines 130 and 140 zero out any residual garbage from memory above RAMTOP, before the program is placed there, using a short ML routine called ZE+. This routine zeroes a page at a time, and the format is "X=USR(ADR(ZE$),[beginning address],[# of pages to clear]." So Line 140 clears 16 pages starting at RAMTOP. Notice that Line 140 also clears page 6. *Fast Move* uses the first 87 bytes of page 6 as pointers and indicators of certain conditions, such as wraparound. If your own program will use part of page six, use ZE$ at your own discretion.

In Line 150, you inform *Fast Move* whether you're using a 400/800 machine or the XL/XE type. It uses this information to know how to handle players 2 and 3, which have no corresponding joysticks in the XL/XE machines. See the section below on combining players, for an advantage of the XL/XE machines.

Line 160 turns on players but no missiles. The use of missiles is so application-dependent that including a handler for them would almost certainly limit their uses.

Line 170 gives all players priority over all playfields. In the demo showing multicolored players, 32 was added to the origi-

nal priority value, allowing the third color to appear.

Line 175 enables DMA (Direct Memory Access) and defines its use with a standard-size playfield, players only, and single-line resolution. For double-line resolution, we subtract 16 from the value poked.

Line 180 defines the colors of the players.

Line 190 defines all players as being of normal width. Poking an address with 1 will double that player's width, and poking with 3 will quadruple it. Players do not all have to be of the same width.

Line 200 defines the screen limits of player movement. Poking addresses 1548-1551 will set, respectively, left, right, top and bottom limits. For example, players may be restricted to the bottom half of the screen by poking 1550 with a value equal to half a player's vertical range. For single-line resolution players, this would be 128; for double-line, it would be 64.

Line 210 defines the initial horizontal positions of the players. Each player requires two memory locations, the OS's Horizontal Position Register and a pseudo-register. They must both contain the same values at all times because the OS registers cannot be read, only written to. *Fast Move* uses Locations 1544-1547 (for players 0-3) for the pseudo-registers, and the FOR-NEXT loop pokes the same values into the OS registers. Positions 50-200 will be on the playfield, on most TVs. Poking positions lower or higher will cause the player to be off-screen, from where it may be moved onto the screen by the user program. Once on the screen, it will be subject to the screen limits that were set in line 200.

Line 220 defines the length of each player. Locations 1556-1559 are poked with the vertical length of players 0-3, respectively. Assigning a player a length of 0 will crash the program.

Line 230 sets the initial speed (location 1610) of players whose speed will be changeable from the keyboard. These players must have their individual speed locations (1606-1609 for players 0-3 respectively) poked with 0. Players that will not need to have their speeds changed from the keyboard, or that will move at a speed different from the others, should have their speed poked directly into their individual locations. For example, in the demo, the initial keyboard speed is 1, players 0 and 1 will move at keyboard speed (their speed locations are poked with 0), player 2 will have a speed of 2, and player 3 will have a speed of 3. These locations may also be poked with different values from your own program while it is running. *Location 1610 must contain a positive nonzero value unless all four individual speed locations contain positive nonzero values*, or the program will crash.

In line 240, address 1605 is poked with 0 to indicate single-line resolution, or with 1 for double-line. The rest of the line must stay the same.

Line 250 establishes the vertical difference between the top movement limit and the bottom, for use in the vertical wraparound function.

Line 255 tells the ML routine how many players it must move. If you wish to control some players by means other than *Fast Move*, *Fast Move* must have the lower-numbered ones. For example, if you poke 1616,3, *Fast Move* will take players 0-2 and leave player 3 for you to control.

In Line 260, poke 1599,1 for horizontal wraparound. Poke with 0 to stop at screen limits. Address 1600 controls vertical wraparound.

Line 270 describes the standard P/M setup for single-line resolution.

Line 280 reads the data defining player 0's shape into the player's initial vertical position. The format is: FOR A=[PLAYER#]+[POSITION] TO [PLAYER#]+[POSITION]+[PLAYER LENGTH-1]. For example, player 0 is ten bytes long, and when it first appears it covers positions 48-57. The position of the first byte is considered to be the position of the player. Lines 290-310 define players 1-3. POSITION must be between one- and 255-player length for single-line resolution; for double-line, it must be between one- and 127-player length. Vertical positions from about 30 to about 220 will be on the screen for single-line resolution (15 to 110 for double-line). Players initially positioned off the screen may be moved onto the screen by the user program, after which they will be subject to the screen limits.

Lines 320-350 poke the initial positions of players 0-3 into their respective pointers in page six.

Fast Move will allow your players to stay one shape during your whole program, or to change shape whenever they change direction. In order to accomplish the latter, you must first create all of the shapes you wish to use, up to a possible maximum of eight per player, and store them in specified memory locations so the ML routine can copy them onto the screen when they are needed.

Line 360 reads the data for all eight shapes of player 0 and pokes it into a designated part of the reserved area; if you refer back to Line 220, you'll see that all the players were defined as being ten lines long. So 80 bytes of data are poked into the storage area for player 0. Lines 370-390 do the same for players 1-3.

Note that each player's data is placed into a different page, beginning at the second byte of the page. Because each player does need a separate page, and there are exactly three pages in the "unused" area between PMBASE and the missile area, the data for the first three players was stored here, with the fourth player data taking the page before PMBASE.

So, at 256 bytes per page, player 0's data starts at (PMBASE+2 pages+1 byte), player 1 starts at (PMBASE+1 page+1 byte), player 2 starts at (PMBASE+1 byte), and player 3 starts at (PMBASE-1 page+1 byte). These are not mandatory locations for the shape data, and in fact may not be used with double-line resolution, because then the unused P/M area is only 384 bytes long. Any other data storage location must be coordinated with lines 400-470 in the BASIC program. The four pages used for data storage must be consecutive in memory, with player 0 having the highest page and player 3 the lowest.

Line 400 installs the page six pointers to tell the ML routine where each of the player 0 shapes are stored. Since each player has a separate page for its shape data, each pointer in page six has to contain only the low byte of the address it points to. There is one pointer in each of eight addresses from 1561 to 1568, and the shapes they point to, respectively, starting with 1561, are up, up-left, up-right, down, down-left, down-right, left, right. Because Line 400 is so important to the operation of the shape-changing, we'll dissect the line.

POSHAPE=1561: This cannot be changed unless you also change the ML routine.

FOR I=PMBASE+513 TO PMBASE+513+X STEP Y: This FOR-NEXT loop computes the first data address of each shape for player 0. X=7*length of player 0. Y=length of player 0. The first time through the loop, the address of the up shape is poked into POSHAPE; next time through, the address of the up-left shape is poked into POSHAPE+1, and so on, until the first address of each shape has been poked into the pointer table. Note that in Line 360, 80 bytes of data were read and poked into the data storage area for player 0. In Line 400, because the player is ten bytes long, every tenth address of that data storage area is poked into the page six pointers for player 0. These pointers tell *Fast Move* where to find each one of the shapes.

POKE POSHAPE,I-INT(I/256)256: As stated above, only the low byte of each data ad-

dress is poked into the pointers.

Lines 410-460 install the shape pointers for players 1-3.

Line 470 tells *Fast Move* the page number where player 0's data is stored. It figures out the page numbers for the other three players.

To use players that do not change shape, lines may be deleted, as in the combined players demo, or the appropriate pointers in addresses 1561-1592 may be poked with 0s. If shape-changing is desired, but eight different shapes are not (say, if you wanted only horizontal and vertical shapes, but no diagonal ones), zeroing the pointers for diagonal movement will prevent the shapes from changing during diagonal movement. Specifically, zeroing the diagonal pointers for player 0 would mean poking 0 into addresses 1562, 1563, 1565, and 1566. And, of course, these pointers may be changed by your BASIC program while it is running.

Line 490 jumps to the subroutine, which reads the ML routine into the string MOVE$.

In Line 500, the USR call shows an argument of ADR(MOVE$)+11. If loading the ML routine somewhere other than in a string, simply use an argument equal to the load address+11.

Line 510 uses addresses 1603 and 1604, the Alternate Movement Indicators (AMI) for players 2 and 3, to move the arrows around on the demo. Since all of the values returned by a joystick are positive integers equal to or less than 15, Poking the AMI's with random numbers less than 15 causes the players to move at random. When a number is poked that is not a valid joystick value, the player does not move. AMI's for players 0 and 1 are 1601 and 1602, respectively.

When *Fast Move* is used with an Atari 400 or 800, and players are being moved via the AMI's, a joystick plugged into any port will override the AMI. With an XL/XE, the sticks for players 0 and 1 will override the AMI's but players 2 and 3 may be moved only via the AMIs.

Data Lines 520-550 contain the data used to display all the players at the beginning of the demo (one player per DATA statement). This is the data that is read in Lines 280-310. If your BASIC program did not need all four players on-screen at the beginning, some or all of these lines would not be here. If shape changing is not used, the leading and trailing zeroes in the DATA statements are not required. However, when using shape-changing, all shapes for the same player must be the same length. This can require DATA statements to be filled out with 0s, especially when some shapes are vertical and some are horizontal for the same player. Also, the first byte of each shape must be 0. Let's look at the DATA statements for player 0.

Lines 560-630 contain the eight shapes for player 0. Line 560 contains nine bytes that actually create the up shape and a first byte of 0. So, player 0 must be defined in Lines 220 and 280 as being ten bytes long. Line 570 has only six bytes that actually create the up-left shape, so it must be filled out on both ends with 0s. And, in Line 520, the initial shape must have enough 0s to make it a total of ten bytes long. Note that the DATA statement for the first shape of player 0 does not match the DATA statement for player 0 in Line 520, because the first shape for each player is the up shape, and player 0's initial shape in the opening screen was facing to the right. Lines 280-310 merely provided an initial display, because the players have to start somewhere. They could have been placed off-screen and moved on-screen by *Fast Move*.

Lines 32000-32590 read the ML routine into MOVE$.

Combining Players

The XL/XE machines may have a slight disadvantage as far as the number of joystick ports they have, but with *Fast Move*, they have a slight advantage. If the XL/XE indicator is allowed to contain a 0 (indicating an 800 machine) when used on an XL or XE, and if both "combine" indicators are set to 1 in Line 195, all players will respond to stick 0 or AMI 0. This means they may all be combined into one six-color player, which requires only one joystick or AMI to control it. (Nothing earth-shattering, but it could be handy.)

When combining all four players in this fashion, there is one limitation that must be observed in addition to the other instructions laid out below for combining players: The even-numbered players must have the same horizontal position, and the odd-numbered players must have the same horizontal position. For example, all four players may have their Horizontal Position Registers set at 100 (they would be stacked more or less vertically) or players 0 and 2 could be at horizontal position 100 while players 1 and 3 could be at Position 105.

Now let's look at the lines we changed in the demo to see the combined players.

Line 170 changes player priority so the overlap area will be a third color.

Line 195 enables players 0 and 1 to combine by poking 1 into Location 1612, and enables players 2 and 3 to combine by poking Location 1613. Two players may be positioned together on the screen, but if the appropriate address is not poked with 1, the two players will separate when moved.

Line 210 changes the horizontal positions of players 1 and 3 so they will combine on the screen with players 0 and 2.

Line 220 changes the lengths of players 0 and 1 to match their new appearance.

Line 225 establishes the difference between the horizontal positions of two players that have been combined. Location 1614 is for players 0 and 1, 1615 is for players 2 and 3. If this line is included when players are not being combined, it won't hurt anything.

Line 230 changes the speed of player 3 to match that of player 2.

Lines 280 and 290 read the new longer length of players 0 and 1 into P/M memory.

Lines 508-518 add the sound effects to the combined-players demo. Lines 508 and 509 turn on the high tone whenever player 0-1 is moved, and the other lines turn on different sounds for each direction player 2-3 moves.

Lines 520-550 contain data for the new appearance of all four players. Notice the large number of 0s in players 0 and 1, though they don't change shape while moving. This is because two players that are combined must be the same length, and they must begin at the same vertical position and end at the same vertical position. Horizontal position is unimportant; two players can be combined and not even be touching each other.

Double-Line Resolution

Line 175 enables DMA for double-line resolution.

Line 200 sets new vertical screen limits because a double-line resolution player has a vertical range of only 128 bytes instead of 256.

Line 240 tells the ML routine we are using double-line resolution.

Line 270 is the standard P/M setup for double-line resolution.

I've tried to make this program as versatile and universal as possible. I would be interested in hearing comments, questions, or complaints about *Fast Move*, and will do my best to answer them. A

```
WO 1 REM ******************************
CV 2 REM *      FASTMOVE V.1.01      *
HM 3 REM *     by John W. Little     *
ZD 4 REM *                           *
BT 5 REM *       COPYRIGHT 1989      *
PM 6 REM *    BY ANALOG COMPUTING    *
WU 7 REM ******************************
BC 12 REM
BM 14 REM **INTRO SCREENS**
BK 16 REM
RS 18 DIM ZE$(28):DIM MOVE$(1072)
TB 20 GRAPHICS 17:POSITION 0,5
WD 30 ? #6;"     FASTMOVE      ":? #6:? #
   6;"  by john little":FOR I=0 TO 2000:N
   EXT I
NM 40 GRAPHICS 17:POSITION 0,5:? #6;"AFTE
   R PLAYERS APPEAR":? #6:? #6
YH 60 REM FOLLOWING SPECIAL CHARACTERS AR
   E [CNTRL][T],[CNTRL][U]
NF 70 ? #6;"ml routine will load":? #6;"i
   n about 8■ seconds":FOR I=0 TO 3000:NE
   XT I
```

```
BI 72 REM
NM 74 REM **MAKE ROOM FOR P/M AND ROUTINE
   **
BQ 76 REM
PZ 80 RAMTOP=PEEK(106):POKE 106,RAMTOP-16
   :RAMTOP=PEEK(106)
QA 90 GRAPHICS 17
IO 100 PMBASE=RAMTOP+8:POKE 54279,PMBASE:
   PMBASE=PMBASE*256
HL 110 REM STRING DATA 104,104,133,209,10
   4,133,208,104,104,133,207,160,0,152,17
   0,145,208,200,208,251,230,209,232
FC 120 REM DATA 228,207,208,244,96
QV 130 ZE$(1,28)="hh,Qh,Phh,O ♥*x,PHP♠Foh
   dOPT♦"
RU 140 X=USR(ADR(ZE$),1536,1):X=USR(ADR(Z
   E$),RAMTOP*256,16)
RA 142 REM
ZM 144 REM **400/800 OR XL/XE?**
RM 146 REM
UT 150 POKE 1611,0:REM 1=XL/XE
RC 152 REM
GS 154 REM **P/M INFO**
RO 156 REM
UY 160 POKE 53277,2
IO 170 POKE 623,1
GB 175 POKE 559,34+8+16
HQ 180 POKE 704,38:POKE 705,151:POKE 706,
   55:POKE 707,200
RI 182 REM
KU 184 REM **SET-UP FOR ML ROUTINE**
RU 186 REM
KJ 190 POKE 53256,0:POKE 53257,0:POKE 532
   58,0:POKE 53259,0
FT 200 POKE 1550,30:POKE 1551,220:POKE 15
   48,50:POKE 1549,200
JD 210 POKE 1544,95:POKE 1545,115:POKE 15
   46,135:POKE 1547,155:FOR I=0 TO 3:POKE
    53248+I,PEEK(1544+I):NEXT I
AK 220 POKE 1556,10:POKE 1557,10:POKE 155
   8,9:POKE 1559,9
FV 230 POKE 1610,1:POKE 1606,0:POKE 1607,
   0:POKE 1608,2:POKE 1609,3
JL 240 POKE 1605,0:POKE 1554,PEEK(1550):P
   OKE 1555,PEEK(1551)
WU 250 POKE 1598,(PEEK(1551)-PEEK(1550))
OE 255 POKE 1616,4
VS 260 POKE 1599,1:POKE 1600,1
RF 262 REM
SN 264 REM **SET-UP P/M AREA**
RR 266 REM
DZ 270 P0=PMBASE+1024:P1=P0+256:P2=P1+256
   :P3=P2+256
DF 280 FOR A=P0+48 TO P0+48+9:READ B:POKE
    A,B:NEXT A
EO 290 FOR A=P1+48 TO P1+48+9:READ B:POKE
    A,B:NEXT A
EE 300 FOR A=P2+48 TO P2+48+8:READ B:POKE
    A,B:NEXT A
FN 310 FOR A=P3+48 TO P3+48+8:READ B:POKE
    A,B:NEXT A
QW 312 REM
DI 314 REM **SET-UP PLAYER LOCATION POINT
   ERS**
RI 316 REM
ZF 320 POKE 1537,INT((P0+48)/256):POKE 15
   36,P0+48-(PEEK(1537)*256)-1
KF 330 POKE 1539,INT((P1+48)/256):POKE 15
   38,P1+48-(PEEK(1539)*256)-1
JA 340 POKE 1541,INT((P2+48)/256):POKE 15
   40,P2+48-(PEEK(1541)*256)-1
UA 350 POKE 1543,INT((P3+48)/256):POKE 15
   42,P3+48-(PEEK(1543)*256)-1
RE 352 REM
JO 354 REM **SET-UP PLAYER SHAPES**
RQ 356 REM
VE 360 FOR I=PMBASE+513 TO PMBASE+513+79:
   READ A:POKE I,A:NEXT I
FU 370 FOR I=PMBASE+257 TO PMBASE+257+79:
   READ A:POKE I,A:NEXT I
HE 380 FOR I=PMBASE+1 TO PMBASE+1+71:READ
    A:POKE I,A:NEXT I
UC 390 FOR I=PMBASE-255 TO PMBASE-255+71:
   READ A:POKE I,A:NEXT I
RM 392 REM
HU 394 REM **SHAPE-FINDING POINTERS**
RY 396 REM
WN 400 P0SHAPE=1561:FOR I=PMBASE+513 TO P
   MBASE+513+70 STEP 10:POKE P0SHAPE,I-IN
   T(I/256)*256:P0SHAPE=P0SHAPE+1:NEXT I
RD 410 P1SHAPE=1569:FOR I=PMBASE+257 TO P
   MBASE+257+70 STEP 10:POKE P1SHAPE,I-IN
   T(I/256)*256:P1SHAPE=P1SHAPE+1
FY 420 NEXT I
SA 430 P2SHAPE=1577:FOR I=PMBASE+1 TO PMB
   ASE+1+63 STEP 9:POKE P2SHAPE,I-INT(I/2
   56)*256:P2SHAPE=P2SHAPE+1
GC 440 NEXT I
OO 450 P3SHAPE=1585:FOR I=PMBASE-255 TO P
   MBASE-255+63 STEP 9:POKE P3SHAPE,I-INT
   (I/256)*256:P3SHAPE=P3SHAPE+1
GG 460 NEXT I
PG 470 POKE 1593,INT((PMBASE+513)/256)
RF 480 REM
LA 482 REM **READ IN DATA FOR ML ROUTINE*
   *
RR 484 REM
UP 490 GOSUB 32000
RW 495 REM
PA 497 REM **START ML ROUTINE**
SI 499 REM
ZO 500 A=USR(ADR(MOVE$),ADR(MOVE$)+11)
QW 502 REM
PJ 504 REM **DEMO**
RI 506 REM
EC 510 POKE 1603,INT(RND(0)*15):FOR I=0 T
   O 50:NEXT I:POKE 1604,INT(RND(0)*15):F
   OR I=0 TO 50:NEXT I:GOTO 510
QY 512 REM
RE 514 REM
PG 520 DATA 0,68,255,255,255,255,255,68,0
   ,0
PI 530 DATA 0,68,255,255,255,255,255,68,0
   ,0
UQ 540 DATA 0,16,16,16,16,146,84,56,16
US 550 DATA 0,16,16,16,16,146,84,56,16
TG 560 DATA 0,60,60,94,60,60,60,60,94,60
QO 570 DATA 0,0,40,240,122,188,30,40,0,0
EQ 580 DATA 0,0,20,15,94,61,120,20,0,0
RW 590 DATA 0,60,94,60,60,60,60,94,60,60
EB 600 DATA 0,0,20,15,94,61,120,20,0,0
QD 610 DATA 0,0,40,240,122,188,30,40,0,0
CI 620 DATA 0,34,255,255,255,255,255,34,0
   ,0
PJ 630 DATA 0,68,255,255,255,255,255,68,0
   ,0
TD 640 DATA 0,60,60,94,60,60,60,60,94,60
QL 650 DATA 0,0,40,240,122,188,30,40,0,0
EN 660 DATA 0,0,20,15,94,61,120,20,0,0
RT 670 DATA 0,60,94,60,60,60,60,94,60,60
ER 680 DATA 0,0,20,15,94,61,120,20,0,0
QT 690 DATA 0,0,40,240,122,188,30,40,0,0
CF 700 DATA 0,34,255,255,255,255,255,34,0
   ,0
PG 710 DATA 0,68,255,255,255,255,255,68,0
   ,0
PV 720 DATA 0,16,56,84,146,16,16,16,16
IY 730 DATA 0,240,192,160,144,8,4,2,1

AY 740 DATA 0,15,3,5,9,16,32,64,128
UU 750 DATA 0,16,16,16,16,146,84,56,16
QM 760 DATA 0,1,2,4,8,144,160,192,240
WE 770 DATA 0,128,64,32,16,9,5,3,15
QO 780 DATA 0,16,32,64,255,64,32,16,0
RE 790 DATA 0,8,4,2,255,2,4,8,0
PS 800 DATA 0,16,56,84,146,16,16,16,16
IV 810 DATA 0,240,192,160,144,8,4,2,1
AV 820 DATA 0,15,3,5,9,16,32,64,128
UR 830 DATA 0,16,16,16,16,146,84,56,16
QJ 840 DATA 0,1,2,4,8,144,160,192,240
WB 850 DATA 0,128,64,32,16,9,5,3,15
QL 860 DATA 0,16,32,64,255,64,32,16,0
RB 870 DATA 0,8,4,2,255,2,4,8,0
KT 32000 RESTORE 32060
GX 32010 FOR I=1 TO 1072:READ Z:MOVE$(I,I
   )=CHR$(Z):NEXT I
CY 32020 READ Z:IF Z<>-1 THEN ? "ERROR IN
    CODE! CHECK DATA STATEMENTS!":END
DT 32040 RETURN
TC 32060 DATA 104,104,170,104,168,169,7,3
   2,92,228,96,216,169,0,141,17,6,168,173
   ,252
CN 32070 DATA 2,201,31,208,7,169,1,141,74
   ,6,208,42,201,30,208,7,169,2,141,74
JC 32080 DATA 6,208,31,201,26,208,7,169,3
   ,141,74,6,208,20,201,24,208,7,169,4
FA 32090 DATA 141,74,6,208,9,201,29,208,5
   ,169,5,141,74,6,173,18,6,141,14,6
RD 32100 DATA 173,19,6,141,15,6,192,1,208
   ,8,173,76,6,240,13,136,240,10,192,3
KH 32110 DATA 208,27,173,77,6,240,1,136,1
   73,69,6,240,16,173,14,6,56,233,128,141
BB 32120 DATA 14,6,24,109,62,6,141,15,6,1
   73,75,6,240,4,192,2,176,7,185,120
XC 32130 DATA 2,201,15,208,5,185,65,6,240
   ,106,141,24,6,172,17,6,185,20,6,133
XW 32140 DATA 205,185,70,6,240,8,141,16,6
   ,208,9,24,144,156,173,74,6,141,16,6
JG 32150 DATA 152,24,109,17,6,141,83,6,13
   3,203,169,6,133,204,160,0,177,203,141,
   81
JD 32160 DATA 6,200,177,203,141,82,6,162,
   0,173,24,6,201,14,240,43,232,201,10,24
   0
WO 32170 DATA 38,232,201,6,240,33,232,201
   ,13,240,28,232,201,9,240,23,232,201,5,
   240
PB 32180 DATA 18,232,201,11,240,13,232,20
   1,7,240,8,208,3,24,144,171,24,144,125,
   172
QN 32190 DATA 17,6,208,24,189,25,6,240,31
   ,205,58,6,240,26,141,58,6,141,84,6
AP 32200 DATA 173,57,6,141,85,6,208,102,1
   92,1,208,27,189,33,6,240,3,205,59,6
XZ 32210 DATA 240,125,141,59,6,141,84,6,1
   73,57,6,56,233,1,141,85,6,208,71,192
YB 32220 DATA 2,208,30,189,41,6,240,99,20
   5,60,6,240,94,141,60,6,141,84,6,173
DZ 32230 DATA 57,6,56,233,2,141,85,6,208,
   40,24,144,156,192,3,208,70,189,49,6
HU 32240 DATA 240,65,205,61,6,240,60,141,
   61,6,141,84,6,173,57,6,56,233,3,141
UH 32250 DATA 85,6,208,6,24,144,54,24,144
   ,216,160,0,200,173,84,6,133,203,173,85
AT 32260 DATA 6,133,204,177,203,141,86,6,
   173,81,6,133,203,173,82,6,133,204,173,
   86
QT 32270 DATA 6,145,203,196,205,208,221,2
   24,3,144,24,224,6,144,102,240,103,224,
   7,240
WU 32280 DATA 102,238,17,6,172,17,6,204,8
   0,6,208,191,76,98,228,173,81,6,205,14
DH 32290 DATA 6,240,83,162,0,160,1,173,81
   ,6,133,203,173,82,6,133,204,177,203,13
   6
LK 32300 DATA 145,203,196,205,200,200,144
   ,245,206,81,6,160,0,173,83,6,133,203,1
   69,6
FP 32310 DATA 133,204,173,81,6,145,203,20
   5,14,6,240,34,232,236,16,6,208,203,172
   ,17
PY 32320 DATA 6,173,24,6,201,10,240,12,20
   1,6,240,11,208,163,24,144,160,24,144,8
   9
NG 32330 DATA 24,144,82,24,144,81,173,64,
   6,240,146,164,205,173,81,6,133,203,173
   ,82
VY 32340 DATA 6,133,204,177,203,170,169,0
   ,145,203,173,81,6,24,109,62,6,56,229,2
   05
SP 32350 DATA 141,81,6,133,203,138,145,20
   3,136,208,16,173,83,6,133,203,169,6,13
   3,204
SF 32360 DATA 173,81,6,145,203,208,183,17
   3,81,6,56,237,62,6,24,101,205,141,81,6
PB 32370 DATA 208,187,24,144,165,144,85,1
   44,86,173,81,6,24,101,205,205,15,6,240
   ,78
RO 32380 DATA 162,0,164,205,173,81,6,133,
   203,173,82,6,133,204,177,203,200,145,2
   03,136
HF 32390 DATA 136,16,247,238,81,6,160,0,1
   73,83,6,133,203,169,6,133,204,173,81,6
JT 32400 DATA 145,203,24,101,205,205,15,6
   ,240,28,232,236,16,6,208,202,172,17,6,
   173
GX 32410 DATA 24,6,201,9,240,95,201,5,240
   ,88,208,166,24,144,86,24,144,80,173,64
TI 32420 DATA 6,240,155,164,205,173,81,6,
   133,203,173,82,6,133,204,177,203,170,1
   69,0
BB 32430 DATA 145,203,173,81,6,56,237,62,
   6,24,101,205,141,81,6,133,203,138,145,
   203
AF 32440 DATA 136,208,16,173,83,6,133,203
   ,169,6,133,204,173,81,6,145,203,208,18
   7,173
PI 32450 DATA 81,6,24,109,62,6,56,229,205
   ,141,81,6,208,187,169,1,208,168,24,144
FN 32460 DATA 104,162,0,172,17,6,192,1,20
   8,15,173,76,6,240,36,136,185,8,6,24
EA 32470 DATA 109,78,6,208,17,192,3,208,2
   2,173,77,6,240,17,136,185,8,6,24,109
FZ 32480 DATA 79,6,200,153,8,6,153,0,208,
   208,199,185,8,6,205,12,6,208,8,173
HO 32490 DATA 63,6,240,186,173,13,6,56,23
   3,1,153,8,6,153,0,208,192,0,208,31
WO 32500 DATA 173,76,6,240,59,185,8,6,24,
   105,1,205,13,6,208,48,185,8,6,56
PB 32510 DATA 237,78,6,208,33,24,144,44,2
   4,144,151,192,2,208,29,173,77,6,240,24
ZV 32520 DATA 185,8,6,24,105,1,205,13,6,2
   08,13,185,8,6,56,237,79,6,153,8
KO 32530 DATA 6,153,0,208,232,236,16,6,20
   8,214,240,166,162,0,172,17,6,192,1,208
TQ 32540 DATA 15,173,76,6,240,73,136,185,
   8,6,24,109,78,6,208,17,192,3,208,24
MB 32550 DATA 173,77,6,240,54,136,185,8,6
   ,24,109,79,6,200,153,8,6,153,0,208
RC 32560 DATA 169,0,240,202,192,0,208,14,
   173,76,6,240,26,185,8,6,24,109,78,6
AQ 32570 DATA 208,12,173,77,6,240,12,185,
   8,6,24,109,79,6,205,13,6,240,8,185
ZW 32580 DATA 8,6,205,13,6,208,8,173,63,6
   ,240,154,173,12,6,24,105,1,153,8
YD 32590 DATA 6,153,0,208,232,236,16,6,20
   8,143,240,134,-1

NG 32330 DATA 24,144,82,24,144,81,173,64,
   6,240,146,164,205,173,81,6,133,203,173
   ,82
VY 32340 DATA 6,133,204,177,203,170,169,0
   ,145,203,173,81,6,24,109,62,6,56,229,2
   05
SP 32350 DATA 141,81,6,133,203,138,145,20
   3,136,208,16,173,83,6,133,203,169,6,13
   3,204
SF 32360 DATA 173,81,6,145,203,208,183,17
   3,81,6,56,237,62,6,24,101,205,141,81,6
PB 32370 DATA 208,187,24,144,165,144,85,1
   44,86,173,81,6,24,101,205,205,15,6,240
   ,78
RO 32380 DATA 162,0,164,205,173,81,6,133,
   203,173,82,6,133,204,177,203,200,145,2
   03,136
```

```
HF 32390 DATA 136,16,247,238,81,6,160,0,1
   73,83,6,133,203,169,6,133,204,173,81,6
JT 32400 DATA 145,203,24,101,205,205,15,6
   ,240,28,232,236,16,6,208,202,172,17,6,
   173
GX 32410 DATA 24,6,201,9,240,95,201,5,240
   ,88,208,166,24,144,86,24,144,80,173,64
TI 32420 DATA 6,240,155,164,205,173,81,6,
   133,203,173,82,6,133,204,177,203,170,1
   69,0
BB 32430 DATA 145,203,173,81,6,56,237,62,
   6,24,101,205,141,81,6,133,203,138,145,
   203
AF 32440 DATA 136,208,16,173,83,6,133,203
   ,169,6,133,204,173,81,6,145,203,208,18
   7,173
PI 32450 DATA 81,6,24,109,62,6,56,229,205
   ,141,81,6,208,187,169,1,208,168,24,144
FN 32460 DATA 104,162,0,172,17,6,192,1,20
   8,15,173,76,6,240,36,136,185,8,6,24
EA 32470 DATA 109,78,6,208,17,192,3,208,2
   2,173,77,6,240,17,136,185,8,6,24,109
FZ 32480 DATA 79,6,200,153,8,6,153,0,208,
   208,199,185,8,6,205,12,6,208,8,173
HO 32490 DATA 63,6,240,186,173,13,6,56,23
   3,1,153,8,6,153,0,208,192,0,208,31
WO 32500 DATA 173,76,6,240,59,185,8,6,24,
   105,1,205,13,6,208,48,185,8,6,56
PB 32510 DATA 237,78,6,208,33,24,144,44,2
   4,144,151,192,2,208,29,173,77,6,240,24
ZV 32520 DATA 185,8,6,24,105,1,205,13,6,2
   08,13,185,8,6,56,237,79,6,153,8
KO 32530 DATA 6,153,0,208,232,236,16,6,20
   8,214,240,166,162,0,172,17,6,192,1,208
TQ 32540 DATA 15,173,76,6,240,73,136,185,
   8,6,24,109,78,6,208,17,192,3,208,24
MB 32550 DATA 173,77,6,240,54,136,185,8,6
   ,24,109,79,6,200,153,8,6,153,0,208
RC 32560 DATA 169,0,240,202,192,0,208,14,
   173,76,6,240,26,185,8,6,24,109,78,6
AQ 32570 DATA 208,12,173,77,6,240,12,185,
   8,6,24,109,79,6,205,13,6,240,8,185
ZW 32580 DATA 8,6,205,13,6,208,8,173,63,6
   ,240,154,173,12,6,24,105,1,153,8
YD 32590 DATA 6,153,0,208,232,236,16,6,20
   8,143,240,134,-1
```

LISTING 2: BASIC

```
01        .OPT NO LIST
0100 ;FASTMOVE V1.01 BY J.LITTLE
0105 ;COPYRIGHT 1989
0106 ;BY ANALOG COMPUTING
0110 ;***************
0120 ;MOVES ALL 4 PLAYERS DURING VBI
0130 ;WITH OR WITHOUT JOYSTICK
0140 ;***************
0150 ;*FAST
0160 ;*SMOOTH
0170 ;*RELOCATABLE
0180 ;*VARIABLE SPEED FOR EACH PLAYER
0190 ;***************
0200 ;CHANGES SHAPES OF PLAYERS
0210 ;TO MATCH DIRECTION OF MOVEMENT
0220 ;***************
0250 ;OPTIONAL HORIZONTAL OR VERTICAL
0260 ;
0270 ;***************
0280 ;DOUBLE OR SINGLE LINE RESOLUTION
0290 ;***************
0300 ;ALL PARAMETERS POKED IN BASIC
0310 ;***************
0320 ;PLAYERS 0,1 OR 2,3 MAY BE
0330 ;COMBINED TO MAKE MULTI-COLOR
0335 ;PLAYERS
0340 ;***************
0350 ;WORKS WITH 400/800 OR XL/XE OS'S
0360 ;***************
0370      ;
0380      ;
0390 ;***************
0400 ;* OS REGISTERS *
0410 ;***************
0420      ;
0430 HPOS0 = $D000    ;P0 HORZ POSITION
0440 STICK = $0278    ;P0 JOYSTICK
0470      ;
0480      ;
0490 ;************
0500 ;* ZERO PAGE *
0510 ;************
0520      ;
0530 PAGE0 = $CB      ;TEMP
0540 LENGTH = $CD     ;CURRENT PLAYER
0570      ;
0580 ;*******************
0590 ;* PAGE SIX CONSTANTS *
0600 ;*******************
0610      ;
0615 ;POINTERS TO ADDRESSES OF PLAYERS
0620 P0  =    $0600   ;PLAYER0
0630 P1  =    $0602   ;PLAYER1
0640 P2  =    $0604   ;PLAYER2
0650 P3  =    $0606   ;PLAYER3
0655      ;
0656 ;HORZ POSITION PSEUDO-REGISTERS
0660 HVAR0 = $0608    ;PLAYER0
0670 HVAR1 = $0609    ;PLAYER1
0680 HVAR2 = $060A    ;PLAYER2
0685 HVAR3 = $060B    ;PLAYER3
0686      ;
0687 ;PLAYERS' SCREEN BOUNDARIES
0688 SCRNLEFT = $060C ;LEFT
0689 SCRNRIGHT = $060D ;RIGHT
0690 SCRNTOP = $060E ;TOP
0691 SCRNBTM = $060F ;BOTTOM
0692      ;
0693 ;FOR CHANGING SPEED OF PLAYERS
0694 SPDVAR = $0610
0695 ;FOR ROTATING PLAYERS DURING VBI
0696 PCOUNTER = $0611
0697      ;
0698 SAVETOP = $0612 ;TEMP STORAGE
0699 SAVEBTM = $0613 ;TEMP STORAGE
0700 LEN =    $0614   ;LENGTH OF PLYR 0
0710 LEN1 =   $0615   ;LENGTH OF PLYR 1
0720 LEN2 =   $0616   ;LENGTH OF PLYR 2
0730 LEN3 =   $0617   ;LENGTH OF PLYR 3
0735 TEMPSTICK = $0618 ;HOLD STICK VAL
0740 ;
0750 ;POINTERS TO ADDRESSES FOR
0755 ;DIRECTIONAL SHAPES
0760 P0SHAPE0 = $0619 ;UP SHAPE
0770 P0SHAPE1 = $061A ;UP LEFT SHAPE
0780 P0SHAPE2 = $061B ;UP RIGHT
0790 P0SHAPE3 = $061C ;DOWN
0800 P0SHAPE4 = $061D ;DOWN LEFT
0810 P0SHAPE5 = $061E ;DOWN RIGHT
0820 P0SHAPE6 = $061F ;LEFT
0830 P0SHAPE7 = $0620 ;RIGHT
0840 P1SHAPE0 = $0621 ;UP SHAPE
0850 P1SHAPE1 = $0622 ;UP LEFT SHAPE
0860 P1SHAPE2 = $0623 ;UP RIGHT
0870 P1SHAPE3 = $0624 ;DOWN
0880 P1SHAPE4 = $0625 ;DOWN LEFT
0890 P1SHAPE5 = $0626 ;DOWN RIGHT
0900 P1SHAPE6 = $0627 ;LEFT
0910 P1SHAPE7 = $0628 ;RIGHT
0920 P2SHAPE0 = $0629 ;UP SHAPE
0930 P2SHAPE1 = $062A ;UP LEFT SHAPE
0940 P2SHAPE2 = $062B ;UP RIGHT
0950 P2SHAPE3 = $062C ;DOWN
0960 P2SHAPE4 = $062D ;DOWN LEFT
0970 P2SHAPE5 = $062E ;DOWN RIGHT
0980 P2SHAPE6 = $062F ;LEFT
0990 P2SHAPE7 = $0630 ;RIGHT
1000 P3SHAPE0 = $0631 ;UP SHAPE
1010 P3SHAPE1 = $0632 ;UP LEFT SHAPE
1020 P3SHAPE2 = $0633 ;UP RIGHT
1030 P3SHAPE3 = $0634 ;DOWN
1040 P3SHAPE4 = $0635 ;DOWN LEFT
1050 P3SHAPE5 = $0636 ;DOWN RIGHT
1060 P3SHAPE6 = $0637 ;LEFT
1070 P3SHAPE7 = $0638 ;RIGHT
1080 ;
1085 ;HI-BYTE OF SHAPE ADDRESSES
1090 SHAPEPAGE = $0639
1100 ;
1105 ;LO-BYTE ADDRESS OF CURRENT SHAPE
1110 P0SHADR = $063A ;PLAYER 0
1120 P1SHADR = $063B ;PLAYER 1
1130 P2SHADR = $063C ;PLAYER 2
1140 P3SHADR = $063D ;PLAYER 3
1200 ;
1250 ;
1260 VDIFF = $063E    ;SCRNBTM-SCRNTOP
1270 HWRAP = $063F    ;HORZ WRAP-AROUND
1280 VWRAP = $0640    ;VERT WRAP-AROUND
1290 ;
1300 LOOK0 = $0641    ;PLAYER0 AMI
1310 LOOK1 = $0642    ;PLAYER1 AMI
1320 LOOK2 = $0643    ;PLAYER2 AMI
1330 LOOK3 = $0644    ;PLAYER3 AMI
1340 ;
1341 DBLRES = $0645   ;DOUBLE-LINE RES
1342 ;
1350 SPDVAR0 = $0646 ;SPEED PLAYER0
1360 SPDVAR1 = $0647
1370 SPDVAR2 = $0648
1380 SPDVAR3 = $0649
1390 ;
1400 COMMON = $064A   ;STORE SPDVAR
1405 XLIND = $064B    ;XL/XE OS
1410 COMBIN01 = $064C ;COMBINE P0&P1
1420 COMBIN23 = $064D
1422      ;
1425 ;HORZ DIFFERENCE BETWEEN P0 & P1
1430 DIFF01 = $064E
1440 DIFF23 = $064F
1450 NUMPLYRS = $0650 ;# OF PLAYERS
1460 LOCATION = $0651 ;OF CURRNT PLYR
1470 POINTER = $0653 ;TO LOCATION
1480 PS  =    $0654   ;CURRNT SHAPE ADR
1485 TEMP =   $0656
1490 ;
1495 ;
1500 ***TO SAVE SPACE IN PAGE ZERO,
1501 ;"LOCATION", "POINTER", AND "PS"
1502 ;ARE ROTATED INTO PAGE 0 AS
1503 ;NEEDED FOR INDIRECT ADDRESSING.
1505 ;
1506 ;
1510      .TITLE "FASTMOVE V1.01"
1520      .SET 2,77
1530      .SET 3,0
1540      .SET 4,66
1550      .TAB 12,16,24
1560      *=  37672    ;ORIGIN
1570      ;
1580      PLA
1590      ;
1660      ;
1670 ;INITIALIZE VBI ROUTINE
1680      PLA          ;START ADR HIBYTE
1690      TAX          ;
1700      PLA          ;LOBYTE
1710      TAY          ;
1720      LDA #7
1730      JSR $E45C
1740      RTS
1750      ;
1760 ;START OF VBI ROUTINE
1770 START CLD
1780      LDA #0       ;FIRST PASS...
1790      STA PCOUNTER ;OF VBI
1800      TAY          ;STICK INDEX
1810      LDA 764      ;SPEED CHANGE?
1820 ;
1830 ;
1840 ;COMMON SPEED CHOICE FROM KEYBD
1850 ;
1860 ONE CMP #31       ;SPEED = 1?
1870      BNE TWO      ;IF NOT,CHECK 2
1880      LDA #1       ;IF SO,CHANGE...
1890      STA COMMON   ;SPEED
1900      BNE LOOK     ;THEN CHECK STICK
1910 TWO CMP #30
1920      BNE THREE
1930      LDA #2
1940      STA COMMON
1950      BNE LOOK
1960 THREE CMP #26
1970      BNE FOUR
1980      LDA #3
1990      STA COMMON
2000      BNE LOOK
2010 FOUR CMP #24
2020      BNE FIVE
2030      LDA #4
2040      STA COMMON
2050      BNE LOOK
2060 FIVE CMP #29
2070      BNE LOOK
2080      LDA #5
2090      STA COMMON
2100 ;
2110 ;
2120 ;---BEGINNING OF MOVEMENT LOOP---
2130 ;
2140 ;
2150 ;-IF 2 PLAYERS ARE COMBINED,
2151 ;ARRANGEMENTS MUST BE MADE FIRST-
2160 ;
2170 LOOK LDA SAVETOP ;REINSTATE ORIG
2180      STA SCRNTOP ;SCRN TOP,BOTTOM
2190      LDA SAVEBTM ;IN CASE THEY
2200      STA SCRNBTM ;WERE CHANGED.
2205 ;
2210      CPY #1       ;IS THIS PLAYER1?
```

```
2220      BNE CPY3     ;NO,CHECK FOR P3
2230      LDA COMBIN01 ;P0,P1 COMBINED?
2240      BEQ RESOL    ;NO.DOUBLE RES?
2250      DEY          ;YES,READ STICK0
2260      BEQ RESOL    ;REALLY A JMP
2270 CPY3 CPY #3       ;IS THIS P3?
2280      BNE STKCHK   ;NO,GO READ STICK
2290      LDA COMBIN23 ;P2,P3 COMBINED?
2300      BEQ RESOL    ;NO
2310      DEY          ;YES,READ STICK2
2320 ;
2330 ;
2340 ;-IF P1 OR P3 AND DOUBLE-LINE RES
2341 ;ADJUST SCRNTOP AND SCRNBTM
2342 ;TO COMPENSATE
2343 ; FOR FACT THAT PLAYERS DON'T
2344 ;START AT BEGINNING OF PAGE.-
2360 ;
2370 RESOL LDA DBLRES ;DBLE-LINE RES?
2380      BEQ STKCHK   ;NO.
2390      LDA SCRNTOP  ;RAISE SCRNTOP...
2400      SEC          ;128 BYTES SO...
2410      SBC #128     ;UPPER AND LOWER
2420      STA SCRNTOP  ;SCREEN LIMIT...
2430      CLC          ;CHECK WILL WORK.
2440      ADC VDIFF    ;
2450      STA SCRNBTM  ;
2460 ;
2470 ;
2480 ;-CHECK PLYR#(0-4) TO SEE IF
2485 ;CURRENT PLAYER SHOULD BE MOVED-
2490 ;
2500 STKCHK LDA $064B ;XL/XE COMPUTER?
2520      BEQ OLDOS    ;NO,400 OR 800
2530      CPY #2       ;YES;IF P2 OR P3,
2540      BCS XLOS     ;SKIP STICK READ.
2550 OLDOS LDA STICK,Y ;CHECK STICK
2560      CMP #15      ;STICK CENTERED?
2570      BNE STKMOV   ;NO
2580 XLOS LDA LOOK0,Y  ;CHECK AMI
2600      BEQ INTERMRETURN ;NO MOVE
2610 STKMOV STA TEMPSTICK ;SAVE STICK
2620 ;
2630 ;
2640 ;-SET LENGTH,SPEED, AND LOCATION-
2670      LDY PCOUNTER ;
2680      LDA LEN,Y    ;SET LENGTH FOR
2690      STA LENGTH   ;CURRENT PLAYER.
2700      ;
2710      LDA SPDVAR0,Y ;INDIVIDUAL...
2720      BEQ COMSPD   ;SPEED SETTING...
2730      STA SPDVAR   ;OR COMMON SPEED.
2740      BNE UNCOM    ;
2750      ;
2760 INTERMLOOK2 CLC
2770      BCC LOOK
2780      ;
2790 COMSPD LDA COMMON ;KEYBD SPEED
2800      STA SPDVAR   ;INTO SPDVAR.
2810 UNCOM TYA         ;
2820      CLC          ;
2830      ADC PCOUNTER ;INCREMENT...
2840      STA POINTER  ;POINTER...
2841      STA PAGE0
2842      LDA #6
2843      STA PAGE0+1
2850      LDY #0       ;TO...
2860      LDA (PAGE0),Y ;CURRENT...
2870      STA LOCATION ;PLAYER...
2880      INY          ;ADDRESS.
2890      LDA (PAGE0),Y ;
2900      STA LOCATION+1 ;DONE!
2910      ;
2920 ;
2930 ;-DETERMINE DIRECTION OF MOVEMENT
2935 ;AND SHAPE REQUIRED-
2940 ;
2950 ;
2960      LDX #0       ;INDX FOR SHAPE.
2970      LDA TEMPSTICK ;SAVED VALUE.
2980      CMP #14      ;GOING UP?
3000      BEQ SHAPE    ;CHANGING ROUTINE
3010      INX
3020      CMP #10      ;UP LEFT?
3030      BEQ SHAPE
3040      INX
3050      CMP #6       ;UP RIGHT?
3060      BEQ SHAPE
3070      INX
3080      CMP #13      ;DOWN?
3090      BEQ SHAPE
3100      INX
3110      CMP #9       ;DOWN LEFT?
3120      BEQ SHAPE
3130      INX
3140      CMP #5       ;DOWN RIGHT?
3150      BEQ SHAPE
3160      INX
3170      CMP #11      ;LEFT?
3180      BEQ SHAPE
3190      INX
3200      CMP #7       ;RIGHT?
3210      BEQ SHAPE
3220      BNE INTERMRETURN
3290      ;
3300 INTERMLOOK CLC
3310      BCC INTERMLOOK2
3320      ;
3330 INTERMRETURN CLC
3340      BCC INTERMRETURN1
3350      ;
3360      ;
3370 SHAPE LDY PCOUNTER ;P0 CURRENT?
3380      BNE SHAPE1   ;NO,CHECK P1
3382      ;
3385 ;CHECK PLYR0 SHAPE POINTER FOR
3386 ;ADDRESS CONTAINING SHAPE DATA.
3390      LDA P0SHAPE0,X ;
3400      BEQ SSJ      ;ZERO=NO CHANGE.
3410      CMP P0SHADR  ;NEW SHAPE=OLD?
3420      BEQ SSJ
3430      STA P0SHADR  ;
3440      STA PS       ;NO,CHANGE SHAPE
3450      LDA SHAPEPAGE ;
3460      STA PS+1     ;
3470      BNE CHANGE   ;GO GET CHANGED.
3480 SHAPE1 CPY #1
3490      BNE SHAPE2
3500      LDA P1SHAPE0,X
3510      BEQ SSJ
3520      CMP P1SHADR
3530 SSJ BEQ SAMESHAPE
3540      STA P1SHADR
3550      STA PS
3560      LDA SHAPEPAGE
3570      SEC
3580      SBC #1
3590      STA PS+1
3600      BNE CHANGE
3610 SHAPE2 CPY #2
3620      BNE SHAPE3
3630      LDA P2SHAPE0,X
3640      BEQ SAMESHAPE
3650      CMP P2SHADR
3660      BEQ SAMESHAPE
3670      STA P2SHADR
3680      STA PS
3690      LDA SHAPEPAGE
3700      SEC
3710      SBC #2
3720      STA PS+1
3730      BNE CHANGE
3731      ;
3732      ;
3733 INTERMLOOK3 CLC
3734      BCC INTERMLOOK
3735      ;
3736      ;
3740 SHAPE3 CPY #3
3750      BNE SAMESHAPE
3760      LDA P3SHAPE0,X
3770      BEQ SAMESHAPE
3780      CMP P3SHADR
3790      BEQ SAMESHAPE
3800      STA P3SHADR
3810      STA PS
3820      LDA SHAPEPAGE
3830      SEC
3840      SBC #3
3850      STA PS+1
3860      BNE CHANGE
3870      ;
3880      ;
3890 INTERMRETURN1 CLC
3900      BCC RETURN
3910      ;
3920 INTERMLOOK1 CLC
3930      BCC INTERMLOOK3
3940      ;
3950 CHANGE LDY #0    ;REPLACE...
3960 LOOP INY         ;OLD...
3961      LDA PS
3962      STA PAGE0
3963      LDA PS+1
3964      STA PAGE0+1
3970      LDA (PAGE0),Y ;SHAPE...
3971      STA TEMP
3972      LDA LOCATION
3973      STA PAGE0
3974      LDA LOCATION+1
3975      STA PAGE0+1
3976      LDA TEMP
3980      STA (PAGE0),Y ;WITH...
3990      CPY LENGTH   ;NEW...
4000      BNE LOOP     ;SHAPE.
4010 SAMESHAPE CPX #3 ;DIRECTION INDEX
4020      BCC MOVEUP   ;FOR X<3
4070      CPX #6       ;FOR 2<X>6
4080      BCC INTERMOVE ;DOWN
4140      BEQ INTERMLEFT ;FOR X=6
4150      CPX #7
4160      BEQ INTERMRIGHT
4170      ;
4180 ;--RETURN ROUTINE;PLACED IN
4181 ;MIDDLE OF PROGRAM TO FACILITATE
4182 ;BRANCHING. RETURNS TO CHECK
4190 ;NEXT PLAYER OR EXITS VBI AFTER
4191 ;LAST PLAYER--
4200 ;
4210 RETURN INC PCOUNTER
4220      LDY PCOUNTER ;WHO'S NEXT?
4230      CPY NUMPLYRS ;ALL PLAYERS?
4240      BNE INTERMLOOK1 ;NO,GO AGAIN.
4250      JMP $E462    ;RETURN FROM VBI
4260      ;
4270 ;
4280 ;-UPWARD MOVEMENT ROUTINE-
4290 ;
4300 ;
4310 MOVEUP LDA LOCATION ;IS PLYR AT
4320      CMP SCRNTOP  ;TOP OF SCRN?
4330      BEQ UWRAP    ;WRAP-AROUND?
4340      LDX #0       ;NO,INIT SPDVAR
4350 UPSPEED LDY #1    ;INIT BYTE-COUNT
4351      LDA LOCATION
4352      STA PAGE0
4353      LDA LOCATION+1
4354      STA PAGE0+1
4360 UPMORE LDA (PAGE0),Y ;MOVE...
4370      DEY          ;ONE BYTE...
4380      STA (PAGE0),Y ;UPWARD
4390      CPY LENGTH   ;FINISHED MOVING?
4400      INY
4410      INY
4420      BCC UPMORE   ;NO,KEEP MOVING.
4430      DEC LOCATION ;MOVE FINISHED.
4440      LDY #0       ;STORE NEW...
4441      LDA POINTER
4442      STA PAGE0
4443      LDA #6
4444      STA PAGE0+1
4450      LDA LOCATION ;PLAYER...
4460      STA (PAGE0),Y ;ADDRESS.
4470      CMP SCRNTOP  ;TOP OF SCRN?
4480      BEQ UWRAP    ;YES,WRAP-AROUND?
4490      INX          ;IF NOT,
4500      CPX SPDVAR   ;CHECK SPEED AND
4510      BNE UPSPEED  ;MOVE AGAIN.
4520      ;
4530      LDY PCOUNTER ;VERT MOVE DONE.
4540      LDA TEMPSTICK ;CHECK FOR
4550      CMP #10      ;DIAGONAL MOVE
4560      BEQ INTERMLEFT
4570      CMP #6
4580      BEQ INTERMRIGHT
4590      BNE RETURN
4600      ;
4610      ;
4620 INTERMRETURN4 CLC
4630      BCC RETURN
4640      ;
4650 INTERMOVE CLC
4660      BCC MOVEDOWN
4670      ;
4680 INTERMLEFT CLC
4690      BCC INTERMLEFT6
4700      ;
4710 INTERMRIGHT CLC
4720      BCC INTERMRIGHT6
4730      ;
4740 ;
4750 ;-DETERMINE IF VERTICAL
4751 ;WRAP-AROUND IS DESIRED AND
4752 ;IMPLEMENT FOR UPWARD MOVEMENT-
4760 ;
4770 ;
4780 UWRAP LDA VWRAP ;WRAP-AROUND?
4790      BEQ RETURN   ;NO
4800      LDY LENGTH
4801 ULOOP LDA LOCATION
4802      STA PAGE0
4803      LDA LOCATION+1
4804      STA PAGE0+1
4805 ;
4810 ;ONE BYTE AT A TIME...
4811 ;STORE PLAYER DATA IN XREG...
4812 ;AND ZERO OUT...
4813 ;LOCATION BEING MOVED FROM.
4820      LDA (PAGE0),Y ;
4830      TAX          ;
4840      LDA #0       ;
4850      STA (PAGE0),Y ;
4855 ;
4860      LDA LOCATION ;OLD LOCATION...
4870      CLC          ;
4880      ADC VDIFF    ;PLUS SCRN LENGTH
4890      SEC          ;MINUS
4900      SBC LENGTH   ;PLAYER LENGTH...
4910      STA LOCATION ;= NEW LOCATION.
4912      STA PAGE0
4920      TXA          ;STORE DATA IN
4930      STA (PAGE0),Y ;NEW LOCATION.
4940      ;
4950      DEY          ;MOVE FINISHED?
4960      BNE ULOOP2   ;IF NOT,PREPARE
```

```
4970      LDA POINTER ;TO MOVE
4971      STA PAGE0   ;NEXT BYTE.
4972      LDA #6
4973      STA PAGE0+1
4980      LDA LOCATION ;PUT NEW ADR IN
4990      STA (PAGE0),Y ;ADR POINTER.
5000      ;
5005 ;
5010      BNE INTERMRETURN4 ;NEXT PLYR.
5020 ULOOP2 LDA LOCATION ;NEW LOCATION
5030      SEC          ;MINUS
5040      SBC VDIFF    ;SCREEN HEIGHT
5050      CLC          ;PLUS
5060      ADC LENGTH   ;PLAYER LENGTH
5070      STA LOCATION ;=OLD LOCATION.
5080      BNE ULOOP    ;MOVE NEXT BYTE.
5090      ;
5100      ;
5110 INTERMRETURN3 CLC
5120      BCC INTERMRETURN4
5130      ;
5131      ;
5132 INTERMLEFT6 BCC INTERMLEFT1
5133      ;
5134      ;
5135 INTERMRIGHT6 BCC INTERMRIGHT1
5136      ;
5140      ;
5150 ;-DOWNWARD MOVEMENT ROUTINE-
5160 ;
5170 ;
5180 MOVEDOWN LDA LOCATION ;IS PLYR AT
5190      CLC
5200      ADC LENGTH
5210      CMP SCRNBTM ;BOTTOM OF SCRN?
5220      BEQ DWRAP   ;YES,VERT WRAP?
5230      LDX #0      ;NO,INIT SPD INDX
5235 ;
5240 DOWNSPEED LDY LENGTH ;MOVE FIRST
5241      LDA LOCATION
5242      STA PAGE0
5243      LDA LOCATION+1
5244      STA PAGE0+1
5250 DOWNMORE LDA (PAGE0),Y ;BYTE OF..
5260      INY          ;PLAYER...
5270      STA (PAGE0),Y ;DOWN.
5280      DEY
5290      DEY
5300      BPL DOWNMORE ;GET NEXT BYTE.
5310      INC LOCATION ;MOVE FINISHED.
5320      LDY #0       ;STORE NEW...
5321      LDA POINTER
5322      STA PAGE0
5323      LDA #6
5324      STA PAGE0+1
5330      LDA LOCATION ;PLAYER...
5340      STA (PAGE0),Y ;ADDRESS.
5350      ;
5360      CLC
5370      ADC LENGTH
5380      CMP SCRNBTM ;AT SCREENBOTTOM?
5390      BEQ DWRAP   ;IF SO,WRAP
5400      INX         ;IF NOT,
5410      CPX SPDVAR  ;CHECK SPEED
5420      BNE DOWNSPEED ;AND MOVE AGAIN
5430      ;
5440      LDY PCOUNTER
5450      LDA TEMPSTICK ;CHECK STICK
5460      CMP #9      ;FOR DIAG MOVE
5470      BEQ LEFT
5480      CMP #5
5490      BEQ INTERMRIGHT2
5500 JTR2 BNE INTERMRETURN3
5510      ;
5520 INTERMLEFT1 CLC
5530      BCC LEFT
5540      ;
5550 INTERMRIGHT1 CLC
5560      BCC INTERMRIGHT2
5570 ;
5580 ;
5590 ;-IF VERTICAL WRAP-AROUND DESIRED
5595 ;IMPLEMENT FOR DOWNWARD MOVEMENT-
5600 ;
5610 ;
5620      ;
5630 DWRAP LDA VWRAP ;WRAP-AROUND?
5640      BEQ INTERMRETURN3 ;NO WRAP
5650 ;WORKS SAME AS UPWARD WRAP.
5660      LDY LENGTH
5670 DLOOP LDA LOCATION
5671      STA PAGE0
5672      LDA LOCATION+1
5673      STA PAGE0+1
5680      LDA (PAGE0),Y
5690      TAX
5700      LDA #0
5710      STA (PAGE0),Y
5720      LDA LOCATION
5730      SEC
5740      SBC VDIFF
5750      CLC
5760      ADC LENGTH
5770      STA LOCATION
5771      STA PAGE0
5780      TXA
5790      STA (PAGE0),Y
5800      DEY
5810      BNE DLOOP2
5811      LDA POINTER
5812      STA PAGE0
5813      LDA #6
5814      STA PAGE0+1
5830      LDA LOCATION
5840      STA (PAGE0),Y
5850      BNE JTR2
5860 DLOOP2 LDA LOCATION
5870      CLC
5880      ADC VDIFF
5890      SEC
5900      SBC LENGTH
5910      STA LOCATION
5920      BNE DLOOP
5930      ;
5940 INTERMRETURN2 LDA #1
5950      BNE JTR2
5960      ;
5970 INTERMRIGHT2 CLC
5980      BCC INTERMRIGHT3
5990      ;
6000 ;
6010 ;-MOVE LEFT ROUTINE-
6020 ;
6030 ;
6040 LEFT LDX #0      ;INIT SPEED INDEX
6050      LDY PCOUNTER ;INIT PLAYR INDX
6060 ;
6070 ;-ARRANGE FOR COMBINED PLAYERS-
6080 ;
6090 LEFT4 CPY #1     ;IS THIS PLAYER1?
6100      BNE CPY3L   ;IF NOT,CHK P3.
6110      LDA COMBIN01 ;P0,P1 COMBINED?
6120      BEQ LEFT1   ;NO.
6130      DEY         ;YES.
6140      LDA HVAR0,Y ;FIND PLAYER 0...
6150      CLC         ;AND ATTACH P1...
6160      ADC DIFF01  ;AT OFFSET.
6170      BNE LEFT3   ;REALLY A JMP.
6180 CPY3L CPY #3
6190      BNE LEFT1
6200      LDA COMBIN23
6210      BEQ LEFT1
6220      DEY
6230      LDA HVAR0,Y
6240      CLC
6250      ADC DIFF23
6260 LEFT3 INY
6270      STA HVAR0,Y ;STORE P1 OR P3
6280      STA HPOS0,Y ;IN NEW POSITION.
6290      BNE INTERMRETURN2 ;NEXT PLYR
6300 ;
6310 ;-IF P1 OR P3 ARE COMBINED
6311 ;PLAYERS,THEIR LOOP ENDS HERE-
6320 ;
6330 ;-GENERAL LEFT MOVEMENT ROUTINE-
6340      ;
6350 LEFT1 LDA HVAR0,Y ;CHECK HORZ POS
6360      CMP SCRNLEFT ;IS PLYR AT EDGE
6370      BNE LEFT2   ;NO,KEEP MOVING
6380      LDA HWRAP   ;WRAP-AROUND?
6390 BTR BEQ INTERMRETURN2 ;NO
6400      LDA SCRNRIGHT ;YES,PLACE PLYR
6401      ;           AT RIGHT EDGE
6410 LEFT2 SEC        ;
6420      SBC #1      ;MOVE LEFT 1
6430      STA HVAR0,Y ;SAVE NEW POS
6440      STA HPOS0,Y ;SET POS REG
6450 ;
6460 ;
6470 ;-ADJUSTING P0 OR P2 POSITION FOR
6471 ;WRAP-AROUND OF COMBINED PLAYER-
6480 ;
6490 ;
6500 CPY0L CPY #0     ;IS THIS P0?
6510      BNE CPY2L   ;NO,CHK FOR P2.
6520      LDA COMBIN01 ;P0,1 COMBINED?
6530      BEQ SPDL    ;NO.
6540      LDA HVAR0,Y ;IS PLAYER 0...
6550      CLC         ;
6560      ADC #1      ;
6570      CMP SCRNRIGHT ;AT RIGHT EDGE?
6580      BNE SPDL    ;NO.
6590      LDA HVAR0,Y ;YES, MOVE P0...
6600      SEC         ;LEFT ENOUGH...
6610      SBC DIFF01  ;TO FIT P1 IN.
6620      BNE LEFT5   ;JMP
6630      ;
6640      ;
6650 INTERMRIGHT3 CLC
6660      BCC RIGHT
6670      ;
6680 INTERMLEFT4 CLC
6690      BCC LEFT4
6700      ;
6710      ;
6720 CPY2L CPY #2     ;IS THIS PLAYER2?
6730      BNE SPDL
6740      LDA COMBIN23
6750      BEQ SPDL
6760      LDA HVAR0,Y
6770      CLC
6780      ADC #1
6790      CMP SCRNRIGHT
6800      BNE SPDL
6810      LDA HVAR0,Y
6820      SEC
6830      SBC DIFF23
6840 LEFT5 STA HVAR0,Y
6850      STA HPOS0,Y
6860 ;
6870 ;
6880 SPDL INX
6890      CPX SPDVAR  ;CHECK SPEED
6900      BNE INTERMLEFT4 ;MOVE AGAIN
6910 JTR BEQ BTR      ;MOVE NEXT PLAYER
6920      ;
6930      ;
6940 ;-MOVE RIGHT ROUTINE-
6950 ;
6960 ;-IF P1 OR P3 IS COMBINED,TACK IT
6961 ;ONTO P0 OR P2, THEN RETURN. THIS
6962 ;PART OF RIGHT ROUTINE IS
6963 ;IDENTICAL TO LEFT ROUTINE.-
6980 ;
6990 RIGHT LDX #0
7000      LDY PCOUNTER
7010 RIGHT4 CPY #1
7020      BNE CPY3R
7030      LDA COMBIN01
7040      BEQ RIGHT1
7050      DEY
7060      LDA HVAR0,Y
7070      CLC
7080      ADC DIFF01
7090      BNE RIGHT3
7100 CPY3R CPY #3
7110      BNE CPY0R
7120      LDA COMBIN23
7130      BEQ RIGHT1
7140      DEY
7150      LDA HVAR0,Y
7160      CLC
7170      ADC DIFF23
7180 RIGHT3 INY
7190      STA HVAR0,Y
7200      STA HPOS0,Y
7210      LDA #0
7220      BEQ JTR
7230 ;
7240 ;
7250 ;FIND SCRNRIGHT IF COMBINED P0,P2
7260 ;
7270 ;
7280 CPY0R CPY #0     ;IS THIS PLAYER0?
7290      BNE CPY2R   ;NO,CHK P2.
7300      LDA COMBIN01 ;P0,P1 COMBINED?
7310      BEQ RIGHT1  ;NO.
7320      LDA HVAR0,Y ;CURRENT HPOS...
7330      CLC         ;PLUS...
7340      ADC DIFF01  ;P0-P1 OFFSET.
7350      BNE RIGHT5  ;P0P1 @ SCRNRIGHT
7360 CPY2R LDA COMBIN23
7370      BEQ RIGHT1
7380      LDA HVAR0,Y
7390      CLC
7400      ADC DIFF23
7410 ;
7420 ;-IF COMBINED PLAYER IS AT
7421 ;SCRNRIGHT, CHECK FOR HORZ WRAP-
7430 ;
7440 RIGHT5 CMP SCRNRIGHT
7450      BEQ CHKWRAP
7460 ;
7470 ;-UNCOMBINED PLAYER AT SCRNRIGHT?
7480 ;
7490 RIGHT1 LDA HVAR0,Y
7500      CMP SCRNRIGHT
7510      BNE RIGHT2
7520 ;
7530 ;
7540 CHKWRAP LDA HWRAP
7550      BEQ JTR     ;NO WRAP = RETURN
7560      LDA SCRNLEFT
7570 ;
7580 ;-GENERAL MOVE-RIGHT ROUTINE-
7590 ;
7600 RIGHT2 CLC       ;INC CURRENT...
7610      ADC #1      ;POSITION AND PUT
7620      STA HVAR0,Y ;NEW POSITION IN
7630      STA HPOS0,Y ;POSITION REGS.
7640 ;
7650 ;
7660      INX
7670      CPX SPDVAR  ;FAST ENOUGH?
7680      BNE RIGHT4  ;NO,MOVE AGAIN.
7690      BEQ JTR     ;YES, RETURN.
```

A LETTER FROM THE PUBLISHER

It's no secret that the U.S. Atari market isn't as healthy as it could be. The 8-bit computer line has declined in popularity, while the ST, though it has gained a respectable following in Europe, has yet to find its niche in the states. For these reasons, most software companies won't develop products for the Atari systems.

This lack of software support has a subtle, but nonetheless powerful impact on magazines that rely on the Atari market for their well-being. The cold fact is that advertisers for the 8-bit products are nearly nonexistent, and there are precious few advertisers for ST products.

Since, for profitable publications, we depend to a great extent upon advertising, we are left with two choices if our publications are to continue: We can increase the price of our magazines, thus forcing readers to pick up the tab for the lack of advertising, or we can find a way to make the magazines less expensive to produce. We've opted for the latter.

There are, of course, many ways we can cut the magazines' publishing costs: We can reduce the page count. We can get rid of the color. We can pay contributors less. Unfortunately, none of these options, nor others, not mentioned here, makes much of a difference in the long run.

After much thought, we decided that although the Atari market is not capable of supporting two Atari-specific magazines from a single publisher, it *is* active enough to support one. So we're going to combine ANALOG Computing and ST-LOG into a single monthly publication.

Don't panic! When you think about it, the merging of the magazines will allow us to produce a much nicer publication. And since the single magazine will be larger than either of the individual ones, we won't have to cut much from our content. In fact, after doing some analysis, we've discovered that we will be able to offer the same columns, departments and types of features you've come to expect. Little will change, except that everything will come to you under a single cover.

The November issue will be the first combination magazine. Next month we'll give you more details on what the new publication will be like, as well as our plans for the future. (We plan some nice surprises, like a reduction in the cost of magazine disks.)

We believe that merging ANALOG Computing and ST-LOG is the best solution to a tough problem. It allows us to continue publication while giving you your full money's worth. It also gives Atari a chance to prove their claim that in the coming year they will emerge a strong presence in the U.S. When that time comes, we plan to reevaluate the situation and possibly separate the publications once again.

Recently, Atari supporters have had to stick together like never before. We've been there, providing support and information for nearly nine years. And we plan to be there for many more.

Here's to the future!

Lee H. Pappas

Lee H. Pappas
Publisher

B&C ComputerVisions

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

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

800/XL/XE SOFTWARE
ALL TITLES ON DISK

ENTERTAINMENT

12 ADAMS ADVENTURES	14.95
ALIANTS	26.95
ALT. REALITY CITY	26.95
ALT.REAL DUNGEON	26.95
ASSULT FORCE	19.95
AUTO DUEL	35.95
BEYOND CASTLE WOLF	14.95
BISMARK	26.95
BOP & WRESTLE	26.95
BORDINO:1812	22.50
BOULDERDASH CONSTR.SET	17.95
BRUCE LEE	17.95
CASTLE WOLFENSTEIN	14.95
CHAMP. LODE RUNNER	26.95
COSMIC TUNNELS	9.95
D-BUG	7.95
DALLAS QUEST	7.95
DELUSE INVADERS	7.95
FIGHT NIGHT	17.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS	22.50
GUNSLINGER	26.95
HARD HAT MAC	7.95
JAWBREAKER	9.95
KARATEKA	13.50
KNICKERBOCKERS	13.50
KORONIS RIFT	13.50
LAST V-8	8.95
L.A. SWAT/PANTHER	8.95
LEADERBOARD	13.50
LODE RUNNER	13.50
MICROLEAGUE BASEBALL	35.95
NAPOLEON AT WATERLOO	22.50
MONTEZUMA'S REVENGE	14.95
MOUSEQUEST	17.95
MOON SHUTTLE	7.95
NIBBLER	12.95
NINJA	8.95
OGRE	26.95
OIL'S WELL	9.95
O'RILEY'S MINE	9.95
PIRATES OF BARB. COAST	22.50
PITFALL/DEMON ATTACK	13.50
PREPPIE I & II	9.95
RESCUE ON FRACTALAS	13.50
ROME & THE BARBARIANS	17.95
SPITFIRE 40	31.50
STARFLEET I	44.95
STAR RAIDERS II	17.95
SPY VS. SPY III	17.95
STOCKMARKET	22.50
STRIP POKER	26.95
SUMMER GAMES	17.95
TAX DODGE	9.95
TEMPLE OF APSHAI	9.95
THE HULK	5.35
TOMAHAWK (64K)	26.95
TRAILBLAZER	26.95
ULTIMA II	35.95
ULTIMA III	35.95
ULTIMA IV	53.95
UNIVERSE	44.95
ZAXXON (400/800)	13.50

****

MICROPOSE SPECIALS!!

CONFLICT IN VIET NAM	10.95
CXRUSADE IN EUROPE	10.95
DECISION IN DESERT	10.95
F-15 STRIKE EAGLE	14.95
KENNEDY APPROACH	10.95
NATO COMMANDER	10.95
SILENT SERVICE	12.95
SOLO FLIGHT	12.95
TOP GUNNER	10.95

PROGRAMMING

DIAMOND (GEM O/S)	69.95
DOS 2.5	7.95
DOS XE	10.00
DISK I/O	26.95
KYAN PASCAL	62.95
LIGHTSPEED C	35.95
SPARTA DOS X	71.95

PRODUCTIVITY

ANIMATION STATION	79.95
ATARIWRITER	29.95
ATARIWRITER (CART ONLY)	19.95
ATARIWRITER+	44.95
ATARIWRITER 80 REQUIRES XEP80	44.95
ATARI BOOKKEEPER	24.95
ATARI MUSIC II	14.95
AWARDWARE (1050)	13.50
BANK STREET WRITER	14.95
BLAZING PADDLES	31.50
CELEBRITY COOKBOOK	26.95
COMPUTE YOUR ROOTS	35.95
DATAMANAGER	8.95
ELECTRONIC CHECKBOOK	8.95
FAMILY FINANCE	6.95
GUITAR WIZARD	26.95
HOME ACCOUNTANT	19.95
HOME FILING MANAGER	6.95
HOMEPAK	24.95
INVENTORY MASTER	80.95
LETTER WIZARD	29.95
MONEY MANAGER	8.95
MUSIC CONSTRUCTION SET	13.50
NEWSROOM (1050 - 64K)	44.95
NEWS STATION	26.95
NEWS STA. COMPANION	26.95
PAGE DESIGNER	26.95
PAINT	12.95
PRINT POWER (1050)	13.50
PRINTKIT (1050)	13.50
PRINTSHOP	34.95
P.S. COMPANION (64K)	24.95
P.S.GRAPHICS LIBRARY 1	17.95
P.S.GRAPHICS LIBRARY 2	17.95
P.S.GRAPHICS LIBRARY 3	17.95
PROOF READER	17.95
PUBLISHING PRO	35.95
RUBBER STAMP	26.95
SYNTREND	14.95
SUPER MAILER	35.95
THE LOTTO PROGRAM	17.95
TIMEWISE	6.95
TURBOWORD/80 COLUMN REQUIRES XEP80	44.95
VIDEO TITLESHOP (64K)	26.95
GRAPHICS COMPANION	17.95
VIRTUOSO	29.95
VISICALC	24.95

EDUCATION

ATARI LIGHT MODULE (REQ. STARTER SET)	9.95
BUZZWORD	35.95
GRANDMA'S HOUSE (-10)	9.95
HEY DIDDLE (AGE 3-10)	9.95
LINKWORD: SPANISH	22.50
LINKWORD: GERMAN	22.50
LINKWORD: FRENCH	22.50
LINKWORD: ITALIAN	22.50
MASTER TYPE	14.95
PRE-SAT WORD ATTACK	14.95
STATES AND CAPITALS	9.95
SPELLING BEE (AG 5-10)	14.95
TOUCH TYPING	9.95
QUIZ MASTER CONSTR.	8.95
QUIZ MSTR. USA CONSTR.	8.95

AMERICAN EDUCATION:

A+ BIOLOGY G10+	17.95
A+ GERMAN	17.95
A+ GRAMMER G4+	17.95
A+ READING COMP G1-8	17.95
A+ SCIENCE G3-4	17.95
A+ SCIENCE G5-6	17.95
A+ SCIENCE G7-8	17.95
A+ SPANISH	17.95
A+ U.S. GOV. G10+	17.95
A+ U.S. HIST. G5+	17.95
A+ VOCABULARY G4+	17.95
A+ WORLD GEOG. G8+	17.95
A+ WORLD HIST. G8+	17.95

(G = GRADE LEVEL)

ATARI:

CONCENTRATION	9.95
DIVISION DRILL	9.95
GRAPHING	9.95
INSTRUCTIONAL COMPUT	9.95
PLAYER MAKER	9.95
PREFIXES	9.95
SCREEN MAKER	9.95
SECRET FORMULA INTERM	9.95
SECRET FORMULA ADVANC	9.95
SPELL IN CONTEXT 8	9.95

CBS (AGE 3-6):

ASTROGROVER	8.95
BIG BIRD SPEC DELIVE	8.95
ERNIE'S MAGIC SHAPE	8.95

DESIGNWARE:

MATHMAZE (6-11)	26.95
MISSION ALGEBRA (13+)	13.50
SPELLICOPTER (6-11)	35.95

TINK TONK (AGE 4-6):

ABC'S	8.95
COUNT AND ADD	8.95
SMART THINKER	8.95
SPELLING	8.95
SUBTRACTION	8.95
THINKING SKILLS	8.95
ALL 6 TINK TONKS	39.95

UNICORN:

10 LITTLE ROBOTS (PRE-SCHOOL)	26.95
FUN BUNCH (6-ADULT)	26.95
RACECAR RITHMETIC (AGE 6+)	26.95

WEEKLY READER (PRE-SCHOOL):

STICKY BEAR SHAPES	26.95
STICKY BEAR NUMBERS	26.95
STICKY BEAR ABC'S	26.96
STICKY BEAR OPPOSITE	26.95
SB BASKET BOUNCE	26.95
STICKY BEAR BOP	26.95
RUN FOR IT	26.95
PIC BUILDER	26.95

Grade

800/XL/XE SOFTWARE
ALL TITLES ON CARTRIDGE

ENTERTAINMENT

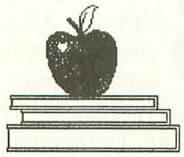
3D TIC-TAC-TOE	9.95
AIRBALL (XL/XE)	24.95
ALIEN AMBUSH	9.95
ACE OF ACES (XL/XE)	24.95
ARCHON	19.95
ASTEROIDS	15.95
ASTRO CHASE	14.95
ATARI TENNIS	9.95
ATLANTIS	14.95
BALL BLAZER	19.95
BARNYARD BLASTER	24.95*
BATTLEZONE	19.95
B.C. QUEST FOR TIRES	19.95
BLUE MAX	19.95
BOULDERS & BOMBS	14.95
CAVERNS OF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

PROGRAMMING

ACTION!	71.95
ACTION! TOOLKIT-DISK	26.95
BASIC XL	53.95
BASIC XL TOOLKIT-DISK	26.95
BASIC XE	71.95
LOGO	29.95
MAC/65	71.95
MAC/65 TOOLKIT-DISK	26.95
PILOT	19.95

PRODUCTIVITY

ATARIWRITER	19.95
FUN WITH ART	14.95
MICROFILER	22.50



EDUCATION

ATARILAB STARTER SET	29.95
MATH ENCOUNTERS	9.95

FISHER PRICE (PRE SCHOOL):

DANCE FANTASY	8.95
LINKING LOGIC	8.95
LOGIC LEVELS	8.95
MEMORY MANOR	8.95

SPINNAKER (AGE 3-10):

ALF IN COLOR CAVES	9.95
ALPHABET ZOO	9.95
DELTA DRAWING	9.95
FACEMAKER	9.95
KIDS ON KEYS	9.95
KINDERCOMP	9.95
STORY MACHINE (XL/XE)	9.95

(AGE 7 - ADULT):

ADV.CREATOR (400/800)	9.95
FRACTION FEVER	9.95

(* = REQ. LIGHT GUN)

****

SPECIAL PRICE
ATARI
XE GAME MACHING
NOW ONLY
$99.95
LIMITED TIME ONLY

LIGHTGUN 34.95

SUPER SPECIALS
RECONDITIONED ATARI MERCHANDISE 30 DAY WARRANTY

800 (48K) COMPUTER $79.95	SPACE AGE JOYSTICK $5.00.	3-1/2" DISKS AS LOW AS 75 CENTS 10 FOR $8.95 100 FOR $80 1000 FOR $750 DOUBLE SIDED WITH OLD DEMO SOFTWARE	1020 COLOR PRINTER/PLOTTER **$19.95** (new in box) 40 COLUMNS WIDE INC. PENS, PAPER, ETC.	ATARI BOOKKEEPER $14.95 - NO BOX	5-1/4" DISKETTES AS LOW AS 20 CENTS 10 FOR $4.00 100 FOR $29.95 1000 FOR $200 MOST ARE UNNOTCHED WITH OLD SOFTWARE
1030 MODEM WITH EXPRESS! $24.95	ATARI TRACKBALL $9.95			ATARI NUMERIC KEYPAD $7.95	

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

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

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

CIRCLE # 101 ON READER'S SERVICE CARD

BASIC EDITOR

by Clayton Walnum

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

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

Typing in the Editor

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

Disk version:

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

(2) Save the program to disk with the command *SAVE "D:EDITORL1.BAS"*.

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

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

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

(6) Load Listing 1 with the command *LOAD "EDITORL1.BAS"*.

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

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

Cassette version:

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

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

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

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

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

(6) Rewind the cassette.

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

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

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

Using the Editor

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

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

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

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

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

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

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

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

Leaving the Editor

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

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

Large Listings

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

The post-processor

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

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

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

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

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

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

Murphy's Law

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

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

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

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

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

Listing 1.
BASIC listing.

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

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

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

Listing 2.
BASIC listing.

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

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

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

THE END USER

Atari's PCES makes the *End User* so exciting.

by Arthur Leyenberger

Recently I was on a business trip that took me to Los Angeles. My business meetings were scheduled for the end of the week, so rather than return to the East Coast on Friday night via a "red eye" flight, I decided to stay the weekend and enjoy the Southern California experience. I spent two days with a childhood buddy.

Normally, LA is not really my kind of place, but I always try to make the best of my business travels; I try to do something a little different in whatever city I happen to be visiting. As it turned out, LA was a great place to do something "a little different."

You see, National Car Rental has a deal they call "California Classics" available only at LA International Airport and in Reno, Nevada. California Classics refers to vintage automobiles that can be rented just like ordinary cars for as long as you want. Instead of renting just another "jelly bean" (like a Taurus or one of its imitators), you can be seen tooling down Wilshire Boulevard in a '57 Chevy, a '62 Caddy convertible or perhaps a '52 Lincoln.

So I dropped off the company-paid-for "no-name" econobox, picked up a 1962 Thunderbird convertible and headed south on the 405 to see my buddy, Mike. If cars were measured in smiles-per-gallon, this T-Bird would be the EPA's number-one choice in America. You wouldn't believe the reactions I got—waves, smiles, thumbs-up—it made me feel good to be alive and well in sunny Southern California.

The weekend was a blast. Cruising had never been this good—top down, blue skies, oldies tunes on the radio—I hadn't felt so good in a long time. After I returned to New Jersey (I did take a "red eye" after all), I started thinking about that 27-year-old T-Bird. I also started thinking about the Atari 8-bit computers.

Driving the T-Bird for two days was a real experience. Aside from the "feel-good factor," the car was, in some ways, showing its age. Sure, it seemed mechanically safe (I should hope so), but it didn't have any of the modern features that many of us take for granted even in inexpensive cars. Fuel economy was in the single digits and no shoulder belts were available.

The Atari 800 sitting on the desk next to me is a lot like that '62 T-Bird. Like the car, the 800 is at least one, if not two generations old. By today's standards, the 8-bit 6502 microprocessor is slow and incapable of the latest "gee-whiz" graphics available on machines like the ST. Just as you would probably find it difficult to obtain parts for the T-Bird, few, if any new 8-bit Atari programs are available.

The analogy could be taken even further, but my point is that the 800 (as well as the XL, XE and XEGS models) fulfills a need just like that T-Bird. The T-Bird is fun, fun, fun and so are the 8-bit Atari machines. More important, 8-bit users know their computers are still functional and can perform the basics of computing and more. Excellent word processors, spreadsheets, telecommunications and graphics programs are still available for the Atari computer, making it useful for both serious and leisure computing activities.

Many 8-bit users have not traded up to an ST or other computer (perish the thought) for one simple reason: Their computers still satisfy their computing needs. A wealth of programming languages makes it an excellent machine for program development, and the graphics are still superior to other machines in its class. A lot can be done with an Atari 8-bit computer, and the hundreds of thousands of users prove it.

I enjoyed driving that '62 T-Bird convertible, just as I enjoy using my Atari 800. I was introduced to the world of microcomputers through the 800. The 8-bit Ataris may be showing their age compared to the latest in computing technology, but they still can compute. And that's what it's all about, isn't it?

Just Another Show

Well, the 40th Consumer Electronics Show is now history. Attendance at the latest Summer CES in Chicago marks my 14th semiannual trip into consumer electronics nirvana. Here, the near- and sometimes long-term future of car audio, home office and photographic products, audio and video hardware, entertainment products, car and home security and home automation is shown for all to see. The equivalent of 17 football fields' worth of exhibit space showcases almost one hundred categories of products.

Atari was at CES, but their emphasis, as before, was on games. That's not bad, just consistent with Atari's new focus. According to Atari, COMDEX (the COMputer Dealers EXposition) is the correct forum for their computer products, whereas CES is appropriate for their entertainment products. However, there was an ST attached to a MIDI program on one outside corner of their booth, as well as a couple of the Atari MS-DOS clone machines.

The big news at the Atari booth was the introduction of the Atari Portable Color Entertainment System (PCES). In Atari's words, it is "the world's first color portable handheld video game system." Actually, they don't have a name for it yet, so I'll refer to it as

the PCES for now.

What is the PCES? Hype aside, it's a hand-held game machine. Using a 3.5-inch built-in color LCD screen, the PCES can display graphics with up to 16 simultaneous colors from a palette of 4,096 colors. Resolution is 160 by 102 pixels, not very good by most standards but adequate when viewed on the small LCD display. Also, it uses a 16-MHz processor, which is faster than other video-game machines, like Nintendo and Sega.

The PCES is a completely self-contained unit. Slightly larger than a videocassette and weighing about a pound, it can be used individually or linked with up to eight other units for multi-player games. The system has 64K RAM and runs on six "AA" batteries. It can also be powered by an AC adapter or used with a cigarette-lighter adapter.

Main controls of the PCES consist of an eight-way "joypad," four fire buttons, two option buttons, a pause and an on/off switch. Other features include a headphone jack for quiet operation, the ability to rotate the screen image 180 degrees so that both right- and left-handed players can play (a logical option given the unit's butterfly shape), four-channel sound and volume and screen-contrast controls.

Games will be available on credit card-sized ROMS that slip into the unit. These game cards, which will sell for "under $35," typically contain 256K bytes of program and data, but are capable of holding as much as two megabytes of information. Epyx's *California Games* (an action game familiar to many Atari 8-bit and ST owners) will be bundled with the unit. Five other games—*Blue Lightning* (a first-person jet fighter game), *Time Quests and Treasure Chests* (an adventure/strategy game), *Gates of Zendocon* (an arcadelike action game), *Impossible Mission* (an action/adventure game) and *Monster Demolition* (an action game)—will also be available at the time of the PCES release.

Atari is working closely with Epyx to develop more titles for the PCES. They also hope to interest third-party developers in the system so that the potential PCES user has dozens of games to choose from. The retail price of the Atari Portable Color Entertainment System is $150. It is scheduled to be available this fall, in time for the Christmas season.

The Inside Story

I wouldn't be able to live with myself unless I told you the inside story on Atari's PCES. After all, that's what makes *End User* so exciting, right? Anyway, here's the scoop: The PCES was developed by Epyx in-house and the rights to it were sold to Atari literally moments before CES began.

Yup, Atari does not acknowledge publicly that Epyx designed and developed the PCES. You see, I and other ANALOG editors got a glimpse of the PCES prototype at the last Consumer Electronics Show, held in January in Las Vegas. We had to sign a nondisclosure agreement with Epyx before we were allowed to see the machine. In fact, ANALOG's sister publication, *VideoGames and Computer Entertainment*, was going to run an exclusive cover story on the PCES in a summer issue.

Epyx told us they were delaying the introduction of their portable game machine, originally scheduled for the summer CES. When we arrived at Chicago's McCormick Place for the first day of CES, we were surprised to see Atari demonstrating the unit. Sources told us that the final contract between Epyx and Atari had not been signed until just hours before the show started. Apparently, negotiations had been going on for some time.

It was obvious from the poor quality of the section of the Atari booth used to present the PCES (a couple of black-and-white posters) that the arrangements were done at the last minute. In addition, Atari placed no pre-announcements of the product in any of the daily trade magazines. Compared to Nintendo, which was also presenting a hand-held video game called the Game Boy, Atari's effort seemed lackluster.

Epyx wasn't discussing the Atari hand-held game deal at all. Rumors suggested that Epyx ran out of cash during the development of the product and was seeking someone or some company to bail them out. Interestingly, one Atari spokesperson told me that Atari now owns 40 percent of Epyx. This, however, was vehemently denied by the Epyx PR person.

The Competition

Nintendo is clearly the biggest name in video games right now. Their booth, some 50,000 square feet, hosted dozens of Nintendo licensees. Also shown at "Nintendo Village" was their new portable game machine, the Game Boy.

The Game Boy is a $90 hand-held unit that weighs about ten ounces and operates on four "AA" batteries. Unlike the Atari portable game machine, the Game Boy uses a monochrome non-backlit LCD screen. The unit features stereo sound and has a headphone jack. In addition, two Game Boys can be joined together via a cable for two-person games, such as baseball and tennis.

Nintendo has the advantage when it comes to game titles. Packed with the Game Boy will be *Tetris*, and other popular titles, such as *Super Mario Brothers*, will be available immediately. Also, the game cards for the Nintendo unit will be priced at "under $20."

It is only natural to compare the Atari PCES with Nintendo's Game Boy. The Atari game is easier to see because of its color and backlighting. In addition, the Atari screen is larger than that of the Game Boy. However, Nintendo has a larger presence in the game market and will, no doubt, launch a major advertising and promotion campaign. Given Atari's track record when it comes to advertising, who knows what they will do to promote the PCES.

Ultimately, it will be interesting to see how each of these portable games fares in the marketplace; a repeat of the early Atari 800 days could happen. You'll recall that in the early 1980s, when the Atari 800 was originally competing with the Commodore 64, the superior sound and graphics of the 800 never overcame the superior marketing and pricing of the Commodore 64. Nintendo is now as powerful (or more so) than Commodore was in its heyday.

The Rest of Atari

In addition to the Atari portable game and a couple of computers here and there, the focus of Atari's booth was games for the 2600, 7800 and XE game systems. Twenty new titles, which will be available by the end of the year were announced for these three systems. Atari has also lowered the prices of two of their game machines to $50 for the 2600 and $70 for the 7800. In addition, Atari has made available light guns for the 2600 and 7800.

The wackiest part of Atari's booth was their display of a full line of calculators. To me, selling calculators is just another sign that Atari, as a corporation, lacks focus. An old adage suggests that if you try to do too many things, you will not do any of them well. Perhaps Atari should listen to this advice and concentrate their limited resources on just a couple of product lines—such as computers and games.

As I've said many times before, the Consumer Electronics Show is always interesting. Of all the new technology that is displayed, there is usually some neat stuff that really appeals to me. And though Atari may no longer be the leader in video games or low-cost/high-power computing, they never fail to surprise me.

***Arthur Leyenberger** is a freelance writer who lives in beautiful New Jersey. He can be reached on CompuServe at 71266,46, or on DELPHI as ARTL.*

(continued from page 8)

```
DD 5190 DATA 19,LOAD Program Too Long
YZ 5200 DATA 20,BAD Device #; #'s 1-7 onl
   y
GN 5210 DATA 21,LOAD File Error;NOT SAVE
   format
EH 5220 DATA 128,BREAK Key Abort during I
   /O
BF 5230 DATA 129,IOCB Error;file already
   OPEN
QL 5240 DATA 130,NONEXISTANT Device;?FILE
   NAME
DG 5250 DATA 131,IOCB Write Only;cant REA
   D it
AA 5260 DATA 132,INVALID Hndlr Cmd;?XIO o
   r IOCB
OW 5270 DATA 133,DEVICE or File not OPEN
YW 5280 DATA 134,BAD IOCB #; 1-7 only in
   BASIC
KB 5290 DATA 135,IOCB Read Only ERR;cant
   write
MA 5300 DATA 136,END OF FILE
UZ 5310 DATA 137,TRUNCATED Record;?INPUT
   line
XR 5320 DATA 138,DEVICE Timeout;?UNIT or
   ?DEV #
MF 5330 DATA 139,DEVICE NAK;?I/O cmd or ?
   cables
DF 5340 DATA 140,LOST data on serial I/O
   bus
LA 5350 DATA 141,CURSOR Out of Range;?GR
   Mode
UD 5360 DATA 142,SERIAL Data Overrun;?too
    fast
UX 5370 DATA 143,SERIAL Bus Data Frame CK
   SM ERR
AH 5380 DATA 144,DEVICE Done;valid cmd ?r
   esponse
VL 5390 DATA 145,INVALID GR Mode command
PP 5400 DATA 146,FUNCTION Not Implemented
BP 5410 DATA 147,NOT enuf RAM for GR Mode
DD 5420 DATA 148,UNRECOGNIZED disk format
    Sparta
RR 5430 DATA 150,DIRECTORY not found-Spar
   ta
KL 5440 DATA 151,FILE Exists-Sparta
GD 5450 DATA 152,NOT binary file-Sparta
RK 5460 DATA 154,LOADER-SYMBOL not define
   d-Sparta
PY 5470 DATA 158,OUT of memory Sparta
GH 5480 DATA 160,DRIVE # ERR;?not in syst
   em
PF 5490 DATA 161,TOO many OPEN FILES;?buf
   fers
VN 5500 DATA 162,DISK Full;XIO#254=format
    new
PW 5510 DATA 163,FATAL System I/O ERR;?ba
   d DOS
HI 5520 DATA 164,FILE # Mismatch;?POINT o
   r file
YU 5530 DATA 165,INVALID FILENAME; ?len ?
   chars
CY 5540 DATA 166,POINT Data Length ERR
KZ 5550 DATA 167,FILE LOCKED;XIO#36=unloc
   k
KU 5560 DATA 168,INVALID Device Command
YN 5570 DATA 169,DISK DIR FULL;XIO#254=fo
   rmat
EG 5580 DATA 170,FILE Not Found;?FILENAME
DI 5590 DATA 171,POINT Invalid; ?FILE upd
   ate
PS 5600 DATA 172,ILLEGAL Append to DOS I
   file
TU 5610 DATA 173,BAD sectors during FORMA
   T
JR 5620 DATA 255,ERROR Manual by Mat*Rat-
   (c) ANALOG
YZ 5630 DATA 0,END OF ERROR MANUAL
EH 9000 DATA -1,End of ERROR Manual data
```

PROGRAM LISTINGS

VideoGames & Computer Entertainment™

12 ALL COLOR ISSUES ONLY $19.95

Save over $15 off the cover price!

- GAME REVIEWS
- ARCADE ACTION
- STRATEGY GUIDES
- TECHNICAL REPORTS
- COMPUTER SOFTWARE

VideoGames & Computer Entertainment™
P.O. BOX 16927, N. HOLLYWOOD, CA 91615

☐ **Yes!** Sign me up for 12 issues for only $19.95—I'll save over $15!

☐ Payment Enclosed — Charge My ☐ VISA ☐ MC

#_____ EXP_____

SIGNATURE_____

NAME_____

ADDRESS_____

CITY_____ STATE_____ ZIP_____

Money back on unused subscriptions if not satisfied!
Foreign—add $10 for postage.

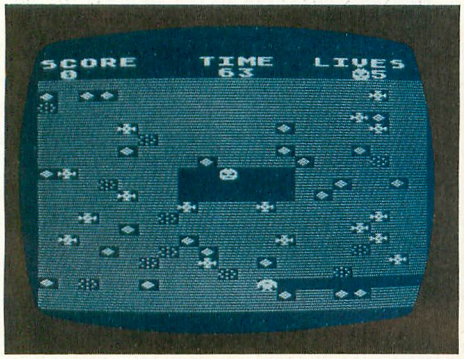
Your first issue will arrive in 6 to 8 weeks.
WATCH FOR IT!!

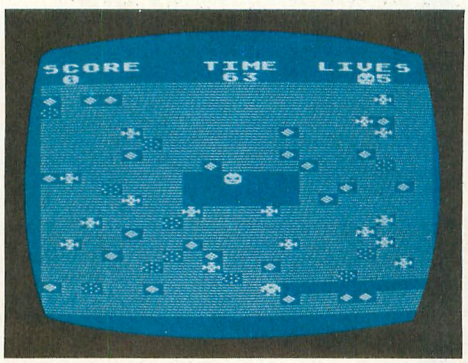
Offer expires November 26, 1989.

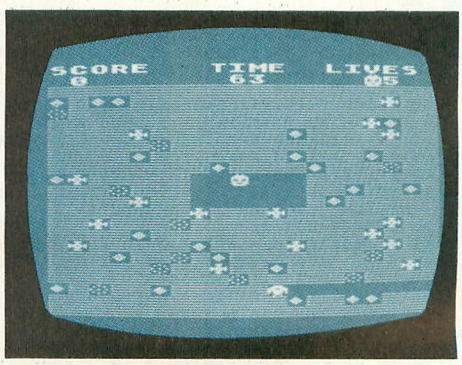
CJWYA

Illustration JENNY ADAMS

If you get them, you will receive a random bonus from 1,000 to 10,000 points.







by Frank Martone

CRUNCHER

Welcome to the amazing world of Tx Cruncher. What is Tx? It's a little demon that just loves to feed on electric energy. Tx lives on an electric grid, happily spending his time munching on a variety of power cells. The different power cells are called Popo, Kentu and Circa.

Popo are diamond shaped and worth 100, points, Kentu are cross shaped and worth 250 points and Circa are circular and worth 500 points. Each progressive screen will have more and more power cells. If you get really lucky, you may be able to find a magic star or two, which is worth a whopping 5,000 points.

Tx does not live alone on the electric grid, however. He has some lovely neighbors, the Hulk Robots, and they just hate it when someone starts eating their power cells. In the beginning there will be only one robot after Tx. But as your score progresses and the damage to their power cells increases, more Hulk Robots will be sent after Tx. A maximum of three will attack him at once. These robots also have the ability to lay out traps on the grid. If Tx snags one of these traps, or if he bumps into a Hulk Robot, you can bet he'll lose one of his five lives.

Tx does have an amazing power, though: He can send himself to Hyperspace one time on each screen. If you press the joystick button, Tx will disappear and reappear in a different spot—hopefully, a safer one, although I have seen him land on a trap sometimes.

You have to stay alive for only 75 seconds on each screen; you will then automatically advance to the next screen. After every three screens you will enter a bonus screen, where you must grab five magic stars while avoiding the traps. If you get them, you will receive a random bonus of from 1,000 to 10,000 points. If you hit a trap, you lose the bonus round.

Go get 'em, Tx!

Frank Martone's interest in computers started in junior high school, where he learned BASIC programming. He got his first computer, an Atari 400, when he was 12. Within two years his first game was published. He is now 21.

LISTING 1: BASIC

```
PD 0 REM TX  C R U N C H E R
DM 1 REM COPYRIGHT (C) 1989 ANALOG COMPUT
   ING
RC 2 REM BY FRANK MARTONE 1/23/89
NI 3 REM
YM 4 GRAPHICS 17:POSITION 0,10:? #6;"A.N.
   A.L.O.G PRESENTS       ":POKE 708,15:SN
   =0:GOSUB 1000:GOSUB 3000
QX 5 GOSUB 400:GRAPHICS 1+16:SETCOLOR 0,8
   ,10:SETCOLOR 2,RND(0)*14,9:TUM=5:SC=0:
   BR$=30:DIFF=400:S1=19:S2=20:CTT=0
JW 6 POKE 756,CH/256:BI=5:S3=1:S4=23:S5=0
   :S6=3:EG=17:GOB=13
BI 7 SCR=PEEK(88)+256*PEEK(89)
ZT 8 FOR G=2 TO 22:POSITION 0,G:? #6;"[redacted]
   [redacted]":SOUND 0,G,10,G:SOUN
   D 0,0,0,0:NEXT G:HYP=1
YH 11 FOR B=1 TO BR$:K1=INT(RND(0)*19):K2
   =INT(RND(0)*20)+3:POSITION K1,K2:? #6;
   "[redacted]":SOUND 0,30,10,10
MT 12 FOR HJ=0 TO 1:NEXT HJ:SOUND 0,0,0,0
   :NEXT B
JU 13 FOR T=1 TO EG:E1=INT(RND(0)*19):E2=
   INT(RND(0)*20)+3:POSITION E1,E2:? #6;"
   ";":SOUND 0,200,10,10
TS 14 FOR HJ=1 TO 3:NEXT HJ:SOUND 0,0,0,0
   :NEXT T
OF 15 FOR C=1 TO GOB:POSITION RND(0)*18,R
   ND(0)*18+3:? #6;"j":SOUND 0,C,6,10:SOU
   ND 0,0,0,0:NEXT C
UI 16 X=9:Y=10:V=10:TIME=0
WN 17 COLOR 0:PLOT 7,10:DRAWTO 12,10:PLOT
    7,11:DRAWTO 12,11:PLOT 7,12:DRAWTO 12
   ,12
KH 18 POSITION 0,0:? #6;"score   time  li
   ves":POSITION X,Y:? #6;" "
NS 19 FOR G=1 TO 80:SOUND 0,G*3,0,10:NEXT
    G:SOUND 0,0,0,0:POSITION 16,1:? #6;"+
   ":TIME=76
ZV 20 SCR=PEEK(88)+256*PEEK(89):POSITION
   9,1:? #6;INT(TIME);" ";:GOSUB 4100
DY 21 ST=STICK(0):TR=STRIG(0):TP=SCR+X+20
   *Y:POKE 709,14:POKE 77,0
LA 22 IF TR=0 AND HYP=1 THEN GOSUB 2051
TI 31 POSITION 1,1:? #6;SC:POSITION 17,1:
   ? #6;TUM:SETCOLOR 3,PEEK(53770),14
YL 32 LOCATE X,Y,BC
LY 33 IF BC=35 THEN FOR G=1 TO 4:FOR R=-1
   5 TO 15 STEP 3:SOUND 0,ABS(R)+35,10,10
   :NEXT R:NEXT G:SC=SC+5000
LO 35 SOUND 1,0,0,0:TIME=TIME-0.5:V=V+1:I
   F TIME<1 THEN GOTO 7000
FL 47 IF BC=247 THEN FOR W=15 TO 0 STEP -
   2:SOUND 0,10,6,W:NEXT W:SC=SC+100
VL 48 IF BC=59 THEN FOR W=15 TO 0 STEP -2
   :SOUND 0,W*3,10,W:NEXT W:SC=SC+250
VO 49 IF BC=106 THEN FOR W=70 TO 11 STEP
   -5:SOUND 0,W*4,10,8:NEXT W:SC=SC+500
KQ 50 POKE SCR+X+20*Y,V
YS 51 POKE 709,0:IF X>18 THEN POKE SCR+X+
   20*Y,0:X=X-1
QK 52 IF X<1 THEN POKE SCR+X+20*Y,0:X=X+1
UD 53 IF Y<3 THEN POKE SCR+X+20*Y,0:Y=Y+1
IO 54 IF Y>22 THEN POKE SCR+X+20*Y,0:Y=Y-
   1
NS 55 IF RND(0)*6<2 THEN GOSUB 500
IO 57 IF RND(0)*5<2 AND SC>20000 THEN GOS
   UB 550
PY 58 IF RND(0)*4<2 AND SC>50000 THEN GOS
   UB 580
QK 60 SOUND 0,0,0,0:IF BC=45 THEN GOTO 20
   00
JD 61 IF X=S1 AND Y=S2 THEN GOSUB 300
LL 62 IF X=S3 AND Y=S4 THEN GOSUB 300
NT 63 IF X=S5 AND Y=S6 THEN GOSUB 300
UJ 64 IF SC>100000 THEN POKE 710,PEEK(537
   70)
ZL 65 IF SC>30000 AND PEEK(53770)=44 THEN
    GOSUB 100
HO 80 IF V>10 THEN V=9
IR 81 POKE 709,0:POKE 709,14:IF RND(0)*DI
   FF<13 THEN POSITION RND(0)*19,RND(0)*1
   8+3:? #6;"-":SOUND 0,255,10,5
SC 82 GOTO 20
NL 99 REM --MAGIC STAR APPEARS!--
AQ 100 POSITION RND(0)*19,RND(0)*18+4:? #
   6;"#"
ZO 101 FOR R=2 TO 20:SOUND 0,R*4,10,10:SO
   UND 1,R*8,10,10:POKE 712,R*5:SOUND 2,R
   ,10,10:POKE 712,R:NEXT R
WT 102 POKE 712,0:FOR T=0 TO 3:SOUND T,0,
   0,0:NEXT T:RETURN
YI 300 POSITION X,Y:? #6;"+":FOR G=14 TO
   1 STEP -1:FOR W=1 TO 25:NEXT W:SETCOLO
   R 0,12,G:SOUND 0,G,10,G
CD 301 SOUND 1,G+2,10,G:NEXT G:SETCOLOR 0
   ,12,14:SOUND 0,0,0,0:SOUND 1,0,0,0
NN 302 FOR EX=4 TO 6:POKE SCR+X+20*Y,EX:F
   OR W=1 TO 30:NEXT W:SOUND 0,10,0,10:NE
   XT EX:POKE SCR+X+20*Y,0:X=9:Y=10
QO 303 FOR S=0 TO 2:SOUND S,0,0,0:NEXT S
XM 304 TUM=TUM-1
RI 305 IF TUM=0 THEN GOTO 5000
OY 306 FOR L=15 TO 1 STEP -0.3:SOUND 1,L,
   6,L:SOUND 2,L,6,L:NEXT L
NP 310 FOR R=0 TO 2:SOUND R,0,0,0:NEXT R
AW 311 POSITION S1,S2:? #6;" ":POSITION S
   3,S4:? #6;" ":POSITION S5,S6:? #6;" "
OW 312 S1=3:S2=19:S3=18:S4=19:S5=3:S6=3:S
   ETCOLOR 0,8,10
ZD 320 RETURN
DK 400 GRAPHICS 18:POKE 711,88:S1=19:S2=1
   8:S3=2:S4=18:S5=0:S6=5
JN 410 POSITION 1,5:? #6;"now entering sc
   reen":SN=SN+1
FO 411 POSITION 9,8:? #6;"0";SN
QU 412 FOR G=1 TO 20
MX 420 FOR D=15 TO 0 STEP -2:SOUND 0,14,1
   0,D:POKE 708,D:NEXT D:NEXT G:RETURN
DV 499 REM ROBOT INTELLIGENCE ROUTINE
FQ 500 SNK=SCR+S1+20*S2:SV=58
VI 510 POKE SNK,0:SOUND 0,20,6,10
ZL 521 IF Y<S2 AND X=S1 THEN S2=S2-1:GOTO
    529
XU 522 IF Y>S2 AND X=S1 THEN S2=S2+1:GOTO
    529
TT 525 IF X<S1 THEN S1=S1-1:SV=58
RH 526 IF X>S1 THEN S1=S1+1
LL 529 POKE SCR+S1+20*S2,SV
ZH 530 RETURN
RH 550 SNK1=SCR+S3+20*S4:SV=58
GJ 555 POKE SNK1,0:SOUND 0,10,6,10
YF 557 IF X<S3 THEN S3=S3-1:SV=58
VT 558 IF X>S3 THEN S3=S3+1
XG 559 IF Y<S4 AND X=S3 THEN S4=S4-1
UN 560 IF Y>S4 AND X=S3 THEN S4=S4+1
NP 561 POKE SCR+S3+20*S4,SV
ZP 570 RETURN
UP 580 SNK2=SCR+S5+20*S6:SV=58
EX 585 POKE SNK2,0:SOUND 0,3,6,5
CL 587 IF X<S5 THEN S5=S5-1:SV=58
ZZ 588 IF X>S5 THEN S5=S5+1
EK 589 IF Y<S6 AND X=S5 THEN S6=S6-1
BR 590 IF Y>S6 AND X=S5 THEN S6=S6+1
QP 591 POKE SCR+S5+20*S6,SV
AI 595 RETURN
CX 1000 REM --CHARACTER SET--
EH 1010 CH=(PEEK(106)-8)*256
BS 1015 FOR I=0 TO 512:POKE CH+I,PEEK(573
   44+I):NEXT I
HM 1020 RESTORE 1100
SX 1030 READ A:IF A<0 THEN RETURN
TY 1040 FOR J=0 TO 7:READ B:POKE CH+A*8+J
   ,B:NEXT J
OQ 1050 GOTO 1030
VE 1099 REM --REDEFINED CHARACTERS--
JJ 1100 DATA 10,20,62,127,93,73,127,99,62
VI 1110 DATA 11,20,62,127,93,73,127,119,6
   2
PH 1115 DATA 12,20,62,127,127,127,127,127
   ,62
SC 1120 DATA 4,0,0,4,0,16,8,64,0
KM 1125 DATA 5,1,66,0,34,8,64,20,128
JU 1130 DATA 6,0,0,4,80,8,20,0,0
AB 1140 DATA 2,255,0,255,0,255,0,255,0
LV 1143 DATA 58,60,126,231,255,189,189,24
   ,36
GO 1146 DATA 27,56,16,186,238,186,16,56,0
UK 1147 DATA 55,0,16,40,84,40,16,0,0
RT 1149 DATA 63,7,5,7,56,40,184,128,128
TT 1150 DATA 42,16,68,0,146,0,68,16,0
DM 1170 DATA 3,146,84,16,238,16,84,146,0
XK 1175 DATA 26,0,0,0,0,0,0,1,3
LJ 1177 DATA 28,0,0,0,0,0,0,128,192
DQ 1178 DATA 29,7,7,7,15,15,15,15,7
KF 1179 DATA 30,255,156,8,8,207,8,156,255
TA 1180 DATA 31,224,224,112,112,112,112,2
   40,224
KU 1181 DATA 32,7,3,7,24,48,24,4,0
MF 1182 DATA 61,189,195,255,126,60,24,24,
   0
YU 1183 DATA 62,224,192,224,24,12,24,32,0
HD 1184 DATA 49,0,0,0,255,0,255,0,0
NQ 1190 DATA 8,170,77,170,77,170,77,170,7
   7
KW 1191 DATA 13,66,129,36,0,0,36,129,66
AB 1192 DATA 7,6,13,14,8,68,66,34,28
RL 1193 DATA 1,0,0,0,0,0,0,0,0
GJ 1194 DATA 1,44,56,66,3,2,4,0,9
DC 1195 DATA 1,14,16,66,4,2,4,4,5
WU 1196 DATA 1,44,55,68,0,0,0,1,6
LK 1197 DATA 1,14,15,18,1,0,1,1,1
UG 1198 DATA 1,57,85,58,7,0,3,8,9
RO 1199 DATA 1,17,15,51,1,0,1,4,4
WX 1200 DATA 1,19,19,99,9,9,9,2,4
BN 1201 DATA 1,14,16,66,4,2,4,4,5
QI 1202 DATA 1,0,0,0,0,0,0,0,0
XJ 1203 DATA 1,19,19,99,9,9,9,2,4
SV 1205 DATA 1,57,85,58,7,0,3,8,9
VZ 1207 DATA 1,44,55,68,0,0,0,1,6
QD 1210 DATA 1,0,0,0,0,0,0,0,0
FB 1211 DATA 1,44,56,66,3,2,4,0,9
FF 1212 DATA 1,44,56,66,3,2,4,0,9
QP 1213 DATA 1,0,0,0,0,0,0,0,0
FN 1214 DATA 1,44,56,66,3,2,4,0,9
RB 1216 DATA 1,0,0,0,0,0,0,0,0
RN 1219 DATA 1,0,0,0,0,0,0,0,0
XD 1220 DATA 1,19,19,99,9,9,9,2,4
XH 1221 DATA 1,19,19,99,9,9,9,2,4
PT 1222 DATA 1,17,15,51,1,0,1,4,4
ST 1223 DATA 1,57,85,58,7,0,3,8,9
XT 1224 DATA 1,19,19,99,9,9,9,2,4
XX 1225 DATA 1,19,19,99,9,9,9,2,4
RE 1226 DATA 1,0,0,0,0,0,0,0,0
QJ 1230 DATA 1,0,0,0,0,0,0,0,0
FH 1231 DATA 1,44,56,66,3,2,4,0,9
QR 1232 DATA 1,0,0,0,0,0,0,0,0
XS 1233 DATA 1,19,19,99,9,9,9,2,4
XW 1234 DATA 1,19,19,99,9,9,9,2,4
QI 1235 DATA 1,17,15,51,1,0,1,4,4
TI 1236 DATA 1,57,85,58,7,0,3,8,9
YI 1237 DATA 1,19,19,99,9,9,9,2,4
YM 1238 DATA 1,19,19,99,9,9,9,2,4
GN 1239 DATA 1,44,56,66,3,2,4,0,9
QU 1242 DATA 1,0,0,0,0,0,0,0,0
EX 1250 DATA -1
OA 2000 FOR R=1 TO 4:POSITION X,Y:? #6;"-
   ":SOUND 0,0,0,0:FOR D=1 TO 4:NEXT D:PO
   SITION X,Y:? #6;" ":SOUND 0,200,10,10
IP 2001 NEXT R
IB 2010 GOSUB 300:GOTO 20
UX 2050 REM HYPERSPACE
FI 2051 POSITION X,Y:? #6;" ":FOR EX=4 TO
    6:POKE SCR+X+20*Y,EX:FOR W=1 TO 5:NEX
   T W:NEXT EX
NE 2052 POSITION X,Y:? #6;" "
GG 2054 FOR R=15 TO 0 STEP -1:SOUND 0,R,1
   0,R:SOUND 1,R*2,10,R:SOUND 1,R*63,10,R
   :POKE 712,R+79:NEXT R:POKE 712,0
NU 2056 POSITION X,Y:? #6;" "
IZ 2057 X=INT(RND(0)*14)+4:Y=INT(RND(0)*1
   7)+5
OC 2058 POSITION X,Y:? #6;" "
KF 2059 FOR EX=4 TO 6:POKE SCR+X+20*Y,EX:
   FOR W=1 TO 5:NEXT W:NEXT EX
MZ 2060 POSITION X,Y:? #6;" "
AI 2061 HYP=0
PZ 2070 GOTO 20
VP 2999 REM --TITLE SCREEN--
XE 3000 GRAPHICS 17:POKE 756,CH/256
FI 3003 POKE 708,122:POKE 711,15:POKE 709
   ,77
YX 3005 X=10:Y=20
ZC 3006 SOUND 0,0,0,0:POSITION X,Y:? #6;"
   +":FOR D=1 TO 15:NEXT D:POSITION X,Y:?
    #6;" ":SOUND 0,20,10,10
EH 3007 Y=Y-1:IF Y=7 THEN FOR D=15 TO 0 S
   TEP -0.3:SOUND 0,D*33,8,D:POKE 712,D+4
   5:NEXT D:GOTO 3026
RY 3008 GOTO 3006
LS 3026 POKE 712,0:POKE 708,14:POKE 710,8
   9
HU 3027 POSITION 2,9:? #6;"   TX CRUNCHER
    "
VD 3028 POSITION 2,11:? #6;"     [redacted] * [redacted]
    "
MD 3029 POSITION 0,13:? #6;"[redacted]
   [redacted]"
JE 3040 POSITION 1,16:? #6;" by frank mar
   tone":POSITION 0,22:? #6;"QQQQQQQQQQQQ
   QQQQQQQQ"
CS 3041 POSITION 0,3:? #6;"QQQQQQQQQQQQQQ
   QQQQQQ"
DB 3047 FOR D=1 TO 1000:NEXT D:GOTO 4000
GV 4000 GRAPHICS 1+16:POKE 756,CH/256:POK
   E 708,14:POKE 710,55:P=5
JS 4001 POSITION 4,2:? #6;" point scale"
CL 4002 POSITION 0,3:? #6;"QQQQQQQQQQQQQQ
   QQQQQQ"
JB 4003 POSITION 0,22:? #6;"QQQQQQQQQQQQQ
   QQQQQQQ"
OC 4004 POSITION 1,P:? #6;"POPO         W
    100":GOSUB 4900
MB 4005 POSITION 1,P:? #6;"KENTU        [redacted]
    250":GOSUB 4900
JR 4006 POSITION 1,P:? #6;"CIRCA        [redacted]
    500":GOSUB 4900
HK 4007 POSITION 1,P:? #6;"MAGIC STAR  #
   5000":GOSUB 4900
SO 4010 POSITION 5,18:? #6;"PRESS START":
   FOR D=1 TO 50:NEXT D
KG 4011 POSITION 5,18:? #6;"press start":
   FOR D=1 TO 50:NEXT D
FI 4015 IF PEEK(53279)=6 OR STRIG(0)=0 TH
   EN GOTO 4030
OR 4020 GOTO 4010
TQ 4030 FOR I=0 TO 19:COLOR 0:PLOT I,0:DR
   AWTO I,22:NEXT I:GOTO 5
RA 4050 GOTO 4050
ZJ 4099 REM --JOYSTICK MOVEMENT ROUTINE--
CA 4100 IF ST=14 THEN UF=1:DF=0:RF=0:LF=0
FB 4111 IF ST=13 THEN DF=1:UF=0:RF=0:LF=0
ZT 4112 IF ST=7 THEN RF=1:DF=0:LF=0:UF=0
DP 4113 IF ST=11 THEN LF=1:RF=0:DF=0:UF=0
MX 4114 IF UF=1 THEN POKE TP,0:Y=Y-1
ES 4115 IF DF=1 THEN POKE TP,0:Y=Y+1
HK 4116 IF RF=1 THEN POKE TP,0:X=X+1
HY 4117 IF LF=1 THEN POKE TP,0:X=X-1
AT 4150 RETURN
PC 4900 FOR D=15 TO 0 STEP -1:SOUND 0,50,
   10,D:NEXT D:P=P+3:RETURN
DW 4999 REM --GAME OVER--
OR 5000 POSITION 17,1:? #6;"0":FOR E=14 T
   O 0 STEP -1:SETCOLOR 2,4,E:FOR R=1 TO
   10:NEXT R:SOUND 0,E,0,10:NEXT E:SN=0
```

(continued on page 67)

M/L EDITOR

For use in machine-language entry.

by Clayton Walnum

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LISTING 1: BASIC LISTING

```
AZ 10 DIM BF(16),N$(4),A$(1),B$(1),F$(15)
   ,F1$(15)
LF 11 DIM MOD$(4)
BN 20 LINE=1000:RETRN=155:BACK5P=126:CHK5
   UM=0:EDIT=0
GO 30 GO5UB 450:PO5ITION 10,6:? "Start or
   Continue? ";:GO5UB 500:? CHR$(A)
ZG 40 PO5ITION 10,8:? "FILENAME";:INPUT F
   $:POKE 752,1:? " "
FE 50 IF LEN(F$)<3 THEN PO5ITION 20,10:?
   "  ":GOTO 40
NF 60 IF F$(1,2)<>"D:" THEN F1$="D:":F1$(
   3)=F$:GOTO 80
KL 70 F1$=F$
TN 80 IF CHR$(A)="5" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$:TRAP 110
HQ 100 FOR X=1 TO 16:GET #2,A:NEXT X:LINE
   =LINE+10:GOTO 100
WM 110 CL0SE #2:OPEN #2,9,0,F1$:GOTO 170
VT 120 TRAP 160:OPEN #2,4,0,F1$:GO5UB 440
   :PO5ITION 10,10:? "FILE ALREADY EXISTS
   !!":POKE 752,0
ZU 130 PO5ITION 10,12:? "ERA5E IT? ";:GO5
   UB 500:POKE 752,1:? CHR$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
   CL0SE #2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
   HEN 130
BH 160 CL0SE #2:OPEN #2,8,0,F1$
IE 170 GO5UB 450:PO5ITION 10,1:? "NOW ON
   LINE: ";LINE:CHK5UM=0
GH 180 L1=3:FOR X=1 TO 16:PO5ITION 13*(X<
   10)+12*(X>9),X+2:POKE 752,0:? "BYTE #"
   ;X;": ";:GO5UB 310
KH 190 IF EDIT AND L=0 THEN BYTE=BF(X):GO
   TO 210
FY 200 BYTE=VAL(N$)
OZ 201 MOD$=N$
BU 210 PO5ITION 22,X+2:? BYTE;"  "
YZ 220 BF(X)=BYTE:CHK5UM=CHK5UM+BYTE*X:IF
   CHK5UM>9999 THEN CHK5UM=CHK5UM-10000
MS 230 NEXT X:CHK5UM=CHK5UM+LINE:IF CHK5U
   M>9999 THEN CHK5UM=CHK5UM-10000
IG 240 PO5ITION 12,X+2:POKE 752,0:? "CHEC
   K5UM: ";:L1=4:GO5UB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
SY 270 PO5ITION 22,X+2:? C;"   "
IL 280 IF C=CHK5UM THEN 300
DI 290 GO5UB 440:EDIT=1:CHK5UM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GO5UB 500:IF (A=A5C("Q") OR A=A5C(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACK5P AND (A<4
   8 OR A>57) THEN 320
DK 331 IF A=RETRN AND N$="" THEN N$=MOD$
TD 335 IF A=RETRN AND L=0 AND X>1 THEN 35
   0
JR 340 IF ((A=RETRN AND  NOT EDIT) OR A=B
   ACK5P) AND L=0 THEN 320
DW 350 IF A=RETRN THEN POKE 752,1:? " ":R
   ETURN
GG 360 IF A<>BACK5P THEN 400
5A 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
A5 380 N$=""
RE 390 ? CHR$(BACK5P);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WK 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHIC5 0:END
YT 430 GO5UB 440:PO5ITION 10,10:? "NO 5UC
   H FILE!":FOR X=1 TO 1000:NEXT X:CL0SE
   #2:GOTO 30
FD 440 POKE 710,48:5OUND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:5OUND 0,0,0,0:RETURN
MY 450 GRAPHIC5 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 5TEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 5TEP 2:POKE DL+X,0
   :NEXT X
ZW 480 POKE DL+41,65:POKE DL+42,PEEK(560)
   :POKE DL+43,PEEK(561):POKE 87,0
AC 490 PO5ITION 2,0:? "analog ml editor":
   POKE 559,34:RETURN
WZ 500 OPEN #1,4,0,"K:":GET #1,A:CL0SE #1
   :RETURN
```

ASTRONAUTS

Reviewed by Matthew J.W. Ratcliff

Happy's Programs Astronauts is a trivia-quiz program for devoted followers of the NASA programs from the Mercury and Gemini missions through the Space Shuttle (mission 51-L). It is actually a rote drill, asking the same types of questions over and over again.

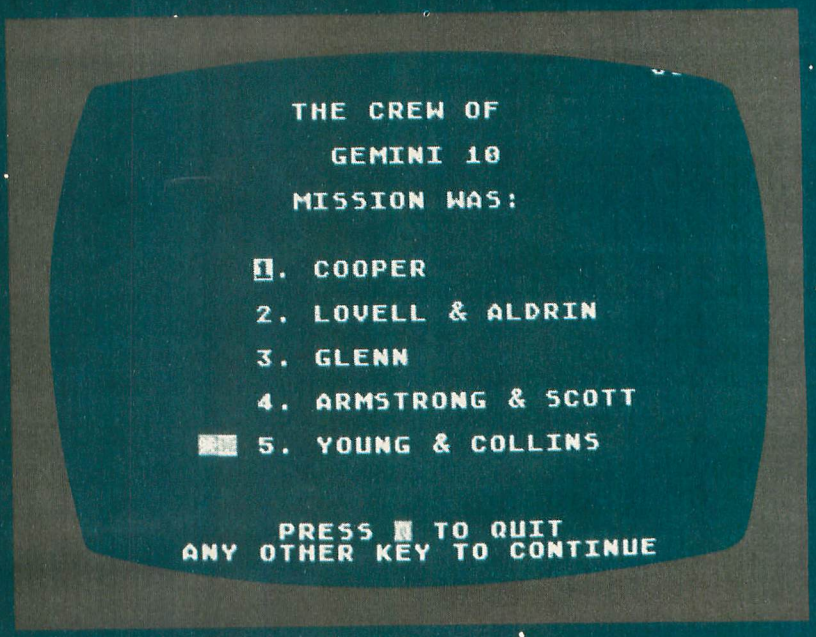
When booted, the main menu presents three selections: You may practice the Mercury/Gemini programs, the Apollo space shots or the Space Shuttle missions. By pressing the appropriate number and Return, or by positioning a highlighter and pressing the fire button, the menu selection is made.

The program's text-menu and question screens can be navigated with either joystick or keyboard control. However, the user is constantly prompted to press a key to continue. At this point a joystick fire button should be equivalent, but instead, the button is simply ignored. The joystick user-interface is inconsistent and frustrating, and for this reason, *Astronauts* is best played entirely with keyboard control.

If Space Shuttle is chosen from the main menu, for example, you are presented with the first question. The questions are selected at random from a database loaded from disk. A typical question would be "Lousma, Fullerton were the crew members of:", followed by five choices of shuttle mission numbers. The only variation on this theme is that in some cases, the mission is given and the crew members are presented multiple choice.

When a question is answered correctly, a rocket blasts off for your entertainment. The drill problems continue until you press the "Q" key at the end of a round to quit. A final tally of questions asked, total correct and percentage score is presented. Pressing any key returns to the main menu.

I found this to be a terribly boring game, not much more exciting than flash cards. If you have a need to learn all the space missions and the names of their crews, *Happy's Programs Astronauts* may prove useful. If trivia is your thing, it might come in handy as a party game. Beyond that, however, I really cannot find anything interesting about this simple quiz program.



Happy's Programs Astronauts
Bensley Consulting
P. O. Box 301
217 W. Walnut St.
Westfield, IL 62474
(217) 967-5465
$19.95

L.A. SWAT

Reviewed by Matthew J. W. Ratcliff

L.A. Swat and *Panther* are two games on one disk from Mastertronic at one affordable price. This game may be found where Commodore 64 games are sold, with the Commodore version on side one and the Atari version on the flip side. Inside the box is a disk, period; all documentation is on the back of the package. The games are fairly simple and easy to learn, so the paperwork won't be missed.

L.A. Swat places you in the West Side of Los Angeles, where a gang has taken over. You lead a team of officers through the streets and shoot the hoodlums as they come at you. Hand grenades are lobbed at you and must be dodged. Occasionally, snipers take pot shots from the rooftops of the buildings that line the streets. Sometimes an overturned car must be worked around.

In a gruesome animated effect, a police officer is clobbered over the head until dead by a hood who catches up with him. Whenever an officer is shot or hit by a grenade, he falls over backward, a well-done effect. One of the officers flanking him takes over the point, until no one is left.

As you walk up the street, shoot along any point of the compass and at 45-degree angles to blow away the gang members coming at you from every direction. At the end of each street you are met by an onslaught of gang members running fast and furious and throwing plenty of grenades. After a short time, one will come running at you holding a hostage. Shoot the bad guy without harming the hostage—your ultimate goal is to save the hostages.

At the end of each street comes the next level, which is faster and tougher to beat. But then, the battle against crime is never finished.

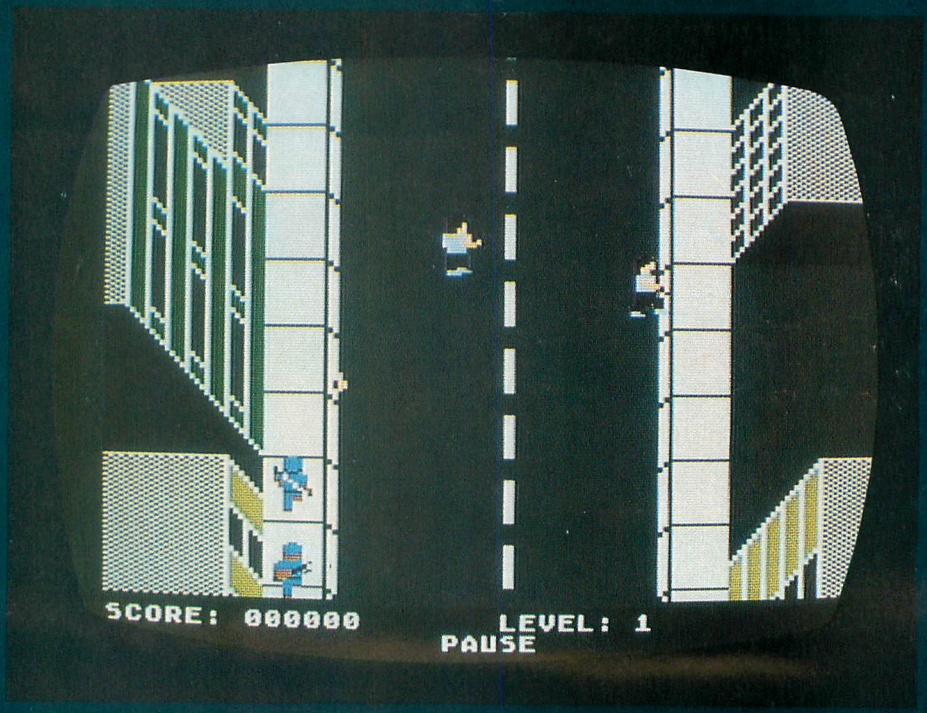
Panther is a smooth-scrolling graphic adventure that is a cross between *Blue Max* and *Choplifter*. Since you are the "one and only poor sucker who can fly a Panther ground attack craft," you must fly diagonally across three different screens in order to save the city of Xenon.

At various depots along the way, you will see people waving at you. Land your Panther craft nearby and pick up as many stranded souls as possible. Don't dally, because a swarm of attack craft will beset you in short order. The object of the game is to collect as many lost people of Xenon as possible and score as many kills of enemy aircraft as you can.

The sound effects and graphics are quite good in *Panther*. There are some haunting effects and music that are superb and make the game a lot more fun to play. The biggest problem with this game is the way the enemy "swarms" you. Often, all three attack craft will come at you at once, with no space on the screen between any of them. Your firing speed at them is very sluggish, as it takes a while to judge the altitude of the attack craft by their shadows. You cannot outfly them; they stick with you until either you kill them or they kill you.

Panther would be an excellent game if each ship you evaded simply flew off the screen and came back later. And if the ships flew in predictable formation and further apart, *Panther* would be playable. As it is, however, it is simply a frustrating joystick exercise with some good graphics and musical effects.

Both *L.A. Swat* and *Panther* are fairly well-done games with a few basic flaws. *L.A. Swat* is quite playable, but without much depth. It is a fairly challenging game that will hold your interest for a while. *Panther* could have been much more. I really wanted to like this game, with its wonderful music and detailed graphics, but it seriously lacks in the playability department. Still, for the low price tag, these are fair rainy-day games.



L.A. Swat and Panther
Mastertronic
711 West 17th St., Unit G9
Costa Mesa, CA 92627
(714) 833-8710
48K Disk, $9.99


SKULL
ISLAND

SKULL ISLAND

Mystified, you wake to find yourself half buried beneath the sands of a strange and vast island. Looking seaward, your eyes gaze upon the remains of your wrecked ship. As you struggle to recall the past events that brought you to this seemingly tropical paradise, your hazy memory pinpoints only a violent storm at sea and your attempts to keep your once-seaworthy vessel afloat.

Certain only that you are somewhere in the South Pacific, you climb to your feet, trying to get a bearing on your present location. Suddenly, your eyes fall upon a wooden board lying on the beach some yards away. As you approach it, you realize it is some kind of a sign. Brushing away the sand from the rotting wood, you reveal the words "Welcome to Skull Island."

Your pulse quickens as you realize the meaning behind these words. In legend, many attempted to sail to Skull Island to discover the magic it possessed, but no man ever returned from his adventure.

The sounds of the jungle surround you. Destiny calls. You set out to escape Skull Island—or die trying.

BY JOHN PATUTO

The Game

Skull Island is a text-adventure game that offers a lot more than just a little challenge. Time was taken in writing and developing this software so that it would have great detail and enough excitement for hours of enjoyment. Two years of programming, problem solving, testing and retesting were spent to create *Skull Island* and to ensure that all who played the game would find it as exciting as possible.

On the outside, *Skull Island* may appear to be an ordinary text-adventure game, but what the inside holds sets it apart from all others. Incorporated into this game are musical tunes and sound effects, along with deliberate background coloring to aid you in your adventure through the vast areas of Skull Island. This, along with an exciting character set and compelling scenario, will show you why *Skull Island* is a game that will not soon be forgotten.

Typing it In

Listing 1 is the main game program. Simply type it as it appears, making certain that all the lines are correct. (Use BASIC Editor II, found elsewhere in this issue, to check your work.) Because of the use of some control characters, you should be very careful to enter them in correctly. One misplaced control character will adversely affect the execution of the program, and it may very well make the game unsolvable. Once you have completed typing Listing 1, save it to disk under the filename D:SKULL.1, but do not run it yet.

Now type Listing 2. As you can see, Listing 2 contains many DATA lines. These lines contain the heart of *Skull Island*. Again, if anything is mistyped, chances are you will not be able to run *Skull Island* successfully.

After typing in Listing 2, save it to disk under the filename D:SKULL.2, then run it. It will prompt you to have the disk with SKULL.1 ready. Once you have it in place, hit Return, and in a few moments it will create two files (D:STRING1 and D:STRING2). You can now run D:SKULL.1 and play the game.

In future play, you need not use D:SKULL.2 again. Simply make sure that D:SKULL.1, D:STRING1 and D:STRING2 are all on the same disk.

Game Play

Skull Island is an adventure game in which you must enter commands to solve riddles and problems in order to win. The program recognizes a variety of words in a standard two-word verb-noun sentence. It also has an array of single-word commands for movement and other special tasks. I'll have more on that later.

When entering-in a two-word command, such as GET WOOD or READ PAPER, you need to type only the first three letters of each word (i.e. GET WOO and REA PAP) for quicker input. Of course, spelling out the entire word will not adversely affect the program in any way. One small hint here: If you need to use GO TO, type it as one word (GOTO).

Along with the two-word commands are single-letter commands for special tasks. These are 'N'orth, 'S'outh, 'E'ast, 'W'est, 'D'own, 'I'nventory, 'H'elp, 'Q'uit, 'X' Present Score, 'Y' Change to Atari Character Set and 'Z' Change to Skull Island Set.

The object of the game is to wander around the island searching for a way to escape. You will find many objects scattered around the island that may prove useful in your escape. There is something unusual about Skull Island, and that is the great wall that surrounds it, making it almost impossible to leave. Yet legend has it that there are magical ways of lowering the wall in certain sections of the island. It is up to you to discover the secret.

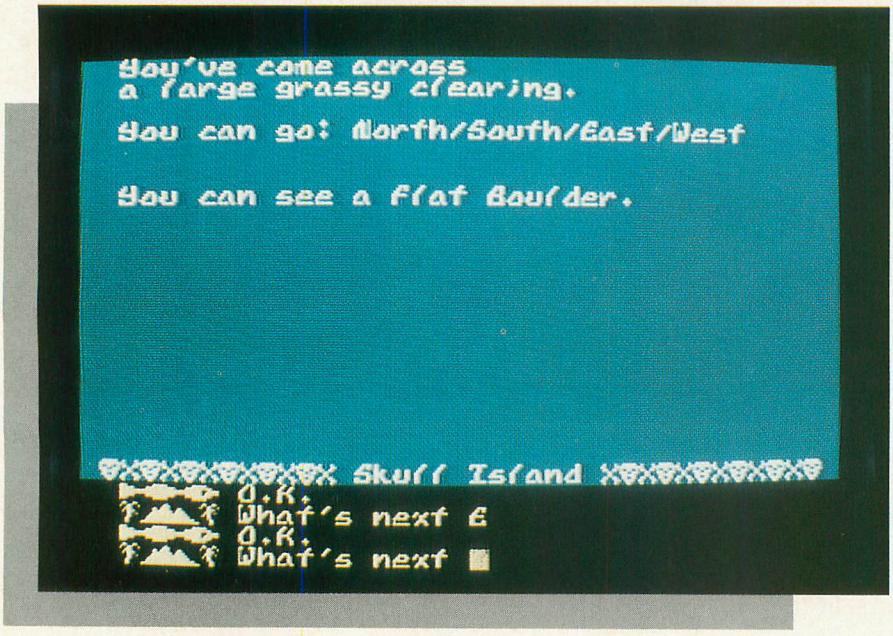
You can carry up to six objects at once; simply hitting "I" will give you an updated list of your inventory. Of course, you can drop any object at any time, but be careful; some things are worth hanging on to.

The Help command is unique. In other adventures, when you ask for help you are usually given a clue or hint as to what to do next. In *Skull Island* you are given a musical tune. It is up to you to decipher what this tune is trying to tell you. There is, however, another way of receiving help: Each time you enter a new section of the island, you are told which directions you can travel in (i.e. north, west. . .?). You will notice the "?" When you see this, the location you are in may or may not be special in some way. Again, it is up to you to discover the meaning behind each question mark.

Also, whenever you see a new object or enter another section, be sure you *examine* everything. If you do not, you may find yourself hopelessly wandering throughout the island.

Finally, *Skull Island* keeps track of your score and number of moves. You are allowed a maximum of 300 moves, and your highest possible score is 285. To view your score and number of turns at any time, enter the "X" command. If you are curious, the smallest number of moves possible to solve this adventure is 82.

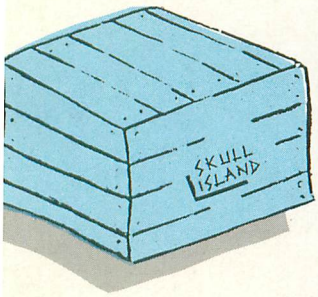
Good luck!



John Patuto is currently studying computer science and communications at William Paterson College in Wayne, New Jersey. His current projects include a series of business programs for the Atari 130XE, an 8-bit Star Trek *simulator, and somewhere down the line, the sequel to* Skull Island.

LISTING 1: BASIC

```
GS 0 POKE 580,1:POKE 16,112:POKE 53774,11
   2:C=1:D=6:E=C+C:A=C-C:H1=D-C
MJ 1 DIM QQ$(1),DLI$(64),L$(43),F$(42),R$
   (16),N$(12),V$(12),DR$(24),DIR$(6),IT$
   (16),VV$(63),NN$(105),INV$(D),II$(D)
LH 2 DIM SW$(12),V1$(3),N1$(3),I0$(D),I1$
   (D),I2$(D),I3$(D),I4$(D),I5$(D),I6$(D)
   ,I7$(D),I8$(D),I9$(D),I10$(D),I11$(D)
MJ 3 DIM I12$(D),I13$(D),I14$(D),I15$(D),
   I16$(D),I17$(D),I18$(D),I19$(D),I20$(D
   ),I21$(D),I22$(D),I99$(D),I16I$(D)
EA 4 DIM YN$(3),OK$(10),WN$(19),C$(31),SS
   $(65),S1$(22),WS$(33),PW$(20),PW1$(20)
UV 8 DIM I25$(D):GOSUB 29000:DON=32009:GS
   =2650:NTH=2605:DHT=2610:NRM=2615:HUH=2
   600
YR 9 PRO=3500:PK=21:OK=49:ILVE=2500:ILNO=
   2550:OKV=54:OKN=2999:LOC=100:NLC=59:IT
   S=1699:DCH=4444:TW=PK-C
ZC 10 OPEN #C,4,A,"K:":POKE 703,4:POKE 75
   6,145:POKE 752,C:POKE 709,14:GOSUB 12:
   GOTO 13
JP 11 POSITION C,19:? #E;">>>>>>>>>>>> Sk
   ull Island <<<<<<<<<<<<";:RETURN
TJ 12 POSITION C,19:? #E;"@&@&@&@&@&@& Sk
   ull Island &@&@&@&@&@&@";:RETURN
TW 13 Q=USR(ADR(DLI$),ADR(DLI$)+32):POKE
   54286,192:POKE 559,34:GOTO 15
RD 14 FOR I=15 TO A STEP -2.5:SOUND A,AT,
   10,I:NEXT I:SOUND A,A,A,A:RETURN
HP 15 POP :POP :QX=A:POSITION E,A:GOSUB L
   OC:? #E:? #E;"You can go: ";DR$:DIR=LE
   N(DIR$)
NY 16 POSITION E,D:? #E;"You can see ";:I
   F BG THEN DL=300
RC 17 IF I99$="XXXXXX" THEN ? #E;"nothing
    interesting.":GOTO PRO
TD 18 II$=I99$:GOSUB IT5:GOTO PRO
VL 20 POP :POP :AT=43:QX=A:N$="":V$="":JP
   =A:R$="":FOR Q=A TO E:GOSUB 14:NEXT Q:
   ? WN$;:NCH=A:VCH=A:VT=A:TRAP TW
CZ 21 POKE 702,64:POKE 694,A:IF PEEK(764)
   =255 THEN GOSUB 17000:GOTO PK
MC 22 VT=A:B=PEEK(764):IF B=60 THEN GOTO
   PK
HT 23 IF B=39 OR B=154 THEN GOTO PK
YS 25 GET #C,R:IF R=155 THEN 50
IK 26 IF (R<32 OR R=127 OR R=125 OR R>154
    AND R<160 OR R>252) AND R<>27 THEN GO
   TO PK
RD 27 IF R=126 AND JP=A THEN GOTO PK
PB 28 IF R=27 THEN R=32
TQ 30 IF R=126 THEN ? "←←2 ←";:R$(JP)="":
   JP=JP-C:GOTO PK
RM 31 GOSUB 35:JP=JP+C:? "←";CHR$(R);"2";
   :R$(JP)=CHR$(R)
MR 32 GOTO PK
BV 35 IF JP=16 THEN GOTO PK
AF 36 RETURN
FQ 49 ? OK$:TR=TR+C:RETURN
WK 50 JP=A:R1=LEN(R$):? "← ":IF R1=A THEN
    V$=" ":GOTO ILVE
SD 51 TRAP 60:Q=USR(L,ADR(R$),ADR(" "),R1
   ):V$=R$(C,Q-C):N$=R$(Q+C,R1):TRAP ILVE
   :V1$=V$(C,3):TRAP ILNO:N1$=N$(C,3)
OA 52 TRAP 15:FOR CH=C TO LEN(VV$) STEP 3
   :VCH=VCH+C:IF V1$=VV$(CH,CH+E) THEN GO
   TO OKV
QE 53 NEXT CH:GOTO ILVE
YD 54 FOR CH=C TO LEN(NN$) STEP 3:NCH=NCH
   +C:IF N1$=NN$(CH,CH+E) THEN GOTO OKN
SE 55 NEXT CH:GOTO ILNO
IA 57 FOR I=D TO 15:X1=USR(ADR(F$),I):NEX
   T I:RETURN
AX 59 XX=A:YY=A:DGR=A:FOR I=A TO 15:X1=US
   R(ADR(F$),I):NEXT I:GOTO 15
GJ 60 GOSUB 75:R$=R$(C,C):FOR CH=C TO 5:I
   F R$=SW$(CH,CH) THEN 70
KU 63 NEXT CH:GOSUB OK:IF R$="I" THEN II$
   =INV$:POSITION E,13:? #2;"You are carr
   ying ";:GOSUB IT5-9:GOTO PRO
FV 64 IF R$="H" THEN 31900
GX 65 IF R$="Q" THEN QU=5:? "K":POKE 756,
   145:GOSUB 12:GOTO DON
CV 66 IF R$="X" THEN GOSUB 1800:GOTO PRO
PR 67 IF R$="Y" THEN ? "K":OK$="♦>>>> O.K
   .":WN$="♣♦●●♦♣ What's next 2":POKE 756,
   224:GOSUB 11:GOTO PRO
FW 68 IF R$="Z" THEN ? "K":OK$="├┼├┼┤ O.K
   .":WN$="●▗ ▄▗▟ What's next 2":POKE 756,
   145:GOSUB 12:GOTO PRO
IP 69 IF R$="X" THEN 15999
RW 70 FOR CH=C TO DIR:IF R$=DIR$(CH,CH) T
   HEN GOSUB OK:GOSUB 2700:GOTO NLC
WA 71 NEXT CH:? "You can't go in that dir
   ection.":GOTO PRO
WU 75 IF LEN(R$)=C THEN 90
BT 76 IF LEN(R$)=E THEN V$=R$:GOTO ILVE
IM 80 V1$=R$(C,3):FOR CH=C TO 33 STEP 3:I
   F V1$=WS$(CH,CH+E) THEN RETURN
BT 81 NEXT CH:GOTO 92
LE 90 FOR CH=C TO 12:IF R$=SW$(CH,CH) THE
   N RETURN
ZW 91 NEXT CH:V$=R$:GOTO ILVE
AA 92 FOR CH=C TO LEN(VV$) STEP 3:IF V1$=
   VV$(CH,CH+E) THEN ? OK$(C,D);R$;" what
   ?":POP :GOTO PRO
LP 93 NEXT CH:FOR CH=C TO LEN(NN$) STEP 3
   :IF V1$=NN$(CH,CH+E) THEN ? OK$(C,D);"
   Do what to the ";R$;"?":POP :GOTO PRO
AC 94 NEXT CH:V$=R$:GOTO ILVE
EU 100 POKE 710,40:POKE 709,14:POKE 712,A
   :? #E;"You are lying on the lagoon of"
   :? #E;"a strange island. A sign says:"
VZ 101 ? #E;"'WELCOME TO SKULL ISLAND'":E
   L=200:WL=1100:I99$=I0$:DR$="East/West"
   :DIR$="EW":H1=5:RETURN
YF 115 I0$=I99$:RETURN
RJ 150 POKE 710,36:POKE 712,C:POKE 709,14
   :? #E;"You are walking along what is":
   ? #E;"known as 'Death Beach'.":H1=7
LT 160 I99$=I1$:EL=1450:WL=1400:DR$="East
   /West":DIR$="EW":RETURN
YV 165 I1$=I99$:RETURN
NJ 200 POKE 710,18:POKE 709,14:? #E;"You'
   re lost deep within a large":? #E;"den
   se rain forest.":NL=1450:SL=250:H1=8
YM 210 I99$=I2$:EL=550:WL=100:DR$="North/
   South/East/West/ ?":DIR$="NSEW":IF LOC
   =200 THEN RETURN
NY 211 GOTO 1460
YS 215 I2$=I99$:RETURN
LU 250 POKE 710,210:POKE 709,14:? #E;"You
   've come across a field that":? #E;"ha
   s dozens of Palm Trees.":NL=200:H1=E
EI 260 I99$=I3$:DR$="North/ ?":DIR$="N":R
   ETURN
ZI 265 I3$=I99$:RETURN
XQ 300 POKE 710,48:POKE 709,14:? #E;"You
   are walking on a forbiddened":? #E;"an
   cient Burial Ground of past":H1=7
DB 301 ? #E;"Mohandian Kings.":SL=350:I99
   $=I4$:DR$="South/ ?":DIR$="S":BG=D:RET
   URN
ZF 315 I4$=I99$:RETURN
UO 350 POKE 710,8:POKE 709,C:? #E;"You ar
   e walking amidst hundreds":? #E;"of er
   upting geysers.":NL=300:EL=400
CS 360 I99$=I5$:DR$="North/East":DIR$="NE
   ":XX=RND(C):YY=TR:H1=C:RETURN
ZV 365 I5$=I99$:RETURN
IR 370 ? "A geyser erupted in your face."
   :GOTO DON
EM 400 POKE 710,32:POKE 709,10:? #E;"You'
   ve come across a clearing":? #E;"which
    has dozens of giagantic":I99$=I6$
YV 401 POKE 712,A:? #E;"Totem Poles.":NL=
   450:EL=950:WL=350:DR$="North/East/West
    ?":DIR$="NEW":H1=E:RETURN
ZS 415 I6$=I99$:RETURN
YF 450 POKE 710,66:POKE 709,14:? #E;"You'
   re walking aside a volcano":? #E;"in a
    dried-up lava fisure.":NL=500:SL=400
SQ 460 I99$=I7$:EL=900:DR$="North/South/E
   ast/ ?":DIR$="NSE":XX=RND(C):YY=TR:H1=
   C:RETURN
AI 465 I7$=I99$:RETURN
PP 470 ? "A lava-avalanche has buried you
   .":GOTO DON
AA 500 POKE 710,128:POKE 709,14:? #E;"A f
   riendly native welcomes":? #E;"you int
   o his village.":NL=550:SL=450:H1=E
EO 510 I99$=I8$:DR$="North/South/ ?":DIR$
   ="NS":RETURN
AF 515 I8$=I99$:RETURN
YY 550 POKE 710,148:POKE 709,14:? #2;"You
   've come across":? #E;"a large grassy
   clearing.":NL=600:SL=500:EL=650:H1=E
JQ 560 I99$=I9$:WL=200:DR$="North/South/E
   ast/West":DIR$="NSEW":RETURN
AV 565 I9$=I99$:RETURN
DV 600 IF  NOT TOR THEN LOC=1350:GOTO LOC
LR 601 GOTO 1500
KG 610 NL=1500:SL=550:I99$=I10$:DR$="Nort
   h/South/ ?":DIR$="NS":XX=RND(C):YY=TR:
   H1=C:RETURN
SO 615 I10$=I99$:RETURN
LM 620 ? "The lion made you his next meal
   .":GOTO DON
YW 650 POKE 710,38:POKE 709,14:? #E;"Quic
   k sand traps surround you":? #E;"as yo
   u step onto this shaky land."
FC 660 I99$=I11$:SL=700:WL=550:DR$="South
   /West":DIR$="SW":XX=RND(C):YY=TR:H1=C:
   RETURN
TF 665 I11$=I99$:RETURN
KR 670 ? "You've sunk into the quicksand.
   ":GOTO DON
NC 700 POKE 712,C:POKE 710,96:POKE 709,14
   :? #E;"You've come across what seems":
```




```
   ? #E;"to be a deserted village."
IN 710 I99$=I12$:NL=650:SL=750:DR$="North
   /South/ ?":DIR$="NS":H1=3:RETURN
TD 715 I12$=I99$:RETURN
ZT 750 POKE 710,36:POKE 712,32:POKE 709,1
   4:? #E;"As you walk apon this sand":?
   #E;"dune, a hasty wind thrashes up."
NO 760 I99$=I13$:NL=700:SL=1000:EL=800:WL
   =850:DR$="North/South/East/West":DIR$=
   "NSEW":H1=E:RETURN
TU 765 I13$=I99$:RETURN
TV 800 POKE 710,34:POKE 712,36:POKE 709,1
   4:? #E;"You've come upon an area fille
   d":? #E;"with bamboo plants. A sign"
FK 801 ? #E;"here says: 'Bamboo burns wel
   l'.":I99$=I14$:WL=750:DR$="West/ ?":DI
   R$="W":H1=E:RETURN
TS 815 I14$=I99$:RETURN
UW 850 POKE 712,C:POKE 710,4:POKE 709,14:
   ? #E;"You are walking along a large":?
    #E;"fisure made of sharp rocks."
UB 860 H1=E:I99$=I15$:SL=900:EL=750:DR$="
   South/East":DIR$="SE":RETURN
UJ 865 I15$=I99$:RETURN
WM 900 POKE 710,20:POKE 709,C:? #E;"A fou
   l stench is in the air as you":? #E;"c
   ome across a dingy swamp area.":H1=7
UL 910 POKE 712,A:I99$=I16$:NL=850:SL=950
   :EL=1000:WL=450:DR$="North/South/East/
   West/ ?":DIR$="NSEW":RETURN
UH 915 I16$=I99$:RETURN
QE 950 POKE 710,14:POKE 709,C:? #E;"You'r
   e at the southern edge of":? #E;"the i
   sland. There's a strange":H1=D
SE 951 ? #E;"feeling in the air.":I99$=I1
   6I$:NL=900:WL=400:DR$="North/West/ ?":
   DIR$="NW":BG=A:RETURN
YD 965 I16I$=I99$:RETURN
WA 1000 POKE 712,C:POKE 710,18:POKE 709,1
   4:? #E;"Animals are all around you":?
   #E;"as you enter a savage jungle."
NL 1010 I99$=I17$:NL=750:SL=1050:WL=900:D
   R$="North/South/West":DIR$="NSW":H1=7:
   RETURN
QX 1015 I17$=I99$:RETURN
QI 1050 POKE 710,245:POKE 709,14:? #E;"Yo
   u're at the only section of":? #E;"the
    island where you can make":H1=E
FT 1051 ? #E;"something of it.":NL=1000:I
   99$=I18$:DR$="North/ ?":DIR$="N":RETUR
   N
RU 1065 I18$=I99$:RETURN
FH 1100 POKE 710,128:POKE 712,128:POKE 70
   9,14:? #E;"You're now swimming in the"
   :? #E;"lagoon.  Havin' fun ?":EL=100
FJ 1110 I99$=I25$:DR$="East/ ?":DIR$="E":
   DGR=5:YY=RND(C):XX=TR:H1=C:RETURN
MU 1115 ? "It's gone forever now!":I99$="
   XXXXXX":GOSUB 5000:POKE 709,14:RETURN
JY 1120 ? "You just drowned.":GOTO DON
CV 1150 POKE 710,128:POKE 712,148:POKE 70
   9,14:? #E;"You are now in your boat.":
   ? #E;"It's been a tiring day.":DGR=5
YC 1160 I99$=I25$:DR$="?":DIR$=" ":YY=RND
   (0):XX=TR:H1=4:RETURN
CM 1170 ? "You failed to complete your jo
   urney.":GOTO DON
IZ 1200 ? #E;"You are on top of a":? #E;"
   very tall Palm Tree."
OA 1210 DL=250:I99$=I19$:DGR=5:DR$="Down"
   :DIR$="D":H1=9:RETURN
RR 1215 I19$=I99$:RETURN
NF 1250 ? #E;"You are on top of a":? #E;"
   strange platform.":DL=950:I99$=I25$:DG
   R=5:DR$="Down":DIR$="D":H1=D:RETURN
PC 1265 GOSUB 5000:GOSUB 18000:? "It's be
   en zapped to another location!"
JO 1268 FOR CH=C TO D:IF I99$(CH,CH)="X"
   THEN I99$(CH,CH)=CHR$(NCH):GOSUB LOC+1
   5:I99$=I25$:LOC=1250:RETURN
AK 1269 NEXT CH:GOSUB 18000:GOTO 1268
KN 1300 ? #E;"You are inside a native's":
   ? #E;"hut. Someone's here."
OV 1310 EL=500:I99$=I20$:DR$="East":DIR$=
   "E":H1=E:RETURN
PG 1315 I20$=I99$:RETURN
SQ 1350 POKE 710,C:POKE 709,E:? #E;"It's
   to dark to see":? #E;"anything in here
   .":SL=550:I99$=I25$:DGR=5
WC 1360 DR$="South ... and hurry!":DIR$="
   S":YY=RND(C):XX=TR:H1=C:RETURN
RX 1365 GOTO 1115
OH 1370 ? "You fell and broke your neck."
   :GOTO DON
QR 1400 POKE 710,128:POKE 712,128:? #E;"Y
   ou are now swimming in":? #E;"the ocea
   n. Sharks are all":? #E;"around you."
LR 1410 DGR=5:I99$=I25$:EL=150:DR$="East/
    ?":DIR$="E":YY=RND(C):XX=TR:H1=C:RETU
   RN
RK 1415 GOTO 1115
VW 1420 ? "You were just attacked by shar
   ks.":GOTO DON
OA 1450 GOTO 200
VD 1460 SL=200:WL=150:H1=8:I99$=I21$:DR$=
   "South/West/ ?":DIR$="SW":RETURN
QF 1465 I21$=I99$:RETURN
CG 1500 POKE 710,C:POKE 709,14:? #E;"You'
   re walking dangerously":? #E;"thru the
    Caves of Death.":SL=600:I99$=I22$
VL 1510 H1=C:DR$="South/ ?":DIR$="S":IF L
   OC=1500 THEN RETURN
PY 1511 GOTO 610
QA 1515 I22$=I99$:RETURN
VV 1690 IF II$="XXXXXX" THEN ? #E;"nothin
   g at all.":RETURN
GR 1691 QX=C
AE 1699 FOR IC=C TO D:IF II$(IC,IC)="X" T
   HEN NEXT IC
DG 1700 IF  NOT QX THEN IN8=A
WB 1701 RESTORE 2099+ASC(II$(IC,IC)):READ
    IT$:? #E;"a ";IT$;".":IF  NOT QX THEN
    IN8=IN8+C:GOTO 1776
RN 1710 FOR CH=IC+C TO D:IF II$(CH,CH)<>"
   X" THEN RESTORE 2099+ASC(II$(CH,CH)):R
   EAD IT$:GOTO 1730
GB 1720 NEXT CH:RETURN
DH 1730 IF QX THEN ? #E,"        A ";IT$;"
   .":GOTO 1750
AQ 1740 ? #E,"  A ";IT$;"."
SK 1750 IF  NOT QX THEN IN8=IN8+C
SB 1751 GOTO 1720
JY 1776 IF IC=6 THEN RETURN
SS 1777 GOTO 1710
AP 1800 ? "You've scored ";SC;" points in
    ";TR;" turns":RETURN
QL 2099 DATA Coconut Shell
QN 2100 DATA Piece of Paper
EK 2101 DATA Piece of Wood
ZC 2102 DATA Coconut
UD 2103 DATA Black Pearl
GU 2104 DATA Diamond
FV 2105 DATA Sharp Rock
ES 2106 DATA Crystal Skull
CG 2107 DATA Rusty Saw
BT 2108 DATA Piece of Bamboo
KF 2109 DATA Bag
WD 2110 DATA Bunch of Nails
DL 2111 DATA Machete
LY 2112 DATA Piece of Bark
MH 2113 DATA Torch
NZ 2114 DATA Giant Volcano
HA 2115 DATA Tall Palm Tree
FP 2116 DATA lot of Trees
OC 2117 DATA Strange Platform
LY 2118 DATA Burial Platform
FJ 2119 DATA Symbolic Pole
KX 2120 DATA Native Hut
JH 2121 DATA Native Girl
YD 2122 DATA Flat Boulder
YZ 2123 DATA Hungry Lion
UE 2124 DATA Bell
AE 2125 DATA Native
OI 2126 DATA Stalk of Bamboo
PE 2127 DATA Large Stone
XQ 2128 DATA Small Fire
YI 2129 DATA Hole
LP 2500 FOR CH=C TO LEN(V$):V$(CH,CH)=CHR
   $(ASC(V$(CH,CH))+128):NEXT CH
QR 2501 ? "I can't '";V$;"' something.":G
   OTO PRO
JO 2550 FOR CH=C TO LEN(N$):N$(CH,CH)=CHR
   $(ASC(N$(CH,CH))+128):NEXT CH
OH 2551 ? "I don't know what a '";N$;"' i
   s.":GOTO PRO
KL 2600 ? "I don't understand what you me
   an.":GOTO PRO
UH 2605 ? "I don't see it here.":GOTO PRO
YY 2610 ? "You can't, you don't have it!"
   :GOTO PRO
FG 2615 ? "You can't, there's no room her
   e.":GOTO PRO
TQ 2650 ? "That's impossible.":GOTO PRO
VG 2700 IF R$="N" THEN LOC=NL
CS 2701 IF R$="S" THEN LOC=SL
IK 2702 IF R$="E" THEN LOC=EL
IW 2703 IF R$="W" THEN LOC=WL
HG 2704 IF R$="D" THEN LOC=DL
BI 2705 RETURN
WN 2999 NCH=NCH-C:IF VCH=TW+C THEN VCH=C:
   GOTO 3003
UH 3000 IF VCH=13 THEN VCH=12
KX 3001 IF VCH>12 THEN VCH=VCH-C
DR 3002 ON VCH GOTO 3900,3900,4024,4024,4
   056,4075,4100,4105,4109,4109,4150,4175
   ,4185,4200,4225,4225,4250,4275,4300
FQ 3003 ON VCH GOTO 4325
DI 3100 IF NCH=E OR NCH=13 OR NCH=11 THEN
    SHIP=SHIP-5
```

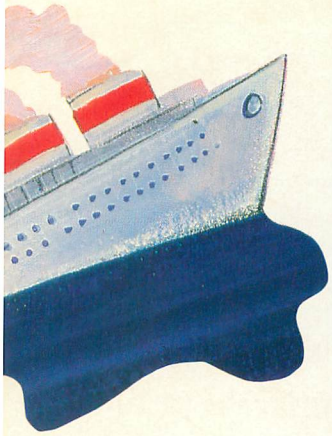
```
XH 3101 IN7=IN7-C:RETURN
UW 3333 FOR I=50 TO C STEP -C:SOUND C,50-
   I,10,10:NEXT I
SY 3334 FOR I=15 TO A STEP -0.5:SOUND C,7
   5+I,8,I:NEXT I:RETURN
FN 3400 FOR I=C TO 4:X=ASC(SS$(I,I)):SOUN
   D E,X,10,15:FOR CH=C TO 10:NEXT CH:SOU
   ND E,A,A,A:FOR CH=15 TO A STEP -0.75
AZ 3401 POKE 712,CH:SOUND E,X,10,CH:NEXT
   CH:NEXT I:RETURN
MQ 3450 FOR CH=C TO 22 STEP E:X=ASC(S1$(C
   H,CH)):SOUND E,X,10,8:SOUND 3,X/E,10,D
   :FOR I=C TO ASC(S1$(CH+C,CH+C))*E
EM 3455 NEXT I:SOUND E,A,A,A:NEXT CH:SOUN
   D 3,A,A,A:RETURN
KO 3475 FOR CH=5 TO 10:X=ASC(SS$(CH,CH)):
   SOUND E,X,10,6:SOUND 3,X/E,10,8:FOR I=
   C TO CH:NEXT I:SOUND 3,A,A,A:NEXT CH
ON 3478 SOUND E,A,A,A:RETURN
SD 3500 IF TOR AND TR>TOR+3 THEN TOR=A:?
   "Your torch just went out.":GOSUB 3550
   :GOTO 3506
IO 3501 IF  NOT FIRE THEN 3506
DH 3502 IF TR<FIRE+4 THEN 3506
WO 3503 FIRE=A:? "The fire went out.":FOR
    CH=2 TO D:IF I9$(CH,CH)="↓" THEN I9$(
   CH,CH)="X":GOTO 3505
KJ 3504 NEXT CH
PD 3505 IF LOC=550 THEN GOTO NLC
KK 3506 IF TR>15 AND  NOT FOOD THEN ? OK$
   (C,D);"You have starved to death.":GOT
   O DON
IX 3507 IF TR=300 THEN ? OK$(C,D);"You ha
   ve taken to many turns.":GOTO DON
LP 3510 IF  NOT YY OR YY=INT(YY) THEN 352
   0
UW 3511 IF TR=XX THEN 3515
QL 3512 IF YY<0.8 THEN GOTO LOC+TW
KY 3513 YY=RND(C):GOTO TW
QP 3515 TR=TR+C:GOTO TW
PL 3520 IF  NOT XX THEN GOTO TW
WW 3525 IF TR=YY THEN GOTO TW
HL 3530 IF XX<0.13 THEN GOTO LOC+TW
WC 3535 IF XX<0.5 AND  NOT SKU THEN GOTO
   LOC+TW
KI 3540 XX=RND(C):GOTO TW
AC 3550 IF LOC=600 OR LOC=1500 THEN LOC=1
   350:GOTO NLC
BE 3551 RETURN
FI 3900 IF NCH>14 THEN GOTO G5
XF 3950 IF IN7=D THEN ? "You are carrying
    to much.":GOTO PRO
UW 3975 FOR CH=C TO D:IF I99$(CH,CH)=CHR$
   (NCH) THEN GOSUB OK:I99$(CH,CH)="X":GO
   TO 4000
SV 3976 NEXT CH:FOR CH=C TO D:IF INV$(CH,
   CH)=CHR$(NCH) THEN ? "You already have
    that item.":GOTO PRO
OA 3977 NEXT CH:GOTO NTH
II 4000 IF NCH<>11 THEN 4006
PS 4004 FOR HC=C TO D:IF INV$(HC,HC)="◣"
   THEN NAI=C:SHIP=SHIP+5:GOTO 4006
KP 4005 NEXT HC:? "Not without something
   to put them in.":I99$(CH,CH)=" ":GOTO
   PRO
MH 4006 IN7=IN7+C:SC=SC+10:IF NCH=E OR NC
   H=13 THEN SHIP=SHIP+5
KT 4012 IF NCH=4 OR NCH=5 OR NCH=7 THEN S
   C=SC+10
NN 4013 IF NCH=7 THEN SKU=5
BG 4015 GOSUB LOC+15:FOR CH=C TO D:IF INV
   $(CH,CH)="X" THEN INV$(CH,CH)=CHR$(NCH
   ):GOTO 4021
JQ 4020 NEXT CH
BU 4021 GOSUB 57:POSITION E,13:? #2;"You
   are carrying ";:II$=INV$:GOSUB ITS-9:Q
   X=0:GOTO 16
FR 4022 GOTO PRO
IR 4024 IF IN8=D THEN GOTO NRM
FR 4025 IF NCH>14 THEN GOTO G5
ZP 4026 FOR CH=C TO D:IF INV$(CH,CH)=CHR$
   (NCH) THEN 4028
FY 4027 NEXT CH:GOTO DHT
ZX 4028 IF (NCH=5 AND LOC<>1300) OR (NCH=
   4 AND LOC<>700) OR (NCH=7 AND LOC<>105
   0) THEN SC=SC-TW:IN7=IN7-C:GOTO 4030
RP 4029 GOTO 4031
NE 4030 GOSUB 3333:? "It shattered into a
    million pieces!":GOSUB DCH:GOTO 4021
LG 4031 IF NCH=4 OR NCH=5 OR NCH=7 THEN G
   OSUB OK:GOSUB 4500:GOTO 4038
NL 4032 IF LOC=550 AND NCH=D THEN GOSUB 4
   400
ZS 4033 IF NCH=3 AND LOC<>1250 THEN 4045
TU 4034 IF NCH=10 AND NAI AND IN8>4 THEN
   GOTO NRM
IU 4035 IF NCH=10 AND NAI THEN SC=SC-TW:S
   HIP=SHIP-5:GOTO 4039
LP 4036 GOSUB 3100:GOSUB OK:FOR CH=C TO D
   :IF I99$(CH,CH)="X" THEN I99$(CH,CH)=C
   HR$(NCH):GOSUB LOC+15:GOTO 4038
KV 4037 NEXT CH
TD 4038 SC=SC-10:GOSUB DCH:GOSUB 4053:GOT
   O 4021
PJ 4039 GOSUB 4051:FOR CH=C TO D:IF INV$(
   CH,CH)="◣" THEN INV$(CH,CH)="X"
NO 4040 IF INV$(CH,CH)=" " THEN INV$(CH,C
   H)="X"
QB 4041 NEXT CH:FOR CH=C TO D:IF I99$(CH,
   CH)="X" THEN I99$(CH,CH)="◣":GOSUB LOC
   +15:GOTO 4043
KE 4042 NEXT CH
KT 4043 FOR CH=C TO D:IF I99$(CH,CH)="X"
   THEN I99$(CH,CH)=" ":IN7=IN7-E:GOSUB L
   OC+15:GOSUB OK:GOTO 4021
KM 4044 NEXT CH
BN 4045 IF IN8>4 THEN GOTO NRM
XF 4046 GOSUB OK:GOSUB 3333:FOR CH=C TO D
   :IF I99$(CH,CH)="X" THEN I99$(CH,CH)="
   ♥":GOTO 4048
KY 4047 NEXT CH
EZ 4048 FOR CH=C TO D:IF I99$(CH,CH)="X"
   THEN I99$(CH,CH)="┤":GOSUB LOC+15:IN8=
   IN8+E:GOTO 4050
LG 4049 NEXT CH
IW 4050 GOSUB DCH:IN7=IN7-C:GOTO 4021
LH 4051 IF LOC=1250 THEN ? "You can't dro
   p the bag of nails here.":GOTO PRO
AZ 4052 RETURN
AW 4053 IF NCH=14 AND TOR THEN ? "The tor
   ch went out.":TOR=A:IF LOC=1500 OR LOC
   =600 THEN LOC=1350:GOTO NLC
BH 4054 RETURN
SD 4056 IF NCH=C THEN 4060
DK 4057 IF NCH<>18 THEN GOTO HUH
VL 4058 IF LOC<>400 THEN GOTO NTH
AO 4059 GOSUB OK:? "It says: Girls fancy
   precious stones":? ,"↑Men prefer dark
   beads.":GOTO PRO
DS 4060 FOR CH=C TO D:IF INV$(CH,CH)="├"
   THEN GOSUB OK:? "↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑It read
   s:":GOTO 4062
FM 4061 NEXT CH:GOTO DHT
LY 4062 ? "To lower the weir around this"
   :? "atoll, leave a glister on a stone.
   ":GOTO PRO
JF 4075 IF NCH<15 OR NCH>17 THEN ? "You c
   an't climb that!":GOTO PRO
LK 4076 ON NCH-14 GOTO 4077,4080,4090
YK 4077 IF LOC<>450 THEN GOTO NTH
BR 4078 LOC=300:GOSUB OK:? "You slid over
    the top to a new place.":GOTO NLC
DZ 4080 IF LOC=200 OR LOC=1450 THEN ? "Yo
   u can't climb these trees.":GOTO PRO
WN 4081 IF LOC<>250 THEN GOTO NTH
UU 4082 GOSUB OK:LOC=1200:GOTO NLC
JT 4090 IF LOC<>300 AND LOC<>950 THEN GOT
   O NTH
GM 4091 GOSUB OK:IF LOC=300 THEN BG=5
DA 4092 LOC=1250:GOTO NLC
NG 4100 IF NCH=20 AND LOC=1300 THEN ? "Sh
   e's not into that!◙":GOTO PRO
HX 4101 IF NCH<>3 THEN ? "You can't eat t
   hat !":GOTO PRO
WC 4102 FOR CH=C TO D:IF INV$(CH,CH)=CHR$
   (NCH) THEN GOSUB OK:? "Thanks, but it
   tasted kind of funny.":GOTO 4104
FE 4103 NEXT CH:GOTO DHT
IB 4104 IN7=IN7-C:SC=SC-5:FOOD=5:GOSUB DC
   H:GOTO 4021
DS 4105 IF NCH<>29 THEN GOTO HUH
KP 4106 IF FOOD THEN ? "You already drank
    the milk.":GOTO PRO
GC 4107 FOR CH=C TO D:IF INV$(CH,CH)="♥"
   THEN GOSUB OK:? "Thank you, it was del
   icious.":FOOD=5:GOTO PRO
FY 4108 NEXT CH:GOTO DHT
AU 4109 IF NCH>29 AND (NCH<>31 AND NCH<>3
   2) THEN GOTO HUH
GK 4110 GOSUB 15000:IF NCH=16 AND (LOC=20
   0 OR LOC=1450) THEN 4119
SU 4111 IF NCH=16 AND LOC=250 THEN 4130
DH 4112 IF NCH=17 AND (LOC=300 OR LOC=950
   ) THEN 4130
WF 4113 IF NCH>15 THEN NCH=NCH+E
WG 4114 FOR CH=C TO D:IF INV$(CH,CH)=CHR$
   (NCH) OR I99$(CH,CH)=CHR$(NCH) THEN 41
   16
ML 4115 NEXT CH:GOTO NTH
PW 4116 GOSUB OK:IF NCH>16 THEN 4118
QO 4117 ON NCH+C GOTO 4120,4122,4121,4123
   ,4124,4124,4125,4124,4121,4119,4146,41
   42,4126,4121,4128,4130
PI 4118 ON NCH-19 GOTO 4122,4123,4131,413
   3,4132,4119,4119,4119,4119,4140,4134
```

```
LU 4119 ? "There's nothing interesting ab
   out it.":GOTO PRO
AZ 4120 IF  NOT FOOD THEN ? "It's full of
    coconut milk.":GOTO PRO
XF 4121 ? "Looks like it could be useful.
   ":GOTO PRO
CH 4122 ? "It has something written on it
   .":GOTO PRO
EM 4123 ? "Sounds like something's inside
   .":GOTO PRO
ZY 4124 ? "It's very fragile.":GOTO PRO
VL 4125 ? "It looks like flint.":GOTO PRO
TI 4126 IF SHA THEN ? "It seems quite sha
   rp.":GOTO PRO
NN 4127 ? "Seems kind of dull.":GOTO PRO
RF 4128 IF TOR THEN ? "It's burning brigh
   tly.":GOTO PRO
LH 4129 ? "It isn't burning.":GOTO PRO
RD 4130 ? "It looks climable.":GOTO PRO
JR 4131 ? "She'd like a sign of affection
   .":GOTO PRO
PI 4132 ? "It seems quite ferocious !":GO
   TO PRO
BM 4133 ? "It has ashes around it.":GOTO
   PRO
JG 4134 FOR CH=C TO D:IF I99$(CH,CH)="\"
   THEN ? "There's a Crystal Skull in the
   re.":GOTO PRO
RU 4135 NEXT CH:? "It's quite empty.":GOT
   O PRO
XY 4140 ? "It seems quite hot.":GOTO PRO
UD 4142 ? "They could be useful.":GOTO PR
   O
HC 4145 ? "It's quite sea-worthy.":GOTO P
   RO
UK 4146 IF NAI THEN ? "It's full of nails
   .":GOTO PRO
DZ 4147 FOR CH=C TO C:GOTO 4135
CZ 4150 IF NCH<>19 THEN GOTO HUH
UZ 4151 IF LOC<>500 THEN GOTO NTH
AU 4152 LOC=1300:GOSUB OK:GOTO NLC
NE 4175 IF NCH<>TW AND NCH<>33 THEN GOTO
   G5
GK 4176 GOSUB 12000:IF LOC<>1300 THEN ? "
   I don't see her around here.":GOTO PRO
IE 4177 IF  NOT KIS THEN ? "She'd like a
   gift first.":GOTO PRO
TL 4178 IF FC THEN 4184
KC 4179 IF IN8=D THEN GOTO NRM
ZU 4180 GOSUB OK:GOSUB 3400:POKE 709,14:?
    "That was quite pleasing. She":? "tha
   nks you by droping a saw.":FC=5
RL 4181 FOR CH=E TO D:IF I99$(CH,CH)="X"
   THEN I99$(CH,CH)="◢":GOSUB LOC+15:SC=5
   C+15:GOTO 4021
KS 4182 NEXT CH
AB 4184 ? "Being the savage that she is,
   she":? "stabs you for being too kinky
   !":GOTO DON
VJ 4185 IF NCH<>14 THEN ? "You can't ligh
   t that on fire !":GOTO PRO
FP 4186 IF  NOT FIRE OR LOC<>550 THEN ? "
   I don't see a fire around here.":GOTO
   PRO
ZA 4187 FOR CH=C TO D:IF INV$(CH,CH)="_"
   THEN 4189
GW 4188 NEXT CH:GOTO DHT
TM 4189 GOSUB OK:SC=SC+15:? "It's now lit
   , but not for long.":TOR=TR:GOTO PRO
ZO 4200 IF NCH<>23 THEN GOTO HUH
VO 4201 IF LOC<>700 THEN GOTO NTH
WY 4202 IF IN8=D THEN 4179
SW 4203 GOSUB OK:GOSUB 4212:FOR CH=E TO D
   :IF I99$(CH,CH)="└" THEN GOTO PRO
YQ 4204 NEXT CH:? "Suddenly, a native app
   ears";:FOR CH=E TO D:IF I99$(CH,CH)="X
   " THEN I99$(CH,CH)="└":GOTO 4206
KI 4205 NEXT CH
XV 4206 GOSUB LOC+15:FOR CH=E TO D:IF I12
   $(CH,CH)="┤" THEN 4210
BJ 4207 NEXT CH:? ".":SC=SC+5:GOTO 4021
WL 4210 ? " and takes":? "the pearl. He s
   ays 'Dig a hole where":? "the ground c
   an measure time.'":SC=SC+25
AX 4211 I99$(CH,CH)="X":GOSUB LOC+15:GOTO
    4021
PZ 4212 FOR QX=C TO 3:FOR I=15 TO A STEP
   -0.5:SOUND C,75,10,I:SOUND A,75,10,I:N
   EXT I:NEXT QX:RETURN
PZ 4225 IF NCH<>25 AND NCH<>16 THEN GOTO
   HUH
LN 4226 IF NCH=25 THEN 4235
HY 4227 IF LOC=250 THEN ? "You can't do t
   hat to this tree.":GOTO PRO
GI 4228 IF LOC<>200 AND LOC<>1450 THEN ?
   "I don't see any trees here.":GOTO PRO
ET 4229 IF CHOP THEN ? "You already did."
   :GOTO PRO
RQ 4230 FOR CH=C TO D:IF INV$(CH,CH)="◢"
   THEN GOSUB OK:SC=SC+15:CHOP=5:GOTO 423
   2
NE 4231 NEXT CH:? "You can't do that empt
   y handed.":GOTO PRO
CS 4232 IF LOC=200 THEN I2$(C,C)="▔":I21$
   (C,C)="X":I99$=I2$
UM 4233 IF LOC=1450 THEN I21$(C,C)="▔":I2
   $(C,C)="X":I99$=I21$
KP 4234 GOSUB 10000:? "As this tree falls
   , so do all the":? "rest in the dark f
   orest.":GOTO 4021
UD 4235 IF LOC<>800 THEN ? "I don't see a
    Stalk here.":GOTO PRO
UX 4236 IF CTB THEN ? "You already did.":
   GOTO PRO
OQ 4237 FOR CH=C TO D:IF INV$(CH,CH)="▗"
   THEN 4239
SQ 4238 GOTO 4231
XG 4239 IF  NOT SHA THEN ? "The Machete's
    not sharp enough.":GOTO PRO
IQ 4240 CTB=5:GOSUB OK:SC=SC+5:I14$(C,C)=
   "▗":I99$=I14$:GOTO 4021
DA 4250 IF NCH<>28 THEN GOTO HUH
XY 4251 IF DGR THEN ? "You can't dig a ho
   le here.":GOTO PRO
CG 4252 IF IN8=D OR (LOC=750 AND IN8>4) T
   HEN GOTO NRM
TY 4253 FOR CH=C TO D:IF INV$(CH,CH)="♥"
   THEN 4255
LE 4254 NEXT CH:? "You don't have anythin
   g to dig with.":GOTO PRO
JJ 4255 GOSUB 4261:FOR CH=C TO D:IF I99$(
   CH,CH)="€" THEN ? "There's already one
    here.":GOTO PRO
ES 4256 NEXT CH:GOSUB OK:FOR CH=C TO D:IF
    I99$(CH,CH)="X" THEN I99$(CH,CH)="€":
   GOSUB LOC+15:GOTO 4258
LF 4257 NEXT CH
QZ 4258 IF LOC<>750 THEN GOTO 4021
RI 4259 ? "A Crystal Skull is inside the
   hole!":SC=SC+5:FOR CH=CH+C TO D:IF I99
   $(CH,CH)<>"X" THEN NEXT CH
JI 4260 I99$(CH,CH)="\":SC=SC+5:GOSUB LOC
   +15:GOTO 4021
EX 4261 IF  NOT FOOD THEN ? "Not until yo
   u do something first.":GOTO PRO
BG 4262 RETURN
AA 4275 IF NCH<>12 THEN GOTO HUH
DE 4276 IF LOC<>850 THEN ? "You can't do
   that here.":GOTO PRO
UW 4277 FOR CH=C TO D:IF INV$(CH,CH)="▗"
   THEN GOSUB OK:SHA=5:? "Your Machete is
    now sharp.":GOTO PRO
GV 4278 NEXT CH:GOTO DHT
FZ 4300 IF NCH<31 THEN GOTO HUH
WE 4301 IF LOC<>1050 THEN ? "You can't do
    that here.":GOTO PRO
WE 4302 IF  NOT FENC THEN ? "You can't do
    that...Yet!!":GOTO PRO
TH 4303 IF SHIP<>15 THEN ? "You're missin
   g an essential item.":GOTO PRO
UP 4304 IF IN7>4 THEN ? "Drop the things
   you don't need first.":GOTO PRO
TZ 4305 SC=SC+TW:GOSUB OK:LOC=1150:INV$="
   XXXXXX":GOTO NLC
NQ 4325 IF NCH<>30 AND NCH<>34 THEN GOTO
   HUH
YN 4326 GOSUB 13000:GOSUB OK:IF LOC=1150
   THEN SC=SC+50:? "K":POKE 756,145:GOSUB
    12:GOTO 4330
PX 4327 ? "Sleeping on Skull Island is ri
   sky.":? "A head-hunter has just scalpe
   d you.":GOTO DON
DQ 4330 FOR I=A TO 15:X1=USR(ADR(F$),I):N
   EXT I:FOR CH=E TO 15:POSITION 3,CH:? #
   2;"@@@ XXX CONGRATULATIONS!!! XXX @@@"
LW 4331 POKE 710,CH*E:SOUND E,CH*3,10,8:S
   OUND 3,CH*D,10,D:NEXT CH:POKE 710,128:
   FOR I=C TO 40:NEXT I
PO 4332 FOR CH=C TO 31:I=ASC(C$(CH,CH)):S
   OUND E,I,10,10:SOUND 3,I*3,10,4:FOR QX
   =C TO D:NEXT QX
CP 4333 SOUND E,A,A,A:NEXT CH:SOUND 3,A,A
   ,A
OZ 4334 X1=USR(ADR(F$),12):POSITION 4,13:
   ? #E;"You have survived a journey thru
   ":? #E;"  Skull Island.";:WIN=D
NG 4341 ? #E;" You are now living":? #E;"
    on Paradise Isle with the native":?
   #E;"  girl as your wife.":GOTO DON
CZ 4400 FOR CH=E TO D:IF I99$(CH,CH)="▗"
   THEN I99$(CH,CH)="↓":GOSUB LOC+15:GOSU
   B OK:GOSUB DCH:SC=SC+TW:GOTO 4402
FW 4401 NEXT CH:RETURN
FH 4402 GOSUB 3334:IN8=IN8-C:? "As the ro
   ck strikes the boulder, it":? "lights
   the bamboo on fire.":FIRE=TR
```

```
LW 4403 IN7=IN7-C:GOTO 4021
TF 4444 FOR CH=C TO D:IF INV$(CH,CH)=CHR$
   (NCH) THEN INV$(CH,CH)="X":GOTO 4446
KY 4445 NEXT CH
CH 4446 IF NCH=11 THEN NAI=A
ZW 4447 IF NCH=7 THEN SKU=A
CC 4448 RETURN
AU 4500 IN7=IN7-C:IF NCH=4 THEN 4510
KV 4501 IF NCH=5 THEN 4520
UU 4502 ? "The Skull disappears into a cl
   oud of":? "smoke as the fence lowers h
   ere.":GOSUB 5000:FENC=5:SC=SC+TW
ZH 4503 POKE 709,14:SKU=A:RETURN
BZ 4510 FOR CH=C TO D:IF I99$(CH,CH)="└"
   THEN ? "He thanks you and says: 'Dig a
    hole":? "where the ground";:GOTO 4512
HT 4511 NEXT CH:IN7=IN7+C:? "↑";:GOTO 403
   6
TL 4512 ? " can measure time.'":SC=SC+TW:
   RETURN
WZ 4520 ? "She takes it gratefully.":KI5=
   5:RETURN
AZ 5000 FOR CH=A TO 200 STEP 3:POKE 709,C
   H:SOUND A,CH,10,10:NEXT CH:SOUND A,A,A
   ,A:RETURN
MQ 10000 FOR CH=C TO 4:JP=D:QX=12:GOSUB 1
   0010:JP=8:QX=8:GOSUB 10010:NEXT CH:GOS
   UB 3333:RETURN
JZ 10010 FOR X=JP+5 TO JP STEP -E:GOSUB 1
   0065:NEXT X:FOR X=JP TO JP+4 STEP 5:GO
   SUB 10065:NEXT X:SOUND 3,A,A,A
MI 10011 SOUND C,A,A,A:FOR R=C TO D:NEXT
   R:RETURN
CM 10065 SOUND 3,X*3,C,QX:SOUND C,X,8,QX*
   0.7:RETURN
EM 12000 IF NCH=TW THEN RETURN
GK 12001 GOTO HUH
QH 13000 IF NCH=30 THEN RETURN
XB 13005 ? "Watch it!▯":GOTO PRO
WB 15000 IF NCH<>29 AND NCH<>31 AND NCH<>
   32 THEN RETURN
NS 15005 IF NCH<>29 THEN 15100
LW 15010 FOR CH=C TO D:IF (INV$(CH,CH)="♥
   " OR I99$(CH,CH)="♥") AND  NOT FOOD TH
   EN POP :GOTO 15500
OR 15015 NEXT CH:GOTO NTH
IH 15100 IF LOC<>1150 THEN GOTO NTH
IX 15110 GOSUB OK:GOTO 4145
XY 15500 GOSUB OK:? "It looks drinkable."
   :GOTO PRO
NH 15999 IF TR<>D THEN GOTO PRO
SP 16000 TRAP 16000:POKE 752,A:FOR CH=C T
   O 14:IF PW1$(CH,CH)=" " THEN ? " ";:NE
   XT CH
NP 16001 ? CHR$(ASC(PW1$(CH,CH))+64);:NEX
   T CH:? " ";:INPUT PW$
TV 16002 FOR CH=C TO LEN(PW$):PW$(CH,CH)=
   CHR$(ASC(PW$(CH,CH))+10):NEXT CH
YY 16003 IF PW$="[_O]^*XOL_VK" THEN GRAPH
   ICS A:TRAP 40000:GOTO 16750
ZG 16004 POKE 752,C:GOTO 32020
OB 17000 IF LOC=950 OR (LOC=1250 AND  NOT
    BG) THEN POKE 712,RND(C)*250:VT=VT+5.
   6E-03
VM 17001 VT=VT+0.01:IF VT<10 THEN RETURN
GR 17005 ? "↑ ":? OK$(C,D);"▯Is anyone ou
   t there?":GOTO TW
JS 18000 ON INT(5*RND(C)+C) GOTO 18100,18
   200,18300,18400,18500
XH 18100 LOC=600:I99$=I10$:RETURN
TU 18200 LOC=1200:I99$=I19$:RETURN
AR 18300 LOC=650:I99$=I11$:RETURN
TG 18400 LOC=450:I99$=I7$:RETURN
RI 18500 LOC=350:I99$=I5$:RETURN
IS 29000 GRAPHICS A:POKE 16,112:POKE 5377
   4,112:POKE 710,A:POKE 752,C:POKE 709,A
   :POSITION 12,10:? "Initializing..."
ZY 29001 FOR I=A TO 15 STEP 0.3:POKE 709,
   I:NEXT I:FOR I=15 TO A STEP -0.3:POKE
   709,I:NEXT I
KK 29002 POKE 559,C
LG 30000 TRAP 30200:CLOSE #C:OPEN #C,4,A,
   "D:STRING2"
RJ 30005 INPUT #C,DLI$:INPUT #C,L$:L=ADR(
   L$):INPUT #C,F$:INPUT #C,C$:INPUT #C,S
   S$
EZ 30010 FOR I=1536 TO 1710:GET #C,Q:POKE
    I,Q:NEXT I:CLOSE #C
YY 30015 OPEN #C,4,A,"D:STRING1":AT=USR(1
   619,36864):CLOSE #C:GOTO 31450
CU 30200 GRAPHICS A:POKE 559,34:? :? "Fil
   e not found.":END
YR 31450 VV$="GETTAKDROGIVREACLIEATDRIEXA
   LOOENTKISHUGLIGRINCUTSAWDIGSHABUIGOT":
   SW$="NSEWDIHQXYZX"
SH 31455 WS$="NORSOUEASWESDOWINVHELQUIXXX
   YYYZZZ":PW1$="┐▗▝┐▗ ▘├┼┼┐▗ ┤"
OG 31460 NN$="SHEPAPWOOCOCPEADIAROCSKUSAW
   BAMBAGNAIMACBARTORVOLTREPLAPOLHUTGIRBO
   ULIOBELNATSTASTOFIRHOLMILSLEBOASHIYOU"
KS 31462 NN$(103,105)="HEL"
WU 31750 I0$="XXXXXX":I1$=I0$:I2$=I0$:I3$
   =I0$:I4$=I0$:I5$=I0$:I6$=I0$:I7$=I0$:I
   8$=I0$:I9$=I0$:I10$=I0$:I11$=I0$
EJ 31751 I12$=I0$:I13$=I0$:I14$=I0$:I15$=
   I0$:I16$=I0$:I17$=I0$:I18$=I0$:I19$=I0
   $:I20$=I0$:I21$=I0$:I22$=I0$:I25$=I0$
QL 31755 INV$=I0$:I1$(C,C)=" ▌":I2$(C,C)="
   ▗":I3$(C,C)="▲":I4$(C,C)="┼":I6$(C,C)=
   "●":I8$(C,C)="▄":I9$(C,C)="┬"
JQ 31760 I10$(C,C)="┴":I12$(C,C)="▌":I14$
   (C,C)="E":I15$(C,C)="/":I16$(C,C)="◣":
   I17$(C,C)="▀":I18$(C,C)="↑":I16I$=I0$
GK 31765 I19$(C,C)="┘":I20$(C,C)="▌":I21$
   (C,E)="▗▄":I22$(C,C)="┐":I16I$(C,C)="─
   ":I5$(C,C)=" ▀":I0$(C,C)="├"
IH 31770 I7$(C,C)="▗":OK$="┼┼┼┼ O.K.":WN
   $="●▗▄▟ What's next ?"
RF 31800 S1$="H●H─H\H●*▗▗▗H▗H▗R▗H▄":GRA
   PHICS E:POKE 559,A:POKE 712,4:POKE 710
   ,4:POKE 709,156
CY 31801 POKE 16,112:POKE 53774,112:POKE
   756,145:POSITION A,E:? #D;"@@@ skull i
   sland @@@":POKE 752,C:POKE 82,0:?
EJ 31802 POSITION 8,4:? #D;"BY":POSITION
   A,6:? #D;"%&% John  Patuto %&%":? " ●●
    Copyright 1989 ●▗▄▟ John Patuto ▌▌"
GR 31803 ? :? "          A.N.A.L.O.G. Comp
   uting":POKE 559,34:GOSUB 31804:POKE 82
   ,2:GOTO 31810
GJ 31804 FOR X=24 TO 65 STEP E:I=ASC(S5$(
   X,X)):FOR R=A TO 3:SOUND R,I+R,10,8:NE
   XT R
WH 31805 FOR I=C TO ASC(S5$(X+C,X+C))/20/
   (H1-4):NEXT I:SOUND E,A,A,A:NEXT X:SOU
   ND C,A,A,A:SOUND 3,A,A,A
XB 31806 SOUND A,A,A,A:RETURN
CS 31810 FOR CH=C TO 20:NEXT CH:OPEN #E,8
   ,A,"S:":POKE 16,112:POKE 53774,112:RET
   URN
GK 31900 IF  NOT FOOD AND TR>D THEN ? "Yo
   u're slowly growing weaker.":GOTO TW
VZ 31910 ON H1 GOSUB 3450,3475,4212,31930
   ,3475,5000,31804,10000,3333
ZO 31920 FOR I=C TO D:NEXT I:GOTO TW
EV 31930 FOR CH=11 TO 23:X=ASC(S5$(CH,CH)
   ):SOUND E,X,10,10:SOUND 3,X/E,10,8:FOR
    I=C TO CH/1.5:NEXT I:SOUND 3,A,A,A
JS 31932 NEXT CH:SOUND E,A,A,A:RETURN
ES 32009 IF  NOT WIN AND  NOT QU THEN GOS
   UB 3450
OT 32010 GOSUB 1800:? "Want to try again
   ?";
DQ 32011 GET #C,X:IF X=89 THEN RUN
WP 32012 IF X<>78 THEN 32011
ME 32020 ? "R":? :FOR CH=C TO E:IF  NOT W
   IN THEN ? "Q      Better Luck Next Time
   !!      Q":NEXT CH:GOTO 32022
KZ 32021 ? "Q & %     Congratulations!!!
    % & Q":NEXT CH:GOTO 32022
XH 32022 SOUND E,RND(A)*255,10,E:IF PEEK(
   53279)<>D THEN 32022
HM 32023 RUN
```

LISTING 2: BASIC

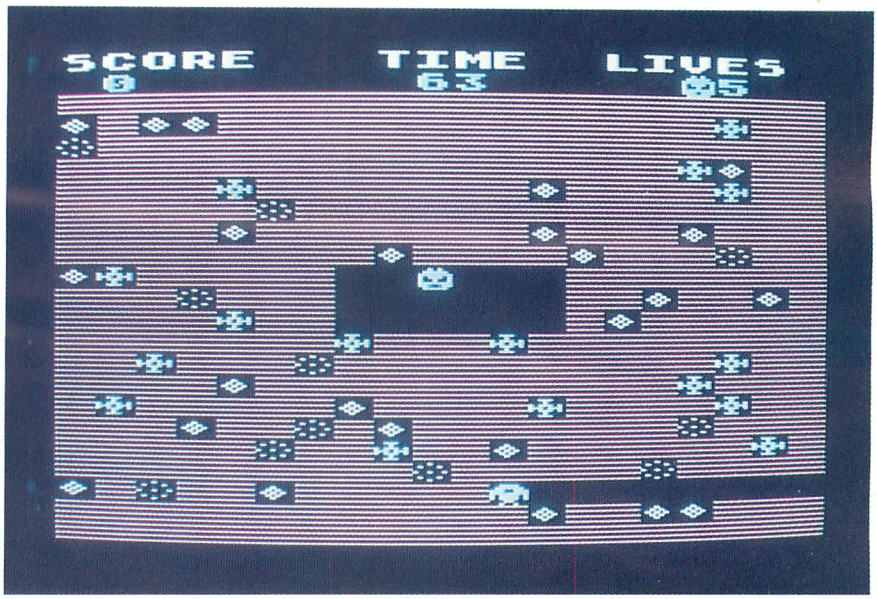
```
WN 0 REM ******************************
WO 1 REM ******************************
PI 2 REM ***  STRING GENERATOR FOR  ***
XJ 3 REM **        SKULL ISLAND       **
PW 4 REM *           (C) 1989          *
PS 5 REM **                            **
EW 6 REM ***      BY JOHN PATUTO      ***
WU 7 REM ******************************
WV 8 REM ******************************
NO 9 REM
IN 10 DIM A$(2):? "R":? :? :? "Place the
   disk to Create the string":? "files in
    Drive #1 and hit [RETURN]";:INPUT A$
IX 15 ? :? :? "One Moment...Creating File
   ...":?
KL 20 CLOSE #1:OPEN #1,8,0,"D:STRING1"
HP 30 FOR X=1 TO 1024:READ A:? #1;CHR$(A)
   ;:NEXT X:? #1:CLOSE #1:OPEN #1,8,0,"D:
   STRING2"
JI 35 FOR X=1 TO 64:READ A:? #1;CHR$(A);:
   NEXT X:? #1
HF 40 FOR X=1 TO 43:READ A:? #1;CHR$(A);:
   NEXT X:? #1
```




```
GQ 50 FOR X=1 TO 42:READ A:? #1;CHR$(A);:
   NEXT X:? #1
FM 60 FOR X=1 TO 31:READ A:? #1;CHR$(A);:
   NEXT X:? #1
KB 65 FOR X=1 TO 65:READ A:? #1;CHR$(A);:
   NEXT X:? #1
NM 67 FOR X=1 TO 175:READ A:? #1;CHR$(A);
   :NEXT X:? #1
DL 70 CLOSE #1:? :? :? "All Done...Now Ru
   n Skull Island...":END
SK 99 READ A:X=X+1:GOTO 99
YY 100 DATA 0,0,0,0,0,0,0,0,12,24,48,48,4
   8,0,48,0,0,102,102,102,0,0,0,0,51
YY 110 DATA 102,204,0,0,0,0,0,204,102,51,
   0,0,0,0,0,24,36,66,153,153,66,36,24,66
   ,195
YF 120 DATA 102,24,24,102,195,66,12,24,48
   ,0,0,0,0,0,28,56,112,112,112,56,28,0,5
   6,28,14
ZS 130 DATA 14,14,28,56,0,0,102,60,255,60
   ,102,0,0,0,24,24,126,24,24,0,0,0,0,0,0
DS 140 DATA 8,28,24,48,0,0,0,126,0,0,0,0,
   0,0,0,0,16,56,16,0,0,6,12,24,48
RJ 150 DATA 96,64,0,28,54,102,206,214,230
   ,124,0,48,240,48,48,24,24,254,0,248,20
   4,6,60,96,198
AB 160 DATA 126,0,124,108,6,63,3,99,126,0
   ,22,54,108,204,254,4,14,0,62,112,224,1
   24,6,198,252
JC 170 DATA 0,62,102,192,252,198,198,252,
   0,254,198,198,6,12,24,48,0,60,102,198,
   124,198,198,124,0
GS 180 DATA 124,198,198,126,6,140,120,0,1
   6,56,16,0,16,56,16,0,8,28,8,0,8,28,24,
   48,6
NN 190 DATA 12,24,48,24,12,6,0,0,0,126,0,
   0,126,0,0,96,48,24,12,24,48,96,0,254,1
   98
OF 200 DATA 12,24,24,0,24,0,126,255,153,2
   31,126,102,60,24,30,54,102,198,254,198
   ,198,0,30,54,102
VB 210 DATA 252,198,198,254,0,30,54,102,1
   92,192,198,254,0,240,216,204,198,198,1
   98,252,0,30,54,96,252
NF 220 DATA 192,198,254,0,30,54,96,252,19
   2,192,192,0,30,54,96,192,222,198,254,0
   ,102,102,198,198,254
PA 230 DATA 198,198,0,254,152,24,48,48,48
   ,254,0,254,24,12,6,198,198,252,0,50,10
   2,204,248,204,198
MD 240 DATA 198,0,24,48,96,192,198,198,25
   4,0,62,122,218,218,218,194,230,0,18,58
   ,122,218,218,218,222
AG 250 DATA 0,28,54,102,198,198,198,124,0
   ,30,54,102,254,192,192,192,0,28,54,102
   ,198,222,198,126,0
IH 260 DATA 30,54,102,254,216,204,198,0,3
   0,54,96,252,6,198,252,0,126,24,48,48,9
   6,96,96,0,38
TC 270 DATA 102,198,198,230,254,124,0,102
   ,102,198,198,198,108,56,0,206,134,134,
   182,182,188,248,0,198,198
GK 280 DATA 108,56,44,230,198,0,38,102,19
   8,254,6,198,252,0,248,140,6,62,96,192,
   254,0,0,30,24
ZU 290 DATA 24,24,24,30,0,0,64,96,48,24,1
   2,6,0,0,120,24,24,24,24,120,0,0,8,28,5
   4
NH 300 DATA 99,0,0,0,0,0,0,0,0,0,255,0,0,
   0,0,0,0,0,0,0,224,252,255,255
NZ 310 DATA 252,224,0,255,0,142,138,142,2
   34,0,255,255,0,238,170,206,170,0,255,0
   ,224,248,159,191,248
MO 320 DATA 224,0,0,0,0,248,248,24,24,24,
   3,7,14,28,56,112,224,192,192,224,112,5
   6,28,14,7
DV 330 DATA 3,1,3,7,15,31,63,127,255,0,0,
   128,192,224,240,252,255,128,192,224,24
   0,248,252,254,255
ZI 340 DATA 15,15,15,15,0,0,0,0,240,240,2
   40,240,0,0,0,0,255,0,236,170,170,234,0
   ,255,255
QG 350 DATA 0,238,68,68,78,0,255,0,0,6,7,
   15,31,63,255,0,28,28,119,119,8,28,0,7,
   8
UL 360 DATA 8,8,8,8,8,7,0,0,0,255,255,0,0
   ,0,0,7,63,255,255,63,7,0,102,153,56
PA 370 DATA 92,26,24,48,48,32,112,241,251
   ,255,255,255,255,255,0,234,74,74,78,0,
   255,224,16,16,16
QQ 380 DATA 16,16,16,224,255,0,200,174,17
   0,174,0,255,102,153,28,58,88,24,12,12,
   255,0,238,138,138
HA 390 DATA 238,0,255,120,96,120,96,126,2
   4,30,0,0,24,60,126,24,24,24,0,0,24,24,
   24,126,60
ON 400 DATA 24,0,0,24,48,126,48,24,0,0,0,
   24,12,126,12,24,0,0,255,0,230,134,96,2
   30,0
XL 410 DATA 255,0,0,56,108,204,140,246,0,
   192,192,220,246,230,198,124,0,0,0,28,5
   4,96,192,126,0
TQ 420 DATA 6,6,30,54,102,198,126,0,0,0,6
   0,102,252,128,254,0,28,54,96,192,248,1
   92,192,0,0
FW 430 DATA 0,60,102,198,126,6,254,192,19
   2,220,246,230,198,198,0,0,24,0,24,24,4
   8,96,0,0,6
FE 440 DATA 0,6,6,6,204,120,192,192,216,2
   40,240,216,206,0,12,24,48,96,96,96,96,
   0,0,0,60
UY 450 DATA 246,182,182,182,0,0,0,220,246
   ,230,198,198,0,0,0,28,54,102,198,124,0
   ,0,0,60,102
NI 460 DATA 198,252,192,192,0,0,120,204,1
   98,126,6,6,0,0,156,182,224,192,192,0,0
   ,0,60,96,252
JP 470 DATA 6,252,0,12,24,254,48,48,48,48
   ,0,0,0,102,102,230,198,124,0,0,0,198,1
   98,198,108
PL 480 DATA 56,0,0,0,198,214,126,108,68,0
   ,0,0,102,220,24,60,102,0,0,0,102,198,1
   98,126,12
XH 490 DATA 248,0,0,254,156,56,114,254,0,
   0,24,60,126,126,24,60,0,24,24,24,24,24
   ,24,24,24
CI 500 DATA 0,126,120,124,110,102,6,0,8,2
   4,56,120,56,24,8,0,16,24,28,30,28,24,1
   6,0
KO 510 DATA 104,104,141,1,2,104,141,0,2,1
   73,48,2,133,203,173,49,2,133,204,160,2
   4,169,130,145,203,169,0,141,243,2,96
BB 520 DATA 0,72,138,72,169,0,162,10,141,
   10,212,141,24,208,142,23,208,230,208
HP 525 DATA 165,208,41,16,74,74,74,141,1,
   212,104,170,104,64
IX 530 DATA 104,104,133,206,104,133,205,1
   04,133,204,104,133,203,169,0,168,133,2
   13,177,203,133,207
RG 540 DATA 104,104,168,136,48,10,165,207
   ,209,205,208,247,200,132,212,96,169,0,
   133,212,96
ZE 550 DATA 104,104,104,170,165,88,133,20
   3,165,89,133,204,216,24,202,48,15,165,
   203,105,40,133
JS 560 DATA 203,165,204,105,0,133,204,24,
   144,238,160,159,169,0,145,203,136,208,
   251,96
CT 570 DATA 80,80,60,52,46,52,60,60,60,60
   ,52,46,44,46,52,0,52,52,46,44,38,44,46
   ,0,46,46,44,46,52,60,60
CW 580 DATA 180,140,120,90,160,120,92,78,
   92,78,130,130,110,130,130,110,130,110,
   80,86,96
CC 590 DATA 96,108,185,200,215,200,165,25
   0,215,250,195,200,185,200,215,200,165,
   250,215,250,195,200
IZ 600 DATA 185,200,215,200,140,150,150,1
   50,140,70,150,70,170,250,185,200,215,2
   00,165,250,215,244
HQ 1440 DATA 72,169,100,141,10,212
UQ 1450 DATA 141,24,208,141,26,208
GR 1460 DATA 169,6,141,9,212,104
SN 1470 DATA 64,104,104,133,204,104
PX 1480 DATA 133,203,169,0,133,205
VH 1490 DATA 169,224,133,206,162,4
RE 1500 DATA 160,0,177,205,145,203
GL 1510 DATA 200,208,249,230,204,230
DE 1520 DATA 206,202,208,240,96,104
BB 1530 DATA 162,16,169,9,157,66
GL 1540 DATA 3,104,157,69,3,104
QB 1550 DATA 157,68,3,169,0,157
OU 1560 DATA 72,3,169,4,157,73
ON 1570 DATA 3,32,86,228,96,104
XA 1580 DATA 162,16,169,5,76,58
PP 1590 DATA 6,9,104,169,0,9,0,133
RJ 1600 DATA 212,169,0,133,213,96
DV 1601 DATA 72,138,72,152,72,169,0,162,0
   ,160,0
NP 1602 DATA 141,10,212,141,26,208
TF 1603 DATA 142,24,208,140,25,208
QB 1604 DATA 169,0,141,22,208,141,10,210,
   169,6,141,9,212,169,0,141,23,208,169,1
   56,141,0,2
OT 1605 DATA 104,168,104,170,104,64,72,16
   9,0,141,10,212,141,26,208,169,104,141,
   10,210,141,0,2,104,64
```

(continued from page 54)

```
CF 5005 FOR D=7 TO 12:POSITION 5,D:? #6;"
        ":NEXT D
RK 5010 POSITION 6,8:? #6;"game over"
AQ 5050 FOR L=20 TO 70:SOUND 0,L,10,10:SO
   UND 1,L,6,10:NEXT L
SO 5061 SOUND 0,0,0,0:SOUND 1,0,0,0:C=0
SH 5062 POSITION 6,8:? #6;"game over"
BH 5065 FOR W=1 TO 30:NEXT W:POSITION 6,8
   :? #6;"          "
AL 5070 FOR W=1 TO 30:NEXT W:C=C+1
CC 5075 POSITION 5,10:? #6;"press start"
FF 5076 IF C=28 THEN FOR I=0 TO 19:COLOR
   0:PLOT 0,I:DRAWTO 19,I:FOR D=1 TO 5:NE
   XT D:NEXT I:GOTO 3000
US 5080 IF PEEK(53279)=6 OR STRIG(0)=0 TH
   EN SN=0:GOTO 5
SN 6000 GOTO 5062
NG 6999 REM --MUSIC--
BM 7000 POSITION 9,1:? #6;"0":POSITION X,
   Y:? #6;"*":POKE 708,14
ZL 7001 SOUND 0,255,10,10:SOUND 1,254,10,
   10:FOR L=1 TO 50:POKE 710,RND(0)*89:NE
   XT L
LB 7002 RESTORE 7100
YH 7005 READ MUSIC
HN 7006 IF MUSIC=255 THEN GOTO 7120
VQ 7010 SOUND 0,MUSIC,10,10
KM 7020 SOUND 1,MUSIC-1,10,10
TH 7025 FOR G=1 TO 5:NEXT G:POKE 712,MUSI
   C:POKE 710,PEEK(53770)
SP 7050 GOTO 7005
BR 7100 DATA 121,96,96,121,91,91,81,81,81
   ,91,108,121,144,128,144,121,108,121,72
   ,64,72,91,81,53,47,45,53,33,35,40
KJ 7110 DATA 45,47,53,60,60,255
EW 7120 SOUND 0,0,0,0:SOUND 1,0,0,0:POKE
   712,0
UJ 7122 FOR I=0 TO 19:COLOR 0:PLOT I,0:DR
   AWTO I,22:NEXT I
ZL 7201 IF DIFF<110 THEN DIFF=120
YJ 7202 CTT=CTT+1
FR 7204 IF CTT=3 THEN GOTO 9000
PZ 7207 EG=EG+7:WB=WB+0.5:DIFF=DIFF-27
XR 7208 GOSUB 400
FV 7210 BR5=BR5+1:GRAPHICS 1+16:POKE 756,
   CH/256:SETCOLOR 2,RND(1)*14,9:SETCOLOR
    0,8,10:GOB=GOB+7:GOTO 7
BC 8999 REM --BONUS SCREEN--
ER 9000 GRAPHICS 17:POKE 710,14:POSITION
   5,5:? #6;"bonus round":CTT=0
MW 9001 POSITION 3,10:? #6;"YOU MUST GRAB
    5"
NI 9002 POSITION 5,12:? #6;"MAGIC STARS"
BO 9003 POSITION 4,14:? #6;"TO GET BONUS"
JP 9005 FOR D=1 TO 100:POKE 708,RND(0)*10
   +50:NEXT D
QH 9010 GRAPHICS 17:POKE 756,CH/256:POSIT
   ION 5,0:? #6;"bonus round":X=10:Y=10:P
   OKE 708,156
LW 9011 POSITION 0,22:? #6;"@@@@@@@@@@@@@
   @@@@@@@":TIME=0:T=10:IF SC>30000 THEN
   T=6:S1=4:S2=16
KI 9012 POSITION 0,3:? #6;"@@@@@@@@@@@@@@
   @@@@@@":IF SC>70000 THEN T=3
PT 9013 POSITION 5,23:? #6;"AVOID TRAPS"
GM 9014 FOR K=5 TO 21 STEP 3
DY 9015 POSITION 3,K:? #6;" -   -   -   -
   "
DQ 9017 NEXT K:SCR=PEEK(88)+256*PEEK(89):
   B5=0
SO 9050 TP=SCR+X+20*Y:ST=STICK(0):POKE 71
   0,14
HY 9051 IF X=S1 AND Y=S2 THEN GOTO 9700
AL 9052 IF SC>60000 AND RND(0)*9<5 THEN G
   OSUB 500
NI 9055 GOSUB 4100:IF X>18 THEN POKE SCR+
   X+20*Y,0:X=X-1
SU 9060 IF X<1 THEN POKE SCR+X+20*Y,0:X=X
   +1
XV 9061 IF Y<5 THEN POKE SCR+X+20*Y,0:Y=Y
   +1
OQ 9062 IF Y>20 THEN POKE SCR+X+20*Y,0:Y=
   Y-1
II 9063 SOUND 0,0,0,0
LL 9064 IF B5=5 THEN GOTO 9200
LU 9070 POKE TP,11:FOR D=1 TO 8:NEXT D:LO
   CATE X,Y,BC:POSITION 7,2:? #6;"COUNT "
   ;B5
TU 9075 IF BC=45 THEN GOTO 9700
RW 9080 POKE TP,0:POKE 710,PEEK(53770)
JP 9100 IF TIME=0 THEN T1=INT(RND(0)*15)+
   3:T2=INT(RND(0)*15)+4:POSITION T1,T2:?
    #6;"@"
KJ 9103 TIME=TIME+1:IF TIME>T THEN POSITI
   ON T1,T2:? #6;" ":FOR D=10 TO 0 STEP -
   1:SOUND 0,10,10,D:NEXT D:TIME=0
YB 9107 IF BC=163 THEN FOR D=1 TO 10:SOUN
   D 0,D+2,8,6:POKE 712,D*3:NEXT D:B5=B5+
   1:SOUND 0,0,0,0:POKE 712,0
TK 9150 GOTO 9050
II 9200 REM
KF 9240 POSITION 3,23:? #6;"CONGRATULATIO
   NS"
VN 9241 POSITION 5,2:? #6;"rANdoM BonUS "
UM 9245 FOR D=1 TO 40:SOUND 0,RND(0)*10,1
   0,10:SOUND 1,RND(0)*20,10,10:NEXT D:A=
   INT(RND(0)*10)*1000+1000:SC=SC+A
SP 9246 POSITION 5,2:? #6;"BONUS ";A;"
   ":SOUND 0,0,0,0:SOUND 1,0,0,0
ZK 9247 FOR D=1 TO 20:FOR F=15 TO 0 STEP
   -1:POKE 708,F:NEXT F:NEXT D
BH 9248 SN=SN+1:GOSUB 400:GRAPHICS 1+16:P
   OKE 756,CH/256:SETCOLOR 2,RND(1)*14,9:
   SETCOLOR 0,8,10:GOTO 7
UK 9250 GOTO 9250
LA 9700 POKE TP,0:POSITION 3,23:? #6;"SOR
   RY NO BONUS"
QT 9704 FOR JJ=30 TO 50
YH 9705 FOR D=-15 TO 15 STEP 3:SOUND 0,AB
   S(D)+JJ,10,10:POKE 708,D+50:NEXT D
OY 9706 NEXT JJ
DZ 9707 SOUND 0,20,6,10:POKE 708,255:FOR
   D=1 TO 40:NEXT D:SOUND 0,0,0,0
AC 9710 SN=SN+1:GOSUB 400:GRAPHICS 1+16:P
   OKE 756,CH/256:SETCOLOR 2,RND(1)*14,9:
   SETCOLOR 0,8,10:GOTO 7
VW 9999 GOTO 9050
```



TX CRUNCHER

INSIDE *this* ISSUE

MORE

BOOT CAMP

+

END USER

DATABASE DELPHI