COMPUTE!'s

ATARI Collection

VOLUME 1

30 original programs for the Atari

Games, applications, utilities, educational programs, and tutorials, all never-before-published, for every Atari home computer.

A **COMPUTE!** Books Publication

$12.95

COMPUTE!'s ATARI Collection

VOLUME 1

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

Printed in the United States of America

ISBN 0-942386-79-5

10 9 8 7 6 5 4 3 2 1

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is one of the ABC Publishing Companies and is not associated with any manufacturer of personal computers. Atari is a trademark of Atari, Inc.

Contents

Foreword

This latest addition to COMPUTE!'s library of books for Atari computer users includes more than 30 never-before-published articles and programs.

COMPUTE!'s Atari Collection, Volume 1 has something for every Atari owner. Whether you have a new Atari 800XL or the older 400, whether you're a beginning or experienced programmer, or just enjoy using your Atari, you'll find enough useful articles and programs to keep you in front of the keyboard for months.

If you enjoy games (and who doesn't?), you'll find "Nessie" filled with photographic fast action as you snap pictures of the elusive monster. If you want to try to outmaneuver and outwit your computer or a friend, "Reversi" and "Memory Match" fit the bill. Even youngsters can play—we've included "Pyramid Math," a two-player math contest, and "Alphabone Hunt" to entertain and educate children.

Do you need practical applications? They're here. "Investment Tracker" helps analyze your investments, and "Disk Catalog Utility" organizes your disk collection.

What if you just want to sit down and program? There are useful tips scattered throughout this book, as well as numerous articles that help you learn how to get more out of your Atari. Do you want to program sound? Then use the editors included with "Three Music Editors for Your Atari" to create notes, chords, or an entire song. Do you need a fast joystick routine in BASIC for your special game? You'll find what you need in "Two Fast and Simple Joystick Routines." And "An Introduction to Atari PEEKs and POKEs" shows you how to quickly and easily enhance your programs with these two important commands.

COMPUTE!'s Atari Collection, Volume 1 is packed with 30 original programs. They've never appeared anywhere else before. And because we've included "The Automatic Proofreader," program entry is virtually mistakeproof.

An entire year has gone by since COMPUTE! Publications released a new book just for Atari users. We're sure you'll agree the wait's been worth it.

Chapter 1

Getting Started

Chapter 1

An Introduction to Atari PEEKs and POKEs

Charles Brannon

If you're a beginning BASIC programmer, you may not realize that there's more to your Atari than BASIC. In fact, the Atari has power that BASIC just doesn't address. For example, you can redefine the character set, so that the letter A appears as an alien invader. Player/missile graphics lets you move and animate images independently of the graphics screen. Custom display lists and display list interrupts give a programmer complete control over the graphics screen, and the POKEY chip gives you more than just four-voice sound.

It's possible to use many of these features in BASIC, though some require machine language and most are beyond the beginning programmer. However, there are many powerful capabilities that anyone can put to good use right away—and since BASIC can't access them directly, you need two special BASIC keywords: PEEK and POKE.

Atari Memory Management

The 6502 microprocessor is the brain of your Atari. It can directly address any of the 65,536 memory locations. Some of this memory is RAM, the read/write memory used to hold data and programs from tape or disk; the rest of the memory is ROM (Read Only Memory) that holds BASIC and the operating system. When the power is turned off, RAM is erased, but the pattern in ROM is not dependent on power. You may already be familiar with ROM and RAM.

But there's also a third kind of memory which isn't really memory at all. Instead, it is a façade used by input/output chips. Input/output chips in the Atari include the GTIA, a graphics chip; ANTIC, which drives the GTIA to produce graphics modes and player/missile graphics; and POKEY, which reads the keyboard and drives serial input/output (used to talk to a disk drive or printer). These chips require information (such as what characters to put on the screen) and can produce information (such as which key is being pressed).

To make things easy, these chips pretend to be memory locations. POKEing to an I/O (input/ouput) memory address alters the action of a chip, and PEEKing will give you information from the chip.

From the point of view of the computer, these are memory locations. In fact, some behave like RAM. For example, you can POKE a number to a location, and then PEEK the location to get the number back.

Most I/O addresses are either read-only or write-only. Read is like ROM. Write-only memory locations can be changed, but you can't PEEK them to see what the current setting is.

Sometimes a given memory location is used for two functions. Writing (POKEing) to the location does one thing, while reading (PEEKing) does another. For example, the color locations in hardware will change the screen color if you POKE them, but PEEKing them returns a meaningless value. To get around this, the operating system (the master program that coordinates all other programs) keeps several *shadow registers* that can be PEEKed and POKEd. These locations are ordinary memory locations, but their values are copied to the hardware color registers every 1/60 second. Many hardware features are accessed through their corresponding shadow registers.

PEEK and POKE

The personality of the operating system is affected by many POKEs, and you can read useful information hidden to BASIC with PEEK.

POKE is used to change memory (although you can POKE to ROM, nothing changes). The format is POKE *address,data*. The *address* is a number from 0 to 65535. Each number accesses a memory cell, which can hold a number from 0 to 255.

A memory cell can be thought of as holding eight tiny switches. If you assign ascending powers of two to each switch, you can use these switches to represent numbers. This convention is called base 2, or binary. For advanced PEEKing and POKEing, an understanding of binary numbers can be most helpful. However, all you need to know to get started is that a memory location can only hold a number from 0 to 255.

PEEK is the converse of POKE, but it is a function rather than a command. PEEK returns a value, and any command that can use a value can use PEEK. For example, consider how PRINT can be used. You can PRINT 4, which displays a number; PRINT TOTAL, which displays the number held by the variable name TOTAL; or PRINT PEEK(53279), which dis-

plays the number held in location 53279. Note, too, that PEEK can be used with POKE (for example, POKE 106,PEEK (106)−8).

Keyboard Control

In BASIC, you can use GET to read a character from the keyboard. But GET always waits for a keystroke. Say you want to periodically check for keyboard input. If no key is pressed, your program continues. But since GET always waits for a keystroke, it will freeze your program until a key is pressed.

Instead of using GET, you can check location 764 with PEEK(764). When it returns a 255, no key has been pressed and you can continue your program loop. Whenever it doesn't return 255, you can use GET to read the ATASCII value of the keystroke.

Another problem with GET is that you must first OPEN a file to the keyboard device. If you only want to wait for a keystroke, you can use something like this:

```
10 PRINT "PRESS ANY KEY TO CONTINUE."
20 IF PEEK(764)=255 THEN 20
30 POKE 764,255
```

The value returned by location 764 is not in ATASCII, the convention used by ASC and CHR$. The number is an internal representation of the key, expressed in terms of what row and column the key is in. Run this small program to see what values keystrokes return. When you press a key, the internal value is shown:

```
10 PRINT PEEK(764):GOTO 10
```

Notice that until you press a key, the value is 255. That's why you can wait for a keystroke by checking until the location no longer holds 255. Also note that when CTRL is held down, the value is greater than 127. If either SHIFT key is used, the value is greater than 63, but less than 128.

This location is used as a one-key buffer. Even if a program is not accepting keyboard input, this location still changes when you press a key. If the program then tries to GET or INPUT from the keyboard, location 764 will provide the keystroke you pressed earlier, even if you are no longer pressing the key.

POKE 764,255 clears any value out of the keyboard buffer. You can even POKE 764 with other values, and watch how these values cause the computer to type out a character automatically.

When you ran the program, you may have noticed that 764 cannot tell if you are holding down a key. Once you press a key, the value in 764 changes and remains changed until you press another key. It will *not* return to 255 when you let go of the key. Sometimes, however, you may want to see if a key is being held down. This small program simulates the action of an organ. When you press a key, a tone sounds as long as you hold down the key and stops when you let go:

```
10 IF PEEK(53775)=255 THEN SOUND 0,0,0,0:GO
   TO 10
20 SOUND 0,100,10,8:GOTO 10
```

Location 53775 holds 255 if no key is being pressed, 251 if a key is down, and 247 if the SHIFT key is being held down.

Controlling the Inverse Key

If you are GETting or INPUTting from the keyboard, you may not want the user to enter inverse or lowercase characters. Since the INVERSE key is easily struck by mistake on the 400/800 models, you need a way to force inverse video off. Alternately, you may want to "press" the INVERSE key for the user, so that everything he types comes out in reverse. Location 694 controls this. POKE it with a 128 to force inverse characters, and 0 to disable inverse. It's a temporary thing, though. If the user hits the INVERSE key again, 694 changes. This location will not affect how text is printed, only how it's received from the keyboard.

A similar location, 702, stores the status of the CAPS/LOWR key. If CAPS lock is on, 702 holds 64, but it holds 0 if the keyboard is in lowercase mode. Location 702 will hold 128 if the keyboard is in CTRL-mode (same as CTRL–CAPS/LOWR). You can use POKE to force the keyboard into the desired mode under program control. Try this:

```
10 DIM A$(10)
20 PRINT "ENTER","0 FOR LOWERCASE":PRINT ,"
   64 FOR UPPERCASE":PRINT ,"128 FOR CONTRO
   L MODE"
```

```
30 INPUT X:IF X<>0 AND X<>64 AND X<>128 THE
   N 30
40 POKE 702,X
50 INPUT A$
60 PRINT A$:PRINT :PRINT
70 GOTO 20
```

Consoling Information

The console keys START, OPTION, and SELECT cannot be read as other keyboard keys can. Whatever you use them for is up to you, but you can easily read them by checking location 53279. Here are the values returned:

7 No console keys held down
6 START key alone
5 SELECT key alone
3 OPTION key alone
0 START, SELECT, and OPTION pressed simultaneously
1 OPTION and SELECT together
2 OPTION and START together
4 SELECT and START together

This simple program demonstrates how that can be used:

```
10 A=PEEK(53279)
30 ON A+1 GOSUB 50,60,70,80,90,100,110
40 GOTO 10
50 PRINT "START+SELECT+OPTION":RETURN
60 PRINT "OPTION+SELECT":RETURN
70 PRINT "OPTION+START":RETURN
80 PRINT "OPTION":RETURN
90 PRINT "SELECT+START":RETURN
100 PRINT "SELECT":RETURN
110 PRINT "START":RETURN
```

Notice that when you press a console key, that key continues to return its value as long as you hold it down, but that it returns to normal (7) when you let go. This rapid-fire repeat is often undesirable. To remove it, add this line to the program:

```
20 IF PEEK(53279)=A THEN 20
```

This waits until you let go of the selected key to print the message on the screen, so you get only one message each time the key is pressed. Nothing will happen until you let go of the key.

Remember how some locations have different functions when read or written? Location 53279 is one of them. When read, it tells you what console keys are being held down. But if you POKE it with a zero, the internal speaker (or the external speaker on XL Ataris) makes a tiny click. Zero makes the speaker cone move out, but the operating system puts an eight (which moves the cone back in) into 53279 every 1/60 second. Rapidly POKEing this location with zeros creates a buzzing noise. Notice how those two functions tied to this location have nothing in common.

While we're on the subject of keyboard POKEs and PEEKs, here's how to disable the BREAK key. You might want to do this to prevent anyone from exiting and listing your BASIC program, or you may want to protect the user from accidentally breaking out of a program. Just use these two POKE statements:

POKE 16,64:POKE 53774,64

You can reenable the BREAK key by changing graphics modes or by pressing SYSTEM RESET. If you don't want the BREAK key reenabled, you must repeat these POKEs after every GRAPHICS command or any OPEN statement.

Although SYSTEM RESET cannot be disabled, you can make someone wish they hadn't pressed it. If you POKE 580 with a value other than 0, the SYSTEM RESET will act as if you turned the power off and on. This is called a cold start, as opposed to the warm start normally performed by this key. POKE 580,0 to reenable warm start.

Screen Play

Although it's easier to use SETCOLOR, you can also POKE directly into the color registers to set colors. POKEing can be faster and more compact, since there is only one number to evaluate instead of four. Locations 708–712 correspond directly to SETCOLOR 0 through SETCOLOR 4. Each location holds both the color and luminance. Just multiply the color number (0–15) by 16 and add in the luminance (0–15). SETCOLOR a,b,c corresponds to POKE 708+a,b*16+c. For example, POKE 712,10 changes the border color to white.

Location 559 can, among other things, turn the screen display on and off. POKE 559,34 is the normal setting. If you POKE 559,0 the screen blanks to the border color.

How can you use this? To speed up programs. Since it takes some time to display the screen, the Atari can run up to 30 percent faster with the screen turned off. You can blank the screen when you perform a long calculation, as long as you warn the user so that he or she doesn't panic when the screen blanks out. You may also want to blank the screen while you are drawing a complex image, then turn the screen back on to make your graphics instantly appear.

You may have heard of locations 82 and 83. These locations are primarily used to let you adjust the width of the screen, since some televisions cannot display the full width of the screen. Location 82 controls the left margin. PEEK will return the current setting, and POKE resets it. The left margin is the number of blank spaces from the edge of the screen. If you want a full 40-column screen, use POKE 82,0.

The right margin, set by location 83, is a number from 0 to 39 and represents the number of spaces from the left side of the screen (not from the left margin). After you change the margins, *subsequent* PRINT statements will conform to the new margin settings. Do not make the left margin greater than the right margin. Why not? Try it and find out! Also beware that if you make the width of the screen too small, you cannot type any commands. In any case, SYSTEM RESET restores the left margin to 2 and the right margin to 39.

Curse That Cursor!

The cursor can be a pesky critter, since it remains on all the time, showing the current PRINT position. It's easy enough to disable it, though—just POKE 752,1. A zero in 752 enables the cursor. After you POKE this location, the cursor will not change until the next PRINT statement moves it, or after you clear the screen. Any change in graphics modes will restore the cursor. SYSTEM RESET also turns the cursor back on.

You can also control how inverse characters appear. A two in location 755 is the normal state. All the dots making up the character will reverse their color. POKE it with a zero, and all inverse characters will not be inverted, but will appear as normal characters. Put a one here, and inverse characters will be invisible. A three makes all inverse characters appear as inverse spaces (opaque). Add four to any of these values, and all text will appear upside-down and mirror-reversed. (This feature was originally used in videogames that projected the

screen onto a mirror.) Since the cursor is just the inverse of whatever character it is sitting on, 755 also affects the appearance of the cursor. Try this short program to see how you can use 755 to make flashing text:

```
100 PRINT "INVERSE CHARACTERS CAN BLINK"
110 POKE 755,2-PEEK(755):FOR W=1 TO 50:NEXT
    W:GOTO 110
```

You can easily read the position of the cursor by checking locations 84 and 85. Location 84 holds the current row (the vertical position of the cursor) and ranges from 0 to 23. The current column, 0–39, is in location 85. You can use POSITION in BASIC to directly move the cursor to an X,Y location, but with POKE you can change the row or column separately. When you change 84 or 85, the cursor does not actually move until a PRINT statement is used.

POKE 85 is the replacement for Atari's missing TAB statement. It makes formatted displays easy. For example, the line

```
Z=Z+6:PRINT TAB(20-19*SIN(Z));CHR$(42):GOTO 10
```

prints a sine wave in Microsoft BASIC. It's easily translated to Atari BASIC:

```
10 Z=Z+6:POKE 85,20-19*SIN(Z):PRINT CHR$(42
   ):GOTO 10
```

In a graphics mode, locations 84 and 85 control the position of the graphics cursor, not the text cursor. The text cursor is set in these modes by location 656 (row) and 657 (column).

You can also change location 201. It holds the number of spaces between comma zones. When you print a list of variables (such as PRINT A,B,C$) each item is tabbed over into a separate zone ten spaces wide. If what you are printing overflows into the next zone, the following item will have to go into the zone after that. You can change the width of the comma zone by POKEing 201. Do not ever put a 0 in this location, or the computer will freeze up when it encounters the comma, forcing you to press SYSTEM RESET or turn off the power to regain control. You may want to change it back to 10 when you are through, or other programs using PRINT may tab strangely.

The Sound of Silence

When you read or write to tape or disk, the speaker beeps and warbles in conjunction with bytes being sent out or pulled in from disk. While this can be a good diagnostic aid (some people can hear the difference between reading and writing, and can tell right away if there's been a read error), it can get on your nerves. Additionally, if you have recorded an audio track to play while the program loads, the beeps can get in the way. POKE 65,0 disables the sound, although you can barely hear it if you turn the volume up. This does not disable the sound made by keystrokes, and has no effect on the SOUND command. Any nonzero value will reenable the input/output sounds.

Special Atari XL POKEs

The 1200XL, the new 600XL and 800XL, and the promised 1450XLD all use the new XL operating system. The new operating system represents a considerable increase in power and flexibility. What this means is that there are more juicy POKEs to try. Remember that none of these POKEs will work with the older 400/800 computers, so if you are writing programs for publication or sharing, keep this in mind.

The most astounding POKE enables fine scrolling in GRAPHICS 0. Just enter POKE 622,255:GRAPHICS 0. If you want a convenient way to watch the scrolling, just enter FOR I=1 TO 1000:? I:NEXT I.

Unlike normal scrolling, which moves the screen text up a full line at a time, fine scrolling moves the characters pixel by pixel. This fine scrolling can adversely affect some programs, so to turn it off, enter POKE 622,0:GRAPHICS 0. Of course, the scrolling works only with GRAPHICS 0.

The 1200XL has additional function keys to control keyboard click, keyboard enable, screen blanking, and the international character set. If you own a 600XL or 800XL, you may not even be aware of these features. First, try POKE 756,204. No immediate changes. Now hold down CTRL and type some letters of the alphabet. Instead of the graphics characters, these keys now produce all kinds of special foreign language symbols. Enter POKE 756,224 and the character set will return to normal. Now you can write multilingual programs without having to redefine the character set.

To disable the keyboard, POKE 621,255. Use POKE 621,0 to reenable it. Don't try this POKE from the keyboard, or you won't be able to type the POKE that restores the keyboard. In any case, SYSTEM RESET will get you out of this mode. It's best to do this POKE under program control. It's useful when you don't want the user to type keys that may interfere with your program.

The XL Atari models all have a HELP key. Although not used by the operating system or BASIC, you can read this key in your own programs, and act on it. Once HELP is pressed, location 732 holds a 17. It will continue to hold 17 until you POKE 732,0. You should check to see if location 732 holds a nonzero value, then POKE 732,0 once you've acted on the key. If SHIFT is held down with HELP, 732 will return a value of 81. A value of 145 indicates that CTRL is used with HELP.

Every time you press a key, the internal speaker (on the 400/800) or the external speaker (XL Ataris) makes a soft blip. This positive audio feedback aids in touch-typing, but some find it annoying. There's no easy way to disable this beep on the 400/800 without cutting the wire to the internal speaker, but you can disable it on XL Ataris. Just POKE 731,255. A value of 0 allows the keyclick to be heard. You can also change the rate at which keys repeat. There are two factors in repeating keys. When you press a key, you don't want it to repeat instantly. Instead, the operating system waits for 4/5 second before it starts the repeating. Once the repeating starts, the other time factor is how quickly the key is repeated. This defaults to about 10 repeats per second (or 1/10-second delay between repeats). In the operating system, these time delays are expressed in multiples of 1/60 second. A value of 60 is a full second, 30 is a half-second, and so on. To change the delay before the key begins to repeat, POKE 729. Location 730 specifies the delay between the key repeats. The default values for 729 and 730 are 48 and 6 respectively.

The power offered by the Atari computers continues to challenge even the most advanced programmers. The locations covered here give a BASIC programmer additional capability, but there's much, much more. Read the other articles in this book and study the PEEKs and POKEs used in the programs for more ideas. If your curiosity is irresistibly piqued, check into COMPUTE! Books' *Mapping The Atari*, a comprehensive guide to memory.

Chapter 1

PEEKing and POKEing Around

Sheila Neece Spencer

This well-designed program will make it easy for you to look into your Atari's memory. You'll also be able to change memory, load ML programs, and even convert hex, decimal, and binary numbers.

As I pored over my Atari manuals one day, it occurred to me how helpful it would be to look at the contents of memory locations in their binary configurations. That would let me see which bits were set and which were not.

One thing led to another, and the result was the program given here. Not only does it let you look at memory locations in hex, decimal, and binary, but it also lets you POKE addresses with binary numbers; convert hex, decimal, and binary numbers; and enter and run a machine language program in hex or decimal—all without leaving the comfort of this one program.

The contents of any address in memory are made up of one byte (eight binary digits). A binary number consists of 1's and 0's only; a 1 indicates that a bit is turned on, while a 0 indicates that it is turned off. The bits are numbered from 0 to 7 from right to left.

Binary is the only language your Atari can actually understand. When you insert a language cartridge into your machine, you are actually providing your Atari with an interpreter which allows you to communicate with it (via a language such as BASIC) in some meaningful and useful way.

Nine Options

Now to the program itself. Type it in, then save a copy *before* you use it.

When you run the program, you'll get a menu with nine options. Option 1 allows you to examine any memory location and see its contents in hex, decimal, and binary.

Option 2 lets you change memory by entering a binary number. I chose to use binary here in order to get the feeling of actually setting bits in the address. I think "bits 0 and 1 set" is a little easier to visualize than "POKE x,3".

Be careful when using option 2. If you POKE the wrong

number into the wrong location, you run the risk of crashing your system. Then you'll have to turn the computer off and on again to regain control.

Some interesting places to make changes are locations 53760–53768 (the sound registers), 53266–53274 (the player/missile graphics color registers), and 53248–53255 (the player/missile graphics position registers). You may see some strange things on your screen when you play around with these registers, but pressing SYSTEM RESET will generally get you out of whatever mess you've gotten into.

Options 3–8 are conversion routines. Note that if you are entering a hex number to be converted, you must *always* enter two digits, even if the first one is a 0. Otherwise, you will get an erroneous answer.

Option 9 lets you put a machine language program into memory. You will be asked to choose between hex and decimal input and to specify a starting address (1536 is usually a good starting point for short machine language programs). You will then input the instructions one by one. Once again, be very careful as you type in the instructions. One wrong digit can crash the system.

When you have entered the last instruction, hit RETURN. The program will prompt you to be sure the RETURN was not an error by asking "Is that all?" If you reply with N, it will return you to the routine and allow you to continue inputting instructions. However, Y will prompt the program to ask you if you wish to run the program you have just entered. At that point, N will return you to the menu.

Understanding PEEK and POKE

For error-free program entry, read "The Automatic Proofreader" in this chapter before typing in this program.

```
CH 5  DIM A$(2),AD$(1),B$(4),C$(5),RESP$(1),BIN
      $(8),MODE$(3),INST$(2),TD(2),N$(8),TITLE$
      (20),BYLINE$(12),NAME$(14)
MC 6  MENU=600:CLICK=6000:B$="0000"
DF 7  GOTO 5000
AN 39 REM DECIMAL TO HEXADEMICAL CONVERSION SU
      BROUTINE
NN 40 N=PEEK(ADDRESS):N1=PEEK(ADDRESS)
BO 60 I=2
PI 70 TEMP=N:N=INT(N/16):TEMP=TEMP-N*16:IF TEM
      P<10 THEN A$(I,I)=STR$(TEMP):GOTO 90
```

```
AH 80  A$(I,I)=CHR$(TEMP-10+ASC("A"))
CG 90  IF N<>0 THEN I=I-1:GOTO 70
AC 91  IF M=3 OR M=8 THEN ? "HEX=";A$(I,2):A$="
       {,}":B$="{4 SPACES}":C$="{5 SPACES}":RET
       URN
ML 95  IF M>4 THEN RETURN
AI 100 REM HEXADECIMAL TO BINARY CONVERSION SU
       BROUTINE
FC 110 IF A$(1,1)="{,}" OR A$(1,1)="0" THEN B$
       ="0000"
OP 120 IF A$(1,1)="1" THEN B$="0001"
PB 130 IF A$(1,1)="2" THEN B$="0010"
PE 140 IF A$(1,1)="3" THEN B$="0011"
PF 150 IF A$(1,1)="4" THEN B$="0100"
PI 160 IF A$(1,1)="5" THEN B$="0101"
PK 170 IF A$(1,1)="6" THEN B$="0110"
PN 180 IF A$(1,1)="7" THEN B$="0111"
PN 190 IF A$(1,1)="8" THEN B$="1000"
PH 200 IF A$(1,1)="9" THEN B$="1001"
AA 210 IF A$(1,1)="A" THEN B$="1010"
AD 220 IF A$(1,1)="B" THEN B$="1011"
AE 230 IF A$(1,1)="C" THEN B$="1100"
AH 240 IF A$(1,1)="D" THEN B$="1101"
AJ 250 IF A$(1,1)="E" THEN B$="1110"
AM 260 IF A$(1,1)="F" THEN B$="1111"
PG 270 IF A$(2,2)="0" THEN C$=" 0000"
PJ 280 IF A$(2,2)="1" THEN C$=" 0001"
PL 290 IF A$(2,2)="2" THEN C$=" 0010"
PF 300 IF A$(2,2)="3" THEN C$=" 0011"
PG 310 IF A$(2,2)="4" THEN C$=" 0100"
PJ 320 IF A$(2,2)="5" THEN C$=" 0101"
PL 330 IF A$(2,2)="6" THEN C$=" 0110"
PO 340 IF A$(2,2)="7" THEN C$=" 0111"
PO 350 IF A$(2,2)="8" THEN C$=" 1000"
AB 360 IF A$(2,2)="9" THEN C$=" 1001"
AK 370 IF A$(2,2)="A" THEN C$=" 1010"
AN 380 IF A$(2,2)="B" THEN C$=" 1011"
AO 390 IF A$(2,2)="C" THEN C$=" 1100"
AI 400 IF A$(2,2)="D" THEN C$=" 1101"
AK 410 IF A$(2,2)="E" THEN C$=" 1110"
AN 420 IF A$(2,2)="F" THEN C$=" 1111"
LK 421 IF M=4 OR M=7 THEN ? "BINARY=";B$;C$:A$
       ="{,}":B$="{4 SPACES}":C$="{5 SPACES}":
       RETURN
FI 422 ? "{4 SPACES}HEX: ";A$(I,2):? "DECIMAL:
        ";N1:? " BINARY: ";B$;C$:? "PEEK(";ADD
       RESS;")=";PEEK(ADDRESS)
LB 440 IF M=1 OR M=2 THEN A$="{,}":B$="
       {4 SPACES}":C$="{5 SPACES}":RETURN
```

```
AE 450  IF M=3 THEN ? "HEX=";A$(I,2):? :A$="
        {,}":RETURN
DL 460  IF M=4 THEN ? "BINARY=";B$;C$:B$="
        {4 SPACES}":C$="{5 SPACES}":RETURN
PH 499  REM BINARY TO DECIMAL CONVERSION SUBROU
        TINE
AH 500  TRAP MENU:B7=VAL(BIN$(1,1)):B6=VAL(BIN$
        (2,2)):B5=VAL(BIN$(3,3)):B4=VAL(BIN$(4,
        4))
MA 505  IF BIN$="00000000" THEN PKR=0:GOTO 526
HC 510  B3=VAL(BIN$(5,5)):B2=VAL(BIN$(6,6)):B1=
        VAL(BIN$(7,7)):B0=VAL(BIN$(8,8))
JO 520  PKR=INT(B7*2^7+B6*2^6+B5*2^5+B4*2^4+B3*
        2^3+B2*2^2+B1*2^1+B0*2^0)+1
NH 525  IF B7 AND  NOT B6 AND  NOT B5 AND  NOT
        B4 AND  NOT B3 AND  NOT B2 AND  NOT B1
        AND  NOT B0 THEN PKR=PKR-1
AM 526  IF M=6 THEN ? "DECIMAL=";PKR:RETURN
PO 527  IF M=8 THEN RETURN
AA 530  POKE ADDRESS,PKR:RETURN
MA 599  REM MENU
NL 600  ? "{CLEAR}":POKE 752,1:POSITION 16,4:?
        "  MENU":?
GG 610  ? :? :? "1 LOOK AT CONTENTS OF MEMORY":
        ? "2 CHANGE CONTENTS OF MEMORY":? "3 CO
        NVERT A DECIMAL NUMBER TO HEX"
DO 615  ? "4 CONVERT A DECIMAL NUMBER TO BINARY
        ":? "5 CONVERT A HEX NUMBER TO DECIMAL"
        :? "6 CONVERT A BINARY ";
FE 616  ? "NUMBER TO DECIMAL":? "7 CONVERT A HE
        X NUMBER TO BINARY":? "8 CONVERT A BINA
        RY NUMBER TO HEX"
CO 617  ? "9 ENTER A SERIES OF POKES":? "  IN S
        UCCEEDING MEMORY LOCATIONS"
PG 620  TRAP MENU:INPUT M:? "{CLEAR}":ON M GOTO
          1000,1050,1100,1100,1200,1300,1400,150
        0,1600
EJ 699  REM HEXADEMICAL TO DECIMAL CONVERSION S
        UBROUTINE
BC 700  FOR Q=1 TO 2
CP 701  IF INST$(Q,Q)="0" THEN TD(Q)=0
DF 705  IF INST$(Q,Q)="1" THEN TD(Q)=1
DD 710  IF INST$(Q,Q)="2" THEN TD(Q)=2
DK 715  IF INST$(Q,Q)="3" THEN TD(Q)=3
DI 720  IF INST$(Q,Q)="4" THEN TD(Q)=4
DP 725  IF INST$(Q,Q)="5" THEN TD(Q)=5
DN 730  IF INST$(Q,Q)="6" THEN TD(Q)=6
EE 735  IF INST$(Q,Q)="7" THEN TD(Q)=7
EC 740  IF INST$(Q,Q)="8" THEN TD(Q)=8
```

```
EJ 745 IF INST$(Q,Q)="9" THEN TD(Q)=9
HF 750 IF INST$(Q,Q)="A" THEN TD(Q)=10
HM 755 IF INST$(Q,Q)="B" THEN TD(Q)=11
HP 756 IF INST$(Q,Q)="C" THEN TD(Q)=12
IC 757 IF INST$(Q,Q)="D" THEN TD(Q)=13
IF 758 IF INST$(Q,Q)="E" THEN TD(Q)=14
II 759 IF INST$(Q,Q)="F" THEN TD(Q)=15
LC 760 NEXT Q:INST=TD(1)*16+TD(2):IF M=5 THEN
       ? "DECIMAL=";INST
HO 770 RETURN
KK 999 REM "1" LOOK AT CONTENTS OF MEMORY
GF 1000 ? :? :? "What is the address you want
        to see?":INPUT ADDRESS:GOSUB 40:GOTO 1
        000
PA 1040 TRAP 40000
KK 1049 REM "2" CHANGE CONTENTS OF MEMORY
OD 1050 ? :? :? "Please enter the address you
        wish to  poke data into";:INPUT ADDRES
        S
FM 1060 ? :? "Now enter the binary configurati
        on youwant in the register (bits are n
        um-{3 SPACES}bered from 7 to 0 left ";
EE 1070 ? "to right).":INPUT BIN$:GOSUB 500:?
        :? "PEEK(";ADDRESS;")=";PEEK(ADDRESS):
        ? :? :GOTO 1050
DC 1099 REM "3" AND "4" CONVERT A DECIMAL # TO
         HEX or CONVERT A DECIMAL # TO BINARY
IL 1100 ? :? :? "Enter the number to convert";
        :INPUT N:GOSUB 60:GOTO 1100
CN 1199 REM "5" CONVERT A HEX # TO DECIMAL
PH 1200 ? :? :? "Enter the number to convert (
        enter 2  digits)";:INPUT INST$:GOSUB 7
        00:GOTO 1200
AP 1299 REM "6" CONVERT A BINARY # TO DECIMAL
GN 1300 ? :? :? "Enter the number to convert";
        :INPUT BIN$:GOSUB 500:GOTO 1300
AH 1399 REM "7" CONVERT A HEX # TO BINARY
PJ 1400 ? :? :? "Enter the number to convert (
        enter 2  digits)";:INPUT A$:GOSUB 110:
        GOTO 1400
AJ 1499 REM "8" CONVERT A BINARY # TO HEX
ED 1500 ? :? :? "Enter the number to convert";
        :INPUT BIN$:GOSUB 500:N=PKR:GOSUB 60:G
        OTO 1500
NA 1599 REM "9" ENTER A SERIES OF POKES IN SUC
        CEEDING MEMORY LOCATIONS
LL 1600 ? :? :? "Will you be inputting instruc
        tions in (D)ecimal or (H)exadecimal";:
        INPUT MODE$
```

```
FA 1610 ? :? "What is the starting address (in
         deci-mal, please)";:INPUT ADDRESS:ADD
        RESS1=ADDRESS
ML 1620 IF MODE$(1,1)="H" THEN HEX=1
ML 1625 IF MODE$(1,1)="D" THEN HEX=0
EH 1630 ? :? "Now enter the instructions, one
        by{4 SPACES}one."
CK 1640 IF HEX THEN TRAP 1700:INPUT INST$:GOSU
        B 700:GOTO 1660
ND 1650 TRAP 1700:INPUT INST
PD 1660 POKE ADDRESS,INST:ADDRESS=ADDRESS+1:IF
         HEX THEN 1640
NE 1680 GOTO 1650
OE 1700 ? :? "Is that all";:INPUT RESP$:IF RES
        P$(1,1)<>"Y" THEN 1640
OO 1720 ? :? "Do you want to run the program y
        ou{4 SPACES}have just entered";:INPUT
        RESP$:IF RESP$(1,1)="Y" THEN 1750
DK 1740 GOTO MENU
IK 1750 TRAP MENU:X=USR(ADDRESS1)
FF 4999 REM OPENING TITLE
CM 5000 ? "{CLEAR}":POKE 752,1:TITLE$="FUN WIT
        H PEEK & POKE":BYLINE$="presented by":
        NAME$="COMPUTE! BOOKS"
OK 5370 LN=INT(LEN(TITLE$)/2):FOR Z=1 TO LN:PO
        SITION 17-Z,7:? "*";TITLE$(LN-Z+1,LN+Z
        );"*":GOSUB CLICK
IP 5400 NEXT Z:LN=INT(LEN(BYLINE$)/2):FOR Z=1
        TO LN:POSITION 17-Z,9:? "**";BYLINE$(L
        N-Z+1,LN+Z);"**":GOSUB CLICK
EL 5440 NEXT Z:LN=INT(LEN(NAME$)/2):FOR Z=1 TO
         LN:POSITION 17-Z,11:? "**";NAME$(LN-Z
        +1,LN+Z);"**":GOSUB CLICK:NEXT Z
MM 5500 FOR DLAY=1 TO 500:NEXT DLAY:GOTO MENU
KI 6000 FOR TICK=0 TO 3:POKE 53279,0:NEXT TICK
        :RETURN
```

Chapter 1

Two Fast and Simple Joystick Routines

Stephen Levy

These routines will make it easy for you to incorporate responsive joystick control into your programs.

You finally feel you know enough Atari BASIC to write your own game. You have some great ideas for games which use joysticks, and you've seen some joystick routines in other programs. But every time you try to duplicate a method, the routine seems ridiculously slow.

Most joystick routines written by beginning programmers contain numerous IF-THEN statements. It's those IF-THEN statements, as well as the actual placement of the routine, that make the joystick response seem unbearably slow.

Put It Up Front

Beginners often place joystick routines at high line numbers (for instance, 10000) and use statements such as GOSUB 10000 when the joysticks need to be read. It works, but it needlessly slows the joystick routine operation.

A much better approach is to place the routine at the beginning of the program. Joystick routines located near the start of any BASIC program will always run faster than the same routines placed later in the program.

The reason for this is simple. In order to carry out a statement like GOSUB 10000, the computer must start at the beginning of the program and check every line number in order until it finds line 10000. If you place the same routine at line 10 instead of 10000, the computer can find your routine sooner, without first checking dozens or hundreds of intervening line numbers.

The two routines described here are located at low line numbers. Thus, the program to which they are appended will have little or no effect on the speed at which the routines read the sticks.

Using Arrays

Program 1 places all the information needed to respond to joystick input into two arrays. By placing the information in arrays, the program will always have the information in

memory before it is needed. That means that the computer does not have to figure out what to do each time it reads the joystick.

What has been done is to place a 0, 1, or −1 into each element of the arrays. X=STICK(0) will give X a value based on the position of the joystick in the first port. If the joystick is in the center position, X will be equal to 15. When the arrays were created, the fifteenth element for both row and column arrays was given a value of zero (COL(15)=0; ROW(15)=0). Therefore, when COL(15) is added to the current column position, there will be no change—just as there will be no change in the row position (line 20). Similarly, if the joystick is pushed up, the row will be decreased by one but the column will remain the same. Pushing the joystick up will return a value of 14 for X. Thus COL(14)=0 and ROW(14)=−1.

Here are the values of each array element:

COL(1)	= 0	ROW(1)	= 0
COL(2)	= 0	ROW(2)	= 0
COL(3)	= 0	ROW(3)	= 0
COL(4)	= 0	ROW(4)	= 0
COL(5)	= 1	ROW(5)	= 1
COL(6)	= 1	ROW(6)	= −1
COL(7)	= 1	ROW(7)	= 0
COL(8)	= 0	ROW(8)	= 0
COL(9)	= −1	ROW(9)	= 1
COL(10)	= −1	ROW(10)	= −1
COL(11)	= −1	ROW(11)	= 0
COL(12)	= 0	ROW(12)	= 0
COL(13)	= 0	ROW(13)	= 1
COL(14)	= 0	ROW(14)	= −1
COL(15)	= 0	ROW(15)	= 0

To see how it works, take a closer look at the routine. Lines 100 and 110 begin the creation of the array by placing zeros into elements 1–4, since those numbers are not used for joystick reading. Line 120 uses the DATA statements on lines 140 and 150 to place the proper values into array elements 5–15. Line 130 sets up the screen and some important values.

On line 40 and again on line 130 the statement TRAP 40 appears. TRAP is used here to avoid the need to check boundaries of the screen each time the routine loops through. TRAP 40 will move the program to line 40 each time an error condition is reached. When the PLOT command on line 30 tries

to plot to a location which is out-of-bounds, TRAP detects the error condition. Without the TRAP the program would stop. But in this routine the program will move to line 40 and check the values of the column (C) and the row (R) and reset them to within the legal limits.

Once everything is set up, this routine uses only lines 20 and 30. The routine moves to another line only if you try to move the cursor out of bounds. Line 30 erases the old cursor by PLOTting in the background color, COLOR 0; then it PLOTs the new location with COLOR 1 and saves the location in A and B. At the end of line 30 the routine moves back to line 20 to read the joystick again.

GOTO STICK

Program 2 uses some of the same techniques as Program 1. The main difference is that the necessary responses to joystick movement are not stored in arrays. Instead, the instructions (that is, which way to move the cursor) are placed on the line number that is returned by reading the joystick. In other words, GOTO STICK(0) will read the joystick and go to the line that corresponds to the joystick movement. If the joystick is in the center position, the routine goes to line 15 and repeats itself.

In order for this routine to work properly, the correct line numbers must be used. Line 14 will move the cursor up; line 13 will move the cursor down; line 11 will move it left; line 7 will move it right; and so on.

Of course, you are not limited to using lines 5–15. The statement could be GOTO STICK(0)*10 and require the use of lines 50–150. GOSUB STICK(0) is also an acceptable statement. Again, the loop requires the use of only three lines if you don't move the cursor out of bounds.

Program 1. Joystick Reading with Arrays

For error-free program entry, read "The Automatic Proofreader" in this chapter before typing in this program.

```
HC 10 GOSUB 100
OB 20 X=STICK(0):C=C+COL(X):R=R+ROW(X):IF X=15
      THEN 20
DJ 30 COLOR 0:PLOT A,B:COLOR 1:PLOT C,R:A=C:B=
      R:GOTO 20
EI 40 TRAP 40:IF C=-1 THEN C=0:GOTO P
LO 50 IF C=80 THEN C=79:GOTO P
```

```
JD 60 IF R=-1 THEN R=0:GOTO P
NN 70 IF R=48 THEN R=47:GOTO P
NL 100 DIM ROW(15),COL(15)
FN 110 FOR X=1 TO 4:ROW(X)=0:COL(X)=0:NEXT X
OA 120 FOR X=5 TO 15:READ C,R:COL(X)=C:ROW(X)=
       R:NEXT X
PA 130 GRAPHICS 21:P=30:TRAP 40:POKE 712,132:P
       OKE 709,122:RETURN
HM 140 DATA 1,1,1,-1,1,0,0,0,-1,1
IN 150 DATA -1,-1,-1,0,0,0,0,1,0,-1,0,0
```

Program 2. Joystick Reading with GOTO

```
NC 2 GOTO 70
IO 5 C=C+1:R=R+1:GOTO P
JB 6 C=C+1:R=R-1:GOTO P
BJ 7 C=C+1:GOTO P
JE 9 C=C-1:R=R+1:GOTO P
LO 10 C=C-1:R=R-1:GOTO P
EG 11 C=C-1:GOTO P
GE 13 R=R+1:GOTO P
GH 14 R=R-1:GOTO P
JO 15 GOTO STICK(0)
DM 20 COLOR 0:PLOT A,B:COLOR 1:PLOT C,R:A=C:B=
      R:GOTO 15
JA 30 TRAP 30:IF C=80 THEN C=79:GOTO P
HD 40 IF C=-1 THEN C=0:GOTO P
NL 50 IF R=48 THEN R=47:GOTO P
JD 60 IF R=-1 THEN R=0:GOTO P
FE 70 P=20:GRAPHICS 21:POKE 712,116:POKE 709,1
      6:TRAP 30:GOTO 15
```

Chapter 1

Three Music Editors for Your Atari

David Florance

Atari's SOUND statement lets you create a multitude of sounds. And sound editors like the ones presented here make the job easy. You'll be amazed at just what your Atari can play.

Atari BASIC, unlike many other BASICs, includes a statement to create sound. It's possible to create music with one to four voices—the statement is even easy to use. Its syntax is:

SOUND ***voice,pitch,distortion,volume***

where *voice* is a number from 0 to 3, *pitch* from 0 to 255, *distortion* from 0 to 15 (10 is a pure tone, 8 is noise), and *volume* from 0 to 15. If you're using more than one voice at a time, the sum of the volumes should not exceed 16.

With just that short introduction, let's jump right into creating our music masterpieces.

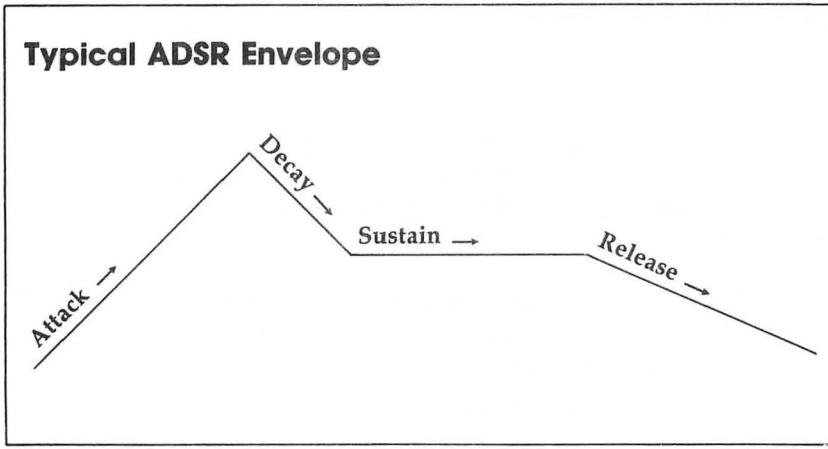
An Envelope

Some critics of Atari sound generation claim it's not comparable to a sound synthesizer. However, when you look closer at the inner workings of Atari sound, you'll quickly find that it's much more powerful and flexible than first imagined.

The first sample program, "Envelope," shows some of the sound possibilities of the Atari. When the program is run, you're asked to supply the envelope parameters. Experiment with different values, but be sure to stay within the stated limits. The figure shows a typical attack/decay/sustain/release (ADSR) envelope.

The attack value controls how fast a sound rises from silence to maximum volume. Decay is the rate at which it declines from the maximum to its sustained volume. Sustain indicates how long the sound will be held, and release controls how quickly it fades into silence.

The technique used in Envelope is simple. By using FOR-NEXT loops we can control the envelope of a sound. Since the Atari's BASIC loop timing will change depending on program length, this technique is not perfect. For our purposes, however, it works well enough to simulate the ADSR control of a true sound synthesizer.



Melodies

Program 2, "Melodies," assists in composing melodies. Program 3, "Player 1," plays the songs you create with Program 2. Be sure to enter and save both Programs 2 and 3 before using the former.

These two programs let you create melodies of up to 48 notes in length (it's possible to enter more than 48 notes, but the DATA will scroll off the screen). For each note, enter a pitch value from those displayed on the screen, and a duration value (0 to 255). The Atari then plays the note, though not with the specified duration. When all the notes are entered, type 0 for the pitch. You'll now hear the composition. If you don't like it, you can easily edit the note(s).

After your melody is finished, the program lets you save the song data if desired. If you press Y when prompted, the Atari prints your note values on the screen, in numbered DATA lines. At this point you can either write the values down, or enter NEW and load Program 3. If you do the latter, be careful not to disturb the DATA lines, which will be added to Program 3. When the Atari has finished loading, cursor up and press RETURN at each numbered line. The song data is now part of the program. Run the program to replay the song, or save it (with a new name) to preserve your song for posterity.

Table 1. Pitch Values

C	243	121	60	F#	173	85	42
C#	230	114	57	G	162	81	40
D	217	108	53	G#	153	76	37
D#	204	102	50	A	144	72	35
E	193	96	47	A#	136	68	33
F	182	91	45	B	128	64	31

Note: Any number from 0 to 255 is an acceptable pitch value.

Creating Chords

Creating chords is very similar to creating notes. Type in and save Program 4, "Chords," and Program 5, "Player 3." These programs let you control three voices instead of one. Just as before, you enter pitch and duration values for each note. Durations, however, do not have the same values as before. The chord editor uses eight duration values (see Table 2). The first note will be assigned to voice 1, the second to voice 2, and so on. The Atari will play each note when you enter it, but without the specified duration.

When all the notes are entered, enter a pitch of 0 to replay the chord(s). Next, you can save the chord data as a file. You'll need to specify the device and filename for disk (no quote marks are needed).

To play your chords or preserve the data, run Program 5, Player 3. You'll be prompted for the filename of your chord(s). Type in the filename, and when the READY prompt appears, delete lines 1–4, which append the data to the player program, and enter RUN again. There are your chords, just as you entered them, again ready to be saved as a separate program.

Table 2. Duration Values

1=quarter note	(1 beat)
2=half note	(2 beats)
3=dotted half note	(3 beats)
4=whole note	(4 beats)
5=quarter & whole	(5 beats)
6=half & whole	(6 beats)
7=dotted half & whole	(7 beats)
8=two whole notes	(8 beats)

All Four Voices

The chord editor used only three voices. "Song Editor" and "Player 4" allow you to write songs using any combination of one, two, three, or four voices. Now you can write complex songs, controlling the pitch and duration of each note, and inserting rests wherever desired.

The program will prompt you to enter the number of the voice you want to use, then the pitch and duration values for each note. To switch voices, type in 1 for the pitch. Enter a pitch of 0 for rests, and a pitch of 2 to hear your work. When finished, the song you've composed is written to disk or cassette, and can be heard by using the Player 4 program. Use the same procedure as you did with Player 3. Try adding a new line to Player 4, 9999 RUN, for continuous music.

Program 1. Envelope

For error-free program entry, read "The Automatic Proofreader" in this chapter before typing in this program.

```
NO 10  DIM A$(50),P(10),P$(10),R$(10),CH(10),ER
       $(38):X=20:Y=1:ER$="{19 SPACES}"
AL 20  CH(1)=3:CH(2)=255:CH(3)=14:CH(4)=15:CH(5
       )=32767:CH(6)=14:CH(7)=CH(4):CH(8)=CH(4)
       :CH(9)=CH(2):CH(10)=1
HJ 30  FOR T=1 TO 10:P(T)=0:NEXT T
NM 50  PRINT CHR$(125)
LL 55  POKE 709,0:POKE 710,14:POKE 712,14
ID 100 REM MAINPRG
KO 110 FOR N=1 TO 10:READ A$:PRINT A$:PRINT :N
       EXT N
EA 120 FOR T=1 TO 10
CL 130 POSITION X,Y:INPUT P$:TRAP 600
MG 135 P=VAL(P$):P(T)=INT(P)
JH 137 IF P(T)>CH(T) OR P(T)<0 OR ASC(P$)<48 A
       ND ASC(P$)>57 THEN POSITION X,Y:PRINT E
       R$:GOTO 130
KO 140 Y=Y+2:NEXT T
DA 190 IF P(6)>0 THEN 300
ON 200 SOUND P(1),P(2),P(3),P(4)
GB 210 IF P(10)=1 THEN ON P(1)+1 GOSUB 220,230
GJ 215 GOTO 260
KM 220 POKE 53768,132:RETURN
HO 230 POKE 53768,34:RETURN
LP 260 FOR Y=1 TO P(5)
DJ 269 NEXT Y
EP 270 SOUND P(1),P(2),P(3),0
LE 280 GOSUB 500:GOTO 200
```

```
EK 300  REM SUBPRG
NP 305  IF P(6)=0 THEN P(6)=1
FQ 310  FOR Y=1 TO 15 STEP P(6)
HE 320  SOUND P(1),P(2),P(3),Y
CO 330  NEXT Y
IE 332  SOUND P(1),P(2),P(3),15
IK 335  FOR T=1 TO P(5):NEXT T
KO 340  FOR T=15 TO P(8) STEP -1:SOUND P(1),P(2
        ),P(3),T:FOR F=1 TO P(7):NEXT F
CP 345  NEXT T
AD 350  FOR T=P(8) TO 0 STEP -1:SOUND P(1),P(2)
        ,P(3),T:FOR D=15 TO P(9) STEP -1:NEXT D
        :NEXT T:SOUND P(1),P(2),P(3),0
KP 400  GOSUB 500:GOTO 300
FL 500  REM PROMPT
NC 505  PRINT "AGAIN(Y/N)";:TRAP 40000:INPUT R$
BC 507  IF R$=CHR$(89) THEN POSITION 2,21:PRINT
         ER$:POSITION 2,21:RETURN
DM 510  PRINT "MORE(Y/N)";:INPUT R$
IA 520  IF R$=CHR$(89) THEN CLR :GOTO 10
GP 530  END
IH 600  IF P$=CHR$(155) THEN 140
GF 610  GOTO 140
IG 1000  DATA VOICE(0-3),PITCH(0-255),DISTORTIO
         N(0-14),VOLUME(0-15)

AD 1010  DATA DURATION(0-32767),ATTACK(0-14),DE
         CAY(0-15),SUSTAIN(0-15),RELEASE(0-15),
         FILTER(0=OFF/1=ON)
```

Program 2. Melodies

```
NP 10  COUNTER=1535:POKE 709,132:POKE 712,132:P
       OKE 710,132
NH 15  DIM A(12),B(12),C(12),D(7),R(10),P$(10),
       ER$(20),G$(10)
JO 20  ER$="{18 SPACES}"
GI 100  REM OCTAVE 1
LB 105  RESTORE 11500
EP 110  FOR X=1 TO 12:READ A:A(X)=A:NEXT X
GK 200  REM OCTAVE 2
LD 205  RESTORE 11510
FD 210  FOR X=1 TO 12:READ B:B(X)=B:NEXT X
GM 300  REM OCTAVE 3
LF 305  RESTORE 11520
FH 310  FOR X=1 TO 12:READ C:C(X)=C:NEXT X
CC 400  REM NOTE NAMES
LH 405  RESTORE 11530
CP 410  FOR X=1 TO 7:READ D:D(X)=D:NEXT X
```

```
AM 500 PRINT CHR$(125)
CD 510 FOR X=1 TO 7:PRINT CHR$(D(X))
LL 512 IF X=3 THEN 517
CI 515 PRINT
DE 517 NEXT X
JM 520 POSITION 2,2
BP 530 FOR X=1 TO 7
CK 532 IF X=3 OR X=7 THEN 535
FM 533 PRINT CHR$(D(X));CHR$(35)
CK 535 PRINT
DG 537 NEXT X
BP 540 Y=1:FOR X=1 TO 12:POSITION 9,Y:PRINT A(
       X),B(X),C(X):Y=Y+1
DB 550 NEXT X
KE 560 PRINT :FOR X=1 TO 29:PRINT CHR$(20);:NE
       XT X
CL 600 POKE 709,190
KH 610 TRAP 610:POSITION 2,16:PRINT ER$:POSITI
       ON 2,16:PRINT "PITCH ";:INPUT P
KN 615 IF P=0 THEN GOSUB 3000:GOSUB 5000:GOTO
       10
HK 617 IF P<0 OR P>255 THEN 610
IL 620 TRAP 620:POSITION 2,16:PRINT ER$:POSITI
       ON 2,16:PRINT "DURATION ";:INPUT D
GC 625 IF D<0 OR D>255 THEN 620
BG 1000 SOUND 1,P,10,15
HG 1010 FOR X=1 TO 300:NEXT X
OC 1020 SOUND 1,P,10,0
BJ 1025 IF E=1 THEN RETURN
AG 1030 GOSUB 2000
JF 1040 GOTO 610
IN 2000 POKE COUNTER,P
FL 2010 POKE COUNTER+128,D:COUNTER=COUNTER+1
KE 2020 RETURN
CO 3000 FOR X=1535 TO COUNTER-1
JH 3010 SOUND 1,PEEK(X),10,15
HC 3020 FOR T=1 TO PEEK(X+128):NEXT T
DE 3030 NEXT X:SOUND 1,PEEK(X),10,0
BI 3040 IF E=1 THEN RETURN
DM 5000 PRINT CHR$(125)
PL 5010 PRINT "DO YOU LIKE THE SONG(Y/N)";:INP
       UT G$
BA 5020 IF G$="Y" THEN GOSUB 6000
AM 5025 IF G$="N" THEN GOSUB 8000
DM 5030 PRINT "MORE";:INPUT G$
BI 5035 IF G$="Y" THEN GOSUB 8000
AM 5040 GOSUB 3000
NG 6000 PRINT "WOULD YOU LIKE THE DATA(Y/N)";:
       INPUT G$
JF 6010 IF G$="Y" THEN 6500
```

```
MK 6015 GOTO 5000
IJ 6020 PRINT :PRINT LN+10;" DATA 256":END
LC 6500 PRINT CHR$(125):LN=10000:SC=10
MI 6502 PRINT LN;" DATA ";
DL 6505 FOR X=1535 TO COUNTER-1
NM 6510 PRINT PEEK(X);",";PEEK(X+128);
FM 6515 IF COUNTER-1>1535+SC AND X>1535+SC THE
        N GOSUB 7000:GOTO 6550
GO 6520 IF X=COUNTER-1 THEN 6550
AG 6530 PRINT ",";
KC 6550 NEXT X:GOTO 6020
NC 7000 LN=LN+10:PRINT :PRINT :PRINT LN;" DATA
         ";:SC=SC+12:RETURN
DE 8000 PRINT CHR$(125):FOR X=1535 TO COUNTER-
        1
DL 8010 PRINT "NOTE #";X-1534;" ";PEEK(X);"
        {3 SPACES}DURATION #";X-1534;" ";PEEK(
        X+128)
GB 8020 NEXT X
EC 8030 PRINT :PRINT :PRINT "PRESS E TO HEAR S
        ONG AGAIN ";:INPUT G$
IE 8040 IF G$="A" THEN 8050
LB 8045 RETURN
PN 8050 E=1:GOSUB 3000
KJ 8060 PRINT "WHICH NOTE TO EDIT";:INPUT NE
BE 8070 E=1:GOSUB 400:E=0:POKE 1535+NE-1,P:POK
        E 1535+NE-1+128,D
LA 8080 RETURN
CG 11500 DATA 243,230,217,204,193,182,173,162,
         153,144,136,128
KJ 11510 DATA 121,114,108,102,96,91,85,81,76,7
         2,68,64
GH 11520 DATA 60,57,53,50,47,45,42,40,37,35,33
         ,31,29
AM 11530 DATA 67,68,69,70,71,65,66
```

Program 3. Player 1

```
HC 10 DIM A(128),B(128):PRINT CHR$(125):POKE 7
   09,192:POKE 710,45:POKE 712,45
CH 20 X=0
KM 30 X=X+1
JO 40 READ A:IF A=256 THEN GOTO 100
AN 45 A(X)=A:READ B:B(X)=B
AB 50 GOTO 30
IN 100 FOR N=1 TO X-1
EG 110 SOUND 1,A(N),10,5
MD 120 FOR DELAY=1 TO B(N):NEXT DELAY
CB 130 NEXT N
KI 10000 DATA 256
```

Program 4. Chords

```
JC 15  DIM A(112),B(112),C(112),D(112),R(120),P
       $(10),ER$(20),G$(20),FN$(12)
AD 17  DIM SOP(112),ALT(112),TEN(112)
OI 20  ER$="{18 SPACES}":NU=0
CK 30  POKE 710,85:POKE 712,85:POKE 709,132
PC 40  FOR X=1 TO 112:SOP(X)=0:ALT(X)=0:TEN(X)=
       0
GI 100  REM OCTAVE 1
LB 105  RESTORE 11500
EP 110  FOR X=1 TO 12:READ A:A(X)=A:NEXT X
GK 200  REM OCTAVE 2
LD 205  RESTORE 11510
FD 210  FOR X=1 TO 12:READ B:B(X)=B:NEXT X
GM 300  REM OCTAVE 3
LF 305  RESTORE 11520
FH 310  FOR X=1 TO 12:READ C:C(X)=C:NEXT X
CC 400  REM NOTE NAMES
LH 405  RESTORE 11530
CP 410  FOR X=1 TO 7:READ D:D(X)=D:NEXT X
AM 500  PRINT CHR$(125)
CD 510  FOR X=1 TO 7:PRINT CHR$(D(X))
LL 512  IF X=3 THEN 517
CI 515  PRINT
DE 517  NEXT X
JM 520  POSITION 2,2
BP 530  FOR X=1 TO 7
CK 532  IF X=3 OR X=7 THEN 535
FM 533  PRINT CHR$(D(X));CHR$(35)
CK 535  PRINT
DG 537  NEXT X
BP 540  Y=1:FOR X=1 TO 12:POSITION 9,Y:PRINT A(
       X),B(X),C(X):Y=Y+1
DB 550  NEXT X
KE 560  PRINT :FOR X=1 TO 29:PRINT CHR$(20);:NE
       XT X
DB 570  POKE 709,190
LN 600  FOR V=1 TO 3:POSITION 2,22:PRINT "VOICE
        #";V
JP 610  POSITION 2,16:PRINT ER$:POSITION 2,16:P
       RINT "PITCH ";:INPUT P
JC 615  IF P=0 THEN GOSUB 3000:GOTO 5000
HK 617  IF P<0 OR P>255 THEN 610
IC 620  POSITION 2,16:PRINT ER$:POSITION 2,16:P
       RINT "DURATION ";:INPUT D
PP 625  IF D<1 OR D>8 THEN 620
BG 1000  SOUND 1,P,10,15
HG 1010  FOR X=1 TO 300:NEXT X
```

```
OC 1020 SOUND 1,P,10,0
NO 1030 ON V GOSUB 2000,6000,7000
HO 1040 NEXT V:NU=NU+3:GOTO 600
EH 2000 FOR I=1 TO D
JJ 2010 J1=J1+1:SOP(J1)=P:NEXT I
KE 2020 RETURN
LG 3000 FOR X=1 TO NU
DC 3010 SOUND 0,SOP(X),10,5
CD 3020 SOUND 1,ALT(X),10,5
CP 3025 SOUND 2,TEN(X),10,5
HG 3027 FOR T=1 TO 100:NEXT T
CN 3030 NEXT X:SOUND 0,SOP(X),10,0:SOUND 1,ALT
        (X),10,0:SOUND 2,TEN(X),10,0
KH 3040 RETURN
NA 5000 PRINT CHR$(125):GOSUB 9000:END
EL 6000 FOR I=1 TO D
PA 6010 G=G+1:ALT(G)=P:NEXT I
KI 6020 RETURN
EM 7000 FOR I=1 TO D
PE 7010 F=F+1:TEN(F)=P:NEXT I
KJ 7020 RETURN
NJ 9000 PRINT "WOULD YOU LIKE THE DATA(Y/N)";:
        INPUT G$
JM 9010 IF G$="Y" THEN 9015
KD 9012 END
BG 9015 PRINT "REMEMBER TO DESIGNATE DEVICE(E.
        G. 'D:' FOR DISK)":PRINT "AS PART OF Y
        OUR FILENAME."
OB 9017 PRINT "FILENAME";:INPUT FN$:GOTO 9500
AO 9020 PRINT #1;CHR$(155);LN+10;" DATA 256,25
        6,256":END
IM 9500 PRINT CHR$(125):LN=10000:SC=8
PJ 9501 OPEN #1,8,0,FN$
FK 9502 PRINT #1;LN;" DATA ";
MG 9505 FOR X=1 TO NU
KH 9510 PRINT #1;SOP(X);",";ALT(X);",";TEN(X);
FF 9515 IF NU>1+SC AND X>1+SC THEN GOSUB 9600:
        GOTO 9550
JJ 9520 IF X=NU THEN 9550
JI 9530 PRINT #1;",";
KI 9550 NEXT X:GOTO 9020
EM 9600 LN=LN+10:PRINT #1;:PRINT #1:PRINT #1;L
        N;" DATA ";:SC=SC+12:RETURN
CG 11500 DATA 243,230,217,204,193,182,173,162,
         153,144,136,128
KJ 11510 DATA 121,114,108,102,96,91,85,81,76,7
         2,68,64
GH 11520 DATA 60,57,53,50,47,45,42,40,37,35,33
         ,31,29
AM 11530 DATA 67,68,69,70,71,65,66
```

Program 5. Player 3

```
IF 1  DIM FN$(12):PRINT CHR$(125):POKE 709,190:
      POKE 712,180:POKE 710,180
CP 2  PRINT "REMEMBER TO DESIGNATE DEVICE (E.G.
       'D:' FOR DISK":PRINT "WHEN ENTERING FILE
      NAME."
AC 3  PRINT "FILENAME";:INPUT FN$
GK 4  ENTER FN$
BN 10  DIM A(128),B(128),C(128)
CH 20  X=0
KM 30  X=X+1
HL 40  READ A,B,C:IF A=256 THEN GOTO 100
BL 45  A(X)=A:B(X)=B:C(X)=C
AB 50  GOTO 30
IN 100  FOR N=1 TO X-1
CF 110  SOUND 0,A(N),10,5:SOUND 1,B(N),10,5:SOU
        ND 2,C(N),10,5
HD 120  FOR DELAY=1 TO 100:NEXT DELAY
CB 130  NEXT N
```

Program 6. Song Editor

```
CF 10  DIM A(12),B(12),C(12),D(7),R(10),P$(10),
       ER$(40),FN$(14)
OM 15  POKE 710,23:POKE 712,23:POKE 709,23
JO 20  ER$="{35 SPACES}"
AB 30  DIM SOP(128),ALT(128),TEN(128),BAS(128)
PK 40  DIM COUNTER(128),G(4),W(100),VL(128),FL(
       128),I(128),R$(128)
AP 45  C0=0:C1=0:C2=0:C3=0:LN=10000:SC=12
PI 47  COUNTER=0
EA 50  FOR I=0 TO 128
MB 52  SOP(I)=0:ALT(I)=0:TEN(I)=0:BAS(I)=0
BB 53  VL(I)=0
PC 55  NEXT I
GI 100  REM OCTAVE 1
LB 105  RESTORE 11500
EP 110  FOR X=1 TO 12:READ A:A(X)=A:NEXT X
GK 200  REM OCTAVE 2
LD 205  RESTORE 11510
FD 210  FOR X=1 TO 12:READ B:B(X)=B:NEXT X
GM 300  REM OCTAVE 3
LF 305  RESTORE 11520
FH 310  FOR X=1 TO 12:READ C:C(X)=C:NEXT X
CC 400  REM NOTE NAMES
LH 405  RESTORE 11530
CP 410  FOR X=1 TO 7:READ D:D(X)=D:NEXT X
AM 500  PRINT CHR$(125)
```

```
CD 510 FOR X=1 TO 7:PRINT CHR$(D(X))
LL 512 IF X=3 THEN 517
CI 515 PRINT
DE 517 NEXT X
JM 520 POSITION 2,2
BP 530 FOR X=1 TO 7
CK 532 IF X=3 OR X=7 THEN 535
FM 533 PRINT CHR$(D(X));CHR$(35)
CK 535 PRINT
DG 537 NEXT X
BP 540 Y=1:FOR X=1 TO 12:POSITION 9,Y:PRINT A(
       X),B(X),C(X):Y=Y+1
DB 550 NEXT X
EA 555 IF F1=1 THEN F1=0:RETURN
KE 560 PRINT :FOR X=1 TO 29:PRINT CHR$(20);:NE
       XT X
DB 570 POKE 709,190
KJ 600 TRAP 600:POSITION 2,16:PRINT ER$:POSITI
       ON 2,16:PRINT "VOICE";:INPUT V
BJ 605 IF V<0 OR V>3 THEN 600
OE 607 POSITION 2,20:PRINT "VOICE ";V
GD 608 X=0
KH 610 TRAP 610:POSITION 2,16:PRINT ER$:POSITI
       ON 2,16:PRINT "PITCH ";:INPUT P
HI 615 IF P<0 OR P>255 THEN 610
DF 617 IF P=1 THEN X=1:G(V)=COUNTER:COUNTER=0:
       GOTO 600
OC 618 IF P=2 THEN 6000
IL 620 TRAP 620:POSITION 2,16:PRINT ER$:POSITI
       ON 2,16:PRINT "DURATION";:INPUT D
PL 630 IF D<1 OR D>8 THEN 620
PH 640 ON V+1 GOSUB 1000,2000,3000,4000
GL 650 GOTO 610
AG 1000 GOSUB 5000
FH 1010 FOR Y=1 TO D
FK 1020 SOP(X)=P:X=X+1:NEXT Y:RETURN
AH 2000 GOSUB 5000
FI 2010 FOR Y=1 TO D
EK 2020 ALT(X)=P:X=X+1:NEXT Y:RETURN
AI 3000 GOSUB 5000
FJ 3010 FOR Y=1 TO D
FC 3030 TEN(X)=P:X=X+1:NEXT Y:RETURN
AJ 4000 GOSUB 5000
FK 4010 FOR Y=1 TO D
EC 4030 BAS(X)=P:X=X+1:NEXT Y:RETURN
JE 5000 SOUND V,P,10,8:FOR T=1 TO 200:NEXT T:S
       OUND V,0,10,0:COUNTER=COUNTER+D:RETURN
AN 6000 GOSUB 7000
IL 6003 TRAP 40000:GOSUB 8000:GOSUB 8900:GOSUB
        8800
```

```
CO 6005 FOR L=0 TO COUNTER
CI 6010 SOUND 0,SOP(L),10,4
BJ 6020 SOUND 1,ALT(L),10,4
CB 6030 SOUND 2,TEN(L),10,4
BG 6040 SOUND 3,BAS(L),14,4
EI 6050 FOR T=1 TO 13:NEXT T
FH 6060 NEXT L
LF 6070 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,0,
        0,0:SOUND 3,0,0,0
NB 6090 GOTO 9000
PC 7000 REM TEST FOR COUNTER
CD 7010 IF G(0)>G(1) THEN G(1)=G(0)
CI 7020 IF G(1)>G(2) THEN G(2)=G(1)
CN 7030 IF G(2)>G(3) THEN G(3)=G(2)
CM 7040 IF G(3)>G(1) THEN G(1)=G(3)
CH 7050 IF G(0)>G(1) THEN G(1)=G(0)
PD 7060 COUNTER=G(1)
KO 7070 RETURN
HJ 8000 PRINT CHR$(125):FL=1
DM 8010 FOR I=0 TO 3
EP 8020 PRINT "WOULD YOU LIKE TO HEAR VOICE ";
        I;"<Y/N>";:INPUT R$
EJ 8030 IF R$="Y" THEN ON I+1 GOSUB 8100,8200,
        8300,8400
FE 8040 NEXT I
CO 8050 FOR I=1 TO COUNTER
PM 8055 IF C0=1 THEN SOUND 0,SOP(I),10,10
OP 8057 IF C1=1 THEN SOUND 1,ALT(I),10,10
PJ 8059 IF C2=1 THEN SOUND 2,TEN(I),10,10
OD 8061 IF C3=1 THEN SOUND 3,BAS(I),10,10
FD 8065 FOR T=1 TO 25:NEXT T
HP 8067 NEXT I:SOUND 0,0,0,0:SOUND 1,0,0,0:SOU
        ND 2,0,0,0:SOUND 3,0,0,0
AL 8069 C0=0:C1=0:C2=0:C3=0
LG 8077 RETURN
ME 8100 C0=1:RETURN
MG 8200 C1=1:RETURN
MI 8300 C2=1:RETURN
MK 8400 C3=1:RETURN
BK 8800 PRINT "READY TO HEAR IT ALL<Y/N>";:INP
        UT R$
KK 8810 IF R$="Y" THEN 6005
OE 8820 PRINT "WOULD YOU LIKE TO EDIT<Y/N>";:I
        NPUT R$
LC 8830 IF R$="Y" THEN 9800
NG 8840 GOTO 9000
LB 8900 PRINT "ANOTHER COMBINATION<Y/N>";:INPU
        T R$
HA 8905 IF R$="Y" THEN GOSUB 8000:GOTO 8900
```

```
LF 8910 IF R$="N" THEN RETURN
AJ 9000 PRINT CHR$(125):POSITION 2,16:PRINT "W
        OULD YOU LIKE TO SAVE<Y/N>";:INPUT R$
KI 9002 IF R$="Y" THEN 9015
NJ 9005 GOTO 8820
OM 9015 PRINT CHR$(125):PRINT "REMEMBER TO DES
        IGNATE DEVICE(e.g.'D:FILE')"
OB 9017 PRINT "FILENAME";:INPUT FN$:GOTO 9500
NH 9020 PRINT #1;CHR$(155);LN+10;" DATA 256,25
        6,256,256":END
IM 9500 PRINT CHR$(125):LN=10000:SC=8
PJ 9501 OPEN #1,8,0,FN$
FK 9502 PRINT #1;LN;" DATA ";
ED 9505 FOR X=1 TO COUNTER
AM 9510 PRINT #1;SOP(X);",";ALT(X);",";TEN(X);
        ",";BAS(X);
NC 9515 IF COUNTER>1+SC AND X>1+SC THEN GOSUB
        9600:GOTO 9550
BG 9520 IF X=COUNTER THEN 9550
JI 9530 PRINT #1;",";
KI 9550 NEXT X:GOTO 9020
EM 9600 LN=LN+10:PRINT #1;:PRINT #1:PRINT #1;L
        N;" DATA ";:SC=SC+12:RETURN
EI 9800 PRINT CHR$(125)
GP 9805 POSITION 2,18:PRINT "WHICH VOICE TO ED
        IT";:INPUT EV
CM 9810 PRINT CHR$(125):ON EV+1 GOTO 9820,9840
        ,9860,9880
EC 9820 FOR X=0 TO COUNTER
OH 9822 PRINT "NOTE #";X;" ";SOP(X)
MC 9824 IF X>20 THEN POSITION 12,1
MJ 9826 IF X>40 THEN POSITION 24,1
HC 9828 NEXT X

LP 9830 FOR X=1 TO COUNTER:SOUND 0,SOP(X),10,4
        :FOR T=1 TO 100:NEXT T:NEXT X
KD 9832 SOUND 0,0,0,0
GK 9834 TRAP 9834:PRINT "WHICH NOTE TO EDIT";:
        INPUT EN:F1=1:GOSUB 500
HF 9836 POSITION 2,18:PRINT "NEW NOTE #";EN;:I
        NPUT NN:SOP(EN)=NN:PRINT "MORE";:INPUT
         R$
LM 9837 IF R$="Y" THEN 9820
NL 9838 GOTO 6000
EE 9840 FOR X=0 TO COUNTER
NI 9842 PRINT "NOTE #";X;" ";ALT(X)
ME 9844 IF X>20 THEN POSITION 12,1
ML 9846 IF X>40 THEN POSITION 24,1
HE 9848 NEXT X
```

```
JC 9850 FOR X=1 TO COUNTER:SOUND 1,ALT(X),10,4
        :FOR T=1 TO 100:NEXT T:NEXT X
KG 9852 SOUND 1,0,0,0
GD 9854 TRAP 9854:PRINT "WHICH NOTE TO EDIT";:
        INPUT EN:F1=1:GOSUB 500
GG 9856 POSITION 2,18:PRINT "NEW NOTE #";EN;:I
        NPUT NN:ALT(EN)=NN:PRINT "MORE";:INPUT
         R$
MA 9857 IF R$="Y" THEN 9840
NN 9858 GOTO 60000
EG 9860 FOR X=0 TO COUNTER
OA 9862 PRINT "NOTE #";X;" ";TEN(X)
MG 9864 IF X>20 THEN POSITION 12,1
MN 9866 IF X>40 THEN POSITION 24,1
HG 9868 NEXT X
JL 9870 FOR X=1 TO COUNTER:SOUND 2,TEN(X),10,4
        :FOR T=1 TO 100:NEXT T:NEXT X
KJ 9872 SOUND 2,0,0,0
HC 9874 TRAP 9874:PRINT "WHICH NOTE TO EDIT";:
        INPUT EN:F1=1:GOSUB 500
GO 9876 POSITION 2,18:PRINT "NEW NOTE #";EN;:I
        NPUT NN:TEN(EN)=NN:PRINT "MORE";:INPUT
         R$
ME 9877 IF R$="Y" THEN 9860
NP 9878 GOTO 60000
EI 9880 FOR X=0 TO COUNTER
NB 9882 PRINT "NOTE #";X;" ";BAS(X)
MI 9884 IF X>20 THEN POSITION 12,1
MP 9886 IF X>40 THEN POSITION 24,1
HI 9888 NEXT X
IN 9890 FOR X=1 TO COUNTER:SOUND 3,BAS(X),10,4
        :FOR T=1 TO 100:NEXT T:NEXT X
KM 9892 SOUND 3,0,0,0
HG 9894 TRAP 9894:PRINT "WHICH NOTE TO EDIT";:
        INPUT EN:F1=1:GOSUB 500
FP 9896 POSITION 2,18:PRINT "NEW NOTE #";EN;:I
        NPUT NN:BAS(EN)=NN:PRINT "MORE";:INPUT
         R$
MI 9897 IF R$="Y" THEN 9880
CG 11500 DATA 243,230,217,204,193,182,173,162,
         153,144,136,128
KJ 11510 DATA 121,114,108,102,96,91,85,81,76,7
         2,68,64
GH 11520 DATA 60,57,53,50,47,45,42,40,37,35,33
         ,31,29
AM 11530 DATA 67,68,69,70,71,65,66
```

Program 7. Player 4

```
IC 1 DIM FN$(14):PRINT CHR$(125):POKE 709,160:
     POKE 712,170:POKE 710,170
CP 2 PRINT "REMEMBER TO DESIGNATE DEVICE (E.G.
      'D:' FOR DISK":PRINT "WHEN ENTERING FILE
     NAME."
AC 3 PRINT "FILENAME";:INPUT FN$
GK 4 ENTER FN$
HJ 10 DIM A(128),B(128),C(128),D(128)
CH 20 X=0
KM 30 X=X+1
OL 40 READ A,B,C,D:IF A=256 THEN GOTO 100
MD 45 A(X)=A:B(X)=B:C(X)=C:D(X)=D
AB 50 GOTO 30
IN 100 FOR N=1 TO X-1
BE 110 SOUND 0,A(N),10,4:SOUND 1,B(N),10,4:SOU
       ND 2,C(N),10,4:SOUND 3,D(N),10,4
HD 120 FOR DELAY=1 TO 100:NEXT DELAY
CB 130 NEXT N
```

Chapter 1

Exploring Atari Variables

Bob Powell

For many programmers, variables are not too exciting. But you'll be able to create much more effective programs if you understand how they are stored and how they can be manipulated.

There are three types of variables that can be used by Atari programmers.

Scalars. Common numerical variables, represented by a variable name such as X, Y, PAY, HIT, etc., are called scalars. The value of a scalar variable is assigned within your program (for example, by X=7, INPUT Y, PAY=A+B, or HIT=X+15*Y). Each scalar value occupies six bytes of memory.

Arrays. Arrays are sets of numbers represented by a variable name followed by the element number in the set (for example, A(3), SCORE(20), or INCOME(10,12)). Before using an array, you must dimension it with the maximum size expected (for example, DIM SCORE(22)). That sets aside six bytes of memory for each array element; in other words, DIM SCORE(22) reserves 22*6 or 132 bytes.

Strings. String variables are extremely versatile, and you will find many uses for them. Strings are represented by a variable name followed by the dollar sign (for example, A$, VTAB$, or BLANK$) and must also be dimensioned so the computer can reserve memory for the string. Each *character* in a string variable is stored as a one-byte ATASCII code (ranging from 0 to 255), so entering DIM A$(35) will reserve 35 bytes of memory for A$.

The *variable name table* holds a list of all variable names that have been entered. They are stored as ATASCII code numbers in the same sequence that they were encountered in your program. Each variable name also identifies the type of variable. Scalars are stored with 128 added to the ATASCII value of the last byte in the name. For example, the name of the variable HIT is described in three bytes with values 72, 73, and 212 (212=84+128). Arrays are stored with a left parenthesis with 128 added (40+128, or 168) as the last byte in the name. Strings are stored with the dollar sign plus 128 (36+128 or 164) as the last byte.

The memory address for the start of the variable name

table can be determined with PEEKs into 130 (LO) and 131 (HI) or with NTAB=PEEK(130)+256*PEEK(131).

Run the following program to see the entries in the name table. It prints the value stored in the first 24 bytes of the name table, one byte for each character in each variable name. Check the results against the ATASCII listing in your manual, and don't forget that 128 has been added to the last character. The first number printed is 216, which is the first variable X (ATASCII code 88) plus 128, since X is also the last character in the first variable name. Line 50 prints the names with the last character as an inverse. Note: Before running this and subsequent program examples, enter NEW to clear previous variables from the tables before entering the program.

For error-free program entry, read "The Automatic Proofreader" elsewhere in this chapter before typing in this program.

```
FK 1 REM .PROGRAM TO PRINT VARIABLE NAME TABLE
EN 10 X=12:Y=35:ZZ=12345:DIM A(12),HIT(4,9),SM
      ALL$(35),BIG$(612)
GB 20 NTAB=PEEK(130)+256*PEEK(131):REM .FIND N
      AME TABLE STARTING ADDRESS.
PJ 30 FOR BYTE=0 TO 23:? PEEK(NTAB+BYTE);",";:
      NEXT BYTE
JF 40 ? :? :?
KC 50 FOR BYTE=0 TO 23:? CHR$(PEEK(NTAB+BYTE))
      ;:NEXT BYTE
```

The *variable value table* contains eight bytes of data for each variable in the name table. The eight bytes have different meanings for each variable type, as shown in Table 1.

Table 1. Bytes in Variable Value Table

Variable Type	Byte Number							
	0	1	2	3	4	5	6	7
Scalar	0	var #	6-byte binary coded decimal (BCD) value					
Array	65	var #	offset LO	offset HI	first DIM+1 LO	first DIM+1 HI	second DIM+1 LO	second DIM+1 HI
String	129	var #	offset LO	offset HI	length LO	length HI	DIM LO	DIM HI

To find the start of the variable value table, PEEK into 134(LO) and 135(HI) or VTAB=PEEK(134)+256*PEEK(135). Enter and run the following program to see the variable number, name, and eight bytes of data for each variable in the program. As in the previous program, line 10 contains example variables for experimentation. By the way, this program can be easily modified and appended to your own programs to list variables for reference.

```
LG 1 REM .PROGRAM TO PRINT VARIABLE VALUE TABL
     E
LG 5 GRAPHICS 0
EN 10 X=12:Y=35:ZZ=12345:DIM A(12),HIT(4,9),SM
     ALL$(35),BIG$(612)
GF 15 NTAB=PEEK(130)+256*PEEK(131):REM .FIND N
     AME TABLE STARTING ADDRESS.
MN 20 VTAB=PEEK(134)+256*PEEK(135):REM .FIND V
     ALUE TABLE STARTING ADDRESS.
JN 25 ? "VAR{3 SPACES}VARNAME{5 SPACES}VTAB DA
     TA":?
DE 30 I=0:FOR VARNUM=0 TO 12:? "#";VARNUM;"
     {4 SPACES}";
CP 35 POSITION 9,VARNUM+2:FOR BYTE=0 TO 100:VA
     RCHR=NTAB+BYTE
LM 40 IF PEEK(VARCHR)>128 THEN ? CHR$(PEEK(VAR
     CHR)-128),:NTAB=VARCHR+1:POP :GOTO 55
HB 45 ? CHR$(PEEK(VARCHR));
NI 50 NEXT BYTE
GB 55 POSITION 17,VARNUM+2:FOR BYTE=0 TO 7:? P
     EEK(VTAB+VARNUM*8+BYTE);
LF 60 IF BYTE<7 THEN ? ",";
FH 65 NEXT BYTE:?
QE 70 I=I+1:NEXT VARNUM
```

The *array/string table* contains the actual data for each element in an array and each character in a string. When BASIC encounters a string or array in your program, it first checks the name table for the variable number, then looks to the value table to see what it is. Finally, it takes the value of OFFSET (LO+256*HI) and reaches that many bytes past the start of the array/string table for the actual data. The start of the array/string table is determined by PEEKing 140(LO) and 141(HI) or ATAB=PEEK(140)+256*PEEK(141).

The following program prints out data in the array/string table for SMALL$. Note that SMALL$ is the sixth variable entered in the program; thus its variable number is 5 (the first variable, X, is variable number 0).

```
BP 1  REM .PROGRAM TO PRINT ARRAY/STRING DATA F
      OR SELECTED STRING (SMALL$)
EN 10  X=12:Y=35:ZZ=12345:DIM A(12),HIT(4,9),SM
       ALL$(35),BIG$(612)
DM 20  SMALL$="COMPUTE! BOOKS"
NM 30  VTAB=PEEK(134)+256*PEEK(135)
AK 40  ATAB=PEEK(140)+256*PEEK(141):REM .FIND S
       TART OF ARRAY/STRING DATA AREA
BA 50  LO=PEEK(VTAB+5*8+2):REM .PEEK IN VTAB FO
       R LO-BYTE OF SMALL$ OFFSET
PO 60  HI=PEEK(VTAB+5*8+3):REM .PEEK IN VTAB FO
       R HI-BYTE OF SMALL$ OFFSET
IJ 70  OFFSET=LO+256*HI
LH 80  FOR BYTE=0 TO 16:A=PEEK(ATAB+OFFSET+BYTE
       ):? A,CHR$(A):NEXT BYTE
```

Using Strings to Store Designs

So much for how variables are managed. What can be done with this knowledge? Plenty. Obviously, you can PEEK into the tables and determine the status of various variables. Less obvious but equally useful, you can POKE in new values. Also, for arrays and strings, you can change the value of OFFSET and the computer will use the new address for the data instead of the array/string table. This is particularly useful when dealing with strings, as you'll see in a moment.

As mentioned earlier, strings are very versatile. You know that strings are nothing more than a series of ATASCII numbers. Each number has a value from 0–255 and occupies one byte in memory. A string of 1000 characters will occupy 1000 adjacent bytes in memory.

That suggests an interesting application. A handy way to reserve 1000 bytes (or any other amount) of memory for your use is to dimension a string, say A$(1000). You could set up your own table of values within the string by indexing every 10 or 100 addresses. The starting address of your string can be quickly found by the ADR(A$) function. Entering ?PEEK(ADR(A$)+99) will return the contents of the hundredth byte in A$. Using strings in this manner is a common

way to store integer data (values less than 256) and machine language subroutines. Note that when a string is used in this way, it will have a peculiar appearance when printed; what you see on the screen will be the ATASCII characters corresponding to the byte values for the data.

Now let's see what happens when you change the OFFSET value for a string. Recall that after BASIC builds the variable tables, you can go in with POKEs and alter them. If you dimension a string, say A$, to be the first variable encountered in a BASIC program, it will be listed as the first variable (0) in the tables (don't forget to enter NEW to clear tables first before entering your program). Hence, in the variable value table (VTAB), bytes 2 and 3 are the LO and HI values, respectively, for the OFFSET of A$ data from ATAB start.

In other words, the starting address of A$ data is ATAB+OFFSET or ATAB+PEEK(VTAB+2)+256*PEEK(VTAB+3). POKE new LO-HI values into locations (VTAB+2) and (VTAB+3); the computer will store A$ data to ATAB+NEWOFFSET instead of the array/string table.

Remembering that memory is just a long continuous place to store numbers from 0 to 255, let's change A$'s OFFSET and position the string to start at the display memory. Afterwards, any change in A$ will still change numbers in A$'s data table. But since A$'s data is stored in display memory, you will change the display as well.

You can use that to create a screen design by putting characters into A$. That may seem a little mysterious, but the computer is actually looking at the display memory area 60 times a second for screen data and couldn't care less where the data came from.

The following program defines a string (A$), determines new LO-HI values for A$'s OFFSET, and then POKEs the new values into the variable value table for A$. Line 90 stores some data in A$ which is immediately displayed on the screen.

```
HJ 1 REM .PROGRAM TO STORE STRING DATA TABLE F
    OR A$ IN SCREEN MEMORY AREA
MM 5 DIM A$(200),B$(200)
HM 10 GRAPHICS 3:SETCOLOR 0,4,6
NL 20 VTAB=PEEK(134)+256*PEEK(135)
MB 30 ATAB=PEEK(140)+256*PEEK(141)
CA 40 SCREEN=PEEK(88)+256*PEEK(89)
```

```
MK 50 OFFSET=SCREEN-ATAB:REM .FIND DISTANCE FR
      OM ATAB START TO SCREEN MEMORY START
KI 60 HI=INT(OFFSET/256):REM .FIND HI-BYTE OF
      NEW OFFSET
DK 70 LO=OFFSET-256*HI:REM .FIND LO-BYTE OF NE
      W OFFSET
FL 80 POKE VTAB+2,LO:POKE VTAB+3,HI:REM .POKE
      IN NEW LO-HI POINTER FOR A$
BM 90 A$="COMPUTE! BOOKS":END
PF 100 FOR I=1 TO 200:A$(I,I)=CHR$(200):NEXT I
GJ 110 END
DN 200 A$(1)=CHR$(0):A$(200)=CHR$(0):A$(2)=A$
```

While the GRAPHICS 3 picture is on the screen, type GOTO 100 to see one of many possible effects. Line 100 is a BASIC loop that fills A$ memory with the number 200.

Speeding Things Up

A much faster trick can be used to work at machine language speed. All characters in a string can be rapidly set to the same value by the statement A$(1)=CHR$(X):A$(SIZE)=CHR$(X): A$(2)=A$, where X=0–255 and SIZE is equal to the maximum string length. For example, A$(200) can be set to all zeros by executing A$(1)=CHR$(0):A$(200)=CHR$(0):A$(2) =A$. Try typing GOTO 200 while you're still in GRAPHICS 3 and you will see the screen go blank (but note that a small, upper portion of the screen will still have some data since ATAB has now moved a few bytes in the manual mode). ATAB would not have moved had you stayed in the program mode. Now try changing A$ to contain all 255's by using CHR$(255). Try 185, 70, and other numbers of your choice.

By now, you've probably noticed that each screen byte defines four adjacent color pixels on the screen in combinations of four colors (including background). This is true for the multicolor graphics modes of 3, 5, and 7. In GRAPHICS 8, each screen byte defines eight adjacent pixels of a single color. In multicolor modes, on and off patterns of bits 0 and 1 of a screen byte select the color for the rightmost pixel associated with that byte. Bits 2 and 3 set the next pixel to the left and so forth. Go ahead and experiment some more by changing A$ to different values.

Press SYSTEM RESET (don't enter NEW this time or it will destroy your program) and PRINT A$ to see what it looks

like. Now run the program again and while in GRAPHICS 3 enter B$="FFFFFFFFF". Then set A$(100,110)=B$ to set the characters 100–110 of A$ equal to B$. Try A$(120,130)=B$ and experiment with different positions in A$ and values of B$. You can put A$(X,X+LEN(B$))=B$ in a loop and vary X to create interesting effects. The screen can be changed very rapidly by having several strings for different shapes or designs and setting A$ (or parts of A$) equal to the choices. These strings can be complete pictures, and you can flip from one to another as simply as saying A$=B$ and then A$=C$.

One more possibility is to first position the string in display (or screen) memory. Then using conventional PLOT and DRAWTO commands, create a design of some sort; the graphic results will be automatically stored in your string.

Of course, you can save the string to disk or tape like any other string, but you are really saving the screen. To do so, put A$ in screen memory and then set B$=A$ after your design is completed. Finally, save B$.

In order to redisplay your saved picture, set up the graphics mode, change the offset to put A$ in screen memory, recall B$ from tape or disk, and set A$=B$. Presto! Instant picture. Different graphics modes require different amounts of memory, so be sure you have dimensioned a string large enough to cover the screen. The memory required for each mode is given in the following table. The lower row of values shows the number of bytes available when no text window is used.

Table 2. Screen Memory Requirements for Various Graphics Modes

	Graphic Mode											
	0	1	2	3	4	5	6	7	8	9	10	11
Memory	960	400	200	200	400	800	1600	3200	6400			
Memory (mode+16)		480	240	240	480	960	1920	3840	7680	7680	7680	7680

Atari Color Matcher

Ron Tinnell

The Atari has a great color set, but finding the colors you want can be tiresome. This program lets you browse through the 128 colors at your leisure and makes it easy to pick the combinations you want.

"Color Matcher" uses graphics mode 2 to put a three-color test pattern on the screen. It can show you two foreground colors and any one background color at one time.

Using the space bar and the cursor control keys, you have full control over all three colors. The cursor-left and cursor-right keys control color hue, while the cursor-up and cursor-down keys control luminosity. Pressing these keys steps through color or luminosity values by 1. You don't need to use the CTRL key; simply press the cursor key to change the values.

The display shows the hue and luminosity values that correspond to the colors currently on the screen. An indicator points to the variable parameter that is currently active; press the space bar to select the test color you want to change.

When entering the program, be careful with the PRINT statements in lines 160, 170, and 420. Each contains a double space.

Color Matcher

For error-free program entry, read "The Automatic Proofreader" elsewhere in this chapter before typing in this program.

```
DF 120 DIM H(3),L(3)
AO 130 GRAPHICS 2:H(1)=1:L(1)=14:H(2)=8:L(2)=8
       :H(3)=3:L(3)=8:F=1
FJ 140 SETCOLOR 4,1,14:SETCOLOR 0,8,8:SETCOLOR
        2,3,8
GC 150 POSITION 5,0:PRINT #6;"BKGD COL1"
EN 160 POSITION 1,2:PRINT #6;"HUE   1"
IP 170 POSITION 1,3:PRINT #6;"LUM   14"
KF 180 POSITION 11,2:PRINT #6;"8"
KH 190 POSITION 11,3:PRINT #6;"8"
EE 200 POSITION 2,6:PRINT #6;"A B C D"
AP 210 POSITION 3,7:PRINT #6;"E F G"
BL 220 POSITION 4,8:PRINT #6;"HIJ"
FI 230 POSITION 2,9:PRINT #6;"#########"
IB 240 POSITION 2,10:PRINT #6;"#########"
```

```
JA 250 POSITION 15,0:PRINT #6;CHR$(195);CHR$(2
       07);CHR$(204);CHR$(178)
KE 260 POSITION 16,2:PRINT #6;"3"
KL 270 POSITION 16,3:PRINT #6;"8"
PF 280 POSITION 11,6:PRINT #6;CHR$(205);CHR$(1
       60);CHR$(206);CHR$(160);CHR$(207);CHR$(
       160);CHR$(208)
KH 290 POSITION 12,7:PRINT #6;CHR$(209);CHR$(1
       60);CHR$(210);CHR$(160);CHR$(211)
FJ 300 POSITION 13,8:PRINT #6;CHR$(212);CHR$(2
       13);CHR$(214)
MM 310 POSITION 10,9:FOR M=0 TO 7:PRINT #6;CHR
       $(163);:NEXT M
PF 320 POSITION 10,10:FOR M=0 TO 7:PRINT #6;CH
       R$(163);:NEXT M
LN 400 IF F=4 THEN F=1
NE 405 POSITION (5*F+1),4:PRINT #6;CHR$(94);CH
       R$(94)
LL 410 CLOSE #3:OPEN #3,4,0,"K:":GET #3,K
GN 420 IF K=32 THEN POSITION (5*F+1),4:PRINT #
       6;"  ":F=F+1:GO TO 400
JK 430 IF K=42 THEN H(F)=H(F)+1
JO 440 IF K=43 THEN H(F)=H(F)-1
CE 450 IF H(F)<0 THEN H(F)=15
CH 460 IF H(F)>15 THEN H(F)=0
KK 470 IF K=45 THEN L(F)=L(F)+2
KL 480 IF K=61 THEN L(F)=L(F)-2
CP 490 IF L(F)<0 THEN L(F)=14
CJ 500 IF L(F)>14 THEN L(F)=0
HE 510 POSITION (5*F+1),2:PRINT #6;H(F);" "
HK 520 POSITION (5*F+1),3:PRINT #6;L(F);" "
FK 530 Q=4
MK 540 IF F=2 THEN Q=0
MO 550 IF F=3 THEN Q=2
HB 560 SETCOLOR Q,H(F),L(F)
GJ 570 GO TO 400
```

Chapter 1

The Automatic Proofreader

Charles Brannon

At last there's a way for your computer to help you check your typing. "The Automatic Proofreader" will make entering programs faster, easier, and more accurate.

The strong point of computers is that they excel at tedious, exacting tasks. So why not get your computer to check your typing for you?

"The Automatic Proofreader" will help you type in program listings without typing mistakes. It is a short error-checking program that hides itself in memory. When activated, it lets you know immediately after typing a line from a program listing if you have made a mistake. Please read these instructions carefully before typing in any programs in this book.

Preparing the Proofreader

1. Using the listing below, type in the Proofreader. Be very careful when entering the DATA statements—don't type an l instead of a 1, an O instead of a 0, extra commas, etc.
2. Save the Proofreader on tape or disk at least twice *before running it for the first time.*
3. After the Proofreader is saved, type RUN. It will check itself for typing errors in the DATA statements and warn you if there's a mistake. Correct any errors and save the corrected version. Keep a copy in a safe place—you'll need it again and again, every time you enter a program from this book or *COMPUTE!* magazine.
4. When a correct version of the Proofreader is run, it activates itself. You are now ready to enter a program listing. If you press SYSTEM RESET, the Proofreader is disabled. To reactivate it, just type PRINT USR(1536) and press RETURN.

Using the Proofreader

All listings in this book have a *checksum* found immediately to the left of each line number. *Don't enter the checksum when typing in a program.* It is just for your information.

When you type in a line from a program listing and press RETURN, the Proofreader displays the checksum letters at the top of your screen. *These checksum letters must match the checksum letters in the printed listing.* If they don't, it means

you typed the line differently than the way it is listed. Immediately recheck your typing. You can correct any mistakes you find immediately.

The Proofreader is not picky with spaces. It will not notice extra spaces or missing ones. This is for your convenience, since spacing is generally not important. But occasionally proper spacing *is* important, so be extra careful with spaces, since the Proofreader will catch practically everything else that can go wrong.

Due to the nature of a checksum, the proofreader will not catch all errors. The Proofreader will not catch errors of transposition. In fact, you could type in a line in any order, and the Proofreader wouldn't notice.

There's another thing to watch out for: If you enter the lines by using abbreviations for commands, the checksum will not match up. But there is a way to make the Proofreader check it. After entering the line, LIST it. This eliminates the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way. The only abbreviation that cannot be handled this way is when a ? is used instead of PRINT; they are not the same to the Proofreader.

The Automatic Proofreader

```
100 GRAPHICS 0
110 FOR I=1536 TO 1700:READ A:POKE I,A:CK=C
    K+A:NEXT I
120 IF CK<>19072 THEN ? "Error in DATA Stat
    ements.  Check Typing.":END
130 A=USR(1536)
140 ? :? "Automatic Proofreader Now Activat
    ed."
150 END
1536 DATA 104,160,0,185,26,3
1542 DATA 201,69,240,7,200,200
1548 DATA 192,34,208,243,96,200
1554 DATA 169,74,153,26,3,200
1560 DATA 169,6,153,26,3,162
1566 DATA 0,189,0,228,157,74
1572 DATA 6,232,224,16,208,245
1578 DATA 169,93,141,78,6,169
1584 DATA 6,141,79,6,24,173
1590 DATA 4,228,105,1,141,95
1596 DATA 6,173,5,228,105,0
```

```
1602 DATA 141,96,6,169,0,133
1608 DATA 203,96,247,238,125,241
1614 DATA 93,6,244,241,115,241
1620 DATA 124,241,76,205,238,0
1626 DATA 0,0,0,0,32,62
1632 DATA 246,8,201,155,240,13
1638 DATA 201,32,240,7,72,24
1644 DATA 101,203,133,203,104,40
1650 DATA 96,72,152,72,138,72
1656 DATA 160,0,169,128,145,88
1662 DATA 200,192,40,208,249,165
1668 DATA 203,74,74,74,74,24
1674 DATA 105,161,160,3,145,88
1680 DATA 165,203,41,15,24,105
1686 DATA 161,200,145,88,169,0
1692 DATA 133,203,104,170,104,168
1698 DATA 104,40,96
```

Chapter 2

Games

Chapter 2

Nessie
A Nonviolent Game for Atari

Tom R. Halfhill

"Nessie" is a nonviolent action game that challenges you to snap a clear photograph of the Loch Ness monster. It runs on any Atari computer with at least 16K (tape) or 24K (disk), and a joystick.

For decades, fans of and believers in Scotland's Loch Ness monster have affectionately referred to the mysterious creature as "Nessie"—hence the title of this game.

The game was inspired by a TV documentary on Loch Ness which recounted the hundreds of attempts to photograph the monster. Almost all of these attempts have failed; there exist only a few controversial photos showing parts of fins, shadowy shapes, and blurred figures. The game simulates some of the difficulties faced by would-be photographers of Nessie.

Starting Nessie

When typing "Nessie," omit all REM statements if your computer has only 16K of RAM. It will barely fit in memory if loaded from tape. At least 24K is required for disk.

After you type RUN, the program requires a few seconds to initialize. During this period, special areas of memory are protected, player/missile graphics are set up, game screens are prepared, and machine language routines are loaded into memory (Nessie makes extensive use of machine language, as noted below). About halfway through this waiting period you'll see the camera viewfinder frame and aiming crosshairs appear on the screen.

The next screen which appears lets you select difficulty options and displays the scoring possibilities.

There are two difficulty levels. Toggle between them with the SELECT key. This chooses which lens you want on your camera. By far the easiest option is wide-angle, the default option. A wide-angle lens allows photographers to cover a larger area from their camera position. In Nessie, the wide-angle lens is indicated by a large viewfinder.

Pressing the SELECT key switches between the wide-angle and the telephoto lens. The telephoto is represented by

a much smaller viewfinder. In fact, the telephoto viewfinder barely frames Nessie. This makes the telephoto lens much more difficult to use than the wide-angle. To give you an idea of this difference, Nessie appears actual size within each viewfinder as you switch lenses.

Since the telephoto is harder to use, it scores more points. The lower half of this screen displays the point totals for every possible picture you can take. For example, using the wide-angle lens, a clear, properly framed photo of Nessie scores 2000 points; a photo in which you crop off Nessie with the viewfinder scores 100 points; if you are fooled and take a photo of a fish or an eel instead of Nessie, you get only 50 points; and if you shoot a blurred photo by moving the camera when you press the shutter button, you get zero points. Likewise, a photo of nothing also scores nothing.

All of these point totals are multiplied by ten if you're using the telephoto lens. The game also scores you on the amount of time you take to shoot your pictures. The longer you take, the lower your score.

After selecting your lens, begin the game by pressing the START key. This starts the timer and displays the main game screen.

Getting the Whole Picture

At the top of the main game screen is your camera's film counter, which shows how many pictures remain on your roll of film. You start with a 20-exposure roll. Each time you snap a picture, the film counter decrements.

Your camera viewfinder starts in the center of the screen (which represents Loch Ness). You can move it in any direction with the joystick. Pressing the fire button releases the shutter. The viewfinder frame itself is blue, with a green aiming crosshairs in the center. To take a properly centered photo, you must position the crosshairs over Nessie. If any part of Nessie is touching the viewfinder frame when you snap the shutter, it will register as a cropped photo when the film is developed at the end of the game. A picture of a piece of Nessie is better than nothing—that's why it's worth 100 or 1000 points—but it's not nearly as valuable as a photo of the whole monster. (Let's face it, wouldn't you feel better walking into the *New York Times* with an indisputable picture of Nessie instead of a doubtful snapshot of a dorsal fin?)

For the same reason, you must be careful not to include any other objects in the viewfinder while photographing Nessie. This isn't as easy as it sounds. When you start the game, you'll find that Loch Ness is alive with fish and eels of assorted shapes and colors. If you photograph one of these instead, you've been fooled—and your photo is worth only 50 or 500 points. (The reason you get any points at all is that you might be able to sell the photo to *Field & Stream* or an airline magazine.) The eels are particularly troublesome. They bear an uncanny resemblance to Nessie, which is why so many hopeful photographers over the years have been fooled.

Another hazard to beware of is jittery hands. Nessie is not an easy target—the creature appears at random in the Loch, swims in random directions for a few seconds, and then submerges to appear somewhere else. Meanwhile, you're trying to center the monster in the viewfinder. If you snap the shutter while moving the finder, the picture will be blurred. And that's worth zero points.

When you get down to your last five pictures on the roll of film, the viewfinder frame automatically turns from blue to bright yellow as a warning. This is in case you're too busy to pay attention to the film counter.

Developing the Film

After snapping your last shot, everything freezes for an instant. Then the screen blanks out and the film starts developing. Since the slightest bit of light in the darkroom would spoil it all, the screen is black during this process. After a few moments, the finished pictures appear—gradually developing to full brightness as they would in a darkroom tray.

Each of the 20 finished prints shows what you photographed when you snapped the shutter. They are arranged in the order you shot them, and each is captioned (unless the picture is blank). At the bottom of the screen is your final score, adjusted for the amount of time that elapsed.

To restart Nessie, just snap the shutter button on your joystick. This returns you to the setup screen, where you can change lenses if you wish before playing again.

Programmer's Notes

Nessie is a fast, responsive game because the most critical animation—the movement of the camera viewfinder—is

written entirely in machine language. An ML routine which fills almost all of page 6 in memory (1536 decimal, $600 hex) constantly reads the joystick and moves the finder. All of this is done during the vertical blank interval, that split second when the TV's electron gun returns from the bottom of the screen to the top to begin scanning another screen frame. Since this happens 60 times per second, the viewfinder's movements appear instantaneous and flicker-free.

The viewfinder itself is created with player/missile graphics. Two player objects are used—one for the frame and another for the crosshairs. This allows the collision-checking routine to detect separate collisions between Nessie, the frame, or the crosshairs.

At least 90 percent of the BASIC in Nessie is initialization—once it sets up the game for the first time, most of it is never executed again. Using BASIC for this work made Nessie easier to program, since setup tasks can be tedious in machine language. ML was used only for the time-critical operations.

This is reflected in the main loop, which starts at line 10000 and is really only six lines long (and a few of these lines could be combined to make the loop even shorter). Since the ML routine executes automatically during each vertical blank, repeated calls to the routine via BASIC's USR statement are unnecessary. The only thing BASIC does during the main phase of Nessie is animate the fish, eels, and monster. BASIC also checks the shutter button and handles the picture-taking sequence (clicking the shutter, flashing the screen, storing collision register values in arrays for later analysis). Everything else is in machine language.

Actually there are four ML routines in the program. By far the largest is the main routine in page 6. A second routine switches on the vertical blank interrupt routine when the game starts, and another shuts it off when the game ends. The fourth routine instantly flips player shapes when called by a USR statement. This is used to rapidly change the viewfinder's size when toggling back and forth between the wide-angle and telephoto lenses during the setup screen. This very short but useful routine is from Eric Stoltman's article "Extending Player/Missile Graphics" (*COMPUTE!'s First Book Of Atari Graphics*).

Redefined characters in graphics mode 2+16 are used for Nessie, the fish, and the eels during the main part of the

game. To speed up their animation, the characters are POKEd directly into screen memory, which is faster than using POSITION and PRINT statements.

The final game screen, which shows the developed pictures, uses a modified display list to put strips of different graphics modes on the screen simultaneously. This screen is a mixture of graphics modes 0 and 1.

Nessie

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
DE 200 GOSUB 11000:REM Initialize
GG 210 GOSUB 12000:REM Redefine characters
GH 215 GOTO 13000:REM Setups
BL 220 GOTO 10000:REM Main loop
KF 1000 REM
JB 1005 REM *** MOVE NESSIE & DECOYS ***
KG 1010 REM
CE 1020 POKE SCREEN+COORD(OBJECT),0:NOOCOORD=C
        OORD(OBJECT)+MOVE(INT(RND(0)*9)+1)
CI 1040 IF NOOCOORD<40 OR NOOCOORD>239 THEN CO
        ORD(OBJECT)=INT(RND(0)*200)+40:RETURN
AJ 1060 POKE SCREEN+NOOCOORD,CHAR(OBJECT):COOR
        D(OBJECT)=NOOCOORD:RETURN
KG 2000 REM
GD 2005 REM *** SNAP PHOTO ***
KH 2010 REM
FB 2020 POKE 77,0:FILM=FILM-1:POSITION 15,0:?
        #6;"  ":POSITION 15,0:? #6;FILM:SOUND
        0,0,0,0
BL 2040 FRAME(PHOTO)=PEEK(53252):HAIR(PHOTO)=P
        EEK(53253):BLUR(PHOTO)=STICK(0):PHOTO=
        PHOTO+1
BO 2060 SETCOLOR 4,9,4:IF FILM<6 THEN POKE 704
        ,28
MG 2080 IF FILM=0 THEN SOUND 0,240,10,15:POP :
        TIME=INT((PEEK(18)*65536+PEEK(19)*256+
        PEEK(20))/60):GOTO 2200
AA 2100 BUTTON=STRIG(0):RETURN
LJ 2200 A=USR(ADR(VBOFF$)):SOUND 0,0,0,0:FOR I
        =1 TO 1000:NEXT I:GOTO 14000
NF 10000 REM
DF 10005 REM *** MAIN LOOP ***
NG 10010 REM
EE 10020 POKE HITCLR,0
HF 10040 IF STRIG(0)=1 THEN BUTTON=1
```

```
GJ 10340 IF STRIG(0)=0 AND BUTTON=1 THEN POKE
         712,14:SOUND 0,4,8,15:GOSUB 2000
EL 10380 OBJECT=OBJECT+1:IF OBJECT>6 THEN OBJE
         CT=1
NF 10400 GOSUB 1000:REM Move objects
CB 10420 GOTO 10000
NG 11000 REM
JD 11020 REM *** INIT P/M & ML ***
NK 11040 REM
EJ 11060 GRAPHICS 2+16:SETCOLOR 2,0,0:? #6;"
         {7 SPACES}nessie":? #6:? #6;"
         {4 SPACES}PLEASE WAIT":? #6;"
         {5 SPACES}21 SECONDS"
BP 11080 PM=PEEK(106)-8:PMBASE=256*PM:HITCLR=5
         3278
NG 11100 FOR I=PMBASE+512 TO PMBASE+768:POKE I
         ,0:NEXT I
HF 11120 RESTORE 11280:DIM WIDEFRAME$(20):FOR
         I=1 TO 20:READ A:WIDEFRAME$(I,I)=CHR$
         (A):NEXT I:REM Wide viewfinder
HD 11140 RESTORE 11300:DIM TELEFRAME$(20):FOR
         I=1 TO 20:READ A:TELEFRAME$(I,I)=CHR$
         (A):NEXT I:REM Tele viewfinder
ID 11160 RESTORE 11340:DIM WIDEHAIR$(20):FOR I
         =1 TO 20:READ A:WIDEHAIR$(I,I)=CHR$(A
         ):NEXT I:REM Wide crosshair
IK 11180 RESTORE 11360:DIM TELEHAIR$(20):FOR I
         =1 TO 20:READ A:TELEHAIR$(I,I)=CHR$(A
         ):NEXT I:REM Tele crosshair
CP 11200 POKE 704,130:POKE 705,198:REM Blue vi
         ewfinder & green crosshair
IC 11220 POKE 559,46:POKE 623,1:POKE 53277,3:P
         OKE 54279,PM:POKE 53256,3:POKE 53257,
         3:REM P/M setup
BJ 11240 HORIZ0=116:VERT0=PMBASE+512+61:HORIZ1
         =118:VERT1=PMBASE+640+66:REM Initial
         positions
IN 11245 FOR I=1 TO 20:POKE VERT0+I,ASC(WIDEFR
         AME$(I)):NEXT I:POKE 53248,HORIZ0:REM
          Draw viewfinder
NP 11250 FOR I=1 TO 20:POKE VERT1+I,ASC(WIDEHA
         IR$(I)):NEXT I:POKE 53249,HORIZ1:REM
         Draw crosshair
OJ 11260 REM * VIEWFINDER SHAPES *
BO 11280 DATA 255,255,129,129,129,129,129,129,
         129,129,129,129,129,129,129,129,129,2
         55,255,0
EL 11300 DATA 255,129,129,129,129,129,129,129,
         129,255,0,0,0,0,0,0,0,0,0,0
```

```
KB 11320 REM * CROSSHAIR SHAPES *
DG 11340 DATA 16,16,16,16,124,16,16,16,16,0,0,
         0,0,0,0,0,0,0,0,0
FM 11360 DATA 16,16,124,16,16,0,0,0,0,0,0,0,0,
         0,0,0,0,0,0,0,0
PH 11380 REM * FLIP SHAPE ROUTINE *
NA 11400 DIM FLIP$(25):RESTORE 11420:FOR I=1 T
         O 25:READ A:FLIP$(I,I)=CHR$(A):NEXT I
DG 11420 DATA 104,104,133,204,104,133,203,104,
         133,207,104,133,206,160,0,177,206,145
         ,203,200,192,20,208,247,96
JN 11440 REM * READ JOYMOVE ROUTINE *
KA 11460 RESTORE 11480:FOR I=0 TO 204:READ A:P
         OKE 1536+I,A:NEXT I
OM 11480 DATA 174,120,2,224,14,208
CL 11490 DATA 3,32,172,6,224,6
FH 11500 DATA 208,6,32,172,6,32
FI 11510 DATA 107,6,224,7,208,3
IJ 11520 DATA 32,107,6,224,5,208
FD 11530 DATA 6,32,140,6,32,107
FF 11540 DATA 6,224,13,208,3,32
FO 11550 DATA 140,6,224,9,208,6
CJ 11560 DATA 32,140,6,32,74,6
IL 11570 DATA 224,11,208,3,32,74
FJ 11580 DATA 6,224,10,208,6,32
DJ 11590 DATA 172,6,32,74,6,76
PJ 11600 DATA 98,228,173,253,6,201
MA 11610 DATA 48,240,25,206,253,6
MD 11620 DATA 206,253,6,173,253,6
LH 11630 DATA 141,0,208,206,251,6
MB 11640 DATA 206,251,6,173,251,6
MN 11650 DATA 141,1,208,96,96,173
PC 11660 DATA 253,6,201,176,240,25
MP 11670 DATA 238,253,6,238,253,6
MB 11680 DATA 173,253,6,141,0,208
MN 11690 DATA 238,251,6,238,251,6
LJ 11700 DATA 173,251,6,141,1,208
ND 11710 DATA 96,96,172,252,6,192
MK 11720 DATA 96,176,24,160,0,177
BO 11730 DATA 203,145,205,136,192,0
DD 11740 DATA 208,247,238,252,6,238
MK 11750 DATA 252,6,238,250,6,238
ND 11760 DATA 250,6,96,96,172,252
MI 11770 DATA 6,192,28,144,24,160
BO 11780 DATA 0,177,205,145,203,200
JI 11790 DATA 192,255,208,247,206,252
LM 11800 DATA 6,206,252,6,206,250
GK 11810 DATA 6,206,250,6,96,96
ID 11820 REM * SET UP VBLANK *
```

```
FH 11900 DIM VBSETUP$(11):RESTORE 11920:FOR I=
         1 TO 11:READ A:VBSETUP$(I)=CHR$(A):NE
         XT I
KM 11920 DATA 104,162,6,160,0,169,7,32,92,228,
         96
JE 11940 DIM VBOFF$(11):FOR I=1 TO 11:READ A:V
         BOFF$(I)=CHR$(A):NEXT I
FH 11960 DATA 104,162,228,160,98,169,7,32,92,2
         28,96
CB 11980 DIM LENS$(18):DIM MOVE(9),CHAR(6),COO
         RD(6),FRAME(20),HAIR(20),BLUR(20),PIC
         TURE(20),CAPTION(30):RETURN
NH 12000 REM
PG 12010 REM *** REDEFINE CHARACTERS ***
NN 12015 REM
HI 12020 CHSET=(PEEK(106)-8)*256:FOR I=0 TO 51
         2:POKE CHSET+I,PEEK(57344+I):NEXT I
BE 12021 RESTORE 12025
AI 12022 READ A:IF A=-1 THEN RETURN
FM 12023 FOR J=0 TO 7:READ B:POKE CHSET+A*8+J,
         B:NEXT J
CJ 12024 GOTO 12022
IF 12025 DATA 1,255,255,192,192,192,192,192,19
         2
BC 12026 DATA 3,255,255,3,3,3,3,3,3
EC 12027 DATA 4,3,3,3,3,3,3,3,3
BG 12028 DATA 5,3,3,3,3,3,3,255,255
AG 12029 DATA 6,0,0,0,0,0,0,255,255
IH 12030 DATA 7,192,192,192,192,192,192,255,25
         5
IJ 12031 DATA 8,192,192,192,192,192,192,192,19
         2
PL 12032 DATA 9,0,0,0,0,0,0,213,127
AK 12033 DATA 10,0,0,64,192,85,127,126,0
JH 12034 DATA 11,0,0,2,3,170,254,126,0
CM 12035 DATA 12,0,0,0,0,0,0,171,254
AG 12036 DATA 13,0,0,56,125,222,125,56,0
AK 12037 DATA 14,0,0,28,190,123,190,28,0
EN 12038 DATA 26,5,10,21,42,84,168,80,128
DK 12039 DATA 27,255,255,0,0,0,0,0,0
MM 12040 DATA 32,0,32,48,32,224,224,0,0
HA 12041 DATA -1
NI 13000 REM
IC 13005 REM * SET UP SCREEN *
NJ 13010 REM
LI 13020 SETCOLOR 4,9,4:POKE 756,CHSET/256:? #
         6;CHR$(125):SETCOLOR 3,12,2:SETCOLOR
         0,4,8:SETCOLOR 2,0,0
CE 13040 FILM=20:LENS=1:LENS$="wideangleteleph
         oto":SCREEN=PEEK(88)+PEEK(89)*256
```

```
FN 13060 ? #6;"{7 SPACES}NESSIE":? #6:? #6;" <
         select> ";LENS$(LENS,LENS+8):? #6;"
         {4 SPACES}<start> game"
BD 13062 POSITION 0,8:? #6;CHR$(171);"  NESSIE
         =2000/20000":POSITION 0,9:? #6;CHR$(1
         92);"  CROP  = 100/ 1000"
HM 13064 POSITION 0,10:? #6;CHR$(174);CHR$(172
         );" FOOLED=  50/  500":POSITION 0,11:
         ? #6;CHR$(186);"  BLUR  ={3 SPACES}0"
CM 13070 POSITION 10,7:? #6;CHR$(171)
MF 13080 SOUND 0,0,0,0:POSITION 10,3:? #6;LENS
         $(LENS,LENS+8):IF PEEK(53279)=6 THEN
         13160
ND 13100 IF PEEK(53279)<>5 THEN 13080
GE 13110 LENS=LENS+9:IF LENS>10 THEN LENS=1
FF 13115 IF LENS=1 THEN 13125
EP 13120 IF LENS>1 THEN 13140
IH 13125 A=USR(ADR(FLIP$),VERT0,ADR(WIDEFRAME$
         )):POKE 53256,3:SOUND 0,160,10,15
IE 13128 FOR I=VERT1 TO VERT1+2:POKE I,0:NEXT
         I:VERT1=VERT1+3:A=USR(ADR(FLIP$),VERT
         1,ADR(WIDEHAIR$)):POKE 53257,3
FK 13130 POSITION 9,6:? #6;" ":POSITION 10,7:?
          #6;CHR$(171):HORIZ1=118:POKE 53249,H
         ORIZ1:GOTO 13080
FE 13140 A=USR(ADR(FLIP$),VERT0,ADR(TELEFRAME$
         )):POKE 53256,1:SOUND 0,80,10,15
CJ 13145 VERT1=VERT1-3:A=USR(ADR(FLIP$),VERT1,
         ADR(TELEHAIR$)):POKE 53257,1
FP 13150 POSITION 10,7:? #6;" ":POSITION 9,6:?
          #6;"■":HORIZ1=117:POKE 53249,HORIZ1:
         GOTO 13080
OC 13160 SOUND 0,120,10,15:? #6;CHR$(125):POSI
         TION 0,0:? #6;"■■■nessie FILM=";FILM;
         "■■■"
HL 13180 COLOR ASC("■"):PLOT 0,1:DRAWTO 19,1
HB 13200 MOVE(1)=-20:MOVE(2)=-19:MOVE(3)=1:MOV
         E(4)=21:MOVE(5)=20:MOVE(6)=19
FD 13220 MOVE(7)=-1:MOVE(8)=-21:MOVE(9)=250:CH
         AR(1)=9:CHAR(2)=139:CHAR(3)=77:CHAR(4
         )=201:CHAR(5)=14:CHAR(6)=204
BG 13240 FOR OBJECT=1 TO 6:COORD(OBJECT)=INT(R
         ND(0)*200)+40:POKE SCREEN+COORD(OBJEC
         T),CHAR(OBJECT):NEXT OBJECT
PD 13250 PHOTO=1
BF 13260 POKE 1789,HORIZ0:POKE 1787,HORIZ1
AP 13270 POKE 1788,61:POKE 203,0:REM VERT0 lo
         byte
JL 13280 POKE 204,(PMBASE+512)/256:REM VERT0 h
         i byte
```

```
LG 13290 POKE 1786,194:POKE 207,128:REM VERT1
         lo byte
JJ 13300 POKE 208,(PMBASE+512)/256:REM VERT1 h
         i byte
GI 13310 POKE 205,2:REM Lo byte for vert memor
         y shift
MG 13320 POKE 206,(PMBASE+512)/256:REM Hi byte
          for vert memory shift
GE 13330 IF LENS>1 THEN POKE 1647,192:POKE 168
         0,106:REM Reset range check for telep
         hoto
ML 13340 A=USR(ADR(VBSETUP$)):POKE 18,0:POKE 1
         9,0:POKE 20,0:SOUND 0,0,0,0:GOTO 220
NJ 14000 REM
AO 14020 REM *** DEVELOP FILM ***
NN 14040 REM
NP 14050 POKE 53277,0:POKE 53261,0:POKE 53262,
         0:REM P/M OFF
AN 14060 GRAPHICS 0:SETCOLOR 2,0,0:DLIST=PEEK(
         560)+256*PEEK(561):SCREEN=PEEK(88)+25
         6*PEEK(89)
NF 14065 SETCOLOR 0,0,0:SETCOLOR 1,12,10:SETCO
         LOR 3,0,0
NP 14070 POKE 756,CHSET/256:POKE 752,1:? CHR$(
         125);:POSITION 0,16:POKE 82,0
CC 14080 RESTORE 14100:FOR I=6 TO 26:READ A:PO
         KE DLIST+I,A:NEXT I
MO 14100 DATA 6,6,6,2,2,6,6,6,2,2,6,6,6,2,2,6,
         6,6,2,2,6
ML 14110 FOR I=1 TO 15:READ A:POKE SCREEN+672+
         I,A:NEXT I
NM 14112 FOR I=PMBASE+512 TO PMBASE+768:POKE I
         ,0:NEXT I
GP 14115 DATA 36,37,54,37,44,47,48,41,46,39,0,
         38,41,44,45
LG 14120 FOR II=1 TO 4
JG 14140 FOR I=0 TO 19:READ A:POKE SCREEN+40+I
         ,A:NEXT I
EL 14160 DATA 1,27,3,0,1,27,3,0,1,27,3,0,1,27,
         3,0,1,27,3,0
JM 14180 FOR I=0 TO 19:READ A:POKE SCREEN+60+I
         ,A:NEXT I
FB 14200 DATA 8,0,4,0,8,0,4,0,8,0,4,0,8,0,4,0,
         8,0,4,0
JJ 14220 FOR I=0 TO 19:READ A:POKE SCREEN+80+I
         ,A:NEXT I
HD 14240 DATA 7,6,5,0,7,6,5,0,7,6,5,0,7,6,5,0,
         7,6,5,0
```

```
NP 14260 RESTORE 14160:SCREEN=SCREEN+140:NEXT
         II
OK 14280 RESTORE 14300:SCREEN=PEEK(88)+256*PEE
         K(89):FOR I=0 TO 5:READ A:POKE SCREEN
         +606+I,A:NEXT I
FN 14300 DATA 46,37,51,51,41,37
GH 14350 FOR I=1 TO 20:READ A:PICTURE(I)=A:NEX
         T I
KD 14355 DATA 61,65,69,73,77,201,205,209,213,2
         17,341,345,349,353,357,481,485,489,49
         3,497
GA 14356 FOR I=1 TO 30:READ A:CAPTION(I)=A:NEX
         T I
CF 14357 DATA 34,44,53,50,0,0,35,50,47,48,0,0,
         38,47,47,44,37,36,46,37,51,51,41,37,0
         ,0,0,0,0,0
KM 14360 RESTORE 14510:SCORE=0:SETCOLOR 1,0,0:
         FOR I=1 TO 21
KG 14365 IF I=21 THEN 14520
EG 14370 IF BLUR(I)<>15 THEN POKE SCREEN+PICTU
         RE(I),90:X=0:GOSUB 14500:NEXT I
AO 14380 IF HAIR(I)=4 AND FRAME(I)<>4 THEN SCO
         RE=SCORE+2000:POKE SCREEN+PICTURE(I),
         203:X=18:GOSUB 14500:NEXT I
AG 14390 F=FRAME(I):H=HAIR(I)
AK 14400 IF F>3 AND F<>8 AND F<>9 AND F<>10 TH
         EN SCORE=SCORE+100:POKE SCREEN+PICTUR
         E(I),96:X=6:GOSUB 14500:NEXT I
PF 14420 IF HAIR(I)>0 THEN SCORE=SCORE+50:POKE
          SCREEN+PICTURE(I),76:X=12:GOSUB 1450
         0:NEXT I
GO 14440 X=24:GOSUB 14500:NEXT I
BE 14500 SOUND 0,12*I,10,7:READ A:FOR II=1 TO
         6:POKE SCREEN+PICTURE(I)+A+II,CAPTION
         (II+X):NEXT II
HC 14505 SOUND 0,0,0,0:RETURN :REM Print capti
         ons
MK 14510 DATA 38,42,46,50,54,38,42,46,50,54,38
         ,42,46,50,54,38,42,46,50,54,54:REM Ca
         ption locations
OG 14520 FOR I=0 TO 6:SETCOLOR 0,5,I:FOR II=1
         TO 8:NEXT II:NEXT I
BO 14540 FOR I=0 TO 8:SETCOLOR 3,15,I:FOR II=1
          TO 8:NEXT II:NEXT I
ED 14550 FOR I=0 TO 10:SETCOLOR 1,12,I:FOR II=
         1 TO 8:NEXT II:NEXT I
PO 14560 RESTORE 14570:FOR I=0 TO 6:READ A:POK
         E SCREEN+632+I,A:NEXT I
MG 14570 DATA 51,35,47,50,37,0,29
```

```
CP 14620 RESTORE 14640:FOR I=1 TO 22:READ A:PO
         KE SCREEN+666+I,A:NEXT I
PN 14640 DATA 52,47,0,50,37,48,44,33,57,0,51,4
         6,33,48,0,51,40,53,52,52,37,50
BP 14650 IF LENS>1 THEN SCORE=SCORE*10
EA 14660 ? INT((4/TIME)*SCORE);
DM 14680 IF STRIG(0)=1 THEN 14680
NN 14700 SOUND 0,4,8,15:FOR I=1 TO 3:NEXT I:SO
         UND 0,0,0,0:POKE 53248,0:POKE 53249,0
         :POKE 704,130
BA 14720 GRAPHICS 2+16:HORIZ0=116:VERT0=PMBASE
         +512+61:HORIZ1=118:VERT1=PMBASE+640+6
         6:POKE 53277,3:POKE 559,46
EL 14740 POKE 53256,3:POKE 53257,3:FOR I=1 TO
         20:POKE VERT0+I,ASC(WIDEFRAME$(I)):NE
         XT I:POKE 53248,HORIZ0
NF 14760 FOR I=1 TO 20:POKE VERT1+I,ASC(WIDEHA
         IR$(I)):NEXT I:POKE 53249,HORIZ1:GOTO
          13000
```

Tank

David E. Huff and Douglas C. Huff

With the help of this program, you may well be the first on your block to drive a tank. A joystick and at least 16K RAM are required.

You are the commander of a sophisticated Tracking And Neutralizing Kar (TANK). Your mission? It's too secret to even think about—but before you can get started you must cross enemy territory and pick up vital information from rebel headquarters.

To get that information, you must navigate your tank through a treacherous battlefield strewn with mines and coiled barbed-wire fences—and to make things even more interesting, you have to take on hostile enemy MCs (Mobile Crunchers) as well as enemy aircraft too.

Taking Control

The obstacle-strewn battlefield scrolls from top to bottom as you push your joystick forward. Should you touch any object while threading your way through the minefield, your tank will be destroyed. In addition, you must keep an eye out for those Crunchers as they cross the field in an attempt to ram your tank. You have to blast them, because they cannot be outrun.

Your tank is blue. It will not appear at the bottom of the screen until you move your joystick to the left, right, or up. Enemy Crunchers are red and will attack at random from the sides of the screen.

There are three sets of five fields each. As the field number increases, more obstacles appear on the battlefield. When you finish a set of five battlefields, the Crunchers speed up for the next set.

You start with three tanks, and you get an extra one for each battlefield you cross. Hitting any object (or being flattened by a Cruncher) will cost you one tank, and getting blown up will cost you any points you have accumulated on that particular attempt. Once you are destroyed you must start over at the beginning of the battlefield.

You receive ten points for blasting an enemy Cruncher or for shooting down an aircraft, but the points are not actually awarded until you successfully complete a field. When you run out of tanks press the START key to restart the game.

Hints

When maneuvering your tank, try to stay in the middle of the battlefield. That gives you more time to turn and aim at the approaching enemy vehicles. Try to get through tight spots quickly. If you get caught in a tight spot, you might not be able to turn and fire on the enemy without hitting an obstacle.

Note that your horizontal speed is greater than your vertical speed. This gives you a chance to move away from a mine before turning and firing at the enemy tank. If you just tap the joystick, you can flip directions without moving. Remember that you cannot go backwards, so choose your route carefully.

If you find yourself in a spot where there is no way to squeeze between two objects, shoot down an airplane. That will cause your tank to automatically miniaturize and enable you to squeeze through. But be careful. Your tank will return to normal size without warning just before the next airplane starts to cross the field. The time varies. Your tank could remain small for a long time or jump right back to normal size. As a general rule, when your tank is miniaturized, don't stay close to obstacles any longer than you have to.

Collision detection requires an overlap between objects, so you can get right next to mines without getting destroyed. Even without being miniaturized, your tank can squeeze through many tight spots if you are careful.

ML, BASIC, VBI, and Characters

The program uses a combination of machine language and BASIC to set up the screen, scroll the screen, and move the players. A vertical blank interrupt routine scrolls the playfield vertically, and redefined characters are used for the battlefield objects. A machine language subroutine sets up and moves the players, detects collisions, and makes the explosions appear on the screen. BASIC sets up the battlefields by POKEing characters into screen memory at random locations. BASIC also displays the score and keeps track of the number of tanks left and the number of playfields traversed.

The main machine language routine is stored as a string of characters labeled E$. This machine language routine is relocatable so that it may be stored in string form. Type PRINT E$ and you will see the characters.

The rest of the machine language code, located on page 6 of memory, is not relocatable. It includes the vertical blank

and explosion subroutines, as well as the data for the player shapes. The main program uses absolute addressing to access this player data; thus this data must reside on page 6 in its proper place. Jump-to-subroutine commands are also used by the main program to access these page 6 routines. These routines must always stay at their proper places or they won't be found by the main program.

There are two USR commands in the program. The one at line 350 jumps to a machine language subroutine that clears the screen memory area between playfields. BASIC was too slow to perform this function without delaying the game considerably, so the machine language routine was developed to clear the display memory area quickly and efficiently.

The second USR command jumps to the main machine language program that moves the players on the screen. This routine also makes all the sounds of the tanks and checks for possible collisions between tanks, missiles, planes, and battlefield objects. When a collision is detected, the main program executes a jump-to-subroutine command to the explosion routine on page 6 of memory.

When you finish a battlefield or your tank is destroyed, the main machine language program returns to BASIC. Then BASIC will either set up a new battlefield (if one was successfully completed) or put you back on the same one for another try.

Tank

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
HD 10 REM TANK
MA 20 DIM E$(1117)
HE 30 POKE 106,PEEK(106)-16:GRAPHICS 18:SETCOL
      OR 4,12,2:SETCOLOR 2,12,2
EL 40 POSITION 8,5:? #6;"TANK":SETCOLOR 1,0,12
      :SETCOLOR 3,3,6
CN 50 RAMTOP=PEEK(106):DLLO=PEEK(560):DLHI=PEE
      K(561):DMLO=PEEK(88):DMHI=PEEK(89):DL=DL
      LO+256*DLHI
CO 60 OCHBASE=256*PEEK(756):CHBASE=RAMTOP*256
IO 70 FOR I=0 TO 511:POKE CHBASE+I,PEEK(OCHBAS
      E+I):NEXT I
MN 80 FOR I=0 TO 63:READ D:POKE CHBASE+8+I,D:N
      EXT I:POKE 756,RAMTOP
```

```
CM 90 DATA 2,201,48,73,74,74,74,49,2,25,100,20
      2,81,81,82,140,0,0,36,24,24,36,0,0,0,0,4
      0,16,40,0,0,0
OG 100 DATA 0,0,0,16,56,16,0,0,0,56,68,84,68,5
       6,0,0,255,129,129,153,153,129,129,255,1
       6,40,84,170,84,40,16,0
EC 110 REM PAGE 6 ROUTINES
MG 120 FOR I=1539 TO 1747:READ D:POKE I,D:NEXT
        I
IC 130 DATA 173,120,2,201,14,208,30,173,0,6,20
       1,50,240,23,206,2,6,208,18,169,2,141,2,
       6,206
NO 140 DATA 1,6,173,1,6,201,255,240,6,141,5,21
       2,76,98,228,238,0,6,169,15,141,1,6,141,
       5
OJ 150 DATA 212,160,0,177,203,56,216,233,20,14
       5,203,160,1,177,203,233,0,145,203,76,40
       ,6,0,252,252
DM 160 DATA 120,120,124,124,126,127,126,124,12
       4,120,120,252,252,0,0,0,63,63,30,30,62,
       62,126,254,126
HP 170 DATA 62,62,30,30,63,63,0,0,0,16,16,16,5
       6,56,186,186,254,254,254,254,254,254,25
       4,254,130
NP 180 DATA 130,0,60,24,24,152,216,254,255,254
       ,216,152,24,24,60,0,0,8,16,74,34,72,68,
       16,197
HL 190 DATA 16,4,162,20,69,18,72,18,8,0,160,18
       ,185,144,6,145,205,136,208,248,142,242,
       6,162,100
CH 200 DATA 160,255,136,208,253,142,6,210,202,
       208,245,169,1,141,30,208,169,0,160,18,1
       45,205,136,208,251
HN 210 DATA 173,242,6,133,205,238,250,6,96
GF 300 PMBASE=(RAMTOP+8)*256:VP0=PMBASE+1024:S
       M=PMBASE+239
KK 310 GOSUB 600
MH 325 REM RESTART HERE
BG 330 POKE 1786,0:DIF=2:POKE 1777,DIF:POKE 17
       75,DIF:E=2:DEN=15:FN=1:TN=2
IO 350 CLEAR=USR(ADR(E$),256*(RAMTOP+4))
KM 355 REM SET UP PLAYFIELD
MF 360 E=E+1:IF E=8 THEN E=3:DEN=20:DIF=DIF-1:
       IF DIF<0 THEN DIF=0
HG 370 TN=TN+1:DEN=DEN+5:POKE 1777,DIF:POKE 17
       75,DIF
KK 380 FOR I=2 TO 50:POKE SM-I*20,68:POKE SM-I
       *20+1,68:NEXT I
AN 390 FOR I=0 TO DEN:SOUND 0,240,10,9:R=INT(R
       ND(0)*950)+40:R1=INT(RND(0)*950)+40
```

```
KD 400 POKE SM-R,E+192:POKE SM-R1,E+65:SOUND 0
       ,0,0,0:NEXT I
PD 410 FOR I=1 TO 15:R=INT(RND(0)*950)+40:SOUN
       D 1,12*I,10,10:POKE SM-R,1:POKE SM-R+1,
       2:NEXT I:SOUND 1,0,0,0
EI 420 POKE SM-15,50:POKE SM-16,33:POKE SM-14,
       52:POKE SM-17,52:POKE SM-18,51
PF 430 POKE SM-1099,38:POKE SM-1098,41:POKE SM
       -1097,46:POKE SM-1096,41:POKE SM-1095,5
       1:POKE SM-1094,40
CO 440 POSITION 1,1:? #6;"(24 SPACES)":REM 23 S
       PACES
JL 450 POSITION 1,1:? #6;"SCORE ";10*PEEK(1786
       ):POSITION 14,1:? #6;"TANKS ";TN
BO 500 D=USR(ADR(E$)+24,DL+7,VP0,PMBASE,SM-239
       )
HO 510 SOUND 2,0,0,0:SOUND 3,0,0,0:SOUND 1,0,0
       ,0:SOUND 0,0,0,0
NG 520 IF PEEK(1536)=50 THEN 550
BK 530 TN=TN-1:IF TN=0 THEN 570
AG 540 POSITION 14,1:? #6;"TANKS ";TN;" ":GOTO
        500
CD 550 FN=FN+1:POSITION 14,1:? #6;" FIELD ";FN
       :IF FN=16 THEN 595
II 560 POSITION 1,1:? #6;"SCORE ";10*PEEK(1786
       ):GOTO 350
DP 570 POSITION 14,1:? #6;"PRESS START"
CI 580 IF PEEK(53279)<>6 THEN 580
GN 590 GOTO 330
FA 595 POSITION 15,1:? #6;"A WINNER":GOTO 580
AN 600 FOR I=1 TO 1117:READ D:E$(I)=CHR$(D):NE
       XT I
LB 610 DATA 104,104,133,204,104,133,203,169,0,
       162,14,160,0,145,203,136,208,251,230,20
       4,202,208,246,96
LL 620 DATA 104,104,133,204,104,133,203,104,13
       3,206,104,133,205,104,141,255,6,104,104
       ,160,1,145,203,104,160
IL 630 DATA 0,145,203,169,0,141,5,212,141,0,6,
       141,1,6,169,2,141,2,6,160,3,162,6,169,7
       ,32,92,228
EO 640 DATA 169,0,141,12,208,169,62,141,47,2,1
       69,3,141,29,208,173,255,6,141,7,212,169
       ,1,141,8
HE 650 DATA 208,141,9,208,141,10,208,141,11,20
       8,169,116,141,192,2,169,38,141,193,2,16
       9,52,141,194,2
DO 660 DATA 169,52,141,195,2,72,104,169,0,133,
       205,133,207,141,144,6,165,206,56,233,1,
       133,208,169,0
```

```
AK 670 DATA 141,249,6,141,1,208,141,2,208,141,
       244,6,141,252,6,141,247,6,141,253,6,169
       ,255,141,243
DN 680 DATA 6,141,3,208,141,30,208,169,6,141,2
       37,6,160,0,169,0,145,207,145,205,136,20
       8,249,230,206
LI 690 DATA 238,144,6,173,144,6,201,4,208,233,
       198,206,198,206,198,206,198,206,169,201
       ,133,205,169,100,141
KL 700 DATA 254,6,141,0,208,173,120,2,201,15,2
       40,25,169,255,141,0,210,169,40,141,1,21
       0,173,120,2
FP 710 DATA 201,7,240,11,201,11,240,43,201,14,
       240,78,24,144,70,169,1,141,253,6,160,18
       ,185,74,6
ND 720 DATA 145,205,136,208,248,173,254,6,201,
       190,240,231,238,254,6,173,254,6,141,0,2
       08,24,144,36,240
LI 730 DATA 185,169,2,141,253,6,160,18,185,92,
       6,145,205,136,208,248,173,254,6,201,50,
       240,195,206,254
DN 740 DATA 6,173,254,6,141,0,208,24,144,0,24,
       144,19,240,215,169,3,141,253,6,160,19,1
       85,110,6
LA 750 DATA 145,205,136,208,248,240,0,173,252,
       6,201,0,240,12,201,1,240,69,201,2,240,8
       3,201,3,240
HB 760 DATA 59,173,132,2,201,0,240,4,208,88,24
       0,207,173,253,6,141,252,6,169,50,141,14
       3,6,141,4
AA 770 DATA 210,169,200,141,5,210,173,254,6,24
       ,109,237,6,141,251,6,141,4,208,165,205,
       24,105,9,133
AP 780 DATA 207,169,3,160,0,145,207,24,144,178
       ,240,41,238,251,6,238,251,6,173,251,6,1
       41,4,208,201
FB 790 DATA 192,16,49,48,59,206,251,6,206,251,
       6,173,251,6,141,4,208,201,60,48,31,16,4
       1,24,144
OG 800 DATA 85,240,163,169,0,160,0,145,207,169
       ,3,198,207,198,207,198,207,145,207,165,
       207,201,42,240,2
DM 810 DATA 208,12,169,0,160,0,145,207,141,252
       ,6,24,144,47,173,8,208,201,2,16,4,48,38
       ,240,207
NA 820 DATA 169,0,160,0,145,207,141,252,6,169,
       47,141,7,210,169,136,141,6,210,173,8,20
       8,201,3,240
```

```
NA 830 DATA 13,201,5,240,36,201,9,240,48,208,0
       ,24,144,67,166,205,173,246,6,133,205,23
       0,206,169,0
BF 840 DATA 141,8,208,169,3,141,237,6,32,163,6
       ,198,206,24,144,38,230,206,230,206,166,
       205,32,163,6
LM 850 DATA 198,206,198,206,24,144,95,230,206,
       230,206,230,206,166,205,32,163,6,198,20
       6,198,206,198,206,24
PE 860 DATA 144,96,240,150,144,94,173,247,6,41
       ,1,208,66,173,10,210,201,1,240,2,208,11
       2,173,10,210
NK 870 DATA 201,100,16,2,48,103,238,247,6,230,
       206,166,205,169,1,141,8,208,169,6,141,2
       37,6,173,10
EB 880 DATA 210,201,30,48,249,133,205,141,246,
       6,160,13,185,129,6,145,205,136,208,248,
       134,205,198,206,24
LJ 890 DATA 144,57,144,85,174,249,6,232,224,22
       5,176,13,142,249,6,142,1,208,24,144,38,
       240,160,144,127
DN 900 DATA 206,247,6,166,205,173,246,6,133,20
       5,230,206,169,0,141,249,6,141,1,208,141
       ,246,6,160,17
PP 910 DATA 145,205,136,208,251,134,205,198,20
       6,234,173,240,6,201,0,208,24,173,241,6,
       141,240,6,173,247
FH 920 DATA 6,41,2,208,55,173,10,210,201,2,240
       ,10,208,79,144,63,206,240,6,24,144,71,1
       73,10,210
EM 930 DATA 201,125,16,2,48,62,238,247,6,238,2
       47,6,230,206,230,206,160,18,185,74,6,14
       5,205,136,208
NN 940 DATA 248,198,206,198,206,24,144,35,240,
       142,174,244,6,232,224,230,176,11,142,24
       4,6,142,2,208,24
KE 950 DATA 144,16,144,114,169,0,141,244,6,141
       ,2,208,206,247,6,206,247,6,173,238,6,20
       1,0,208,22
EN 960 DATA 173,239,6,141,238,6,173,247,6,41,4
       ,208,63,173,10,210,201,3,240,8,208,91,2
       06,238,6
NI 970 DATA 24,144,85,173,10,210,201,225,16,2,
       48,76,238,247,6,238,247,6,238,247,6,238
       ,247,6,230
LO 980 DATA 206,230,206,230,206,160,18,185,92,
       6,145,205,136,208,248,198,206,198,206,1
       98,206,24,144,39,240
```

```
NH 990 DATA 138,174,243,6,202,224,40,144,9,142
      ,243,6,142,3,208,24,144,20,169,225,141,
      243,6,141,3
MH 1000 DATA 208,206,247,6,206,247,6,206,247,6
      ,206,247,6,173,12,208,201,0,208,23,173
      ,4,208,201,0
BO 1010 DATA 208,16,173,14,208,201,0,208,9,173
      ,15,208,201,0,208,2,240,36,169,207,141
      ,3,210,169,102
LC 1020 DATA 141,2,210,166,205,32,163,6,162,0,
      160,255,136,208,253,202,208,248,169,0,
      141,3,210,206,250
HC 1030 DATA 6,96,240,151,162,7,160,255,136,20
      8,253,202,208,248,169,0,141,8,210,238,
      143,6,173,143,6
BM 1040 DATA 141,4,210,160,255,136,208,253,173
      ,143,6,201,255,208,5,169,0,141,5,210,1
      69,0,141,1,210
NO 1050 DATA 141,7,210,173,0,6,201,50,240,4,16
      9,0,240,194,96
BC 1100 REM DISPLAY LIST DATA
JP 1110 FOR I=0 TO 21:READ D:POKE DL+I,D:NEXT
      I
EJ 1120 POKE DL+8,PMBASE/256:POKE DL+4,DMLO+20
      :POKE DL+5,DMHI:POKE DL+22,DLLO:POKE D
      L+23,DLHI:RETURN
DN 1130 DATA 112,112,112,66,0,0,103,0,0,39,39,
      39,39,39,39,39,39,39,39,7,65
```

Chapter 2

Dots

Eric Saper

"Dots" is a computer version of the popular pencil-and-paper strategy game that kids play. It's designed for two players; joysticks are required.

If you haven't played dots before, you will probably think there isn't much to the game. The rules are few and simple. Two players take turns drawing horizontal or vertical lines between two adjacent dots on the playing field. The object of the game is to close off more boxes than your opponent. When you close a box, it's identified with your color. Sound simple? It is. But there is a lot of strategy involved. Toward the end of the game, a wrong move may cost you the game.

You have to be careful not to draw the third of four lines needed to complete a particular box, because your opponent can then win that box by drawing the final line. Sooner or later, you will have to give away boxes because there will be no more free space available.

Whenever a player closes a box, he goes again. That makes it possible to win several boxes in one turn. Toward the end of the game, when there are no free spaces left, it's important to choose lines that will give your opponent the fewest boxes. But you must be careful, because giving the fewest is not necessarily the best.

How Big?

The program requires about 8K to load and from 10K to 12K to run, depending on the dimensions of the playing field. It also requires two joysticks.

When you run the program, you will be asked to supply the horizontal and vertical dimensions of the playing field. The smallest either dimension can be is 5; the largest is 20. A 5 × 5 board (16 boxes) will take only a few minutes to play, while a 20 × 20 board (361 boxes) may require an hour or two. A good-sized board is 10 × 10 (81 boxes) or 12 × 12 (121 boxes). The dimensions do not have to be the same, although it is usually preferred.

After dimensions have been specified, the program draws the board and puts a text window at the bottom of the screen. The information in the window gives each player's score and the number of remaining boxes. An asterisk will indicate each player's turn.

Assume you're player 1 and that it's your turn. Press your fire button; the text window disappears and the screen colors change. One of the dots will be white; that's the cursor dot. Push your joystick and the cursor dot will move left-right or up-down.

Move the cursor to one end of the line you want to draw. Press your fire button and the cursor will turn red, indicating that you are ready to make your move. With the cursor red, push your joystick in the direction you wish the line to be drawn, relative to the cursor. If you decide not to draw from that point, press your button again and the cursor will turn white once more.

When you draw the line, a sound will be heard. If you try to draw your line over another line or off the playing field, a buzzer will sound and you can try again. When you close a box, it will fill in with your color. Player 1's boxes will be white, while player 2's boxes will be red.

After every move, the text window will reappear. Remember, if you have just gained a box, it will still be your turn.

At the end of the game the final scores are displayed, with flashing stars surrounding the winner's score. If it is a tie game, both scores will be surrounded. If you want to play again, press the START button.

Dots

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
NE 10 ? "{CLEAR}":POSITION 16,8:? "DOTS":POSIT
      ION 2,14:? "WHAT ARE THE MAXIMUM DIMENSI
      ONS{2 TAB}(5,5)-(20,20)";:INPUT XM,YM
BM 20 IF XM<5 OR XM>20 OR YM<5 OR YM>20 THEN 1
      0
GM 30 DIM A$(4),P(XM,YM),S(1)
PG 40 FOR I=0 TO XM:FOR J=0 TO YM:P(I,J)=0:NEX
      T J:NEXT I
EH 50 X0=INT(80/(XM+1)):Y0=INT(48/(YM+1)):T0=(
      XM-1)*(YM-1)
EB 60 XS=(80-XM*X0+X0)/2-X0-1:YS=(48-YM*Y0+Y0)
      /2-Y0-1
AH 70 GRAPHICS 21:SETCOLOR 0,0,0:SETCOLOR 1,0,
      14:SETCOLOR 4,7,4
HI 80 COLOR 1:FOR X=1 TO XM:FOR Y=1 TO YM
DK 90 PLOT X*X0+XS,Y*Y0+YS:NEXT Y:NEXT X
HK 100 COLOR 2:PLOT X0+XS,Y0+YS:COLOR 1:X=0:Y=
       0:S=0:T=0:S(0)=0:S(1)=0
```

```
KO 110 GRAPHICS 37:POKE 752,1:IF S=0 THEN ? "*
       PLAYER 1 HAS ";S(0):? " PLAYER 2 HAS ";
       S(1):GOTO 130
NB 120 ? " PLAYER 1 HAS ";S(0):? "*PLAYER 2 HA
       S ";S(1)
GD 130 ? :? "BOXES REMAINING = ";T0-T;
JC 135 POKE 77,0
ID 140 IF STRIG(S)=1 THEN 140
IM 145 IF STRIG(S)=0 THEN 145
EO 150 ? :GRAPHICS 53:SETCOLOR 0,0,0:SETCOLOR
       1,0,14:SETCOLOR 2,3,8:SETCOLOR 4,7,4
AO 155 COLOR 2:PLOT (X+1)*X0+XS,(Y+1)*Y0+YS
PF 160 M=STICK(S):X1=X:Y1=Y
IG 170 IF STRIG(S)=0 THEN 330
BG 180 IF M<>14 THEN 210
DO 190 Y=Y-1:IF Y<0 THEN Y=YM-1
GG 200 GOTO 290
OF 210 IF M<>7 THEN 240
DD 220 X=X+1:IF X>XM-1 THEN X=0
GJ 230 GOTO 290
BI 240 IF M<>13 THEN 270
DL 250 Y=Y+1:IF Y>YM-1 THEN Y=0
GM 260 GOTO 290
BH 270 IF M<>11 THEN 160
DJ 280 X=X-1:IF X<0 THEN X=XM-1
CL 290 SOUND 0,100,10,10
GH 300 COLOR 1:PLOT (X1+1)*X0+XS,(Y1+1)*Y0+YS
AH 310 COLOR 2:PLOT (X+1)*X0+XS,(Y+1)*Y0+YS
AC 320 SOUND 0,0,0,0:FOR I=1 TO 30:NEXT I
IE 321 IF STRIG(S)=0 THEN 321
GH 322 GOTO 160
PK 330 SOUND 0,50,10,10
AL 340 COLOR 3:PLOT (X+1)*X0+XS,(Y+1)*Y0+YS
GF 350 SOUND 0,0,0,0
IK 360 IF STRIG(S)=0 THEN 360
PI 370 M=STICK(S):X=X1:Y=Y1
IH 380 IF STRIG(S)=0 THEN 310
BM 390 IF M<>14 THEN 420
DE 400 Y=Y-1:IF Y<0 THEN 510
FE 410 C=4:GOTO 530
OL 420 IF M<>7 THEN 450
AH 430 X=X+1:IF X>XM-1 THEN 510
FG 440 C=3:GOTO 530
BO 450 IF M<>13 THEN 480
AO 460 Y=Y+1:IF Y>YM-1 THEN 510
FI 470 C=2:GOTO 530
BN 480 IF M<>11 THEN 370
DK 490 X=X-1:IF X<0 THEN 510
FB 500 C=1:GOTO 530
```

```
ND 510 SOUND 0,60,12,10:FOR I=1 TO 200:NEXT I:
       SOUND 0,0,0,0
GK 520 GOTO 370
HB 530 A$="0000":A$(5-LEN(STR$(INT(P(X1,Y1))))
       ,4)=STR$(INT(P(X1,Y1)))
AH 540 IF A$(C,C)="1" THEN 510
KE 550 A$(C,C)="1":P(X1,Y1)=VAL(A$)
DN 560 IF C<3 THEN C=2-C:GOTO 580
MC 570 C=6-C
PJ 580 P(X,Y)=P(X,Y)+INT(10^C+0.1)
MG 590 COLOR 1:PLOT (X1+1)*X0+XS,(Y1+1)*Y0+YS:
       DRAWTO (X+1)*X0+XS,(Y+1)*Y0+YS
AF 600 FOR I=100 TO 0 STEP -5
HJ 610 SOUND 0,I,2,10:NEXT I:SOUND 0,0,0,0
PG 620 B=0:IF X=X1 THEN 720
LK 630 IF Y=0 THEN 670
CH 640 K=0:C=4:ZX=X1:ZY=Y1:GOSUB 810:ZX=X:ZY=Y
       :GOSUB 810:IF X1<X THEN C=3:GOTO 660
EM 650 C=1
DD 660 ZX=X1:ZY=Y1-1:GOSUB 810:IF K=3 THEN C=1
       :GOSUB 840
FL 670 IF Y=YM THEN 1030
CE 680 K=0:C=2:ZX=X1:ZY=Y1:GOSUB 810:ZX=X:ZY=Y
       :GOSUB 810:IF X1<X THEN C=3:GOTO 700
FA 690 C=1
CN 700 ZX=X1:ZY=Y1+1:GOSUB 810:IF K=3 THEN C=2
       :GOSUB 840
JF 710 GOTO 1030
LJ 720 IF X=0 THEN 760
CF 730 K=0:C=1:ZX=X1:ZY=Y1:GOSUB 810:ZX=X:ZY=Y
       :GOSUB 810:IF Y1<Y THEN C=2:GOTO 750
EP 740 C=4
DF 750 ZX=X1-1:ZY=Y1:GOSUB 810:IF K=3 THEN C=3
       :GOSUB 840
FJ 760 IF X=XM THEN 1030
CP 770 K=0:C=3:ZX=X1:ZY=Y1:GOSUB 810:ZX=X:ZY=Y
       :GOSUB 810:IF Y1<Y THEN C=2:GOTO 790
FD 780 C=4
DI 790 ZX=X1+1:ZY=Y1:GOSUB 810:IF K=3 THEN C=4
       :GOSUB 840
JF 800 GOTO 1030
BG 810 A$="0000":A$(5-LEN(STR$(INT(P(ZX,ZY))))
       ,4)=STR$(INT(P(ZX,ZY)))
KB 820 IF A$(C,C)="1" THEN K=K+1
HL 830 RETURN
NI 840 B=1:S(S)=S(S)+1:T=T+1:COLOR S+2:POKE 76
       5,S+2
FA 850 FOR I=0 TO 200 STEP 10:SOUND 0,I,10,10:
       NEXT I
```

```
HO 860 FOR I=200 TO 0 STEP -10:SOUND 0,I,10,10
   :NEXT I
LD 870 ON C GOTO 880,910,940,970
AD 880 IF X1<X THEN Z=X1+1:GOTO 900
OM 890 Z=X+1
LG 900 PX=Z*X0+XS+1:PY=Y*Y0+YS+1:PLOT PX,PY:PO
   SITION PX,(Y+1)*Y0+YS-1:GOTO 1000
AA 910 IF X1<X THEN Z=X1+1:GOTO 930
OG 920 Z=X+1
GH 930 PX=Z*X0+XS+1:PY=(Y+1)*Y0+YS+1:PLOT PX,P
   Y:POSITION PX,(Y+2)*Y0+YS-1:GOTO 1000
AJ 940 IF Y1<Y THEN Z=Y1+1:GOTO 960
OK 950 Z=Y+1
LM 960 PX=X*X0+XS+1:PY=Z*Y0+YS+1:PLOT PX,PY:PO
   SITION PX,(Z+1)*Y0+YS-1:GOTO 1000
AP 970 IF Y1<Y THEN Z=Y1+1:GOTO 990
ON 980 Z=Y+1
DI 990 PX=(X+1)*X0+XS+1:PY=Z*Y0+YS+1:PLOT PX,P
   Y:POSITION PX,(Z+1)*Y0+YS-1
KK 1000 XIO 18,#6,0,0,"S":PLOT PX,PY:DRAWTO PX
   +X0-2,PY
LJ 1010 IF T0-T=0 THEN 1060
KD 1020 RETURN
ME 1030 IF B=1 THEN 110
JJ 1040 S= NOT S
JB 1050 GOTO 110
IB 1060 GRAPHICS 18
LC 1070 READ N,T
DD 1080 IF N=-1 THEN 1110
OE 1090 SOUND 0,N,10,10:SOUND 1,N+2,10,10:SOUN
   D 2,N+4,10,10
OA 1100 FOR I=1 TO 20*T:NEXT I:GOTO 1070
KB 1110 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,0,
   0,0
BM 1120 POSITION 3,1:? #6;"final score is"
AH 1130 POSITION 2,5:? #6;"PLAYER 1 HAS ";S(0)
AN 1140 POSITION 2,8:? #6;"PLAYER 2 HAS ";S(1)
OK 1150 IF S(0)>=S(1) THEN P=5:GOSUB 1220
OM 1160 IF S(0)<=S(1) THEN P=8:GOSUB 1220
BO 1170 FOR I=1 TO 50:IF PEEK(53279)=6 THEN 12
   10
OB 1180 NEXT I:SETCOLOR 2,0,0
CA 1190 FOR I=1 TO 50:IF PEEK(53279)=6 THEN 12
   10
CD 1200 NEXT I:SETCOLOR 2,9,4:GOTO 1170
JO 1210 CLR :RESTORE 3130:GOTO 10
DD 1220 POSITION 0,P-1:? #6;"■■■■■■■■■■■■■■■■■■■■
   ■■■■":POSITION 0,P:? #6;"■"
```

```
GM 1230 POSITION 19,P:? #6;"+":POSITION Ø,P+1:
        ? #6;"++++++++++++++++++++"
KH 1240 RETURN
GE 2075 FOR I=1 TO 3ØØ:NEXT I
GC 3130 DATA 162,1,144,1,128,1,144,2,128,1,162
        ,3,81,3,-1,Ø
```

Chapter 2

Reversi

José R. Suárez

"What's this?" you say. "Not another Reversi program!" Yes, it is—but this one features full-color graphics, playing chips that flip over right before your eyes, replay options, and a computer opponent that will truly make you think. The program requires 24K memory, 32K if you use a disk drive.

"Reversi" is played on an 8 × 8 grid. The playing pieces are two-sided disks, black on one side and white on the other. Initially, four pieces are placed on the four center squares, two showing black and two showing white, in an X pattern. Black starts, and the object is to capture as many of the other player's pieces as possible.

You do this by bracketing one or more of your opponent's chips with your chips, and then flipping them over to your color. This can be done in any direction—vertically, horizontally, or diagonally.

A move is legal only if it flips one or more of the other player's pieces. If you have no legal moves, you must pass. The game ends when the board is full or when no more moves are possible. At that time, the player with the most pieces wins.

Joysticks or Keyboard?

Before displaying the game board, the program asks if you want to use joysticks or the keyboard to enter moves. Joysticks (plugged into ports 1 and 2) are the better choice, especially for a two-player game. But for keyboard fans, traditional row-column entry is also available.

If you opt for the keyboard, enter a letter and a number for each move. You may enter either the letter or the number first; the computer will figure out what you mean. To pass, enter the letter P instead of coordinates.

With joysticks, move the cross-shaped cursor to the square you want to capture; then press the trigger. To pass, move the cursor over the word PASS (at the right side of the board) and press the trigger. The black pieces are *always* moved by the joystick in port 1; the white pieces are *always* moved by the stick in port 2. Remember this when playing against the computer.

One or Two Players

Once you select your preferred entry method, there is a short wait while the computer initializes the game board. When the board appears, press SELECT to choose a one- or two-player game, and use the OPTION key to toggle the color of the computer's pieces. Press START to begin the game.

The computer acts as referee and piece-flipper whether you play against it or against another player. It does not allow illegal moves or passes. After a game is completed, the totals are displayed, and the winner declared.

You may then review the game just played by engaging one of the two replay options. AUTOMATIC REPLAY shows you the game over, move by move; just sit back and watch as it develops. MANUAL REPLAY does much the same thing, except that after each move the computer will pause until you press the trigger (if using joysticks) or a key (if using the keyboard). To choose a replay option, press SELECT to cycle through the choices until you reach the option you want. Then press START. These functions should help you locate poor moves and improve your game; you can review the game as many times as you desire.

Setting Things Up

The program begins with a jump to line 14000. Lines 14000–14030 initialize the important variables and tie together all the initialization subroutines. Lines 9000–9110 are the title display, and lines 9500–9600 display the 10- or 14-second board-preparation message.

Lines 1000–1140 then move and modify the character set. Take a look at the expression in line 1005. Variable FAC is set to 0 if you chose the keyboard and to 255 if you chose joysticks. This means that the character set is moved in complementary form (inverse video) if you choose joysticks. This unorthodox font permits some interesting and pleasing graphics.

If joysticks are selected, then player/missile graphics are handled next at lines 2500–2540. All four players and one missile are used. Player 0 is the cursor. Player 1 is the word PASS mapped directly from the character generator. Player 2 is the blue field. Player 3 and missile 3 border the playfield so that no green background shows except on the board.

The board is sent to the screen by lines 15000–15100. All the characters embedded in those lines were redefined with

box-making control characters normally unavailable in GRAPHICS 1 and 2. They form a nice grid when printed out. Lines 15060–15070 print the coordinate system for the keyboard user.

With initialization completed, control passes to lines 8540–8670 where the console keys are read and the selected options displayed. Once the player presses START, the main loop at lines 8000–8430 takes over. This master loop has control over each full game. It calls the scanning and piece-flipping module (1600–1720); the large network of computer-intelligence subroutines (3300–5420); the animation subroutine (6990–6995); score keeping and move storing (3000–3150); illegal-move correction (7100–7130); and message printing (7200–7220).

Artificial Intelligence

You will find that the computer is a challenging opponent. The game board is kept internally as array TBL. The computer assigns a strategic value to each square on the board, and it keeps those values in array STR. They are not static values, however, for they change as the game progresses. The value of a prospective move is based largely on the dynamic value of the square, and (to a much smaller degree) on the number of pieces flipped. Corners and edges have the highest values, while the adjacent squares forming a bridge to these have the lowest values.

The computer is very careful when moves are made on the edges, calling special subroutines to handle them. The value of the number of pieces flipped jumps drastically during the last few moves of the game—and that's when many games are won or lost.

Flipping the pieces smoothly adds to the attractiveness of the display and is quite simply done. Take a look at line 15010. ANIM$ is filled with a series of control characters. These characters have been redefined so that each is a frame of the flipping action. The characters so reshaped are now ellipsoids with progressively shorter minor axes—two of them are just horizontal lines. When these characters are printed out rapidly at the same screen location, they make a little movie of a flipping chip. Color switching is accomplished by using the inverse-video incarnations of the same characters.

Reversi

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
IP 10 GOTO 14000
PN 1000 POKE 106,PEEK(106)-C4:GRAPHICS 17:POKE
    53774,112:POKE 16,64:GOSUB 9500:I=(PE
   EK(106)+C2)*256
OD 1005 FOR J=0 TO 511:POKE I+J,ABS(FAC-PEEK(5
   7344+J)):NEXT J
ED 1010 SET=I:GOSUB 9600:TRAP 1130:RESTORE 101
   00
EI 1020 READ LIST1,LIST2:FOR J=0 TO C7:BYTE=PE
   EK(LIST1+J+57344):POKE LIST2+J+I,BYTE:
   NEXT J:GOTO 1020
HH 1130 RESTORE 10110:FOR J=0 TO C7:READ BYTE:
   POKE 80+J+I,BYTE:READ BYTE:POKE C8+J+I
   ,BYTE:POKE 480+J+I,0:NEXT J
OF 1140 FOR K=C1 TO C5:READ LIST1:FOR J=0 TO C
   7:READ BYTE:POKE LIST1+J+I,BYTE:NEXT J
   :NEXT K:RETURN
JC 1600 FLIP=0:OFF=C9*(NOW=C1):FOR VD=-C1 TO C
   1:FOR HD=-C1 TO C1:IF VD=0 AND HD=0 TH
   EN NEXT HD
GO 1610 TRAP 1720:X=J:Y=I:DX=X:DY=Y:CFL=0
GC 1630 IF TBL(Y+VD,X+HD)=OTHER THEN CFL=C1:X=
   X+HD:Y=Y+VD:GOTO 1630
LE 1640 TRAP 40000:IF TBL(Y+VD,X+HD)<>NOW OR C
   FL=0 THEN 1720
LM 1670 IF PASS=C1 THEN FLIP=C1:POP :POP :RETU
   RN
CH 1680 DX=DX+HD:DY=DY+VD:TBL(DY,DX)=NOW:FOR L
   OOP=C1+OFF TO C9+OFF:K=C1/C1/C1
IF 1690 POSITION DX*C2+C2,DY*C2+C2:? #S;ANIM$(
   LOOP,LOOP):NEXT LOOP:SOUND 0,60,12,C8:
   FLIP=FLIP+C1
FH 1700 SOUND 0,0,0,0:IF DX<>X OR DY<>Y THEN 1
   680
MC 1720 NEXT HD:NEXT VD:RETURN
NL 1740 FOR I=C2 TO 16 STEP C2:POSITION C2,I:?
    #S;"[reverse]            ":NEXT I
AN 1750 NOW=C1:OTHER=C2:PL1$="{J}black{,}move
   {Z}":PL2$="{J}White{,}move{Z}"
GM 1752 RESTORE 10000:FOR I=0 TO C3:FOR J=0 TO
    C3:READ K:STR(I,J)=K:STR(C7-I,J)=K:ST
   R(I,C7-J)=K:STR(C7-I,C7-J)=K
EK 1754 TBL(I,J)=0:TBL(C7-I,J)=0:TBL(I,C7-J)=0
   :TBL(C7-I,C7-J)=0:NEXT J:NEXT I:HOR=C4
   :VER=C4
```

```
LP 1760 POSITION C8,C8:? #S;"{■}◪{J}":POSITION
        C8,10:? #S;"{J}◪{■}":BSCR=C2:WSCR=C2
KH 1770 TBL(C3,C3)=OTHER:TBL(C4,C4)=OTHER:TBL(
        C3,C4)=NOW:TBL(C4,C3)=NOW:RETURN
IL 2000 RESTORE 10200:FOR I=0 TO C1:FOR J=C5 T
        O 15:READ K:DELTA(J,I)=K:NEXT J:NEXT I
        :RETURN
NF 2500 PMBASE=PEEK(106)-C8:POKE 54279,PMBASE:
        POKE 53277,C3:P0=PMBASE*256+512:P1=P0+
        128:P2=P1+128:P3=P2+128
ND 2510 MIS=P0-128:POKE 559,46:K=39:POKE 623,C
        2:RESTORE 2530
EG 2520 FOR I=C1 TO C4:READ LIST1:FOR J=0 TO S
        :POKE J+P1+K,PEEK(57344+LIST1+J):NEXT
        J:K=K+C7:NEXT I:POKE 53258,C1
NL 2530 DATA 384,264,408,408
JF 2540 FOR I=38 TO 69:POKE P2+I,255:NEXT I:FO
        R I=0 TO 119:POKE P3+I,255:POKE MIS+I,
        192:NEXT I:POKE 53260,192:RETURN
KF 2600 FLIP=0:FOR VD=-C1 TO C1:FOR HD=-C1 TO
        C1:IF VD=0 AND HD=0 THEN NEXT HD
DF 2610 TRAP 2650:X=J:Y=I:FL=0:CFL=FL
GG 2630 IF TBL(Y+VD,X+HD)=OTHER THEN X=X+HD:Y=
        Y+VD:FL=FL+C1:GOSUB 2660:GOTO 2630
FA 2640 TRAP 40000:IF TBL(Y+VD,X+HD)=NOW THEN
        FLIP=FLIP+FL:IF CFL THEN DFL=C1
MF 2650 NEXT HD:NEXT VD:RETURN
DF 2660 IF STR(Y,X)=-200 THEN CFL=C1
KP 2670 RETURN
OE 3000 SFL=0:IF NOW=C1 THEN BSCR=BSCR+FLIP+C1
        :WSCR=WSCR-FLIP:GOTO 3145
JH 3010 WSCR=WSCR+FLIP+C1:BSCR=BSCR-FLIP
CA 3145 IF BSCR=0 OR WSCR=0 OR BSCR+WSCR=64 TH
        EN SFL=C2
BJ 3150 GAME(M)=I+J/C10:M=M+C1:RETURN
KK 3200 POSITION 0,23:? #S;"■EVALUATING PASS
        {3 SPACES}";
GD 3203 FLIP=0:FOR I=0 TO C7:FOR J=0 TO C7:IF
        TBL(I,J)=0 THEN GOSUB 1600
CN 3205 IF FLIP>0 THEN POP :POP :PASS=0:GOSUB
        7220:GOSUB 3220:GOTO MAINLOOP
JD 3210 NEXT J:NEXT I:PASS=0:GOSUB 7220
AE 3212 POKE 705,42:LINE$="{8 SPACES}PASS  ":I
        F NOW=C1 THEN LINE$="{8 SPACES}pass  "
PA 3214 DF=C10:GOSUB 7200:GAME(M)=-C1:M=M+C1:S
        FL=SFL+C1:HOR=C4:VER=C4:POKE 705,144:I
        F SFL=C2 THEN 8500
NB 3216 GOTO 8400
```

```
DG 3220 LINE$="{3 SPACES}you{,}must{,}play.":IF
        NOW=C1 THEN LINE$="{3 SPACES}you{,}mu
        st{,}play"
NC 3230 DF=C12:GOSUB 7200:RETURN
AB 3300 FLIP=0:MAX=-210:FOR I=0 TO C7:FOR J=0
        TO C7:IF TBL(I,J)=0 THEN GOSUB 2600:GO
        SUB 3500
CM 3400 NEXT J:NEXT I:RETURN
BN 3500 IF FLIP=0 THEN RETURN
LB 3510 IF STR(I,J)=-70 OR STR(I,J)=45 THEN GO
        SUB 3600+I*C10+J
LJ 3512 IF BSCR+WSCR>59 THEN K=FLIP:GOTO 3520
NO 3515 K=FLIP*C5+STR(I,J):IF DFL THEN DFL=0:I
        F STR(I,J)<>500 THEN K=-200
LM 3520 IF K>MAX THEN MAX=K:HOR=J:VER=I
CI 3525 IF K=MAX THEN IF RND(0)<0.4 THEN HOR=J
        :VER=I
KL 3530 RETURN
II 3601 X=C2:HD=C1:GOTO 3700
LN 3606 X=C5:HD=-C1:GOTO 3700
JM 3610 Y=C2:VD=C1:GOTO 3750
KD 3617 Y=C2:VD=C1:GOTO 3750
NB 3660 Y=C5:VD=-C1:GOTO 3750
NI 3667 Y=C5:VD=-C1:GOTO 3750
IP 3671 X=C2:HD=C1:GOTO 3700
ME 3676 X=C5:HD=-C1:GOTO 3700
MD 3700 IF TBL(I,X)=0 AND TBL(I,X+HD)=NOW THEN
         STR(I,J)=-70:RETURN
NK 3710 TRAP 3730:IF TBL(I,X)=NOW THEN X=X+HD:
        GOTO 3710
KC 3720 TRAP 40000:IF TBL(I,X)=OTHER THEN STR(
        I,J)=-70:RETURN
JG 3730 STR(I,J)=45:RETURN
NK 3750 IF TBL(Y,J)=0 AND TBL(Y+VD,J)=NOW THEN
         STR(I,J)=-70:RETURN
PL 3760 TRAP 3780:IF TBL(Y,J)=NOW THEN Y=Y+VD:
        GOTO 3760
KJ 3770 TRAP 40000:IF TBL(Y,J)=OTHER THEN STR(
        I,J)=-70:RETURN
JL 3780 STR(I,J)=45:RETURN
MD 3800 IF VER+C2<C7 THEN IF STR(VER+C2,HOR)>0
         THEN STR(VER+C2,HOR)=-55
IC 3810 IF VER-C2>0 THEN IF STR(VER-C2,HOR)>0
        THEN STR(VER-C2,HOR)=-55
GE 3820 IF STR(VER+C1,HOR)=-55 THEN STR(VER+C1
        ,HOR)=80:GOSUB 3850
GL 3830 IF STR(VER-C1,HOR)=-55 THEN STR(VER-C1
        ,HOR)=80:GOSUB 3870
KP 3840 RETURN
```

```
PN 3850 TRAP 3860:IF TBL(VER+C1,HOR)=0 AND TBL
        (VER+C2,HOR)=0 AND TBL(VER+C3,HOR)=NOW
         THEN STR(VER+C1,HOR)=-55
BG 3860 TRAP 40000:RETURN
AJ 3870 TRAP 3880:IF TBL(VER-C1,HOR)=0 AND TBL
        (VER-C2,HOR)=0 AND TBL(VER-C3,HOR)=NOW
         THEN STR(VER-C1,HOR)=-55
BI 3880 TRAP 40000:RETURN
MA 3900 IF HOR+C2<C7 THEN IF STR(VER,HOR+C2)>0
         THEN STR(VER,HOR+C2)=-55
JO 3910 IF HOR-C2>0 THEN IF STR(VER,HOR-C2)>0
        THEN STR(VER,HOR-C2)=-55
GG 3920 IF STR(VER,HOR+C1)=-55 THEN STR(VER,HO
        R+C1)=80:GOSUB 3950
GN 3930 IF STR(VER,HOR-C1)=-55 THEN STR(VER,HO
        R-C1)=80:GOSUB 3970
LA 3940 RETURN
PP 3950 TRAP 3960:IF TBL(VER,HOR+C1)=0 AND TBL
        (VER,HOR+C2)=0 AND TBL(VER,HOR+C3)=NOW
         THEN STR(VER,HOR+C1)=-55
BH 3960 TRAP 40000:RETURN
AL 3970 TRAP 3980:IF TBL(VER,HOR-C1)=0 AND TBL
        (VER,HOR-C2)=0 AND TBL(VER,HOR-C3)=NOW
         THEN STR(VER,HOR-C1)=-55
BJ 3980 TRAP 40000:RETURN
AN 4000 IF HOR>0 AND HOR<C7 AND VER>0 AND VER<
        C7 THEN RETURN
PO 4010 IF HOR=0 OR HOR=C7 THEN 4100
DA 4020 IF TBL(VER,HOR+C1)=NOW THEN HD=-C1:GOT
        O 4600
AK 4025 IF TBL(VER,HOR-C1)=NOW THEN HD=C1:GOTO
         4600
EJ 4030 K=STR(VER,HOR+C1):IF K<500 AND K<>-70
        AND K<>-55 THEN STR(VER,HOR+C1)=-K
GE 4035 IF K=-55 THEN STR(VER,HOR+C1)=-60
EO 4040 K=STR(VER,HOR-C1):IF K<500 AND K<>-70
        AND K<>-55 THEN STR(VER,HOR-C1)=-K
GH 4045 IF K=-55 THEN STR(VER,HOR-C1)=-60
CK 4050 GOSUB 5110:RETURN
FB 4100 IF VER=0 OR VER=C7 THEN K=0:GOTO 4200
DN 4110 IF TBL(VER+C1,HOR)=NOW THEN VD=-C1:GOT
        O 4500
BH 4115 IF TBL(VER-C1,HOR)=NOW THEN VD=C1:GOTO
         4500
EJ 4120 K=STR(VER+C1,HOR):IF K<500 AND K<>-70
        AND K<>-55 THEN STR(VER+C1,HOR)=-K
GE 4125 IF K=-55 THEN STR(VER+C1,HOR)=-60
EO 4130 K=STR(VER-C1,HOR):IF K<500 AND K<>-70
        AND K<>-55 THEN STR(VER-C1,HOR)=-K
```

```
GH 4135 IF K=-55 THEN STR(VER-C1,HOR)=-60
CM 4140 GOSUB 5310:RETURN
NF 4200 STR(VER,ABS(HOR-C1))=K:STR(ABS(VER-C1)
        ,HOR)=K:STR(ABS(VER-C1),ABS(HOR-C1))=K
        -30-200*(K=300):RETURN
DN 4500 K=VER+VD:TRAP 4140
KP 4510 IF TBL(K,HOR)=OTHER THEN K=K+VD:GOTO 4
        510
CK 4520 TRAP 40000:IF TBL(K,HOR)<>0 THEN RETUR
        N
EH 4540 Y=K+VD:TRAP 4570:IF TBL(Y,HOR)=0 THEN
        TRAP 40000:GOTO 4570
FD 4550 IF TBL(Y,HOR)=NOW THEN Y=Y+VD:GOTO 455
        0
DC 4560 TRAP 40000:IF TBL(Y,HOR)=OTHER THEN RE
        TURN
FG 4570 STR(K,HOR)=500+100*(STR(K,HOR)<>500):R
        ETURN
CM 4600 K=HOR+HD:TRAP 4140
KH 4610 IF TBL(VER,K)=OTHER THEN K=K+HD:GOTO 4
        610
CP 4620 TRAP 40000:IF TBL(VER,K)<>0 THEN RETUR
        N
EA 4640 Y=K+HD:TRAP 4670:IF TBL(VER,Y)=0 THEN
        TRAP 40000:GOTO 4670
EL 4650 IF TBL(VER,Y)=NOW THEN Y=Y+HD:GOTO 465
        0
DH 4660 TRAP 40000:IF TBL(VER,Y)=OTHER THEN RE
        TURN
FP 4670 STR(VER,K)=500+100*(STR(VER,K)<>500):R
        ETURN
KE 4800 IF TBL(0,0)=NOW THEN X=0:Y=0:HD=C1:VD=
        0:GOSUB 5000:X=0:Y=0:VD=C1:HD=0:GOSUB
        5000
FD 4810 IF TBL(0,C7)=NOW THEN X=C7:Y=0:HD=-C1:
        VD=0:GOSUB 5000:X=C7:Y=0:VD=C1:HD=0:GO
        SUB 5000
FE 4820 IF TBL(C7,0)=NOW THEN X=0:Y=C7:HD=C1:V
        D=0:GOSUB 5000:X=0:Y=C7:VD=-C1:HD=0:GO
        SUB 5000
AD 4830 IF TBL(C7,C7)=NOW THEN X=C7:Y=C7:HD=-C
        1:VD=0:GOSUB 5000:X=C7:Y=C7:VD=-C1:HD=
        0:GOSUB 5000
LA 4840 RETURN
GB 5000 DFL=0:TRAP 5040
KM 5010 X=X+HD:Y=Y+VD:IF TBL(Y,X)<>0 THEN GOSU
        B 5050:GOTO 5010
EN 5020 TRAP 40000:IF DFL=0 AND STR(Y,X)<>500
        THEN STR(Y,X)=400
KJ 5040 RETURN
```

```
KJ 5050 IF TBL(Y,X)=NOW THEN RETURN
KA 5060 DX=X:DY=Y:TRAP 5100
IJ 5070 DX=DX+HD:DY=DY+VD:IF TBL(DY,DX)=OTHER
        THEN 5070
JH 5080 IF TBL(Y,DX)=NOW THEN DFL=C1
KO 5090 RETURN
OA 5100 TRAP 5040:RETURN
KN 5110 FOR X=C1 TO S
MB 5120 IF TBL(VER,X)=0 AND TBL(VER,X-C1)<>0 A
        ND TBL(VER,X+C1)<>0 THEN GOSUB 5150
HK 5130 NEXT X:RETURN
KN 5150 HD=C1:FL=0:GOSUB 5200:IF FL=OTHER THEN
         FL=0:HD=-C1:GOSUB 5200:GOTO 5170
KM 5160 RETURN
GF 5170 IF FL=OTHER THEN STR(VER,X)=100
KO 5180 RETURN
DC 5200 DX=X:TRAP 5220
IP 5210 DX=DX+HD:IF TBL(VER,DX)<>0 THEN FL=TBL
        (VER,DX):GOTO 5210
AO 5220 TRAP 40000:RETURN
LA 5310 FOR Y=C1 TO S
LM 5320 IF TBL(Y,HOR)=0 AND TBL(Y-C1,HOR)<>0 A
        ND TBL(Y+C1,HOR)<>0 THEN GOSUB 5350
HN 5330 NEXT Y:RETURN
NB 5350 VD=C1:FL=0:GOSUB 5400:IF FL=OTHER THEN
         FL=0:VD=-C1:GOSUB 5400:GOTO 5370
KO 5360 RETURN
GE 5370 IF FL=OTHER THEN STR(Y,HOR)=100
LA 5380 RETURN
DI 5400 DY=Y:TRAP 5420
JN 5410 DY=DY+VD:IF TBL(DY,HOR)<>0 THEN FL=TBL
        (DY,HOR):GOTO 5410
BA 5420 TRAP 40000:RETURN
CJ 6990 J=HOR:I=VER:K=C5+C9*(NOW=C1):FOR LOOP=
        K TO K+39:K=K+C1-18*(K=18):POSITION J+
        J+C2,I+I+C2:? #S;ANIM$(K,K)
CO 6995 SOUND 0,C4,0,S*(K=C8 OR K=17):NEXT LOO
        P:RETURN
DA 7000 OPEN #C1,C4,0,"K:":GET #C1,KEY:CLOSE #
        C1:IF KEY=155 THEN 7000
NL 7010 KEY=KEY-128*(KEY>127):KEY=KEY-32*(KEY>
        90):RETURN
DN 7100 POSITION C2*J+C2,I*C2+C2:ON TBL(I,J) G
        OTO 7120,7130
HD 7110 ? #S;"\":RETURN
CC 7120 ? #S;"{J}":RETURN
KD 7130 ? #S;"{■}":RETURN
MA 7200 POSITION 0,23:? #S;"{20 SPACES}";:POKE
        LIST1+26,C7:POKE LIST1+28,S
```

```
KJ 7205 FOR LOOP=C1 TO C3:POSITION 0,24-OPT:?
        #S;LINE$;:SOUND 0,50*LOOP,DF,C10:FOR D
        EL=C1 TO 55:NEXT DEL
DB 7210 POSITION 0,24-OPT:? #S;"{20 SPACES}";:S
        OUND 0,0,0,0:FOR DEL=C1 TO C7:NEXT DEL
        :NEXT LOOP
LH 7220 POKE LIST1+28,C7:POKE LIST1+26,S:POSIT
        ION 17,20:? #S;"{32 SPACES}":RETURN
JB 8000 HOR=C8:VER=11:POKE 53248,HOR*16:CR=P0+
        VER*C8:COL=14*(NOW-C1):OP=14*(NOW*C2-C
        3):POKE 704,COL
HP 8001 SOUND 0,40,C10,15:POKE CR,24:POKE CR+C
        1,126:POKE CR+C2,24:POKE 705,144
DB 8002 POSITION C1,20:? #S;"{4 SPACES}";PL1$(
        C2,11);"  ":SOUND 0,0,0,0
NN 8005 JST=STICK(NOW-C1):TRIG=STRIG(NOW-C1):I
        F JST=15 AND TRIG THEN 8005
ON 8007 POKE 77,0:POKE CR,0:POKE CR+C1,0:POKE
        CR+C2,0:IF TRIG=0 THEN SOUND 0,20,C10,
        15:J=HOR-C4:I=VER-C3:GOTO 8100
GP 8010 HOR=HOR+DELTA(JST,0):VER=VER+DELTA(JST
        ,C1):IF HOR>C12 THEN HOR=HOR-C9
NK 8020 IF HOR<C4 THEN HOR=HOR+C9
NG 8030 IF VER>11 THEN VER=VER-C9
OH 8040 IF VER<C3 THEN VER=VER+C9
MH 8080 IF HOR<C12 AND VER<11 THEN POKE 704,CO
        L-OP*(TBL(VER-C3,HOR-C4)=NOW):GOTO 808
        5
BF 8083 POKE 704,COL:IF VER>C4 AND VER<C9 THEN
         POKE 704,42
IN 8085 CR=P0+VER*C8:SOUND 0,200,C12,C10:POKE
        53248,HOR*16:POKE CR,24:POKE CR+C1,126
        :POKE CR+C2,24
BO 8090 SOUND 0,0,0,0:FOR LOOP=C1 TO 20:NEXT L
        OOP:GOTO 8005
DM 8100 SOUND 0,0,0,0:IF J=C8 AND I>C1 AND I<S
         THEN POKE 705,42:PASS=C1:GOTO 3200
PM 8110 IF I=C8 OR J=C8 THEN 8080
NE 8120 GOTO 8260
AF 8200 GOSUB 7220:SOUND 0,40,C10,15:POSITION
        C1,20:? #S;PL1$(C2);" COL? ";:SOUND 0,
        0,0,0
BO 8210 GOSUB 7000:I$=CHR$(KEY):? #S;I$:IF I$=
        "P" THEN PASS=C1:GOTO 3200
GJ 8240 ? #S;"{13 SPACES}ROW? ";:GOSUB 7000:EXC
        H$=CHR$(KEY):? #S;EXCH$
AP 8242 IF I$>="A" AND I$<="H" THEN 8250
CH 8244 IF EXCH$<"A" OR EXCH$>"H" THEN 8254
BI 8246 J=ASC(EXCH$)-65:IF I$<"1" OR I$>"8" TH
        EN 8254
```

```
MØ 8248 I=ASC(I$)-49:GOTO 8260
PC 8250 J=ASC(I$)-65:IF EXCH$<"1" OR EXCH$>"8"
         THEN 8254
KI 8252 I=ASC(EXCH$)-49:GOTO 8260
GE 8254 LINE$="{5 SPACES}TRY AGAIN":DF=C1Ø:GOS
        UB 72ØØ:GOTO 82ØØ
LG 8260 HOR=J:VER=I:POSITION C2*J+C2,I*C2+C2:?
         #S;PL1$(C1,C1):IF TBL(I,J)=Ø THEN TBL
        (I,J)=NOW:GOSUB 16ØØ:GOTO 835Ø
FE 83ØØ LINE$="  captured{♥}square":IF NOW=C1
        THEN LINE$="  captured{,}square"
BA 8310 DF=C12:GOSUB 72ØØ:GOSUB 71ØØ:GOTO MAIN
        LOOP
MI 8350 IF FLIP>Ø THEN GOSUB 3ØØØ:GOTO 84ØØ
OC 8360 TBL(I,J)=Ø:LINE$="{4 SPACES}illegal
        {♥}move":IF NOW=C1 THEN LINE$="
        {4 SPACES}illegal{,}move"
BK 8365 DF=C12:GOSUB 72ØØ:GOSUB 71ØØ:GOTO MAIN
        LOOP
JE 84ØØ I=NOW:NOW=OTHER:OTHER=I:EXCH$=PL1$:PL1
        $=PL2$:PL2$=EXCH$:IF SFL=C2 THEN 85ØØ
KN 8405 IF PLYRS>C2 THEN 98ØØ
HH 8410 IF PLYRS=C2 OR NOW=C3-CP THEN GOTO MAI
        NLOOP
MK 8415 LINE$="  computer{G}s{,}move":IF NOW=C
        2 THEN LINE$="  computer{G}s{♥}move"
IK 8417 IF OPT=C1 THEN GOSUB 857Ø
KB 8420 SOUND Ø,4Ø,C1Ø,15:POSITION Ø,2Ø:? #S;L
        INE$:SOUND Ø,Ø,Ø,Ø:GOSUB 4ØØØ:GOSUB 48
        ØØ:GOSUB 33ØØ
DI 8421 IF MAX=-21Ø THEN 3212
IJ 8422 IF STR(VER,HOR)=5ØØ THEN K=3ØØ:GOSUB 4
        2ØØ:GOTO 843Ø
JB 8424 IF HOR=Ø OR HOR=C7 THEN GOSUB 38ØØ
JH 8426 IF VER=Ø OR VER=C7 THEN GOSUB 39ØØ
GK 843Ø GOSUB 699Ø:GOTO 826Ø
JN 85ØØ GOSUB 722Ø:POSITION C1,2Ø:? #S;"black
        {DOWN}";:I=BSCR:K=-32:GOSUB 97ØØ:? #S;
        I$;"  white{INS LINE}";:I=WSCR:K=96:GO
        SUB 97ØØ
OD 8510 ? #S;I$:GFL=C1:? #S:IF BSCR>WSCR THEN
        ? #S;"{5 SPACES}black{,}wins";:GOTO 85
        4Ø
MJ 8520 IF WSCR>BSCR THEN ? #S;"{5 SPACES}whit
        e{♥}wins";:GOTO 854Ø
FL 8530 ? #S;"{6 SPACES}tie game";
HB 8540 IF PEEK(53279)>S THEN 854Ø
NL 8550 POKE 77,Ø:K=PEEK(53279):IF K=S THEN SF
        L=Ø:M=Ø:GOSUB 857Ø:GOSUB 174Ø:GOTO 84Ø
        5
```

```
NO 8552 IF K=C3 AND PLYRS=C1 THEN CP=CP+C1-C2*
        (CP=C2):GOSUB 8600:GOTO 8540
CG 8554 IF K=C5 THEN PLYRS=PLYRS+C1-C4*(PLYRS=
        C4):GOSUB 8560:GOSUB 8600
OC 8556 GOTO 8540
GB 8560 IF PLYRS=C3 AND GFL=0 THEN PLYRS=C1
LI 8565 RETURN
OJ 8570 POSITION 0,20:? #S;"{60 SPACES}";:RETUR
        N
CN 8600 IF PLYRS<C3 THEN GOSUB 8670
EI 8601 ON CP+C2*PLYRS-C2 GOSUB 8630,8640,8610
        ,8610,8650,8650,8660,8660
KK 8602 IF PEEK(53279)<>5 THEN 8602
LC 8604 RETURN
BN 8610 ? #S;"{18 SPACES}":RETURN
GH 8630 ? #S;" computer{,}is{,}black":RETURN
AM 8640 ? #S;" computer{,}is{,}white":RETURN
GJ 8650 POSITION 0,20:? #S;"  AUTOMATIC REPLAY
        ":GOTO 8610
AB 8660 POSITION 0,20:? #S;"{3 SPACES}MANUAL R
        EPLAY{4 SPACES}":GOTO 8610
EF 8670 POSITION 0,20:? #S;"{3 SPACES}";CHR$(P
        LYRS+176);" PLAYER GAME{4 SPACES}":RET
        URN
PB 9000 GRAPHICS 0:I=PEEK(560)+256*PEEK(561):P
        OKE I+C3,71:POKE I+C10,5:SETCOLOR C2,C
        10,0:SETCOLOR C4,C10,0
JC 9005 SETCOLOR C1,C10,C8:SETCOLOR C3,C12,C8:
        POKE 53774,112:POKE 16,64:POKE I+14,5:
        POKE I+15,5:POKE I+20,5
LD 9010 POKE I+21,5:POKE 82,0:POKE 83,39:POKE
        752,C1:POSITION C5,0:? " REVERSI
        {20 SPACES}{12 T}{T}"
LJ 9011 FOR J=C1 TO 50:NEXT J:FOR J=19 TO 31:S
        OUND 0,200,C12,C8:POSITION J,0:? " +":
        SOUND 0,0,0,0
FD 9012 FOR K=C1 TO C7:NEXT K:NEXT J
EA 9013 FOR J=C1 TO 30:NEXT J:SOUND 0,20,C10,C
        10:POSITION 32,0:? "{T}";:SOUND 0,0,0,
        0:FOR J=C1 TO 35:NEXT J
HD 9014 FOR J=C1 TO C12:SOUND 0,60,C12,C8:? "
        {T}";:I=I-I+I:SOUND 0,0,0,0:FOR K=C1 T
        O C4:NEXT K:NEXT J:I$=CHR$(34)
BA 9020 POSITION 20,C4:? "select input device
        {2}{38 M}  ":? :? "  1. keyboard"
IJ 9030 ? "{4 SPACES}Enter COLumn, ROW coordin
        ates.":? "{4 SPACES}Enter ";I$;"P";I$;
        " to pass."
```

```
AA 9040 ? :? :? "  2. joysticks":? "
        {4 SPACES}Move the cursor (+) to the s
        quare":? "{4 SPACES}you want and press
         the trigger."
JM 9050 ? "{4 SPACES}To pass, place the cursor
         over":? "{4 SPACES}";I$;"PASS";I$;" a
        nd press the trigger."
DG 9100 GOSUB 7000:TRAP 9100:OPT=VAL(CHR$(KEY)
        ):TRAP 40000:IF OPT<C1 OR OPT>C2 THEN
        9100
JL 9110 GRAPHICS 23:RETURN
IH 9500 I=C10+C4*(OPT-C1):K=-32:GOSUB 9700
CK 9505 POSITION C1,C1:? #S;"PREPARING THE FIR
        ST":? #S;" BOARD WILL TAKE":? #S:? #S;
        " ABOUT ";I$;" seconds."
MD 9510 ? #S:? #S:? #S;" WHEN THE BOARD
        {5 SPACES}":? #S;" APPEARS, PRESS:
        {4 SPACES}":? #S;" select - GAME MODE
        "
CL 9520 ? #S;" option - COLOR OF  ":? #S;" COM
        PUTER'S PIECES  ":? #S;" start - GAME"
        :RETURN
FC 9600 ? #S:? #S:? #S;" patience{N}":RETURN
MN 9700 I$=STR$(I):FOR J=C1 TO LEN(I$):I$(J,J)
        =CHR$(ASC(I$(J,J))+K):NEXT J:RETURN
KI 9800 SOUND 0,40,C10,15:VER=INT(GAME(M)):HOR
        =10*(GAME(M)-VER):POSITION C5,20:? #S;
        PL1$(C2,11):SOUND 0,0,0,0
IA 9805 I=INT((M+C2)/C2):POSITION C9+(I<C10),2
        2:K=128:GOSUB 9700:? #S;I$:FOR K=C1 TO
         C10:NEXT K
D9 9806 IF PLYRS=C4 THEN GOSUB 7000+2830*(OPT-
        C1)
NG 9807 IF VER<0 THEN 3212
GN 9810 GOSUB 6990:GOTO 8260
LA 9830 IF STRIG(0) AND STRIG(C1) THEN 9830
AL 9835 POKE 77,0:RETURN
OB 10000 DATA 500,-70,75,65,-70,-200,0,0,75,0,
         40,20,65,0,20,0
BD 10100 DATA 664,88,648,224,696,256,552,240,5
         20,472,544,488,720,32,704,40,536,48,9
         92,120,656,496,ERROR
JP 10110 DATA 60,24,126,24,255,60,255,231,255,
         231,255,60,126,24,60,24
LG 10120 DATA 16,0,60,126,255,255,126,60,0,24,
         0,0,126,255,255,126,0,0,64,0,0,0,255,
         255,0,0,0,72,0,0,0,255,0,0,0,0
PA 10130 DATA 104,0,0,0,0,255,0,0,0
OD 10200 DATA 1,1,1,0,-1,-1,-1,0,0,0,0,1,-1,0,
         0,1,-1,0,0,1,-1,0
```

```
AF 14000 C1=1:C2=C1+C1:C3=C2+C1:C4=C2+C2:C5=C4
         +C1:C8=C4+C4:C7=C4+C3:S=C3+C3:C10=C5+
         C5:C12=S+S:C9=C5+C4
AC 14010 DIM TBL(C7,C7),STR(C7,C7),DELTA(15,C1
         ),I$(C2),PL1$(C12),PL2$(C12),EXCH$(C1
         2),LINE$(19),ANIM$(18),GAME(75)
CE 14020 PLYRS=C1:CP=C1:GOSUB 9000:FAC=255*(OP
         T-C1):MAINLOOP=8000+200*(OPT=C1):GOSU
         B 1000
BB 14025 IF OPT-C1 THEN GOSUB 2000:GOSUB 2500
OE 14030 SFL=0:X=PEEK(559):POKE 559,0:GOSUB 15
         000:POKE 559,X:GOSUB 8600:GOTO 8540
CF 15000 ? #S;"{CLEAR}":POKE 709,0:POKE 711,14
         :POKE 712,198:POKE 710,176:POKE 708,2
         34:IF OPT=C2 THEN POKE 708,52
FI 15010 ANIM$="{B}{C}{H}{I}{■}{■}{■}{■}{■}
         {■}{■}{■}{■}{M}{H}{C}{B}{J}":POKE 756
         ,SET/256:POSITION 0,19:? #S;"████████
         ████████████":REM 20 INV. SHIFT-
FJ 15020 POSITION C1,C1:? #S;"<^C^C^C^C^C^C^C^
         >":FOR I=C1 TO C7:POSITION C1,(C2*I)
LJ 15030 ? #S;"▞\▞\▞\▞\▞\▞\▞\▞\▞":? #S;" [^+^+
         ^+^+^+^+^+^]":NEXT I
DF 15040 ? #S;" ▞\▞\▞\▞\▞\▞\▞\▞\▞"
DG 15042 ? #S;CHR$(32);CHR$(164);CHR$(222);CHR
         $(165);CHR$(222);CHR$(165);CHR$(222);
         CHR$(165);CHR$(222);CHR$(165);
OE 15044 ? #S;CHR$(222);CHR$(165);CHR$(222);CH
         R$(165);CHR$(222);CHR$(165);CHR$(222)
         ;CHR$(166)
MB 15050 POSITION C5,C5:? #S;"█";:POKE 85,13:?
          #S;"█":POSITION C5,13:? #S;"█";:POKE
          85,13:? #S;"█"
GB 15055 IF OPT=C2 THEN 15080
ED 15060 POSITION 0,0:? #S;"  A B C D E F G H"
         :FOR I=C1 TO C8:? #S:? #S;I;:POKE 85,
         18:? #S;I:NEXT I
MP 15070 ? #S:? #S;"  A B C D E F G H":LET LIS
         T1=-C8:RETURN
DF 15080 POSITION 17,C5:? #S;"+^>":FOR I=S TO
         C12:POSITION 18,I:? #S;"\▞":NEXT I
NA 15085 POSITION 17,13:? #S;CHR$(171);CHR$(22
         2);CHR$(164)
NE 15090 I=PEEK(560)+256*PEEK(561):LET LIST1=I
         :POKE I,S:POKE I+C1,C7
EF 15095 FOR J=C2 TO C4:POKE I+J,PEEK(I+J+C1):
         NEXT J:POKE I+C5,S:POKE I+28,C7:POKE
         707,52
AC 15100 POKE 706,150:POKE 705,144:POKE 53251,
         208:POKE 53255,40:POKE 53250,188:POKE
          53249,192:RETURN
```

Chapter 2

Dollars from Heaven

Steven Cohen

Money may not grow on trees, but after playing this game you'll be convinced that it drops from the sky.

"Dollars from Heaven" uses the vertical blank PM routine (VBLANK PM) by Tom Sak and Sid Meier, in *COMPUTE!'s First Book of Atari Graphics*. It shows how a novice can create a game using programming techniques like the ones described in that volume. In fact, once the tricks have been mastered, it takes only a good idea and a little time.

Understanding Interrupts

The picture on your TV is formed when a beam of electrons draws scan lines across the screen. The beam starts at the top-left corner and moves to the right edge of the screen. It then shuts off for a fraction of a second (the so-called horizontal blank) and moves back to the left edge and down one line to draw the next scan line.

After the last scan line is drawn, the beam shuts off and moves back to the top-left corner. This is the vertical blank, and it repeats 60 times per second. This game uses the blank period to update the positions of the players on the screen. All the user has to do is update the registers that hold the player position; the ML routine does the rest.

In addition, each player can have four different shapes which, when used in a sequence, add to the animation of the character. For a complete discussion of the technique, I recommend that you get a copy of *COMPUTE!'s First Book of Atari Graphics*.

Catching Dollars

After you type in Dollars from Heaven, save it to disk or tape before trying to play. Attempting to save the game after a few rounds have been played will cause strange things to happen when it is reloaded.

The object of the game is to catch dollar signs to buy building materials without getting hit by the bombs. You control the player at the right, moving right or left with the joystick. If you get hit by a bomb, you lose everything you were carrying. In addition, there are nails in the middle of the screen; use the fire button to jump over them.

Once you have collected enough dollars to match the current trade value (3 in the first round), take them to the store to get supplies. Then go to the site where the house is to be built (at the far-left side of the screen) and part of the house will appear. It takes four trips to the store to build a house. You get 10 points for catching a dollar sign, 10 points for installing the first three parts of the house, and 50 points for completing the house.

After the first house is completed, you go to the next round. The trade value becomes four, and the bombs move a little faster. If you get to 1000 points, you get an additional player. The game starts with two players on reserve.

How It Works

Below is a brief explanation of the program.

Line(s)	
90	Initializes player/missile graphics and Vblank PM.
91–100	Draw background.
101–102	Set up variables.
103–120	Draw background.
125–134	Set player color size and starting location.
143–145	Move bombs and dollar sign.
146	Checks to see if player is currently jumping.
147	Checks for start of jump.
148–150	Check joystick and move player.
151	Disables attract mode.
152	Gives Vblank PM new positions.
170–175	Check for collisions.
176	Looks for player at store.
177	Looks for player at house.
195	Animates player.
300–349	Player–bomb collision routine.
500–506	Player–dollar sign collision routine.
600–612	House plotting routine.
800–820	Jumping routine.
1000–3060	Set up Vblank PM and player/missile graphics.

Dollars from Heaven

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
MN 90 POKE 53278,HITCLR:GRAPHICS 5:SETCOLOR 2,
      3,3:SETCOLOR 4,8,4:? "PLEASE WAIT!":GOSU
      B 1000
IL 91 COLOR 1:PLOT 35,39:PLOT 37,39:PLOT 39,39
```

```
KL 92 PLOT 37,38
MA 99 COLOR 2:PLOT 75,39:DRAWTO 75,26:PLOT 79,
      25:DRAWTO 73,27:PLOT 79,39:DRAWTO 79,25
PN 100 PLOT 78,39:DRAWTO 78,26:PLOT 77,39:DRAW
       TO 77,27:PLOT 76,39:DRAWTO 76,27
KE 101 Q=6:MEN=3:D9=3:A1=1:SC=0:FY=0
KK 102 B2=0:D1=0:C0=0
EB 103 PRINT :? :? :? "SCORE=";SC:FOR I=1 TO 8
       99:NEXT I
OA 104 PRINT :? "HOUSE{9 SPACES}NAILS
       {12 SPACES}STORE":?
DH 105 COLOR 1:PLOT 9,5:DRAWTO 9,7:PLOT 8,6:PL
       OT 10,4:DRAWTO 10,8:PLOT 11,4:DRAWTO 11
       ,8:PLOT 12,5:DRAWTO 12,7
JM 106 PLOT 13,6
HN 120 PRINT "ROUND";A1;"{20 SPACES}TRADE=";D9
GL 125 POKE PLY,169:POKE PLL,24
DI 126 POKE PLY+1,15:POKE PLL+1,9:POKE PLX+1,1
       50
BJ 127 POKE PLL+2,9:POKE PLL+3,13:POKE PLY+2,1
       5:POKE PLY+3,15:POKE PLX+2,170:POKE PLX
       +3,70
JJ 134 Z=10:A=10:B=10:DRAW=1
MC 136 I=200:GOTO 148
FP 143 A=A+Q+2:IF A>220 THEN A=5:R=RND(1)*200:
       POKE PLX+2,R:IF R<45 THEN R=170:POKE PL
       X+2,R
OF 144 B=B+4:IF B>220 THEN B=5:R=RND(1)*200:PO
       KE PLX+3,R:IF R<45 THEN R=100:POKE PLX+
       3,R
DF 145 Z=Z+Q:IF Z>220 THEN Z=5:R=RND(1)*200:PO
       KE PLX+1,R:IF R<45 THEN R=70:POKE PLX+1
       ,R
DF 146 IF JP<>0 THEN 800
OE 147 IF STRIG(0)=0 THEN JP=1:GOTO 800
JF 148 IF STICK(0)=15 THEN 152
CF 149 IF STICK(0)=7 THEN I=I+3:IF I>200 THEN
       I=200
PE 150 IF STICK(0)=11 THEN I=I-3:IF I<44 THEN
       I=44
JA 151 POKE 77,0
CK 152 POKE PLY+1,Z:POKE PLY+2,A:POKE PLY+3,B:
       POKE PLX,I:SOUND 0,Z,14,2
MO 170 IF PEEK(53261)=1 THEN 300
KN 171 IF I<126 AND I>102 AND PEEK(53252)>0 TH
       EN 300
NB 172 IF PEEK(53262)=1 THEN 300
NG 174 IF PEEK(53263)=1 THEN 500
MH 175 POKE 53278,HITCLR
```

```
LP 176 IF D1=D9 AND I>180 THEN D1=0:SOUND 2,60
       ,10,8:FOR C1=1 TO 200:NEXT C1:SOUND 2,0
       ,0,0:B1=B1+1
DA 177 IF B1>0 AND I<50 THEN GOSUB 600
AN 185 POKE PDR,DRAW
JB 190 IF STICK(0)=15 THEN 205
DD 195 DRAW=DRAW+24:IF DRAW>73 THEN DRAW=1
GI 205 GOTO 143
NJ 300 Z=5:D=0:JP=0:SOUND 3,0,0,0:EX=0
JF 302 FOR U=100 TO 200 STEP 10:SOUND 0,I,0,15
       :NEXT U
CC 306 POKE PLY+1,Z:POKE PLY+2,Z:POKE PLY+3,Z
GH 307 POKE PLX,200:MEN=MEN-1:D1=0:B1=0:POKE P
       LX+1,R:FOR S=1 TO 200:NEXT S:SOUND 0,0,
       0,0
KF 325 POKE 53278,HITCLR:IF MEN=0 THEN 327
GM 326 GOTO 125
CC 327 ? :? "SCORE=";SC
LA 328 PRINT "GAME OVER{3 SPACES}TO PLAY AGAIN
        PRESS FIRE"
EF 330 COLOR 0
HB 331 PLOT 3,39:DRAWTO 3,33
GN 332 PLOT 0,35:DRAWTO 8,30
KE 333 PLOT 8,30:DRAWTO 15,35
NE 334 PLOT 12,33:DRAWTO 12,39
EG 348 IF STRIG(0)=0 THEN 99
HI 349 GOTO 348
GP 500 D1=D1+1:IF D1>D9 THEN D1=D9
LI 501 B=5:FOR C=1 TO D1:FOR Y=15 TO 0 STEP -1
       :SOUND 2,9,10,Y:NEXT Y:NEXT C
FC 502 SC=SC+10:IF FY=0 AND SC>1000 THEN MEN=M
       EN+1:FY=1:? "{BELL}"
NN 505 R=RND(1)*190:IF R<45 THEN 505
ON 506 POKE PLY+3,B:FOR S=1 TO 20:NEXT S:POKE
       PLX+3,R:SOUND 0,0,0,0:POKE 53278,HITCLR
       :GOTO 143
AO 600 COLOR 2:B2=B2+B1:B1=0:FOR R1=1 TO 20:SO
       UND 3,170,6,10:FOR J=1 TO 3:NEXT J:SOUN
       D 3,0,0,0:FOR J=1 TO 5:NEXT J
FJ 601 NEXT R1
BC 602 IF B2>0 THEN PLOT 3,39:DRAWTO 3,33
AP 603 IF B2>1 THEN PLOT 0,35:DRAWTO 8,30
EH 604 IF B2>2 THEN PLOT 8,30:DRAWTO 15,35
LK 605 IF B2>3 THEN PLOT 12,33:DRAWTO 12,39:CO
       LOR 0:FOR DE=1 TO 100:NEXT DE:GOTO 611
KG 610 SC=SC+10:RETURN
BM 611 IF C0=0 THEN A1=A1+1:Q=Q+2:D9=D9+1:C0=1
       :GOTO 602
JI 612 SC=SC+50:GOTO 102
```

```
EK 800 POKE PLY,PEEK(PLY)-3*JP:IF PEEK(PLY)<15
       0 THEN JP=-JP
MI 805 SOUND 3,PEEK(PLY)-100,10,10
DL 806 DRAW=2
IP 810 IF D<>0 OR EX=1 THEN 815
MB 811 IF STICK(0)<>7 AND STICK(0)<>11 THEN EX
       =1
NI 812 IF STICK(0)=11 THEN D=-3
KM 813 IF STICK(0)=7 THEN D=+3
AF 815 I=I+D:IF I>200 THEN I=200
DI 816 IF I<44 THEN I=44
OG 817 IF PEEK(PLY)=169 THEN JP=0:SOUND 3,0,0,
       0:D=0:EX=0
GK 820 GOTO 151
OG 1000 FOR I=1536 TO 1706:READ A:POKE I,A:NEX
        T I
EN 1020 FOR I=1774 TO 1787:POKE I,0:NEXT I
NC 1030 PM=PEEK(106)-16:PMBASE=256*PM
EM 1040 FOR I=PMBASE+1023 TO PMBASE+2047:POKE
        I,0:NEXT I:DRWBAS=PMBASE+1
DP 1055 FOR J=0 TO 3
GL 1065 FOR K=DRWBAS+J*24 TO DRWBAS+J*24+23:RE
        AD X:POKE K,X:NEXT K:NEXT J
LJ 1066 RESTORE 3050
LM 1067 FOR I=PMBASE+1281 TO PMBASE+1289:READ
        A:POKE I,A:NEXT I
LL 1068 RESTORE 3050
LN 1069 FOR I=PMBASE+1537 TO PMBASE+1545:READ
        A:POKE I,A:NEXT I
LF 1070 RESTORE 3060
LK 1072 FOR I=PMBASE+1793 TO PMBASE+1805:READ
        A:POKE I,A:NEXT I
PF 1075 POKE 704,223:POKE 705,73
PD 1076 POKE 706,44:POKE 707,200
GP 1080 PLX=53248:PLY=1780:PLL=1784
AK 1090 POKE 559,62:POKE 623,4:POKE 1788,PM+4:
        POKE 53277,3:POKE 54279,PM
FM 1091 POKE 53256,1
MF 1095 PDR=1772:POKE 1771,PM
HI 1100 X=USR(1696)
KD 1110 RETURN
KG 2000 REM
GP 2010 DATA 162,3,189,244,6,240,89,56,221,240
        ,6,240,83,141,254,6,106,141
DG 2020 DATA 255,6,142,253,6,24,169,0,109,253,
        6,24,109,252,6,133,204,133
EC 2030 DATA 206,189,240,6,133,203,173,254,6,1
        33,205,189,248,6,170,232,46,255
ED 2040 DATA 6,144,16,168,177,203,145,205,169,
        0,145,203,136,202,208,244,76,87
```

```
PE 2050 DATA 6,160,0,177,203,145,205,169,0,145
        ,203,200,202,208,244,174,253,6
KM 2060 DATA 173,254,6,157,240,6,189,236,6,240
        ,48,133,203,24,138,141,253,6
NE 2070 DATA 109,235,6,133,204,24,173,253,6,10
        9,252,6,133,206,189,240,6,133
GL 2080 DATA 205,189,248,6,170,160,0,177,203,1
        45,205,200,202,208,248,174,253,6
CG 2090 DATA 169,0,157,236,6,202,48,3,76,2,6,7
        6,98,228,0,0,104,169
OF 2100 DATA 7,162,6,160,0,32,92,228,96
KM 3005 REM
KK 3015 DATA 0,12,12,30,0,12,12,0,12,14,30,45,
        13,13,12,28,28,20,52,34,34,34,102,0
KO 3025 DATA 0,12,12,30,0,12,12,0,12,14,14,13,
        26,4,8,12,12,28,24,28,24,20,18,50,0
OH 3035 DATA 0,12,12,30,0,12,12,0,12,14,10,14,
        30,12,8,12,28,28,8,12,12,8,24,0
AA 3045 DATA 0,12,12,30,0,12,12,0,12,12,12,10,
        6,30,12,12,12,12,20,20,18,50,6,0
OI 3050 DATA 126,195,195,126,24,153,90,62,24
LD 3060 DATA 16,124,214,211,208,112,28,22,19,1
        47,214,124,16
```

Chapter 2

Box Hunt

Lenny Norinsky

Are you looking for a fast-paced game that challenges reflexes as well as skill? Then "Box Hunt," for any Atari with the GTIA chip installed, is for you. It will bring new meaning to the phrase "Don't fence me in."

"Box Hunt" is a simple but entertaining game that demonstrates your Atari's ability to plot objects on the screen. You've just been named to the high post of Territorial Defender of Wambaogh, and your mission is a simple one: to erect a network of protective walls to defend your city and confuse the hostile Zuvambian raiders. Your weapon? The speedy Wambaoghian Waller, a sophisticated machine that automatically leaves a wall wherever it goes. All you have to do is guide it across the countryside.

Or so you thought, until you read the fine print in the manual: "This Wambaoghian Waller must stop for supplies every 1000 glunkas." Supplies, huh? You know that means *boxite*—and that means you'll not only have to erect those walls but look for boxes too.

It's not just a job. It's a box hunt!

Guide your Waller using your joystick; it will leave a trail—the wall—behind it. Hit the boxes to run up your score. Every time you hit a box, you get 100 points and several additional boxes appear. The object is to survive as long as you can, hitting as many boxes as possible, before crashing into the border or running into one of your own walls. If you do crash, the game will stop, show your score, and start over.

As you'll quickly discover, Box Hunt produces some fast, reflex-challenging action. That's why I've included a panic button feature too. When you are in a tight space or about to crash, press the trigger on the joystick and your line will be randomly relocated on the screen. Sometimes it's all that will save you. But use it only when you have to, for it can make things worse just as easily as it can make them better!

Box Hunt

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
CL 1  GRAPHICS 0:GOSUB 1100:? :? :? "
      {12 SPACES}BOX HUNT":? :? :? "POINT VALUES
      :":? :? "100 POINTS for a box."
PJ 3  ? "5 POINTS for a line segment.":? "10 PO
      INTS OFF for easy escape"
BC 4  POSITION 20,20:? "PRESS":SOUND 0,100,10,1
      0:POSITION 20,21:? "START":SOUND 0,50,10,
      10:IF PEEK(53279)<>6 THEN 4
HG 5  GOSUB 1000
GC 10 GRAPHICS 11:POKE 710,0:ZX=1:ZY=0:X=40:Y=
      90:COLOR 10:PLOT 0,0:DRAWTO 79,0:DRAWTO
      79,191:DRAWTO 0,191:DRAWTO 0,0
BK 20 A=RND(0)*70+5:B=RND(0)*180+5:COLOR 15:PL
      OT A,B:DRAWTO A+2,B:DRAWTO A+2,B+2:DRAWT
      O A,B+2:DRAWTO A,B
FK 30 COLOR 5:LOCATE X,Y,Z:IF Z=10 OR Z=5 THEN
       100
NA 40 PLOT X,Y:SOUND 0,X,10,4:SOUND 0,Y,10,4:I
      F Z=15 THEN 110
II 50 S=STICK(0):IF S=14 THEN ZX=0:ZY=-1
JC 60 IF S=13 THEN ZX=0:ZY=1
LO 70 IF S=11 THEN ZX=-1:ZY=0
GH 80 IF S=7 THEN ZX=1:ZY=0
BG 85 IF STRIG(0)=0 THEN X=RND(0)*70+5:Y=RND(0
      )*180+5:SC=SC-10
BD 90 X=X+ZX:Y=Y+ZY:SC=SC+5:GOTO 30
JE 100 GRAPHICS 2:GOSUB 1100:? #6:? #6:? #6:?
       #6;"{3 SPACES}score :";SC:FOR I=0 TO 99
       9:NEXT I:RUN
JH 110 X=X+ZX:Y=Y+ZY:SC=SC+100:GOTO 20
FO 1000 GRAPHICS 2:? #6:? #6:? #6;"{3 SPACES}g
        et READY":SOUND 1,190,10,10:GOSUB 1100
LJ 1010 FOR I=1 TO 500:NEXT I:? #6:? #6;"
        {7 SPACES}go!":SOUND 0,121,10,10:SOUND
         1,87,10,10:FOR I=1 TO 200:NEXT I
LC 1020 SOUND 0,0,0,0:SOUND 1,0,0,0:RETURN
KG 1100 REM
CC 1101 POKE 709,31
OH 1102 POKE 710,0
KH 1105 RETURN
```

Chapter 2

Dragon's Den

Ken and JoAnn Davy

Fight monsters and search for gold, but watch out for the traps in "Dragon's Den."

We've always been dreamers, so after we bought our Atari and cassette recorder, we went in search of an adventure. But not just any adventure. We wanted one with several levels and lots of monsters and treasures, and filled with magic, sound, and graphics. Most of all, it had to fit into 16K. We also thought the adventure should change each time it was played, so that even if players did well, the game would still be challenging.

Alas, our searching was in vain. So our next idea was to write our own adventure. After all, how hard could it be to write an adventure game that met all our requirements?

A Year Later

One year and many sleepless nights later the task was done. "Dragon's Den" was complete.

In the game you wander from room to room, through different levels, looking for monsters to kill and gold to win. Each lower level is more difficult than the one above. When you reach the fifth level, you'll meet the dragon.

Your player has three characteristics: strength, dexterity, and hit points. Hit points determine the amount of damage you can survive when fighting monsters.

Strength determines the amount of damage you do to a monster each time you hit it. Your strength is subtracted from the monster's hit points and added to your hit points. When a monster hits you, its strength is subtracted from your hit points. Some monsters are very strong! When both monster and player hit each other, its strength is subtracted and your strength is added to your hit points at the same time.

Dexterity is the measure of how often you can expect to hit the monster. The higher your dexterity, the more often you will score a hit.

There are two play options. The STANDARD PLAYER has a value of seven for each characteristic. Pressing the OPTION key causes the player characteristics to be selected randomly, thus RANDOM PLAYER. A random player could

have characteristic values higher or lower than seven. The game begins when you press START.

Meeting Monsters

All game action is controlled with the joystick. To get from one room to the next, simply move the joystick in the desired direction. Choices are prompted by messages in the text window and include things like going up or down stairs, attacking monsters, and using magic rings or potions.

When you enter a room with a monster, you can attack it by pushing the trigger on the joystick. You can flee the monster by moving away from it. However, the monster will get a free attack.

Every time you kill a monster, your gold will increase. Some monsters will also have magic items. You may possess one ring and one potion at a time. When you see either, you are given the choice of using it or leaving it. To use a ring or drink a potion, press the trigger on the joystick. To leave it, move away.

When you have found one of the rings that can be used for attacks, the computer will give you the normal prompt, PRESS TRIGGER TO ATTACK. If you wish to attack, press the trigger. Otherwise, move the joystick in any direction. You will then get the prompt PRESS TRIGGER TO USE RING. If you wish to flee, move the joystick a second time.

The rings have a limited number of charges, so use them wisely. When you attack with an empty ring, the computer tells you OUT OF CHARGES, and the monster gets a free attack.

A few of the monsters have poison. If you are poisoned by a monster, you die no matter how many hit points you had. If you kill a poisonous monster, you will be given a magic sword which will increase your strength. If you get a second magic sword, your strength will again increase.

If you pass through an empty room you get extra hit points for "resting." The level you're on determines how many hit points you'll get. When you return to a room for a second or third visit, it is unlikely that what you saw the first time will still be there. If you see a stairway in a room, leave, and then come back—there may be a monster in the stairway's place.

Oh, one more thing: Watch out for traps.

How It's Done

Let's look at the listing. Line 10 assigns variable names to frequently used numbers (to save memory), and calls a subroutine that creates redefined characters.

Line 15 makes the dragon show up at the beginning of the program.

Lines 35–75 give you the player option. Memory location 53279 reads the function keys. A value of three means the OPTION key is pressed, and a value of six means the START key is pressed.

Line 90 ends the program if you've gone up to the surface.

Lines 100–105 draw the box for the floor plan. POKE 756,BASE/256+N2 resets the character set. POKE 77 in line 100 disables the rotating colors that appear if there is no keyboard input for several minutes. Lines 120–195 select the room contents and draw the floor plans. Lines 210–230 draw the dragon.

Lines 245–350 animate the player with redefined characters.

Lines 355–380 tell the computer what to do, depending on what's in the room. The strings store names and graphic shapes for the monsters. Lines 385–390 make the screen flash when you meet a monster.

Line 400 sets the strength, dexterity, and hit points for the monsters. It also decides if you're facing a poisonous monster.

Lines 410–555 handle the combat. Lines 565–685 decide what you found, if anything.

Lines 760–790, 795–830 and 835–840 contain subroutines for stairway up, stairway down, and traps.

The subroutine at 845 prints character values on the screen; 850 is the sound routine for the rings; 855 is used when you pick up certain rings. Line 860 disables the BREAK key. Line 865 blows dragon fire.

Lines 870–920 end the game when you win or die. Lines 925–935 start the game again.

Next comes the data for monster names and graphics. Line 960 is a delay loop.

Lines 965–985 place the monster in the room. The position depends on which direction the player came from.

Line 990 prints an often used phrase.

The rest of the program redefines the character set. Be careful typing the DATA statements, or your graphics won't look right.

Dragon's Den

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
ML 5 DIM S$(1),M$(9),L$(17):M$="YOU SEE A"
CC 10 G=1:V=0:J=0:Q=0:E=245:K=330:N0=0:N1=1:N2
   =2:N3=3:N4=4:N5=5:N6=6:N7=7:N8=8:N9=9:GO
   SUB 10000
IG 15 QW=1:GOTO 205
OP 20 QW=0:GRAPHICS N2+16:SETCOLOR N4,N8,N0:GO
   SUB 860
PE 25 POSITION N2,N2:? #6;"THE DRAGON'S DEN":P
   OSITION N9,N4:? #6;"BY":POSITION N2,N6:?
    #6;"KEN & JOANN DAVY"
JJ 30 FOR Z=N1 TO 3000:NEXT Z
CO 35 GRAPHICS N2+16:SETCOLOR N0,N0,14:SETCOLO
   R N4,N3,N6:POSITION N5,N3:? #6;"STANDARD
   "
BF 40 POSITION N6,N7:? #6;"PLAYER":SOUND 0,121
   ,10,8:FOR Z=1 TO 20:NEXT Z:SOUND 0,0,0,0
   :POKE 53729,N8:GOSUB 860
EN 45 X=PEEK(53279):IF X=N6 THEN P=N7:N=N7:H=N
   7:L=N1:GOTO 90
BJ 50 IF X=N3 THEN FOR Z=N1 TO 130:NEXT Z:GOTO
    60
AM 55 GOTO 45
FL 60 POSITION N5,N3:? #6;" RANDOM ":SOUND 0,6
   0,10,8:FOR Z=1 TO 20:NEXT Z:SOUND 0,0,0,
   0
BO 65 POKE 53279,N8:X=PEEK(53279):IF X=N6 THEN
    80
BN 70 IF X=N3 THEN FOR Z=N1 TO 130:NEXT Z:GOTO
    35
BA 75 GOTO 65
JM 80 P=INT(RND(N1)*N6)+N4:N=INT(RND(N1)*N5)+N
   5:H=INT(RND(N1)*N7)+N3:L=N1
NA 85 IF L=N0 THEN 905
MP 90 GRAPHICS N2:POKE 756,BASE+N2:SETCOLOR N4
   ,N8,N2:SETCOLOR N2,N2,N6:SETCOLOR N0,N8,
   N2:SETCOLOR N3,10,N8:A=10:B=N4:GOSUB 860
MK 95 IF L=N5 THEN 200
JA 100 POKE 77,N0:POSITION N1,N0:? #6;"{Q}{16 R}
   {E}":FOR Z=N1 TO N8:POSITION N1,Z:? #6;
   "{Y}":POSITION 18,Z:? #6;"{Y}":NEXT Z
ML 105 POSITION N1,N9:? #6;"{Z}{16 R}{C}"
```

```
EM 110 ? "USE JOYSTICK TO MOVE"
LI 115 GOSUB 845
OO 120 M=INT(RND(N1)*13):W=N0:ON L GOTO 125,14
       5,170,185,210
HM 125 I=M+N2:POSITION N4,N0:? #6;"{W}{5 R}
       {W}{3 R}{W}":POSITION N4,N2:? #6;"{Y}
       {5 SPACES}{Y}{3 SPACES}{Y}":POSITION N4
       ,N3:? #6;"{A}{R} {R}{W} {X} {W}{R}{X}
       {2 R}{D}"
OJ 130 POSITION N1,N4:? #6;"{A}{R} {D}":POSITI
       ON N4,N5:? #6;"{Y}{3 SPACES}{Y}
       {3 SPACES}{Y}":POSITION N4,N6:? #6;"
       {Y}{3 SPACES}{A}{R} {R}{S}{4 R} {D}"

KP 135 POSITION N8,N7:? #6;"{Y}"
FN 140 POSITION N4,N8:? #6;"{Y}{7 SPACES}{Y}":
       POSITION N4,N9:? #6;"{X}{3 R}{X}{3 R}
       {X}":RESTORE 940:GOTO E
EC 145 SETCOLOR N0,12,N2:SETCOLOR N4,12,N2:I=M
       +N9:POSITION N5,N0:? #6;"{W}{4 R}{W}
       {4 R}{W}":POSITION N5,N2:? #6;"{Y}
       {9 SPACES}{Y}"

DD 150 POSITION 10,1:? #6;"{Y}"
MK 155 POSITION N1,N3:? #6;"{A}{R} {R}{S}{R}
       {2 R}{X}{2 R} {R}{S} {R}{D}":POSITION 1
       5,N4:? #6;"{Y}":POSITION N5,N5:? #6;"
       {Y}":POSITION N5,N6:? #6;"{A}{R} {2 R}
       {W}{R} {2 R}{S} {R}{D}"
AP 160 POSITION 5,8:? #6;"{Y}"
GM 165 POSITION 10,N7:? #6;"{Y}{4 SPACES}{Y}":
       POSITION N5,N9:? #6;"{X}{4 R}{X}{4 R}
       {X}":RESTORE 945:GOTO E

FD 170 SETCOLOR N0,N1,N2:SETCOLOR N4,N1,N2:I=M
       +15:POSITION N4,N0:? #6;"{W}{3 R}{W}
       {3 R}{W}":POSITION N4,N2:? #6;"{Y}
       {3 SPACES}{A}{3 R}{D}"
MA 175 POSITION N1,N3:? #6;"{A}{2 R}{X} {R}
       {W}{C}{3 SPACES}{Z}{W}{R} {2 R}{D}":POS
       ITION N7,N4:? #6;"{Y}{5 SPACES}{Y}":POS
       ITION N1,N6:? #6;"{A}{2 R} {2 R}{X}{E}
       {3 SPACES}{Q}{X}{2 R} {R}{D}"
CE 180 POSITION N8,N7:? #6;"{A}{3 R}{D}":POSIT
       ION N8,N9:? #6;"{X}{3 R}{X}":RESTORE 95
       0:GOTO E
EA 185 SETCOLOR N0,N5,N6:SETCOLOR N4,N5,N6:I=M
       +21:POSITION N5,N0:? #6;"{W}{6 R}{W}":P
       OSITION N5,N2:? #6;"{Y}{6 SPACES}{A}
       {2 R}{E}"
```

```
OH 190 POSITION N1,N3:? #6;"{A}{R} {R}{D}
       {Q}{R} {R}{D}  {Z} {R}{D}":POSITION N5,
       N4:? #6;"{Z}{W}{R}{C}{3 SPACES}{Y}":POS
       ITION N1,N6:? #6;"{A}{R} {R}{W}{X}{R}
       {W}{R} {W}{X}{3 R} {R}{D}"
GD 195 POSITION N5,N7:? #6;"{Y}  {Y}  {Y}":POS
       ITION N5,N9:? #6;"{X}{2 R}{X}{2 R}{X}":
       RESTORE 955:GOTO E
GC 200 GRAPHICS N1
MN 205 IF QW=1 THEN GRAPHICS N1+16
IF 210 POKE 756,BASE+N2:SETCOLOR N0,N0,N0:SETC
       OLOR N1,N3,N5:SETCOLOR N2,N7,12:SETCOLO
       R N4,N0,N0:GOSUB 860
ND 215 POSITION N4,N2:? #6;"{F}{M}{J}{F}{G}
       {N}{F}{G}{H}{M}{G}":POSITION N5,N3:? #6
       ;"{H}  {G}{N}{F}  {J}":POSITION N5,N4:?
        #6;"{F}{G}{N} ' {N}{F}{G}":POSITION N4
       ,N5:? #6;"{H}  {H}{G}'{F}{J}  {J}"
CG 220 POSITION N4,N6:? #6;"{H}{J}{H}{T}{J}'
       {H}{T}{J}{H}{J}":POSITION N4,N7:? #6;"
       {H}{J}{V}{M}{G}{N}{F}{M}{B}{H}{J}":POSI
       TION N4,N8:? #6;"{H}{J}{V}{H}{J}{N}{H}
       {J}{B}{H}{J}":POSITION N5,N9:? #6;"{H}
       {J} {J}{X}{H} {H}{J}"
HM 225 POSITION N6,10:? #6;"{H} {J}{N}{H} {J}"
       :POSITION N6,11:? #6;"{H}{F} {U} {G}
       {J}":POSITION N6,12:? #6;"{H}{F} {U}
       {G}{J}":POSITION N6,13:? #6;"{H} {H}
       {U}{J} {J}"
EJ 230 POSITION N6,14:? #6;"{B}{H}{K} {L}{J}
       {V}":POSITION N7,15:? #6;"{V}
       {3 SPACES}{B}":POSITION N7,16:? #6;"
       {J}{3 U}{H}"
EA 235 IF QW=1 THEN FOR Z=1 TO 700:NEXT Z:GOTO
        865
GK 240 GOTO 380
GA 245 POSITION A,B:? #6;"E":SOUND N0,25,N0,N8
       :SOUND N0,N0,N0,N0
HD 250 IF C=N1 THEN GOSUB 960
JL 255 IF STICK(N0)=14 THEN S=14:LOCATE A,B-N1
       ,X:IF X=32 THEN 280
JE 260 IF STICK(N0)=13 THEN S=13:LOCATE A,B+N1
       ,X:IF X=32 THEN 290
ND 265 IF STICK(N0)=N7 THEN S=N7:LOCATE A+N1,B
       ,X:IF X=32 THEN 300
JB 270 IF STICK(N0)=11 THEN S=11:LOCATE A-N1,B
       ,X:IF X=32 THEN 315
HD 275 GOTO 255
```

```
PN 280 POSITION A,B:? #6;"■":POSITION A,B-N1:?
    #6;"■":SOUND NØ,5Ø,NØ,N8:SOUND NØ,NØ,N
    Ø,NØ:GOSUB 96Ø
LC 285 POSITION A,B:? #6;" ":B=B-N1:C=N1:GOTO
    K
PM 29Ø POSITION A,B:? #6;"■":SOUND NØ,5Ø,NØ,N8
    :SOUND NØ,NØ,NØ,NØ:POSITION A,B+N1:? #6
    ;"■":GOSUB 96Ø
LB 295 POSITION A,B:? #6;" ":B=B+N1:C=N1:GOTO
    K
ED 3ØØ POSITION A,B:? #6;"■":GOSUB 96Ø
LG 3Ø5 POSITION A,B:? #6;"■":SOUND NØ,5Ø,NØ,N8
    :SOUND NØ,NØ,NØ,NØ:GOSUB 96Ø
KC 31Ø POSITION A,B:? #6;" ":A=A+N1:C=NØ:GOTO
    K
EL 315 POSITION A,B:? #6;"■":GOSUB 96Ø
LF 32Ø POSITION A,B:? #6;"■":SOUND NØ,5Ø,NØ,N8
    :SOUND NØ,NØ,NØ,NØ:GOSUB 96Ø
OM 325 POSITION A,B:? #6;" ":C=NØ:A=A-N1
PD 33Ø IF D=N1 THEN POSITION A,B:? #6;"■":D=NØ
    :GOTO 355
DH 335 LOCATE A+N1,B,X:LOCATE A-N1,B,X1:LOCATE
    A,B+N1,X2:LOCATE A,B-N1,X3
BI 34Ø IF X<32 AND X1<32 OR X2<32 AND X3=1 OR
    X2<32 AND X3=4 OR X2<32 AND X3=25 OR X2
    <32 AND X3=23 THEN D=N1
IK 345 IF X2<32 AND X3=19 THEN D=N1
GM 35Ø GOTO 245
IE 355 IF M<N7 THEN FOR Z=NØ TO M:READ L$,S$:N
    EXT Z:? M$;L$:GOSUB 965
AD 36Ø IF M=N7 THEN 76Ø
BB 365 IF M=N8 THEN 795
AJ 37Ø IF M=N9 THEN 835
DD 375 IF M>N9 THEN ? :? :? "ROOM EMPTY":H=H+L
    :GOTO 11Ø
NG 38Ø IF L=N5 THEN ? M$;" DRAGON!":I=3Ø:M=N1
GL 385 FOR Z=1 TO 3:FOR Z1=8 TO Ø STEP -1:SETC
    OLOR N1,N2,N6:SOUND NØ,6Ø,1Ø,Z1:SOUND N
    1,47,1Ø,Z1:NEXT Z1
BM 39Ø SETCOLOR N1,12,1Ø:SOUND NØ,NØ,NØ,NØ:SOU
    ND N1,NØ,NØ,NØ:FOR Z2=1 TO 5Ø:NEXT Z2:N
    EXT Z
LB 395 IF L=N5 THEN SETCOLOR N1,N3,N5
NJ 4ØØ T=L+N4:O=I+T:F=NØ:IF L>N1 AND M=N3 THEN
    W=N1
LK 4Ø5 GOSUB 845
AJ 41Ø IF F=N1 THEN ? "IT'S STILL ALIVE!"
BJ 415 ? "PRESS TRIGGER TO ATTACK"
AE 42Ø IF STRIG(NØ)=NØ THEN 49Ø
```

```
EN 425 IF STICK(NØ)<15 AND L=N5 THEN ? "YOU'RE
       CUT OFF!"
NM 430 IF STICK(NØ)<15 THEN 440
GL 435 GOTO 420
EC 440 IF V<N6 OR V>N9 THEN Q=N1:GOTO 505
KB 445 IF V>N5 THEN ? :? :? "PRESS TRIGGER TO
       USE RING":GOSUB 960
EC 450 IF STRIG(NØ)=NØ THEN GOTO 465
GF 455 IF STICK(NØ)<15 THEN Q=N1:GOTO 505
GM 460 GOTO 450
DB 465 IF U<N1 THEN V=NØ:? "OUT OF CHARGES!":F
       OR Z=N1 TO 100:NEXT Z:GOTO 505
LI 470 IF V=N6 THEN O=O-12:U=U-N1:GOTO 850
LP 475 IF V=N7 THEN O=O-22:U=U-N1:GOTO 850
MB 480 IF V=N8 THEN O=O-36:U=U-N1:GOTO 850
MC 485 IF V=N9 THEN O=O+15:U=U-N1:GOTO 850
KG 490 X=INT(RND(N1)*10):IF X<=N THEN O=O-P:H=
       H+P:GOTO 500
DJ 495 FOR Z=NØ TO 15:SOUND NØ,15,NØ,Z:NEXT Z:
       FOR Z=15 TO NØ STEP -1:SOUND NØ,15,NØ,Z
       :NEXT Z:GOTO 505
JG 500 FOR Z=NØ TO 15 STEP +N2:SOUND NØ,15,NØ,
       Z:NEXT Z:SOUND NØ,NØ,NØ,NØ
BI 505 X=INT(RND(N1)*10):IF X<=T THEN H=H-I:GO
       TO 515
DA 510 FOR Z=NØ TO 15:SOUND NØ,15,NØ,Z:NEXT Z:
       FOR Z=15 TO NØ STEP -1:SOUND NØ,15,NØ,Z
       :NEXT Z:GOTO 535
AI 515 IF L=N5 THEN 865
JF 520 FOR Z=NØ TO 12 STEP +N2:SOUND NØ,15,NØ,
       Z:NEXT Z:SOUND NØ,NØ,NØ,NØ
HJ 525 IF J=N8 THEN W=NØ
KI 530 X=INT(RND(N1)*N8):IF W=N1 AND X<L THEN
       ? "POISONED BY MONSTER!":G=NØ:GOTO 895
AE 535 IF H<N1 THEN 895
LK 540 IF Q=N1 THEN S$=" ":GOSUB 965
HK 545 IF Q=N1 THEN Q=NØ:GOTO 110
HB 550 IF O>NØ THEN F=N1:GOTO 405
LG 555 ? :? :? "YOU WON!":S$=" ":GOSUB 965
HJ 560 IF J=N8 THEN W=N1
OF 565 IF W=N1 AND L>N1 AND M=N3 THEN ? :? M$;
       " MAGIC SWORD!":? :P=P+L:S$="j":GOSUB 9
       65
GP 570 IF W=N1 AND L>N1 AND M=N3 THEN FOR Z=1
       TO 300:NEXT Z:S$=" ":GOSUB 965
BL 575 X=INT(100*RND(N1)+N1):IF X>10 THEN 665
IM 580 ? M$;" RING!":U=INT(N5*RND(N1)+N1):GOSU
       B 990
GD 585 S$="k":GOSUB 965
```

```
OP 590 IF STRIG(NØ)=NØ THEN V=X:? :? :Z2=655:O
       N V GOTO 605,610,615,620,625,630,635,64
       Ø,645,650
OM 595 IF STICK(NØ)<15 THEN 660
GN 600 GOTO 590
LJ 605 L=N2:GOTO 855
LG 610 L=N3:GOTO 855
LM 615 L=N4:GOTO 855
AP 620 G=G*N2:? "GOLD DOUBLED!":GOTO Z2
MD 625 G=NØ:? "GOLD LOST":GOTO Z2
NC 630 ? "RING OF ICE!":GOTO Z2
CM 635 ? "RING OF FIRE!":GOTO Z2
FG 640 ? "RING OF DESTRUCTION!":GOTO Z2
CH 645 ? "RING OF LIFE!":GOTO Z2
GI 650 ? "NO EFFECT"
KI 655 FOR Z=N1 TO 500:NEXT Z
PC 660 S$=" ":GOSUB 965
DM 665 X=INT(100*RND(N1)+N1):IF X>N9 THEN 750
LO 670 GOSUB 845
GP 675 IF X=N1 AND J=N8 THEN X=N8
MN 680 ? :? :? M$;" POTION!":J=X:Z2=750:GOSUB
       990
GF 685 S$="1":GOSUB 965
FF 690 FOR Z=1 TO 100:NEXT Z
HF 695 IF STRIG(NØ)=NØ THEN ON J GOTO 710,715,
       720,725,730,735,740,745,745
OA 700 IF STICK(NØ)<15 THEN 750
HJ 705 GOTO 695
MO 710 ? "POISON! TASTED AWFUL!":G=NØ:GOTO 895
DE 715 H=1:? "HIT POINTS LOST!":GOTO Z2
HH 720 H=H+H:? "HIT POINTS DOUBLED!":GOTO Z2
PD 725 N=N+L:? "DEXTERITY IMPROVED!":GOTO Z2
MN 730 N=N-L:? "DEXTERITY LOST!":GOTO Z2
KF 735 P=P+L:? "STRENGTH IMPROVED!":GOTO Z2
HP 740 P=P-L:? "STRENGTH LOST!":GOTO Z2
GN 745 ? "NO EFFECT"
PC 750 S$=" ":GOSUB 965
HM 755 G=G+INT(RND(1)*10)*(L*L*L):GOTO 110
NG 760 GOSUB 845:S$="{H}":? M$;" STAIRWAY UP":
       GOSUB 990
MG 765 GOSUB 965
FE 770 FOR Z=1 TO 100:NEXT Z
BJ 775 IF STRIG(Ø)=Ø THEN L=L-1:FOR Z=120 TO 8
       Ø STEP -5:SOUND NØ,Z,10,N8:FOR Z1=1 TO
       10:NEXT Z1:SOUND NØ,NØ,NØ,NØ:NEXT Z:GOT
       O 85
KC 780 IF STICK(Ø)=15 THEN 775
PK 785 S$=" ":GOSUB 965
GL 790 GOTO 110
```

```
MG 795 GOSUB 845
AL 800 S$="{J}":? M$;" STAIRWAY DOWN":GOSUB 99
       0
MB 805 GOSUB 965
EP 810 FOR Z=1 TO 100:NEXT Z
BJ 815 IF STRIG(N0)=N0 THEN L=L+N1:FOR Z=80 TO
        120 STEP N5:SOUND N0,Z,10,N8:FOR Z1=1
       TO 10:NEXT Z1:SOUND N0,N0,N0,N0:NEXT Z:
       GOTO 90
OG 820 IF STICK(N0)=15 THEN 815
PF 825 S$=" ":GOSUB 965
GG 830 GOTO 110
NA 835 ? :? :? "TRAP!":POSITION A,B:? #6;"i":F
       OR Z=40 TO 150:SOUND N0,Z,10,N8:NEXT Z:
       FOR Z=N1 TO 10:SOUND N0,15,N0,15:NEXT Z
DK 840 SOUND N0,N0,N0,N0:L=L+N1:GOTO 85
GA 845 ? "STR=";P;"  DEX=";N;"  HP=";H;"  GOLD
       =";G:RETURN
AE 850 FOR Z=1 TO 50:SOUND N0,40,10,N8:SOUND N
       1,100,10,N8:NEXT Z:SOUND N0,N0,N0,N0:SO
       UND N1,N0,N0,N0:GOTO 535
JA 855 ? "TELEPORTED TO LEVEL ";L:FOR Z=1 TO 2
       00:NEXT Z:GOTO 85
FC 860 X=PEEK(16):IF X>127 THEN X=X-128:POKE 1
       6,X:POKE 53774,X:RETURN
JG 865 FOR Z1=N1 TO 10:FOR Z=N1 TO 14:SETCOLOR
        N0,N3,Z:SOUND N0,50,N8,15:NEXT Z:SOUND
        N0,N0,N0,N0:NEXT Z1
PO 870 IF QW=1 THEN FOR Z=1 TO 100:NEXT Z:GOTO
        20
AL 875 IF H<N1 THEN 895
GL 880 IF O>N0 THEN F=N1
ON 885 IF O<N1 THEN G=G+10000:? "DRAGON DEAD!
        GOLD=";G:GOTO 910
OK 890 SETCOLOR N0,N0,N0:GOTO 405
GD 895 ? "PLAYER DEAD!":G=N0:? "GOLD=";G:IF L<
       N5 THEN POSITION A,B:? #6;"R"
HB 900 GOTO 915
AG 905 ? "GOLD=";G
LD 910 ? "YOU SURVIVED!"
PE 915 IF G<N1 THEN FOR Z=N0 TO 255 STEP 10:SO
       UND N0,Z,10,N8:FOR Z1=N1 TO N5:NEXT Z1:
       NEXT Z:SOUND N0,N0,N0,N0:GOTO 925

EG 920 FOR Z=255 TO N0 STEP -10:SOUND N0,Z,10,
       N8:FOR Z1=N1 TO N5:NEXT Z:SOUND N0,N0,N
       0,N0
PN 925 ? "PRESS TRIGGER TO PLAY AGAIN"
DJ 930 IF STRIG(0)=0 THEN 35
HG 935 GOTO 930
```

```
AK 940 DATA N ORC,a, GIANT ANT,o, GIANT RAT,w,
       GIANT SPIDER,{RIGHT}, SKELETON,r, ZOMB
       IE,v, TROGLODYTE,u
KI 945 DATA N OGRE,u, BUG BAT,x, GIANT LIZARD,
       {P}, GAS BAG,t, GORILLA,m, GIANT BADGER
       ,{LEFT}, MAN EATING PLANT,q
PF 950 DATA N EVIL FIGHTER,a, WEREWOLF,a, MUMM
       Y,v, GIANT HORNET,{I}, GIANT SNAKE,p, T
       ROLL,l, CYCLOPS,{
IJ 955 DATA N EVIL WIZARD,s, WRAITH,z, MINOTAU
       R,{UP}, GIANT SCORPION,n, VAMPIRE,
       {DOWN}, TITAN,y, DEMON,{O}
LD 960 FOR Z=N1 TO N9:NEXT Z:RETURN
KL 965 IF S=14 THEN POSITION A,B-N1:? #6;S$
KE 970 IF S=13 THEN POSITION A,B+N1:? #6;S$
MK 975 IF S=N7 THEN POSITION A+N1,B:? #6;S$
KF 980 IF S=11 THEN POSITION A-N1,B:? #6;S$
IG 985 RETURN
HA 990 ? "PRESS TRIGGER TO USE":RETURN
ON 10000 BASE=PEEK(106)-8:CHSET=BASE*256:IF PE
         EK(CHSET+512)=229 THEN RETURN
OM 10010 GRAPHICS 18:SETCOLOR N4,N3,N0:SETCOLO
         R N0,N1,10:POSITION N2,N2:? #N6;"THE
         DRAGON'S DEN"
IN 10020 POSITION N2,N5:? #N6;"PLEASE WAIT FOR
         ":POSITION N3,N7:? #N6;"35 SECONDS...
         "
HN 15000 FOR I=N0 TO 1023:POKE CHSET+I,PEEK(57
         344+I):NEXT I
BG 15001 RESTORE 15005
CI 15002 READ A:IF A<N0 THEN RETURN
EH 15003 FOR J=N0 TO N7:READ B:POKE CHSET+A*N8
         +J,B:NEXT J
CL 15004 GOTO 15002
AH 15005 DATA 64,229,22,215,124,30,60,106,161
PP 15006 DATA 73,0,230,234,234,28,120,150,230
DG 15007 DATA 79,56,124,215,253,189,184,68,68
LD 15008 DATA 80,254,225,116,56,62,92,31,23
BK 15009 DATA 89,24,24,24,24,24,24,24,24
LE 15010 DATA 92,56,16,124,124,124,56,56,56
EC 15011 DATA 93,12,30,61,61,60,62,126,18
BI 15012 DATA 94,0,0,0,59,126,255,128,64
PJ 15013 DATA 95,36,149,93,250,255,90,149,37
EC 15014 DATA 97,0,48,120,120,120,48,48,0
MC 15015 DATA 98,0,26,60,88,28,36,64,0
HJ 15016 DATA 99,0,12,30,45,12,58,1,0
KN 15017 DATA 100,0,88,60,26,56,36,2,0
HF 15018 DATA 101,0,48,120,180,48,92,128,0
PC 15019 DATA 102,0,0,0,0,48,120,120,120
```

```
GM 15020 DATA 103,48,48,16,16,0,0,0,0
AK 15021 DATA 104,0,16,38,56,126,32,16,0
EM 15022 DATA 105,0,126,66,66,66,66,126,0
EI 15023 DATA 106,16,16,16,16,16,16,56,16
DE 15024 DATA 107,0,0,28,20,28,0,0,0
BE 15025 DATA 108,0,0,8,8,28,28,0,0
GC 15026 DATA 109,24,60,255,189,189,189,219,24
GP 15027 DATA 110,0,32,80,128,190,255,42,0
LD 15028 DATA 111,0,74,42,237,255,237,42,74
AI 15029 DATA 112,0,8,20,16,12,126,255,0
KM 15030 DATA 113,28,19,18,16,56,16,124,124
KI 15031 DATA 114,8,28,42,8,28,20,20,0
DL 15032 DATA 115,0,13,62,92,60,124,254,0
LH 15033 DATA 116,28,62,127,62,28,215,36,82
OC 15034 DATA 117,48,120,252,180,180,48,48,0
OE 15035 DATA 118,0,48,60,48,56,40,32,0
HB 15036 DATA 119,0,0,60,126,255,128,124,0
JI 15037 DATA 120,0,238,124,16,0,0,0,0
EK 15038 DATA 121,24,126,255,255,255,255,60,60
OD 15039 DATA 122,62,107,255,119,62,28,12,24
FA 15040 DATA 123,56,108,254,254,254,254,56,56
NA 15041 DATA 124,0,0,24,60,126,90,24,0
HE 15042 DATA -1
```

Chapter 2

Memory Match

Dave Miller

Memory (yours, not the computer's) is the key to winning this game of matching shapes. For two players.

In this popular memory game, you are presented with a game board made up of 21 squares. Each square is identified by a letter, and different shapes are hidden behind the squares. The object is to match the hidden shapes by picking the appropriate squares.

Each player picks two squares per turn. If they match, the player's score increases by ten points and the player gets another turn. One extra point is added for each subsequent match; for instance, the tenth pair is worth 20 points when correctly matched. If the two squares you want revealed don't match, the squares will again go blank, no points will be awarded, and the other player takes a turn.

The board also contains one penalty square. The penalty square costs you five points (and forfeits your turn) every time you pick it, so you would be well advised to remember where it is.

The game continues until the final pair of characters have been matched. At that time the program checks to see which player has the most points, or if a tie exists. Pressing the ESC key will end the game, or you can press any other key to play again.

Built-in checks make sure that you enter only valid letters (A–U) when selecting a square. If you do hit an invalid key, you are notified by a message at the top of the screen and can then try again. The program also will not allow you to choose the same square for your first and second guesses in any given turn.

Memory Match

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
OM 10  GRAPHICS 17:POKE 708,44:POKE 709,52:POKE
       712,56
GD 20  DL=PEEK(560)+256*PEEK(561)+4:POKE DL+9,7
AH 30  POSITION 4,8:? #6;"MEMORY MATCH"
FN 40  POSITION 2,15:? #6;"please stand by"
EP 47  REM
```

```
GK 48 REM *** ENABLE CHSET ***
FB 49 REM
MK 50 GOSUB 670:? #6;"{CLEAR}":POKE 756,CHSET/
      256
FA 57 REM
GA 58 REM *** SET UP GAME BOARD ***
FC 59 REM
BM 60 POKE 559,0:POKE 708,142:POKE 709,134:POK
      E 711,30:POKE 712,0
JA 70 DIM CHR(21),COLR(21),MATCH(11):POKE DL+2
      ,7:POKE DL+21,7:POKE DL+9,6
KD 80 PLAYER=0:SCORE1=0:SCORE2=0:COUNT=0:X=0:L
      =33
AF 90 POSITION 3,1:? #6;"MEMORY  MATCH"
OE 100 SCR=PEEK(88)+256*PEEK(89):POKE 16,64:PO
       KE 53774,64
EC 110 FOR A=2 TO 16 STEP 2:FOR B=3 TO 17:POKE
        SCR+A+B*20,75:NEXT B:NEXT A
EG 120 FOR A=1 TO 4:FOR B=42 TO 56:POKE SCR+B+
       X,76:NEXT B:X=X+100:NEXT A
DJ 130 FOR A=63 TO 75 STEP 2:POKE SCR+A,L:L=L+
       1:NEXT A
JM 140 FOR A=163 TO 175 STEP 2:POKE SCR+A,L:L=
       L+1:NEXT A
JP 150 FOR A=263 TO 275 STEP 2:POKE SCR+A,L:L=
       L+1:NEXT A
EP 160 GOSUB 540:POKE 559,34
IC 167 REM
KF 168 REM *** CHARACTER PLACEMENT ***
IE 169 REM
MG 170 RESTORE INT(RND(0)*9)+180:FOR A=1 TO 11
       :READ CHR,COLR:CHR(A)=CHR:COLR(A)=COLR:
       NEXT A
NM 171 FOR A=12 TO 21:READ CHR,COLR:CHR(A)=CHR
       :COLR(A)=COLR:NEXT A
FB 180 DATA 129,14,130,20,131,30,132,186,133,8
       6,134,76,135,40,136,218,0,0,138,54,137,
       198
JK 181 DATA 130,20,138,54,129,14,137,198,131,3
       0,136,218,132,186,135,40,133,86,134,76
FD 182 DATA 131,30,135,40,0,0,138,54,129,14,13
       2,186,130,20,137,198,134,76,136,218,133
       ,86
JM 183 DATA 132,186,138,54,129,14,137,198,134,
       76,133,86,136,218,135,40,131,30,130,20
FF 184 DATA 129,14,0,0,131,30,136,218,130,20,1
       38,54,133,86,135,40,134,76,132,186,137,
       198
JG 185 DATA 138,54,137,198,136,218,135,40,134,
       76,133,86,132,186,131,30,130,20,129,14
```

```
FH 186 DATA 129,14,0,0,131,30,136,218,130,20,1
       38,54,133,86,135,40,134,76,132,186,137,
       198
KA 187 DATA 132,186,138,54,129,14,137,198,134,
       76,133,86,136,218,135,40,131,30,130,20
FJ 188 DATA 129,14,130,20,131,30,132,186,133,8
       6,134,76,135,40,136,218,0,0,138,54,137,
       198
KC 189 DATA 138,54,137,198,136,218,135,40,134,
       76,133,86,132,186,131,30,130,20,129,14
NJ 190 FOR X=1 TO 10:MATCH(X)=0:NEXT X
HO 226 REM
NH 227 REM *** READ AND MATCH ***
PC 228 REM ***(3 SPACES)KEY PRESS(4 SPACES)***
IB 229 REM
KG 230 GUESS=0:TEMP1=0:TEMP2=0
LC 240 GOSUB 570
PF 242 P=PEEK(764):IF P=255 THEN 242
JM 245 FOR S=1 TO 5:SOUND 0,50*(S+GUESS+10),10
       ,14:NEXT S:SOUND 0,0,0,0
LD 250 GOSUB 570
IJ 260 RESTORE 270:TRAP 575:FOR X=1 TO 21:READ
        Y,Z
DN 270 DATA 63,103,21,105,18,107,58,109,42,111
       ,56,113,61,115,57,203
GI 280 DATA 13,205,1,207,5,209,0,211,37,213,35
       ,215,8,303,10,305
DO 290 DATA 47,307,40,309,62,311,45,313,11,315
AL 300 IF P<>Y THEN NEXT X
GH 305 IF  NOT CHR(X) THEN 370
JD 307 FOR C=1 TO 10:IF CHR(X)=MATCH(C) THEN 5
       75
BN 308 NEXT C
GM 310 GUESS=GUESS+1:POKE 66,0
HB 320 IF GUESS=1 THEN TEMP1=CHR(X):POKE 710,C
       OLR(X):POKE SCR+Z,CHR(X):A=Z:GOSUB 570:
       GOTO 240
DM 330 IF GUESS=2 THEN TEMP2=CHR(X):IF TEMP1=T
       EMP2 THEN B=Z:POKE SCR+Z,CHR(X):GOSUB 5
       70:GOTO 360
PK 335 POKE 711,COLR(X):POKE SCR+Z,CHR(X)+64:B
       =Z:GOSUB 570:GOTO 360
CO 340 NEXT X
LN 350 GOSUB 570:GOTO 240
IC 356 REM
ID 357 REM *** PENALTY FOR ***
LJ 358 REM *** BLANK SQUARE ***
IF 359 REM
JA 360 IF TEMP1 AND TEMP2 THEN 430
```

```
AH 370 POKE 66,0:SOUND 0,42,2,12:IF PLAYER<>1
       THEN 400
BE 380 SCORE1=SCORE1-5:POKE 712,52:FOR W=1 TO
       100:NEXT W
CF 390 GOSUB 580:POKE 712,0:POSITION 3,1:? #6;
       "{3 SPACES}penalty{3 SPACES}"
DE 395 FOR W=1 TO 300:NEXT W:POSITION 3,1:? #6
       ;"MEMORY  MATCH":GOTO 530
AP 400 SCORE2=SCORE2-5:POKE 712,52:FOR W=1 TO
       100:NEXT W
BP 410 GOSUB 590:POKE 712,0:POSITION 3,1:? #6;
       "{3 SPACES}penalty{3 SPACES}"
CN 415 FOR W=1 TO 300:NEXT W:POSITION 3,1:? #6
       :"MEMORY  MATCH":GOTO 530
IB 427 REM
PN 428 REM *** CORRECT MATCH ***
ID 429 REM
DF 430 IF TEMP1<>TEMP2 THEN 520
PP 431 SOUND 0,100,10,10:FOR W=1 TO 200:NEXT W
       :SOUND 0,0,0,0:COUNT=COUNT+1
IE 432 MATCH(COUNT)=TEMP1
GM 450 IF PLAYER<>1 THEN 490
MI 460 SCORE1=SCORE1+10+COUNT:GOSUB 580
GL 470 POKE SCR+A-40,0:POKE SCR+B-40,0:IF COUN
       T=10 THEN 600
OD 480 PLAYER=2:GOTO 530
MD 490 SCORE2=SCORE2+10+COUNT:GOSUB 590
GF 500 POKE SCR+A-40,0:POKE SCR+B-40,0:IF COUN
       T=10 THEN 600
NM 510 PLAYER=1:GOTO 530
EL 520 FOR W=1 TO 500:NEXT W
IC 527 REM
HM 528 REM ***  ERASE CHARACTER ***
IE 529 REM
PD 530 SOUND 0,0,0,0:GOSUB 540:POKE SCR+A,0:PO
       KE SCR+B,0:GOTO 230
ID 537 REM
ND 538 REM *** UPDATE PLAYER'S TURN ***
IF 539 REM
GN 540 POKE 66,0:PLAYER=PLAYER+1:IF PLAYER>2 T
       HEN PLAYER=1
MA 550 IF PLAYER=1 THEN POSITION 0,20:? #6;"pl
       ayer {Q}":POSITION 12,20:? #6;"
       {8 SPACES}":RETURN
ND 560 IF PLAYER=2 THEN POSITION 12,20:? #6;"p
       layer {R}":POSITION 0,20:? #6;"
       {8 SPACES}":RETURN
IA 570 POKE 764,255:POKE 66,1:RETURN
EP 571 POKE 764,255:RETURN
IC 572 REM
```

```
IF 573 REM *** INVALID KEY PRESSED ***
IE 574 REM
LC 575 POSITION 3,1:? #6;" invalid key ":FOR W
       =1 TO 200:NEXT W
EH 576 POSITION 3,1:? #6;"MEMORY  MATCH":GOTO
       240
IH 577 REM
OG 578 REM *** UPDATE CURRENT SCORE ***
IJ 579 REM
GI 580 POSITION 0,22:? #6;"{5 SPACES}":POSITIO
       N 2,22:? #6;SCORE1:RETURN
NA 590 POSITION 12,22:? #6;"{5 SPACES}":POSITI
       ON 14,22:? #6;SCORE2:RETURN
IJ 597 REM
AM 598 REM ***{4 SPACES}END OF GAME{4 SPACES}*
       **
IL 599 REM
HO 600 POKE 66,0:FOR W=1 TO 100:C=PEEK(53770):
       POKE 712,C:NEXT W:POKE 712,0
DG 610 IF SCORE1>SCORE2 THEN POSITION 1,22:? #
       6;" wins"
GI 620 IF SCORE1<SCORE2 THEN POSITION 13,22:?
       #6;" wins"
NP 630 IF SCORE1=SCORE2 THEN POSITION 8,22:? #
       6;" tie"
KF 635 POSITION 0,20:? #6;"player {Q}":POSITIO
       N 3,0:? #6;"esc TO END OR"
BF 640 POSITION 12,20:? #6;"player {R}":POSITI
       ON 3,1:? #6;"PRESS any KEY":GOSUB 571
PL 650 P=PEEK(764):IF P=255 THEN 650
IK 655 IF P=28 THEN CLR :GOSUB 571:POKE 66,0:G
       RAPHICS 0:END
BK 660 GOSUB 571:GRAPHICS 17:CLR :DL=PEEK(560)
       +256*PEEK(561)+4:GOTO 50
IH 667 REM
DN 668 REM ***{3 SPACES}REDEFINE CHSET  ***
IJ 669 REM
AN 670 CHSET=(PEEK(106)-8)*256:IF PEEK(CHSET+9
       *8)=28 THEN RETURN
KA 680 FOR I=0 TO 512:POKE CHSET+I,PEEK(57344+
       I):NEXT I:RESTORE 720
LH 690 READ A:IF A=-1 THEN SOUND 0,0,0,0:RETUR
       N
HG 700 FOR J=0 TO 7:READ B:SOUND 0,B,10,10:POK
       E CHSET+A*8+J,B:NEXT J
HA 710 GOTO 690
AN 720 DATA 1,126,126,126,60,24,24,24,60
KN 730 DATA 2,15,15,15,15,31,127,127,7
ED 740 DATA 3,0,12,25,51,255,24,12,6
```

```
QL 750  DATA 4,24,28,30,16,17,255,127,63
OD 760  DATA 5,20,93,107,62,20,28,20,119
CD 770  DATA 6,28,126,52,62,60,48,127,124
KM 780  DATA 7,102,255,219,255,102,60,36,102
IB 790  DATA 8,60,103,253,244,224,116,29,15
PK 800  DATA 9,28,127,255,126,28,8,8,255
PD 810  DATA 10,0,0,0,120,200,254,255,102
LC 820  DATA 11,52,44,52,44,52,44,52,44
FC 830  DATA 12,126,219,165,219,219,165,219,126
BE 840  DATA -1
```

Chapter 3

Education

Alphabone Hunt

Glenn M. Varano

Learn the alphabet while sniffing out bones in this delightful alphabetizing game for children ages five to nine.

I use an Atari 400 in my first-grade classroom, but finding quality programs at the appropriate level has proved to be difficult. As a result, I've developed several educational programs myself. Here is one of them, an alphabetical order program (for ages five to nine) called "Alphabone Hunt."

After a brief initialization, during which characters are redefined and player/missile graphics are enabled, the screen shows a dog in a doghouse and ten bones scattered about the yard. A starting letter is shown at the bottom of the screen, and the child must locate the next ten letters in order.

Letters are revealed by moving the dog (with the joystick) until it touches the middle of a bone. That reveals a hidden letter. If it is the next letter in the alphabetical sequence, the child pushes the fire button; if the child is correct, the letter will take its place at the bottom of the screen. The dog wraps around the screen horizontally but may not cross the fences at the top and bottom.

After an incorrect answer, there will be a time penalty and the dog is returned to the doghouse. Obviously, the child will do best by remembering the locations of as many letters as possible.

The game continues until all letters have been placed in order. Total time will be displayed, and a short song will be played. Pushing the START button during play resets the game to the beginning.

I've included numerous REM statements for those who would like to enhance the game. All REM statements can be safely omitted to save on typing and memory.

Alphabone Hunt

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
PB 10 GOSUB 32000:CLR
GC 15 GOSUB 10000:GOSUB 20000:GOSUB 20900:GOSU
      B 6000:POKE 20,0:POKE 19,0:POKE 18,0:GOT
      O 100:REM RESET TIMER
OF 40 S=STICK(0):REM JOYSTICK ROUTINE----FROM
      COMPUTE!'S 2ND BOOK OF ATARI--PAGE 2
```

```
IP 50 DX=(S=5 OR S=6 OR S=7)-(S=9 OR S=10 OR S
      =11)
MB 60 DY=(S=5 OR S=9 OR S=13)-(S=6 OR S=10 OR
      S=14)
LC 65 POKE 53278,1:REM RESET COLLISION REGISTE
      R
EH 70 RETURN
GP 100 S=0:GOSUB 40
BC 105 IF PEEK(53279)=6 THEN GAME=1:GOTO 15:RE
       M START BUTTON TO RESTART GAME
PN 110 IF  NOT (DX OR DY) THEN 100
CM 115 POKE 53279,0
GH 120 X=X+DX:IF X>212 THEN X=35
FL 122 IF X<35 THEN X=212:REM LETS DOG WRAP AR
       OUND HORIZONTALLY
HA 125 POKE 53248,X:REM HORIZ POSITION
EK 130 Y=Y+DY:IF Y>85 THEN Y=85
NC 132 IF Y<20 THEN Y=20:REM LIMITS DOG VERTIC
       AL MOVEMENT
FA 133 IF DX<0 THEN P0$(Y,Y+8)=T$:GOTO 140
BN 135 P0$(Y,Y+8)=S$:REM CHANGES DOG FACING
BH 140 IF PEEK(53252)<>1 OR DX=0 THEN 100:REM
       NO COLLISION
IK 149 REM ***COLLISION WITH BONE***
LN 150 K=((X-56)/16)+1:IF K<>INT(K) THEN 100:R
       EM CHECKS FOR CORRECT HORIZ POS
IL 160 ON K GOSUB 501,502,503,504,505,506,507,
       508,509,510
FC 170 POSITION XX,YY:? #6;BL$:POSITION XX,YY:
       ? #6;CHR$(LTR(K)):FOR T=1 TO 200:NEXT T
       :POSITION XX,YY:? #6;"!":GOTO 7000
LN 500 REM ***VARIABLES FOR POSITIONS***
CP 501 XX=1:YY=8:RETURN
CM 502 XX=3:YY=2:RETURN
FP 503 XX=5:YY=11:RETURN
GH 504 XX=7:YY=16:RETURN
DF 505 XX=9:YY=2:RETURN
JB 506 XX=11:YY=13:RETURN
GE 507 XX=13:YY=4:RETURN
JI 508 XX=15:YY=14:RETURN
JJ 509 XX=17:YY=12:RETURN
GF 510 XX=19:YY=5:RETURN
BE 3999 REM ***GAME OVER***
IL 4000 TIME=INT((PEEK(19)*256+PEEK(20))/60):S
        EC=TIME-(INT(TIME/60)*60)
DF 4005 POSITION 9,10:? #6;INT(TIME/60);":";:I
        F SEC>9 THEN ? #6;SEC:GOTO 4010
CH 4006 ? #6;"0";SEC
AL 4010 GOSUB 6000
CH 4020 GAME=1:GOTO 15
```

```
PH 5999 REM ***SONG***
LB 6000 RESTORE 6100
KB 6010 READ N,D
FG 6020 IF N=-1 THEN SOUND 0,0,0,0:RETURN
KL 6030 SOUND 0,N,10,10:FOR S=1 TO D:NEXT S
MK 6040 GOTO 6010
BM 6100 DATA 81,64,60,64,81,64,96,64,121,128,7
        2,32,81,32,96,32,81,64,108,256
HA 6110 DATA 81,64,64,64,72,64,81,64,91,128,64
        ,32,72,64,81,256
MD 6150 DATA -1,-1
GD 6499 REM ***GOOD SOUND***
GH 6500 FOR N=25 TO 1 STEP -1:SOUND 0,N,10,10:
        NEXT N:SOUND 0,0,0,0
KM 6510 RETURN
MI 6999 REM ***CHECK FOR LOW LETTER***
GM 7000 FOR L=1 TO 10
KD 7010 IF CHR$(LTR(K))<>LOW$(M,M) AND STRIG(0
        )=0 THEN GOSUB 8000:GOTO 100
GP 7015 IF CHR$(LTR(K))=LOW$(M,M) AND STRIG(0)
        =0 THEN GOSUB 7500:GOTO 100
FL 7018 NEXT L
JD 7020 GOTO 100
OE 7499 REM ***MOVES LETTER TO BOTTOM OF SCREE
        N***
OI 7500 M=M+1:POSITION XX,YY:? #6;" ":POSITION
         M+3,21:GOSUB 6500:? #6;CHR$(LTR(K))
OI 7510 IF M>10 THEN GOTO 4000:REM GAME OVER
KO 7520 RETURN
NP 7999 REM ***PENALTY***
CB 8000 FOR N=1 TO 500:SOUND 0,N,8,10:NEXT N:S
        OUND 0,0,0,0:Y=45:X=129:GOSUB 20110
KJ 8010 RETURN
LC 9999 REM ***DRAW SCREEN***
FJ 10000 IF GAME>0 THEN ? #6;CHR$(125):GOTO 10
         010
JE 10005 GRAPHICS 17:SETCOLOR 4,0,4:SETCOLOR 1
         ,2,14:SETCOLOR 0,15,10:SETCOLOR 2,8,0
HH 10010 POKE 756,PEEK(106)+5:REM RESETS POINT
         ER TO NEW CHARACTER SET
BM 10015 POSITION 3,2:? #6;"!{5 SPACES}!":POSI
         TION 19,5:? #6;"!":REM !=BONE
FE 10020 POSITION 1,8:? #6;"!":POSITION 13,4:?
          #6;"!"
OF 10030 POSITION 11,13:? #6;"!":POSITION 15,1
         4:? #6;"!"
IK 10040 POSITION 5,11:? #6;"!":POSITION 7,16:
         ? #6;"!"
NE 10050 POSITION 17,12:? #6;"!"
```

```
NM 10060 POSITION 9,7:? #6;"■■■":POSITION 9,8:
         ? #6;"■■■":POSITION 9,9:? #6;"■ ■":RE
         M DOGHOUSE
HP 10070 POSITION 0,19:? #6;"#{C}#{C}#{C}#{C}#
         {C}#{C}#{C}#{C}#{C}#{C}#{C}#{C}":REM FENCE
ML 10075 POSITION 0,0:? #6;"#{C}#{C}#{C}#{C}#
         {C}#{C}#{C}#{C}#{C}#{C}#{C}#{C}"
NK 10090 RETURN
FE 19999 REM ***PM INIT***USE STRINGS TO DEFIN
         E AND MOVE PLAYERS VERTICALLY
CM 20000 IF GAME>0 THEN 20110
ML 20010 DIM X$(1)
BG 20020 A=ADR(X$)
EL 20040 B=INT((A-512)/1024+1)*1024
PO 20050 DIM F$(B-A+511)
GM 20060 DIM P0$(128),P1$(128),T$(8),S$(8),LTR
         (10),BL$(1),REPEAT$(10),LOW$(10):REPE
         AT$="{10 SPACES}":BL$=" "
BL 20070 POKE 559,46:POKE 53277,3:POKE 54279,I
         NT(B/256)
KG 20080 POKE 623,4:REM SETS PLAYFIELD PRIORIT
         Y OVER PLAYER
BL 20090 POKE 704,50:REM PLAYER0(DOG) COLOR
CK 20100 S$="{,}DN{TAB}┤D■{,}":T$="{,}":T$(2)=
         CHR$(34):T$(3)="r{DELETE}>":T$(6)=CHR
         $(34):T$(7)="w{,}":REM STRINGS FOR DO
         GS
PH 20110 P0$=CHR$(0):P0$(128)=CHR$(0):P0$(2)=P
         0$:REM CLEARS PLAYER0
NF 20120 RETURN
EF 20899 REM ***RESTART INIT***
FG 20900 REPEAT$="{10 SPACES}":M=1
NA 20910 Y=50:X=129:REM STARTING POSITION OF D
         OG
FG 20920 P0$(Y,Y+LEN(S$))=S$
LH 20930 POKE 53248,X
GJ 20999 REM ***RANDOMLY SELECTS LETTERS FOR B
         ONES
LE 21000 R=INT(RND(0)*16)
JM 21005 FOR K=1 TO 10
AD 21010 J=INT(RND(0)*10)+1
EI 21020 IF REPEAT$(J,J)="*" THEN 21010
FA 21030 REPEAT$(J,J)="*":LTR(K)=R+97+J:LOW$(J
         ,J)=CHR$(R+97+J)
IB 21040 NEXT K
HM 21050 POSITION 4,21:? #6;CHR$(R+97)
NJ 21060 RETURN
PD 31999 REM
```

```
DA 32000 ? CHR$(125):POKE 106,PEEK(106)-9:GRAP
         HICS 18:START=(PEEK(106)+5)*256:POKE
         756,START/256:POKE 752,1
JG 32010 POSITION 3,2:? #6;"ALPHABONE HUNT":PO
         SITION 9,5:? #6;"BY":POSITION 5,6:? #
         6;"G.M. VARANO"
MI 32015 POSITION 3,9:? #6;"please stand by"
NP 32020 FOR Z=0 TO 1023:POKE START+Z,PEEK(573
         44+Z):NEXT Z:RESTORE 32100
JF 32030 READ X:IF X=-1 THEN RESTORE :RETURN
PB 32040 FOR Y=0 TO 7:READ Z:POKE X+Y+START,Z:
         NEXT Y:GOTO 32030
KH 32100 DATA 40,128,192,224,240,248,252,254,2
         55
DD 32101 DATA 8,0,102,255,255,255,102,0,0
OB 32102 DATA 56,1,3,7,15,31,63,127,255
PA 32103 DATA 256,255,255,255,255,255,255,255,
         255
HC 32104 DATA -1
```

Pyramid Math

Stephen Levy

Colorful graphics and exciting sound make "Pyramid Math" an excellent math tutor for young children. It's a fun-to-play game too.

"Pyramid Math" is a simple and straightforward math contest for two players. As each player answers a problem correctly, he or she builds another part of the pyramid. The winner is the player who first completes a pyramid by answering ten problems correctly.

Players choose addition, subtraction, multiplication, or division problems, and the numbers used in the problems are generated randomly. The upper limits can be changed by adjusting the italicized number, as shown below.

Note that the actual upper limit is one less than the number given. For example, if the italicized number is 50, the upper limit is 49. Be careful with subtraction; do not make Q2's limit greater than Q1's. Note that division is presently set for one-digit answers and one-digit divisors. Q1 is the divisor and Q2 is the dividend; in other words, problems take the form Q2/Q1.

Operation	Line	
Add	610	Q1 = INT(RND(0)*_50_):Q2 = INT(RND(0)*_50_)
Subtract	810	Q1 = INT(RND(0)*_50_):Q2 = INT(RND(0)*_40_)
Multiply	1010	Q1 = INT(RND(0)*_12_):Q2 = INT(RND(0)*_12_)
Divide	1210	Q1 = INT(RND(0)*_10_):Q2 = INT(RND(0)*_9_)*Q1

If you prefer to use different colors, change line 433. Remember, SETCOLOR 0,8,14 sets the print color to white.

Notice that the problems appear in the text window in large print, while the pyramid is done in graphics mode 3. This is accomplished by lines 410, 420, and 430.

Pyramid Math

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
GA 50  DIM YES$(3),BUZZ$(1),CLEAR$(1),PLAYER1$(
       20),PLAYER2$(20):BUZZ$=CHR$(253):CLEAR$=
       CHR$(125):OPEN #1,4,0,"K:"
MN 60  DIM P$(10):PLAYER1$="  ":PLAYER1$(20)="  "
       :PLAYER1$(2)=PLAYER1$:PLAYER2$=PLAYER1$
BH 100 REM INTRODUCTION
```

```
PP 110 GRAPHICS 18:SETCOLOR 4,4,2:SETCOLOR 0,1
       3,14:TRAP 10000:SETCOLOR 1,12,3
FN 120 POSITION 4,3:PRINT #6;"PYRAMID MATH":PO
       SITION 2,5:PRINT #6;"A TWO PLAYER GAME"
ME 130 POSITION 3,10:PRINT #6;"COMPUTE!'S book
       S":FOR W=1 TO 2500:NEXT W
PK 150 GRAPHICS 1:SETCOLOR 4,13,13:SETCOLOR 2,
       13,13:POSITION 5,5:PRINT #6;"DO YOU NEE
       D":TRAP 10000
DK 155 POSITION 4,7:PRINT #6;"INSTRUCTIONS?"
DM 160 INPUT YES$:IF YES$(1,1)<>"N" AND YES$(1
       ,1)<>"Y" THEN GOTO 10000
KM 170 IF YES$(1,1)="N" THEN 300
LK 180 GRAPHICS 18:SETCOLOR 4,8,4:SETCOLOR 2,1
       3,7
CM 190 POSITION 4,0:PRINT #6;"INSTRUCTIONS":PO
       SITION 0,3:PRINT #6;"THIS IS A TWO PLAY
       ER"
HA 195 POSITION 2,4:PRINT #6;"GAME, PLAYER ONE
       ":POSITION 2,5:PRINT #6;"GOES FIRST, ea
       ch"
MK 200 POSITION 0,6:PRINT #6;"player will be g
       iven":POSITION 0,7:PRINT #6;"a problem
       to solve."
CO 210 POSITION 2,8:PRINT #6;"a correct answer
       ":POSITION 0,9:PRINT #6;"builds the pla
       yer's"
AB 215 POSITION 6,10:PRINT #6;"PYRAMID":POSITI
       ON 3,11:PRINT #6;"PRESS RETURN"
NG 220 IF PEEK(764)=255 THEN 220
MB 225 POKE 764,255:GRAPHICS 18:SETCOLOR 4,6,5
NP 230 POSITION 0,2:PRINT #6;"THE FIRST PLAYER
        TO":POSITION 1,4:PRINT #6;"FINISH BUIL
       DING THE"
JK 235 POSITION 3,6:PRINT #6;"PYRAMID WINS"
CP 240 POSITION 3,9:PRINT #6;"press return"
OE 245 IF PEEK(764)=255 THEN 245
CP 250 POKE 764,255
HN 300 GRAPHICS 2:SETCOLOR 4,9,4:SETCOLOR 0,9,
       12
GF 305 TRAP 10010:LN=300
CF 310 POSITION 4,0:PRINT #6;"PLAYER one's":PO
       SITION 4,1:PRINT #6;"NAME PLEASE..."
AJ 320 INPUT P$;PLAYER1$(1,LEN(P$))=P$:POSITIO
       N 5,2:PRINT #6;PLAYER1$:PRINT CLEAR$
IJ 325 HH=ASC(PLAYER1$(1,1)):IF HH<65 OR HH>90
        THEN 10010
EJ 330 POSITION 4,5:PRINT #6;"PLAYER two's":PO
       SITION 4,6:PRINT #6;"NAME PLEASE..."
```

```
BC 340  INPUT P$:PLAYER2$(1,LEN(P$))=P$:POSITIO
        N 5,7:PRINT #6;PLAYER2$:PRINT CLEAR$
IM 345  HH=ASC(PLAYER2$(1,1)):IF HH<65 OR HH>90
         THEN 10010
BJ 360  GRAPHICS 17:SETCOLOR 4,5,10
GN 370  TRAP 10010:LN=360
HG 380  POSITION 4,0:PRINT #6;"WHAT TYPE":POSIT
        ION 4,1:PRINT #6;"OF PROBLEMS":POSITION
         4,2:PRINT #6;"DO YOU WANT?"
JE 390  POSITION 2,4:PRINT #6;"1 ADDING":POSITI
        ON 2,6:PRINT #6;"2 SUBTRACTING"
CL 395  POSITION 2,8:PRINT #6;"3 MULTIPLYING":P
        OSITION 2,10:PRINT #6;"4 DIVIDING"
HO 400  GET #1,HH:IF HH<49 OR HH>52 THEN 10020
IN 403  HH=HH-48
ME 405  TRAP 40000
GE 410  POKE 82,0:GRAPHICS 3:DL=PEEK(560)+PEEK(
        561)*256
AC 420  IF PEEK(DL)<>66 THEN DL=DL+1:GOTO 420
CL 430  POKE DL,70:POKE DL+3,7:POKE DL+4,6:POKE
         DL+5,65:POKE DL+6,PEEK(DL+7):POKE DL+7
        ,PEEK(DL+8)
JA 433  SETCOLOR 4,8,6:SETCOLOR 2,5,12:SETCOLOR
         0,8,14
KG 435  COLOR 3:FOR LINE=11 TO 13:PLOT 0,LINE:D
        RAWTO 39,LINE:NEXT LINE
KN 437  COLOR 2:FOR LINE=14 TO 16:PLOT 0,LINE:D
        RAWTO 39,LINE:NEXT LINE
DE 440  ON HH GOTO 600,800,1000,1200
FJ 600  R=1:ROW1=10:ROW2=10
LA 610  Q1=INT(RND(0)*50):Q2=INT(RND(0)*50)
HE 615  TRAP 10030:LN=615
II 620  IF R=1 THEN GOSUB 10100
IL 630  IF R=2 THEN GOSUB 10110
EM 640  PRINT "  ";Q1;" + ";Q2;" =";:ON R GOTO
        680,720
NH 680  INPUT ANS:PRINT
PC 690  IF ANS=Q1+Q2 THEN GOSUB 6000
NA 700  IF ANS<>Q1+Q2 THEN PRINT BUZZ$:GOSUB 10
        100:PRINT "{5 SPACES}";Q1;" + ";Q2;" =
        ";Q1+Q2:FOR W=1 TO 700:NEXT W
GD 710  R=2:GOTO 610
NC 720  INPUT ANS:PRINT
OO 730  IF ANS=Q1+Q2 THEN GOSUB 6100
NF 740  IF ANS<>Q1+Q2 THEN PRINT BUZZ$:GOSUB 10
        110:PRINT "{5 SPACES}";Q1;" + ";Q2;" =
        ";Q1+Q2:FOR W=1 TO 700:NEXT W
GG 750  R=1:GOTO 610
FL 800  R=1:ROW1=10:ROW2=10
LB 810  Q1=INT(RND(0)*50):Q2=INT(RND(0)*40)
```

```
DD 812 IF Q1<Q2 THEN 810
HI 815 TRAP 10030:LN=815
IK 820 IF R=1 THEN GOSUB 10100
IN 830 IF R=2 THEN GOSUB 10110
FE 840 PRINT "  ";Q1;" - ";Q2;" =";:ON R GOTO
       880,920
BC 880 INPUT ANS
PG 890 IF ANS=Q1-Q2 THEN GOSUB 6000
NI 900 IF ANS<>Q1-Q2 THEN PRINT BUZZ$:GOSUB 10
       100:PRINT "{4 SPACES}";Q1;" - ";Q2;" =
       ";Q1-Q2:FOR W=1 TO 700:NEXT W
GH 910 R=2:GOTO 810
AN 920 INPUT ANS
PC 930 IF ANS=Q1-Q2 THEN GOSUB 6100
NN 940 IF ANS<>Q1-Q2 THEN PRINT BUZZ$:GOSUB 10
       110:PRINT "{4 SPACES}";Q1;" - ";Q2;" =
       ";Q1-Q2:FOR W=1 TO 700:NEXT W
GK 950 R=1:GOTO 810
IE 1000 R=1:ROW1=10:ROW2=10
NH 1010 Q1=INT(RND(0)*12):Q2=INT(RND(0)*12)
MK 1015 TRAP 10030:LN=1015
LD 1020 IF R=1 THEN GOSUB 10100
LG 1030 IF R=2 THEN GOSUB 10110
PK 1040 PRINT " ";Q1;" X ";Q2;" =";:ON R GOTO
       1080,1120
DL 1080 INPUT ANS
BM 1090 IF ANS=Q1*Q2 THEN GOSUB 6000
CG 1100 IF ANS<>Q1*Q2 THEN PRINT BUZZ$:GOSUB 1
       0100:PRINT "{4 SPACES}";Q1;" X ";Q2;"
       = ";Q1*Q2:FOR W=1 TO 700:NEXT W
LJ 1110 R=2:GOTO 1010
DG 1120 INPUT ANS
BI 1130 IF ANS=Q1*Q2 THEN GOSUB 6100
CL 1140 IF ANS<>Q1*Q2 THEN PRINT BUZZ$:GOSUB 1
       0110:PRINT "{4 SPACES}";Q1;" X ";Q2;"
       = ";Q1*Q2:FOR W=1 TO 700:NEXT W
LM 1150 R=1:GOTO 1010
IG 1200 R=1:ROW1=10:ROW2=10
FJ 1210 Q1=INT(RND(0)*10):Q2=INT(RND(0)*9)*Q1
MH 1211 IF Q1=0 OR Q2=0 THEN 1210
MO 1215 TRAP 10030:LN=1215
LF 1220 IF R=1 THEN GOSUB 10100
LI 1230 IF R=2 THEN GOSUB 10110
LN 1240 PRINT Q2;" DIVIDED BY ";Q1;"=";:ON R G
       OTO 1280,1320
DN 1280 INPUT ANS
CD 1290 IF ANS=Q2/Q1 THEN GOSUB 6000
HD 1300 IF ANS<>Q2/Q1 THEN PRINT BUZZ$:GOSUB 1
       0100:PRINT " ";Q1;" INTO ";Q2;" IS ";Q
       2/Q1:FOR W=1 TO 700:NEXT W
```

```
LN 1310 R=2:GOTO 1210
DI 1320 INPUT ANS
BP 1330 IF ANS=Q2/Q1 THEN GOSUB 6100
HI 1340 IF ANS<>Q2/Q1 THEN PRINT BUZZ$:GOSUB 1
        0110:PRINT " ";Q1;" INTO ";Q2;" IS ";Q
        2/Q1:FOR W=1 TO 700:NEXT W
MA 1350 R=1:GOTO 1210
FD 6000 ROW1=ROW1-1:SOUND 0,70,10,10
IC 6010 IF ROW1=7 OR ROW1=4 OR ROW1=1 THEN COL
        OR 1:GOTO 6040
IH 6020 IF ROW1=8 OR ROW1=5 OR ROW1=2 THEN COL
        OR 2:GOTO 6040
HL 6030 COLOR 3
OM 6040 PLOT 10-ROW1,ROW1:DRAWTO ROW1+10,ROW1:
        FOR W=1 TO 20:NEXT W:SOUND 0,0,0,0
DM 6050 IF ROW1<>0 THEN RETURN
IE 6060 POP :P$=PLAYER1$:GOSUB 7000:GOTO 7100
FI 6100 ROW2=ROW2-1:SOUND 0,200,10,8
IH 6110 IF ROW2=7 OR ROW2=4 OR ROW2=1 THEN COL
        OR 1:GOTO 6140
IN 6120 IF ROW2=8 OR ROW2=5 OR ROW2=2 THEN COL
        OR 3:GOTO 6140
HL 6130 COLOR 2
AF 6140 PLOT 29-ROW2,ROW2:DRAWTO ROW2+29,ROW2:
        FOR W=1 TO 20:NEXT W:SOUND 0,0,0,0
DO 6150 IF ROW2<>0 THEN RETURN
IG 6160 POP :P$=PLAYER2$:GOSUB 7000:GOTO 7100
LG 7000 TIMES=0
NF 7005 FOR XX=1 TO 10
OI 7010 POKE 708,PEEK(709):POKE 709,PEEK(710):
        SOUND 0,XX*20,10,8:POKE 710,PEEK(712):
        POKE 712,PEEK(708)
MA 7020 POKE 709,XX*20:FOR W=1 TO 25:NEXT W:TI
        MES=TIMES+1:NEXT XX
HP 7055 IF TIMES<15 THEN 7005
JP 7056 SOUND 0,0,0,0
KN 7060 RETURN
FC 7100 GRAPHICS 18:SETCOLOR 4,8,6:POKE 82,2
CK 7110 FOR XX=1 TO 10:PRINT #6;" ";P$;" WINS!
        ":NEXT XX
OD 7120 FOR W=1 TO 500:NEXT W:TRAP 40000
BN 7130 GRAPHICS 0:POSITION 5,10:PRINT "Play A
        gain":INPUT YES$
EN 7140 IF YES$(1,1)="Y" THEN RUN
MA 7150 GRAPHICS 17:POSITION 5,5:PRINT #6;"GAM
        E OVER"
IB 7160 FOR W=1 TO 400:NEXT W
KG 7170 END
NM 10000 PRINT BUZZ$:PRINT #6;CLEAR$:GOTO 150
```

```
PL 10010 PRINT BUZZ$:GOTO LN
EP 10030 PRINT BUZZ$:PRINT CLEAR$:GOTO LN
DL 10100 PRINT CLEAR$;" ";PLAYER1$;:RETURN
DN 10110 PRINT CLEAR$;" ";PLAYER2$;:RETURN
```

Chapter 3

Dot Drawing

Robert D. Goeman

Here's a drawing program that lets students create their own connect-the-dots pictures. It helps develop visual skills too.

"Dot Drawing" uses less than 2K of memory for the body of the program. When the program is running, the entire memory requirements total less than 13K. The program is entirely in BASIC and can be modified by those with limited programming skill and an Atari reference manual.

The program places a flashing cursor in the upper left-hand corner of an otherwise darkened screen. Pressing the fire button marks the cursor's location as the starting point for a drawing; in the text window, the X and Y coordinates of that location will be displayed.

As the cursor is moved, each subsequent pressing of the fire button enters a new X and Y point and displays the new coordinates in the text window. To connect the points and draw a picture, press D.

After the drawing is complete, the cursor remains at its last position, allowing you to expand your creation. You can clear the screen by pressing BREAK and running the program. If you attempt to save more than 254 X,Y points without drawing the picture, the program will automatically jump to the drawing routine and then return to the beginning of the program.

About the Program

Lines 1–43 set the graphics mode and control cursor movement by reading the joystick. Line 11 initializes the memory location for storing the X and Y coordinates; that line also initializes variable ST, which counts the number of points entered in a given drawing.

Lines 45–50 read the keyboard and the joystick. POKE 764,255 returns the keyboard to a "no keys pressed" condition after D has been read.

Lines 100–110 are used to enter X coordinates which are greater than 255, since numbers larger than 255 cannot be held in a single memory location. Lines 115–116 enter the present coordinate values and then move the present memory locations ahead in preparation for the next set of coordinate

values. Lines 200–215 then reinitialize memory locations and X, Y values for the drawing routine and plot the first point of that routine.

Lines 220–240 do the actual drawing. T−1 is the number of points held in memory locations, minus the first point which has already been plotted.

Dot Drawing

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
PI 1  X=1:Y=1
JP 10 GRAPHICS 8:SETCOLOR 2,16,1:COLOR 5
IN 11 L=1536:L1=1537:L2=1538:T=0:ST=0
NL 14 IF STICK(0)=15 THEN X=X:Y=Y
AN 15 IF STICK(0)=11 THEN X=X-1
BG 19 IF STICK(0)=14 THEN Y=Y-1
GF 20 IF STICK(0)=9 THEN X=X-1:Y=Y+1
JE 25 IF STICK(0)=10 THEN X=X-1:Y=Y-1
GF 26 IF STICK(0)=5 THEN X=X+1:Y=Y+1
GJ 27 IF STICK(0)=6 THEN X=X+1:Y=Y-1
AM 30 IF STICK(0)=13 THEN Y=Y+1
OC 35 IF STICK(0)=7 THEN X=X+1
BJ 40 IF X<1 THEN X=319
BM 41 IF X>319 THEN X=1
CC 42 IF Y<1 THEN Y=189
CF 43 IF Y>189 THEN Y=1
IF 45 IF PEEK(764)=58 THEN POKE 764,255:GOTO 2
      00
AN 50 IF STRIG(0)=0 THEN T=T+1:SOUND 1,30,10,8
      :FOR P=1 TO 25:NEXT P:SOUND 1,0,0,0:GOTO
       100
DN 55 PLOT X,Y:FOR P=1 TO 15:NEXT P:COLOR 0:PL
      OT X,Y:COLOR 5:GOTO 15
NJ 100 IF X<=255 THEN XX=0
ME 105 ST=ST+1:IF ST>=254 THEN 200
PL 110 IF X>255 THEN XX=X-255:X=255
ME 115 POKE L,X:POKE L1,XX:POKE L2,Y
GL 116 L=L+3:L1=L1+3:L2=L2+3
LO 117 X=X+XX:? X;" ";Y:?
DC 120 GOTO 15
HE 200 L=1536:L1=1537:L2=1538
MI 210 X=PEEK(L):XX=PEEK(L1):Y=PEEK(L2)
LG 215 X=X+XX:PLOT X,Y
IO 220 FOR P=1 TO T-1
FF 222 SOUND 1,220,10,8:FOR PP=1 TO 10:NEXT PP
       :SOUND 1,0,0,0
GM 225 L=L+3:L1=L1+3:L2=L2+3
```

```
MK 230  X=PEEK(L):XX=PEEK(L1):Y=PEEK(L2)
EK 235  X=X+XX:DRAWTO X,Y
LP 237  ? X;" ";Y:?
CF 240  NEXT P
EE 242  IF ST>=254 THEN 1
DG 245  GOTO 11
```

Chapter 3

Art Class

Mark Poesch
Tim Kilby
Steve Steinberg

"Art Class" is an outstanding example of the graphics capabilities of your Atari computer. It can be an excellent teaching tool, a fine introduction to computers, or simply great entertainment for a rainy day.

The Atari GTIA chip is getting to be like the weather—everybody talks about it, but nobody does very much with it.

Until now.

"Art Class" is a drawing program utilizing the GTIA's graphics mode 10. It is designed for use by small children, who can use the cursor much like they would use felt coloring pens.

Coloring with the Cursor

We have deliberately kept Art Class simple so children will be comfortable with the program. Even so, it boasts several features that will be of interest to programmers.

The screen display consists of a blank GRAPHICS 10 screen, a "color palette" showing the numbers 0–8 in nine different colors, and two lines showing the prompts for the commands that are available.

A flashing cursor can be seen on the graphics 10 screen. The joystick is used to move the cursor; holding the red trigger button down allows you to draw with the cursor as it moves. Color 0 is the background color and is used for erasing.

You can switch to any of the available palette colors by typing the number corresponding to the desired color (or by typing 0 for the background). Hitting the CLEAR key clears the screen, and the S and L keys enable you to save or load your drawings to and from disk.

In saving and loading programs, the filename must be entered in the form D1:FILENAME. While you can draw with only eight colors at a time, hitting the N key gives you access to all 128 Atari colors. You can choose any nine for your palette. Changing colors is accomplished by moving the joystick left or right until you have the right color in the right place; then hit the RETURN key to enter your choice.

Short and Powerful

Even though Art Class is a short program, it displays some of Atari's best features. For example, a text window with graphics mode 10. The text window will help children learn to read and follow instructions. But where does that text window come from?

The program begins by taking a regular GRAPHICS 8 screen (with its four-line text window) and modifying it as follows. The display list, those instructions that set up the screen display, is modified in lines 270 and 280 for only two GRAPHICS 0 lines in the text window. Also, two blank lines are inserted in the graphics window near the bottom, the second of which includes an instruction for a display list interrupt (DLI). It's that DLI instruction that is the key to having text and GTIA graphics on the same display.

When GTIA modes are initiated, an internal register called PRIOR (53275) is set for the special modes. That register has to be set differently for modes 0–8. Every screen cycle (that is, every 1/60 second) PRIOR's shadow register at location 623 updates the register for GTIA 10 by setting it to a value of 128. The DLI switches PRIOR's setting to 0 as the electron beam scans across the screen just above the text window. Thus, PRIOR is constantly being set and reset according to the position of the electron beam on the screen.

This setting and resetting must be done in machine code for speed and accurate timing, but that's no problem. The machine language routine is POKEd into position in page 6 and activated through BASIC. The DLI pointer at locations 512 and 513 has to be set; finally, the DLI is turned on by POKEing 54286 with 192.

The DATA for the DLI service routine can be found in lines 210 and 220. In addition to resetting PRIOR, the DLI also sets the various colors at the bottom of the screen. In assembly code, this is the DLI service routine:

```
PHA          ;Save values from
TXA          ;S, Y, and A registers
PHA          ;
TYA          ;
PHA          ;
LDA  #0      ;This value will be stored in PRIOR
LDX  #148    ;Color blue for the text window
```

```
LDY  #12      ;White text characters
STA  $D40A    ;Wait for WSYNC horizontal blank to begin
STA  $D01B    ;Reset PRIOR register to 0
STX  $D018    ;Background text window color register
STY  $D017    ;Text luminance color register
LDA  $2C0     ;Load the current border color
STA  $D01A    ;Store in border color register
PLA           ;Restore X, Y,
TAY           ; and A values
PLA           ;
TAX           ;
PLA           ;
RTI           ;Return from interrupt
```

Now that the screen can display both GTIA graphics and normal text in a text window, a creative and useful display can be made. The colored numbers at the bottom of the screen are normal and inverse numerals plotted in GTIA 10 colors. They are drawn by lines 300–400.

One last feature worth mentioning is the screen clear technique. A string S$ originally dimensioned to 1 is redimensioned to 6560, the size (in bytes) of the graphics window. Lines 400–420 do the redimensioning and relocate the string to screen memory location. Then, when the screen needs clearing, zeros are written to the string almost instantaneously (line 640).

Art Class

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
MP 10 CLR :DIM S$(1),B$(1),C$(1),F$(15),T$(9),
      IO$(6),X(10),Y(10)
CN 20 GOTO 200
ID 30 S=STICK(0)-5:N=STRIG(0):X=X+X(S):Y=Y+Y(S
      ):IF X<0 THEN X=79
ON 40 IF X>79 THEN X=0
BF 50 IF Y<0 THEN Y=160
BI 60 IF Y>160 THEN Y=0
JK 70 LOCATE X,Y,Z:COLOR C+12:PLOT X,Y:DRAWTO
      X,Y+3:COLOR C+8:PLOT X,Y:DRAWTO X,Y+3:CO
      LOR C+4:PLOT X,Y:DRAWTO X,Y+3
MG 80 COLOR C:PLOT X,Y:DRAWTO X,Y+3
PI 90 IF  NOT N THEN 110
AG 100 COLOR Z:PLOT X,Y:DRAWTO X,Y+3
KD 110 IF PEEK(764)=255 THEN 30
DP 120 GET #2,K
```

```
FL 130  IF K>47 AND K<58 THEN C=K-48:Z=C:GOTO 3
        Ø
OA 140  IF K=83 THEN 46Ø
NO 150  IF K=76 THEN 5ØØ
OF 160  IF K=78 THEN 54Ø
JP 170  IF K=6Ø OR K=125 THEN 64Ø
DF 180  GOTO 3Ø
MG 190  ? C$;" S> Save Picture{3 SPACES}N> New
        Colors{8 SPACES}L> Load Picture
        {3 SPACES}CLEAR> Clear Screen";:GOTO 3Ø
EC 2ØØ  C$=CHR$(125):B$=CHR$(253):FOR I=Ø TO 34
        :READ D:POKE 1536+I,D:NEXT I
MD 21Ø  DATA 72,138,72,152,72,169,Ø,162,148,16Ø
        ,12,141,1Ø,212,141,27,2Ø8,142,24,2Ø8,14
        Ø,23,2Ø8,173,192
GE 22Ø  DATA 2,141,26,2Ø8,1Ø4,168,1Ø4,17Ø,1Ø4,6
        4
FE 23Ø  FOR I=1 TO 6:READ N:IO$(I,I)=CHR$(N):NE
        XT I
JI 24Ø  DATA 1Ø4,162,16,76,86,228
EO 25Ø  GRAPHICS 8:POKE 7Ø4,12:POKE 7Ø5,7Ø:POKE
         7Ø6,152:POKE 7Ø7,218:POKE 7Ø8,46:POKE
        7Ø9,118:POKE 71Ø,4:POKE 711,78
MI 26Ø  POKE 712,38:D=PEEK(56Ø)+256*PEEK(561)
PJ 27Ø  FOR I=167 TO 17Ø:POKE D+I+14,PEEK(D+I):
        NEXT I:FOR I=173 TO 175:POKE D+I+12,PEE
        K(D+I):NEXT I
FE 28Ø  FOR I=165 TO 18Ø:POKE D+I,15:NEXT I:POK
        E D+171,Ø:POKE D+18Ø,128:POKE 512,Ø:POK
        E 513,6:POKE 54286,192
OJ 29Ø  POKE 623,128:POKE 87,1Ø:N=PEEK(16)-128:
        IF N>=Ø THEN POKE 16,N:POKE 53774,N
MK 3ØØ  X=Ø:Y=164:T$="Ø12345678"
CM 31Ø  FOR N=1 TO 9:S=ASC(T$(N,N))-32:L=57344+
        S*8:FOR I=Ø TO 7:K=255:COLOR C:IF C=Ø T
        HEN COLOR 6:K=Ø
FG 32Ø  D=ABS(PEEK(L+I)-K):IF D>127 THEN D=D-12
        8:PLOT X,Y+I
MB 33Ø  IF D>63 THEN D=D-64:PLOT X+1,Y+I
LJ 34Ø  IF D>31 THEN D=D-32:PLOT X+2,Y+I
LP 35Ø  IF D>15 THEN D=D-16:PLOT X+3,Y+I
GD 36Ø  IF D>7 THEN D=D-8:PLOT X+4,Y+I
FN 37Ø  IF D>3 THEN D=D-4:PLOT X+5,Y+I
FL 38Ø  IF D>1 THEN D=D-2:PLOT X+6,Y+I
PD 39Ø  IF D>Ø THEN PLOT X+7,Y+I
FP 4ØØ  NEXT I:X=X+9:C=C+1:NEXT N:N=PEEK(14Ø)+2
        56*PEEK(141):D=PEEK(134)+256*PEEK(135)
```

```
GH 410 I=PEEK(88)+256*PEEK(89)-N:X=INT(I/256):
       Y=I-X*256:POKE D+2,Y:POKE D+3,X:POKE D+
       4,160:POKE D+5,25
CD 420 POKE D+6,160:POKE D+7,25:FOR I=0 TO 10:
       READ X,Y:X(I)=X:Y(I)=Y:NEXT I
BB 430 DATA 1,4,1,-4,1,0,0,0,-1,4,-1
AO 440 DATA -4,-1,0,0,0,0,4,0,-4,0,0
NA 450 OPEN #2,4,0,"K:":X=39:Y=80:C=1:POKE 752
       ,1:POKE 82,1:GOTO 190
DJ 460 POKE 752,0:? C$;"Enter a filename for t
       his picture.":INPUT F$:POKE 752,1:IF F$
       ="" THEN GOTO 190
MA 470 TRAP 650:CLOSE #1:OPEN #1,8,0,F$:POKE 8
       52,PEEK(88):POKE 853,PEEK(89):POKE 856,
       160:POKE 857,25:POKE 850,11
AH 480 L=USR(ADR(IO$)):FOR I=704 TO 712:PUT #1
       ,PEEK(I):NEXT I
HK 482 CLOSE #1:OPEN #1,4,0,F$
HC 490 POKE 54286,192:? C$;"Picture saved as:
       ";F$:FOR D=1 TO 400:NEXT D:GOTO 190
KP 500 POKE 752,0:? C$;"Enter the picture's fi
       lename.":INPUT F$:POKE 752,1:IF F$="" T
       HEN GOTO 190
HO 510 TRAP 650:CLOSE #1:OPEN #1,4,0,F$
FL 520 POKE 852,PEEK(88):POKE 853,PEEK(89):POK
       E 856,160:POKE 857,25:POKE 850,7:J=USR(
       ADR(IO$))
OE 530 FOR I=704 TO 712:GET #1,C:POKE I,C:NEXT
        I:POKE 54286,192:GOTO 190
EJ 540 ? C$;"Press a number key and then use t
       he{4 SPACES}joystick.  Press RETURN whe
       n finished.";:C=1
JP 550 I=PEEK(704+C)
GO 560 I=I+X(STICK(0)-5):IF I<0 THEN I=255
DD 570 IF I>255 THEN I=0
AF 580 POKE 704+C,I:IF PEEK(764)<>255 THEN 600
HC 590 GOTO 560
PH 600 GET #2,K:IF K=155 THEN 190
HI 610 IF K<48 OR K>56 THEN 560
PM 620 C=K-48
GM 630 GOTO 550
KC 640 S$(1)=CHR$(0):S$(6560)=CHR$(0):S$(2)=S$
       :GOTO 30
GL 650 POKE 54286,192:D=PEEK(195)
GN 660 IF D=165 OR D=130 OR D=146 THEN ? C$;"Y
       ou used an improper or incomplete
       {5 SPACES}filename.  Try again.";:GOTO
       710
```

```
BM 670 IF D=138 OR D=139 THEN ? C$;"Check all
      connections and try again.";:GOTO 710
PL 680 IF D=144 OR D=162 OR D=167 THEN ? C$;"D
      iskette is full or write protected
      {4 SPACES}or file is locked.";:GOTO 710
EG 690 IF D=170 THEN ? C$;"That picture is not
       on file.";:GOTO 710
IA 700 ? C$;"Error ";D;
BC 710 ? B$:FOR D=1 TO 500:NEXT D:GOTO 190
```

Chapter 3

Hyperword

Daniel M. Daly

Fugitive hyperwords have escaped from their dimension and are materializing in ours. Your job is to type their names into the targeting computer and send them back where they belong.

"Hyperword" is a program that brings new excitement to the old job of learning how to type. Each round pits you against five waves of invaders. After the title screen and the warning to GET READY, the first wave of hyperwords appears.

But what are hyperwords? They are groups of random letters (you can specify the size of the group with the SELECT key) that appear two at a time on your screen. Then, when the FIRE! command appears, you must type in the letters, in order, before the hyperword escapes to other dimensions.

A single game consists of five rounds, and you can specify how long each round lasts by using the OPTION key (to select Skill Level) and the SELECT key (to pick a level from 1 to 9). You'd better practice on the lower levels first, though. Those high-level hyperwords come at you pretty quickly.

If you type in the hyperword correctly, it will turn different colors and then fade out of view. The points earned for that word will be displayed in the text window and added to the tally. If there is not a match, a low tone is heard and the timers continue counting down.

A wave can end in one of two ways. If both words are typed in correctly, the next wave starts. However, if you do not type them correctly before time runs out, the words still remaining will fade out, and then the next wave will begin. No points are earned for words that fade out by themselves.

At the end of the fifth wave, a short fanfare sounds, and you'll learn whether or not you've beaten the previous high score. If so, the high score is changed. The next screen tells the player now many words were hit and/or missed, again indicating the score for the last game.

After a short pause, the program returns to the main menu, waits for input via the console keys, and displays the high score for each size/level combination as each combination is chosen.

Fading Words

The fade-in and fade-out subroutines at the beginning of the program add a great deal of visual excitement to the game. They look impressive, but they are fairly straightforward. In fact, they're simple FOR-NEXT loops with a STEP of less than one. The resulting values set the brightness parameter of the SETCOLOR statement, and the result is a gradual fading in or out of the letters of the word.

Hyperword

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
FL 10 GOTO 1000
HA 100 IF ASC(T0$(1))>128 THEN WCR=3:WC=4:WL=6
      :GOTO 110
IE 105 WCR=0:WC=2:WL=8
AL 110 SETCOLOR WCR,WC,0
MN 115 POSITION XPOS,YPOS
FM 120 PRINT #6;T0$
OP 125 FOR LUP=0 TO WL STEP 0.2
MO 130 SETCOLOR WCR,WC,LUP
MM 135 SOUND 2,50,12,LUP:SOUND 3,85,12,LUP
MF 140 NEXT LUP
IN 145 SOUND 2,0,0,0:SOUND 3,0,0,0:RETURN
HM 160 IF ASC(T0$(1))>128 THEN WCR=3:WC=4:WL=6
      :GOTO 170
IK 165 WCR=0:WC=2:WL=8
IE 170 SETCOLOR WCR,WC,WL
AO 175 FOR LDN=WL TO 0 STEP -0.2
MA 180 SETCOLOR WCR,WC,LDN
KE 185 SOUND 2,50,12,LDN:SOUND 3,60,12,LDN
LH 190 NEXT LDN
KM 193 SETCOLOR WCR,0,0
JC 195 SOUND 2,0,0,0:SOUND 3,0,0,0:RETURN
OG 200 SETCOLOR 1,0,0
BL 205 POSITION RGX,RGY
HB 210 PRINT #6;RG$
JI 220 FOR LUP=0 TO 12 STEP 0.15:SETCOLOR 1,0,
      LUP:NEXT LUP
IN 230 FOR LDN=12 TO 0 STEP -0.15:SETCOLOR 1,0
      ,LDN:NEXT LDN
EG 240 POSITION RGX,RGY:FOR W=1 TO LEN(RG$):PR
      INT #6;" ";:NEXT W:SETCOLOR 1,12,10:RET
      URN
MM 250 RG$="get":RGX=5:RGY=3:GOSUB 200:RG$="re
      ady":RGX=8:RGY=6:GOSUB 200:RETURN
```

```
LJ 1000  DIM H$(9),L1$(26),L2$(26),L(26),RG$(20
         ),LD$(26),OPN$(11),BESTSCORE(9,9)
DO 1005  DIM T0$(9),T1$(9),T2$(9),K1$(9),K2$(9)
MK 1010  H$="hyperword":L1$="ABCDEFGHIJKLMNOPQR
         STUVWXYZ":L2$="abcdefghijklmnopqrstuvw
         xyz"
PN 1012  KBD=764:FF=255:FLAG=206:CHECK=1536:DST
         RYD=1561
NK 1013  CONSOL=53279
MA 1015  CDTMV1L=540:CDTMV1H=541:CDTMV2L=542:CD
         TMV2H=543:TXTROW=656:TXTCOL=657
ID 1090  GRAPHICS 17
MI 1100  POSITION 0,5
FK 1120  PRINT #6;H$;"..."
PK 1220  POSITION 0,13
CD 1225  PRINT #6;"HOW WELL CAN"
PN 1230  POSITION 0,15
AM 1235  PRINT #6;"you TYPE???"
PK 1240  POSITION 0,20
MG 1245  PRINT #6;"initializing..."
JO 1247  FOR I=1 TO 9:FOR J=1 TO 9:BESTSCORE(I,
         J)=0:SOUND 3,I*J,10,8:POKE 711,I*J:NEX
         T J:NEXT I
OC 1250  FOR I=1 TO 26:READ A:LD$(I)=CHR$(A):SO
         UND 0,A,10,8:POKE 711,A:NEXT I
LB 1255  FOR I=1536 TO 1560:READ A:POKE I,A:SOU
         ND 1,A,10,8:POKE 711,A:NEXT I
LH 1257  FOR I=1561 TO 1637:READ A:POKE I,A:SOU
         ND 2,A,10,8:POKE 711,A:NEXT I
BO 1260  FOR I=0 TO 3:SOUND I,0,0,0:NEXT I
GF 1305  CPT=1:SL=1
IB 1307  OPN=0:SLCN=CPT:OPN$="TARGET SIZE"
HP 1310  GRAPHICS 18
JE 1330  POSITION 5,0:PRINT #6;H$
EA 1335  POSITION 0,10:PRINT #6;"PRESS START TO
          BEGIN"
NC 1340  POSITION 4,5
PL 1350  PRINT #6;OPN$
HC 1355  POSITION 2,2:PRINT #6;"{17 SPACES}"
NL 1360  POSITION 9,7
BM 1370  PRINT #6;SLCN
BI 1371  IF BESTSCORE(CPT,SL)>0 THEN POSITION 2
         ,2:PRINT #6;"BEST SCORE=";BESTSCORE(CP
         T,SL)
CG 1372  POKE CONSOL,0
KD 1373  FOR PAUSE=1 TO 35:NEXT PAUSE
AP 1375  IF PEEK(CONSOL)=3 THEN 1390
AG 1380  IF PEEK(CONSOL)=5 THEN 1410
AM 1385  IF PEEK(CONSOL)=6 THEN 1500
NM 1387  GOTO 1375
```

```
NH 1390 IF OPN=1 THEN OPN=0:OPN$="TARGET SIZE"
        :SLCN=CPT:GOTO 1340
GE 1400 OPN=1:OPN$="SKILL LEVEL":SLCN=SL:GOTO
        1340
PB 1410 IF OPN=1 THEN SL=SL+1:GOTO 1430
CD 1420 CPT=CPT+1
KJ 1430 IF SL>9 THEN SL=1
DK 1440 IF CPT>9 THEN CPT=1
DA 1450 IF OPN=1 THEN SLCN=SL:GOTO 1355
GA 1460 SLCN=CPT:GOTO 1355
NJ 1500 SCR=0:REM THE GAME STARTS HERE
FG 1510 T1$="":T2$=""
EA 1630 GRAPHICS 2:SETCOLOR 2,0,0:POKE 752,1
JO 1640 WAVES=1:HITS=0
DD 1650 FOR MT=1 TO CPT:PRL=INT(RND(0)*26)+1:T
        1$(MT,MT)=L1$(PRL,PRL):NEXT MT
KN 1655 T1X=INT(RND(0)*(20-(CPT-1))):T1Y=INT(R
        ND(0)*10)
DG 1660 FOR MT=1 TO CPT:PRL=INT(RND(0)*26)+1:T
        2$(MT,MT)=L2$(PRL,PRL):NEXT MT
KO 1663 T2X=INT(RND(0)*(20-(CPT-1))):T2Y=INT(R
        ND(0)*10)
PP 1664 IF ASC(T1$)=ASC(T2$)-160 THEN 1650
OD 1665 IF T2Y=T1Y AND T2X<T1X+(CPT+1) THEN 16
        63
GO 1667 POKE TXTCOL,16:POKE TXTROW,3:PRINT "WA
        VE ";WAVES;
JH 1670 IF WAVES=1 THEN GOSUB 250
CF 1675 HIT=0
OG 1680 T0$=T1$:XPOS=T1X:YPOS=T1Y:GOSUB 100
OC 1700 T0$=T2$:XPOS=T2X:YPOS=T2Y:GOSUB 100
KC 1705 SLTM=60*(2*(10-SL))+75:SLTMH=INT(SLTM/
        256):SLTML=SLTM-256*SLTMH
IP 1710 POKE 694,0:POKE 702,64
CE 1720 KSN=1
PG 1723 POKE TXTROW,1:POKE TXTCOL,2
FP 1725 FOR PROMPT=1 TO CPT:PRINT ".";:NEXT PR
        OMPT
CK 1727 POKE TXTROW,0:POKE TXTCOL,12
AL 1730 PRINT "FIRE!":POKE 755,2
PJ 1735 POKE TXTROW,1:POKE TXTCOL,2
HP 1740 POKE CDTMV2L,SLTML:POKE CDTMV2H,SLTMH:
        POKE CDTMV1L,56:POKE CDTMV1H,4
IJ 1745 POKE KBD,FF
OD 1747 IF PEEK(CDTMV2L)+256*PEEK(CDTMV2H)<75
        THEN 1995
GO 1750 IF PEEK(KBD)=FF THEN 1747
JN 1760 K=PEEK(KBD)
PM 1770 POKE 755,1
BP 1780 DUMMY=USR(CHECK,K,ADR(LD$)-1)
```

```
CK 1781 INDXY=PEEK(FLAG)
FK 1795 IF INDXY=0 THEN 1840
GN 1797 POKE 53279,0
KF 1800 K1$(KSN)=L1$(INDXY,INDXY):K2$(KSN)=L2$
        (INDXY,INDXY)
IH 1810 PRINT K1$(KSN,KSN);
GL 1820 IF KSN=CPT THEN 1845
DN 1830 KSN=KSN+1
DP 1840 POKE 755,2:GOTO 1745
GI 1845 SVETIMEL=PEEK(CDTMV2L):SVETIMEH=PEEK(C
        DTMV2H):SVESCOREL=PEEK(CDTMV1L):SVESCO
        REH=PEEK(CDTMV1H)
EA 1847 WRDSCR=SVESCOREL+256*SVESCOREH
OO 1850 IF K1$=T1$ THEN GOSUB 20000:DUMMY=USR(
        DSTRYD,0):GOSUB 20100:GOSUB 20200:SCR=
        SCR+WRDSCR:GOTO 1880
PF 1860 IF K2$=T2$ THEN GOSUB 20000:DUMMY=USR(
        DSTRYD,3):GOSUB 20100:GOSUB 20210:SCR=
        SCR+WRDSCR:GOTO 1880
HI 1870 SOUND 0,200,12,10:FOR PAUSE=1 TO 25:NE
        XT PAUSE:SOUND 0,0,0,0
FF 1880 POKE TXTROW,2:POKE TXTCOL,20:PRINT "
        {14 SPACES}":IF HIT=3 THEN 1980
FN 1885 KSN=1:POKE TXTROW,1:POKE TXTCOL,2:FOR
        PROMPT=1 TO CPT:PRINT ".";:NEXT PROMPT
PL 1890 POKE TXTROW,1:POKE TXTCOL,2
EA 1900 POKE CDTMV2H,SVETIMEH:POKE CDTMV1H,SVE
        SCOREH:POKE CDTMV2L,SVETIMEL:POKE CDTM
        V1L,SVESCOREL:GOTO 1840
GN 1980 ? "{CLEAR}":WAVES=WAVES+1:IF WAVES<6 T
        HEN 1650
NA 1990 GOTO 2020
FA 1995 IF HIT=0 OR HIT=2 THEN T0$=T1$:GOSUB 1
        60:POSITION T1X,T1Y:FOR I=1 TO CPT:PRI
        NT #6;" ";:NEXT I
DM 2000 IF HIT=0 OR HIT=1 THEN T0$=T2$:GOSUB 1
        60:POSITION T2X,T2Y:FOR I=1 TO CPT:PRI
        NT #6;" ";:NEXT I
NC 2005 GOTO 1980
MN 2020 GRAPHICS 17:RESTORE 10100
HD 2030 FOR I=1 TO 17:READ A,B
CD 2040 SOUND 0,A,10,8:FOR PAUSE=1 TO 12*B:NEX
        T PAUSE:SOUND 0,0,0,0:NEXT I:SOUND 0,0
        ,0,0
JP 2045 FOR PAUSE=1 TO 25:NEXT PAUSE
KF 2050 SOUND 0,121,10,8:SOUND 1,96,10,8:SOUND
         2,81,10,8:SOUND 3,60,10,8
JL 2060 FOR PAUSE=1 TO 15:NEXT PAUSE
BO 2070 FOR I=0 TO 3:SOUND I,0,0,0:NEXT I
CG 2090 PRINT #6;"NICE GOING!!!"
```

```
KK 21ØØ PRINT #6:PRINT #6;"YOU MADE IT THROUGH
        ":PRINT #6:PRINT #6;"SKILL LEVEL ";SL
FO 211Ø IF BESTSCORE(CPT,SL)>=SCR THEN 219Ø
FB 212Ø PRINT #6:PRINT #6;"AND HAVE BEATEN THE
        ":PRINT #6:PRINT #6;"BEST SCORE FOR TH
        IS":PRINT #6:PRINT #6;"SKILL LEVEL!!"
HD 213Ø PRINT #6:PRINT #6;"YOUR SCORE:  ";SCR
BI 214Ø PRINT #6:PRINT #6;"BEST SCORE:  ";BESTS
        CORE(CPT,SL)
NC 2145 FOR PAUSE=1 TO 45Ø:NEXT PAUSE
EF 215Ø BESTSCORE(CPT,SL)=SCR:POSITION 12,14:P
        RINT #6;BESTSCORE(CPT,SL)
KH 216Ø SOUND Ø,121,1Ø,8:SOUND 1,96,1Ø,8:SOUND
         2,81,1Ø,8:SOUND 3,6Ø,1Ø,8
JN 217Ø FOR PAUSE=1 TO 15:NEXT PAUSE
CA 218Ø FOR I=Ø TO 3:SOUND I,Ø,Ø,Ø:NEXT I
NC 219Ø FOR PAUSE=1 TO 45Ø:NEXT PAUSE
HN 22ØØ GRAPHICS 17
LJ 221Ø PRINT #6;"YOU HIT ";HITS;" WORDS,"
JB 222Ø PRINT #6:PRINT #6;"AND MISSED ";1Ø-HIT
        S;"."
JD 223Ø PRINT #6:PRINT #6;"YOUR SCORE IS":PRIN
        T #6:PRINT #6;SCR;" POINTS!"
NB 224Ø FOR PAUSE=1 TO 75Ø:NEXT PAUSE
MN 225Ø GOTO 13Ø7
OP 999Ø DATA 63,21,18,58,42,56,61,57,13,1,5,Ø,
        37,35,8,1Ø,47,4Ø,62,45,11,16,46,22,43,
        23
NK 1ØØØØ DATA 1Ø4,1Ø4,1Ø4,133,2Ø3,1Ø4,133,2Ø5,
         1Ø4,133,2Ø4,16Ø,26,165,2Ø3,2Ø9
DJ 1ØØ1Ø DATA 2Ø4,24Ø,3,136,2Ø8,247,132,2Ø6,96
MC 1ØØ2Ø DATA 1Ø4,1Ø4,1Ø4,133,2Ø3,169,128,133,
         2Ø4,169,17Ø,141,1,21Ø,169,255
PO 1ØØ3Ø DATA 141,Ø,21Ø,166,2Ø4,16Ø,1,136,2Ø8,
         253,2Ø2,2Ø8,248,72,138,72
JC 1ØØ4Ø DATA 166,2Ø3,173,1Ø,21Ø,157,196,2,1Ø4
         ,17Ø,1Ø4,56,233,1,24Ø,3
DH 1ØØ5Ø DATA 76,41,6,72,165,2Ø4,56,233,16,2Ø1
         ,32,24Ø,6,133,2Ø4,1Ø4
LH 1ØØ6Ø DATA 76,39,6,1Ø4,169,Ø,141,1,21Ø,141,
         Ø,21Ø,96
DI 1Ø1ØØ DATA 81,1,6Ø,5,81,1,81,1,81,1,96,1,81
         ,1,81,1,96,1,81,1,6Ø,6,6Ø,1,81,2,81,1
         ,96,2,96,1,121,2
EM 2ØØØØ POKE TXTROW,2:POKE TXTCOL,2Ø
BE 2ØØ1Ø PRINT WRDSCR;" POINTS!!!"
NE 2ØØ2Ø RETURN
HC 2Ø1ØØ POKE TXTROW,2:POKE TXTCOL,2Ø:PRINT "
         {14 SPACES}":RETURN
```

```
GB 20200 T0$=T1$:GOSUB 160:POSITION T1X,T1Y:FO
         R I=1 TO CPT:PRINT #6;" ";:NEXT I:HIT
         S=HITS+1:HIT=HIT+1:RETURN
GG 20210 T0$=T2$:GOSUB 160:POSITION T2X,T2Y:FO
         R I=1 TO CPT:PRINT #6;" ";:NEXT I:HIT
         S=HITS+1:HIT=HIT+2:RETURN
```

Chapter 3

Stock Market

Sul Kattan

The Atari is an excellent computer for action game programming. However, you should not forget that the same features that make it a great games computer also make it ideal for a variety of real-life simulations.

"Stock Market" is a good example of the sophisticated simulations that can be done on your Atari. It can be used as an educational tool or as a game, and it is sure to be enjoyed for many hours. The program uses approximately 8K, so it can be played on any Atari computer.

After you run the program, it will pause for a few seconds before prompting you for the number of players (1–5). From that point, the game is self-explanatory. After the stock codes have been displayed and you have learned the function keys, the screen will go blank for another few seconds. Then the top line will transform into a stock ticker, displaying the three-letter stock codes and their respective costs per share. That's when the fun begins.

Thanks to machine language and the vertical blank interrupt (VBI), the ticker remains active throughout each play session, even during transactions. All fluctuations will be displayed on the ticker, so keep a close eye on it.

When you press the BUY or SELL key, all prices will be frozen. This makes the game fair to all players. However, the CREDIT CHECK key will not freeze prices.

It is helpful to keep a record of what price you paid for certain stocks. The game will keep a record of which stocks you own, and it will list your holdings whenever you buy or sell shares. Also, remember your identification number (1–5), but the game will remember your name.

The names of the companies used in the simulator are completely fictitious. Should such a company name exist, it is purely coincidental.

Stock Market

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
MB 11  SUMMARY=13000
NI 12  GOSUB  14000
DO 13  POKE  16,64:REM  DISABLE  'BREAK'  KEY
```

```
DE 25 GOTO 130
FG 30 RESTORE 100+EC:READ D$,A,B,DD$:RETURN
EG 60 ? :? NAME$(HM*20,HM*20+NL(HM));" OWNS:":
      ?
II 65 FOR EC=1 TO 20:IF SH(HM,EC)=0 THEN NEXT
      EC:GOTO 70
FD 66 ? CHR$(32+3*(AV(EC)=0));
CL 67 ? SH(HM,EC);" shares of ";:GOSUB 30:? D$
      ;" (";DD$;")":NEXT EC
GC 70 ? "PRESENT CREDIT $";W(HM)
EI 80 RETURN
CC 100 REM ** STOCK TITLES
IK 101 DATA ASTERISK ELEC.,22,10,AST
MO 102 DATA AV&V,55,3,AVV
FA 103 DATA BORVAC AIR,45,5,BOV
BN 104 DATA BOWLAND CORP.,25,5,BWL
BI 105 DATA COMLINK COMP.,30,8,COM
GL 106 DATA CROY GENETICS,15,14,CRG
MF 107 DATA DELTON CHEM.,40,6,DLT
GO 108 DATA FAIRVIEW MTR.,46,10,FRV
AG 109 DATA GEM MILLS,23,6,GEM
LC 110 DATA GIBSON AERO.,40,5,GIB
EG 111 DATA I.M.I.,54,22,IMI
OK 112 DATA KATTAN PROD.,39,9,KTN
HM 113 DATA LOCKE CORP.,17,3,LOC
AG 114 DATA METER ONE OIL,35,7,MTO
GL 115 DATA RINGER AMERICA,25,5,RNA
FD 116 DATA SAFE STEEL,12,2,SFS
LB 117 DATA SONER CORP.,43,7,SNR
CP 118 DATA TEXTAR CORP.,36,12,TEX
BA 119 DATA UNION TC,45,20,UTC
NE 120 DATA WEDWAY COMM.,30,5,WED
PE 130 REM ** INITIALIZATION
IN 135 FOR I=TADR TO TADR+199:POKE I,0:NEXT I:
       RESTORE 101
HI 140 DIM STOCK(20),D$(30),AV(20),DD$(3),V$(1
       0),CO(20),CODE$(3)
JD 200 FOR ST=1 TO 20
BB 205 READ D$,A,B,DD$:CO(ST)=B
DF 210 STOCK(ST)=A+INT((RND(0)*B)*100)/100:AV(
       ST)=INT(RND(0)*5000)+3000:NEXT ST
KH 250 TRAP 250:? CHR$(125);"HOW MANY PLAYERS
       (1-5) ";:INPUT PL:TRAP 40000:PL=INT(PL)
       :IF PL<1 OR PL>5 THEN 250
CC 260 DIM NAME$(PL*20+20),W(PL),SH(PL,20),NL(
       PL)
AC 265 ? "EACH PLAYER STARTS WITH $3000"
GP 270 ? :FOR M=1 TO PL:? "NAME OF PLAYER #";M
       ;" ";:INPUT D$:NL(M)=LEN(D$)-1
NM 280 NAME$(20*M,20*M+NL(M))=D$:NEXT M
```

```
EN 285 GOSUB SUMMARY
BL 600 FOR A=1 TO PL:FOR B=1 TO 20:SH(A,B)=0:N
       EXT B:W(A)=3000:NEXT A
JP 640 UU=USR(1585)
MF 650 GOTO 10000
JC 1000 REM ** EXIT MAIN LOOP
PB 1010 EP=PEEK(764):IF EP=255 THEN RETURN
FP 1015 POKE 764,255
PI 1020 IF EP=21 THEN GOSUB 2000
AC 1030 IF EP=62 THEN GOSUB 6000
DB 1040 IF EP=18 THEN GOSUB 12000
KH 1060 RETURN
MB 2000 REM ** BUY STOCKS
CI 2002 TRAP 5000:POKE 755,2
JE 2003 ? CHR$(125):? "{8 SPACES}enter 'RETURN
       ' to exit"
AG 2005 ? :? "WHICH PLAYER WILL BUY STOCKS (1-
       ";PL;")";:INPUT HM:TRAP 40000
KJ 2010 GOSUB 60
HI 2040 ? :? "Enter code of desired stock ";:I
       NPUT CODE$
AJ 2041 IF CODE$="" THEN 2002
MH 2050 FOR CV=1 TO 20:RESTORE 100+CV:READ D$,
       A,B,DD$:IF DD$=CODE$ THEN 2070
BJ 2055 NEXT CV:? "CAN'T LOCATE ";CODE$:GOTO 2
       040
KO 2070 ? CHR$(125):? CV;") ";D$:? :? AV(CV);"
        SHARES AVAILABLE AT":? "$";STOCK(CV);
       " PER SHARE."
DA 2080 ? :? "YOU HAVE $";W(HM);"."
HB 2082 PO=INT(W(HM)/STOCK(CV)):IF PO>AV(CV) T
       HEN PO=AV(CV)
PL 2085 ? "You may buy up to ";PO;" shares."
IJ 2090 TRAP 2002:? :? "HOW MANY SHARES ";:INP
       UT WANT:TRAP 40000
EM 2095 IF WANT>AV(CV) THEN ? "*** ONLY ";AV(C
       V);" SHARES AVAILABLE"
AL 2100 COST=STOCK(CV)*WANT
FM 2110 IF W(HM)-COST<0 THEN ? "*** THAT WOULD
        COST $";COST:GOTO 2080
LL 2120 ? WANT;" shares of ";D$
EL 2130 W(HM)=W(HM)-COST:SH(HM,CV)=SH(HM,CV)+W
       ANT
HB 2135 AV(CV)=AV(CV)-WANT
GP 2140 ? :? "Cost of transaction $";COST
LJ 2150 ? "Player ";HM;"'s present credit $";W
       (HM)
HA 2160 POKE 755,0:? "HIT ANY KEY TO CONTINUE"
EC 2170 IF PEEK(764)=255 THEN 2170
JK 2180 POKE 764,255:GOTO 2002
```

```
GO 5000 ? CHR$(125):POKE 755,0:RETURN
AF 6000 REM ** SELL STOCKS
PO 6010 ? CHR$(125):POKE 755,2:? "{9 SPACES}en
        ter 'RETURN' to exit"
CO 6020 TRAP 8000:? :? "WHO WILL SELL STOCKS
        1-";PL;")";:INPUT HM:TRAP 40000
KP 6030 GOSUB 60
MJ 6040 TRAP 6020
KI 6050 ? :? "ENTER STOCK CODE ";:INPUT CODE$:
        IF CODE$="" THEN 6020
NB 6060 FOR CV=1 TO 20:RESTORE 100+CV:READ D$,
        A,B,DD$:IF DD$=CODE$ THEN 6080
BP 6070 NEXT CV:? "CAN'T LOCATE ";CODE$:GOTO 6
        050
GP 6080 ? CHR$(125):? CV;") ";D$:? :? "YOU OWN
         ";SH(HM,CV);" SHARES."
II 6090 ? :? "CURRENT PRICE PER SHARE $";STOCK
        (CV)
BO 6095 ? "YOUR PRESENT CREDIT $";W(HM)
CD 6100 ? "HOW MANY SHARES DO YOU SELL ";:INPU
        T GIVE
LK 6110 IF GIVE>SH(HM,CV) THEN ? "*** YOU OWN
        ONLY ";SH(HM,CV);" SHARE(S)":GOTO 6100
DC 6120 NET=GIVE*STOCK(CV):? :? "NET GAIN $";N
        ET
OP 6125 SH(HM,CV)=SH(HM,CV)-GIVE
MI 6130 W(HM)=W(HM)+NET:? "PRESENT CREDIT $";W
        (HM)
GA 6140 AV(CV)=AV(CV)+GIVE
IG 6150 ? :? "HIT ANY KEY TO CONTINUE"
EI 6160 IF PEEK(764)=255 THEN 6160
JP 6170 POKE 764,255:GOTO 6000
HB 8000 POKE 755,0:? CHR$(125):RETURN
IH 10000 X=0:REM ** MAIN LOOP
OM 10010 FOR I=0 TO 190 STEP 10:X=X+1
CD 10015 IF PEEK(764)<>255 THEN GOSUB 1000
DD 10019 RESTORE 100+X
AO 10020 ADD=-CO(X)+INT(RND(0)*CO(X)*200)/100
HH 10030 STOCK(X)=STOCK(X)+ADD:STOCK(X)=ABS(ST
        OCK(X))
KI 11005 FOR P=TADR+I TO TADR+I+10:POKE P,0:NE
        XT P:P=TADR
CK 11010 READ D$,A,B,DD$
HL 11020 FOR L=1 TO 3:POKE P+I+L,ASC(DD$(L,L))
        -32:NEXT L
JO 11030 V$=STR$(INT(STOCK(X)*10)/10)
MM 11040 LN=LEN(V$):FOR L=1 TO LN:POKE P+I+L+4
        ,ASC(V$(L,L))-32:NEXT L
OD 11050 NEXT I:GOTO 10000
```

```
EE 12000 REM ** CREDIT CHECK
BN 12010 ? CHR$(125)
AK 12020 ? :? "{15 SPACES}CREDIT CHECK"
KO 12030 ? :?
GM 12040 FOR EPL=1 TO PL
DA 12050 D$=NAME$(EPL*20,EPL*20+NL(EPL))
LI 12060 ? EPL;") ";D$;"{3 SPACES}$";W(EPL):?
         :NEXT EPL:RETURN
FK 13000 REM ** SUMMARY
EL 13010 RESTORE :POKE 752,1:POKE 82,1:? CHR$(
         125);
AK 13020 FOR I=1 TO 20:READ D$,A,B,DD$
HF 13025 IF I=1 THEN ? "avail ";AV(I);"code ";
         DD$;"stock ";D$:NEXT I
GO 13030 ? "{6 SPACES}";AV(I);"{5 SPACES}";DD$
         ;"{6 SPACES}";D$
HI 13040 NEXT I:POKE 703,4:POKE 82,2:POKE 752,
         1
NL 13080 ? CHR$(125);"DURING PLAY YOU MAY:":?
         "{3 SPACES}BUY  STOCKS BY PRESSING 'B
         '"
CO 13090 ? "{3 SPACES}SELL  STOCKS BY PRESSING
          'S'"
CI 13095 ? "OR....(press any key)";
JI 13100 IF PEEK(764)=255 THEN 13100
BN 13110 POKE 764,255:? CHR$(125):? " CHECK  Y
         OUR CREDIT BY TYPING 'C'"
IE 13120 PRINT
EJ 13125 ? "PRESS ANY KEY TO START";
JO 13130 IF PEEK(764)=255 THEN 13130
JB 13140 POKE 764,255
NC 13150 POKE 703,24:? CHR$(125):RETURN
BN 14000 REM ** MACHINE LANGUAGE LOADER
LK 14005 RESTORE 15000:FOR AX=1536 TO 1596:REA
         D AXP:POKE AX,AXP:NEXT AX
FC 14010 POKE 1563,PEEK(88):POKE 1564,PEEK(89)
BK 14020 DIM TAPE$(200):TADR=ADR(TAPE$)
AF 14030 IT=INT(TADR/256)
NL 14040 POKE 1560,TADR-256*IT:POKE 1561,IT
ME 14045 POKE 1576,PEEK(548):POKE 1577,PEEK(54
         9)
NK 14050 RETURN
PM 14055 REM -- THE FOLLOWING NUMBERS ARE MACH
         INE LANGUAGE
OE 14056 REM -- CODES.  TYPE CAREFULLY!
OF 14057 REM
LD 15000 DATA 173,47,6,205,10,210
MF 15002 DATA 176,31,174,48,6,232
KP 15004 DATA 224,200,208,2,162,0
```

```
JE 15006 DATA 142,48,6,160,0,189
MG 15008 DATA 5,52,153,64,156,232
BD 15010 DATA 224,200,240,8,200,192
PA 15012 DATA 40,208,240,76,62,233
FG 15014 DATA 162,0,76,34,6,200
IP 15016 DATA 80,104,169,0,162,6
FJ 15018 DATA 141,36,2,142,37,2
IB 15020 DATA 96
```

Chapter 3

Adding Excitement to Educational Programs

Barry Sperling

The Atari is an outstanding computer for use in the classroom, particularly when onscreen printing is made more exciting with modes 1 and 2.

All students like to see their names in lights. Unfortunately, even when you put a large, multicolored name request in graphics mode 2, the name that the student types in appears in tiny white letters near the bottom of the screen.

How can you get that name on the top where it belongs?

My first thought was to alter the display list, the program in RAM that controls the ANTIC chip. It governs which mode line will appear at what level on the screen. GRAPHICS 2, for example, has ten lines of double-width, double-height letters followed by four lines of normal size letters (GRAPHICS 0).

The fifth and sixth bytes of the list carry the address of the data that will be put on the screen. I decided to change those bytes to point to the text window. Anything typed in the text window would then appear on the large screen; the technique is shown in the short program given below.

```
10 GRAPHICS 2:DL=PEEK(560)+256*PEEK(561)
20 POKE DL+4,PEEK(660):POKE DL+5,PEEK(661)
30 ? "TYPE IN A NUMBER.":INPUT A
```

Line 10 sets the screen for large type with a text window. Locations 560 and 561 in RAM hold the address of the first byte of the display list in standard low byte/high byte format. Therefore, DL+4 is the fifth byte of the list.

Locations 660 and 661 hold the start of the text window (upper left byte). POKEing their values into the data address for the main screen allows your typing to appear on the top in large letters.

Unfortunately, as you can see when you try it, your handiwork appears in both the screen and the window. What do you do? One solution would be to find some permanently empty memory and point the text window to that. Another would be to make the window invisible with SETCOLOR 1,0,0 and SETCOLOR 2,0,0, making the characters and background black. Unfortunately, this wipes out two of the pos-

sible colors that you might want to use, leaving only the background and two character colors.

Can you get all the colors while staying in BASIC? You can, but it is more complicated. I envisioned a screen with all five colors available, no question mark for a prompt, and the input appearing with large letters. The following program gives you all of that and throws in audible feedback for good measure.

Feel free to change the programs to suit your own needs and preferences. You might want to accept longer first names, for example, or add a redefined character set to sprinkle a few Martians around the screen. A loop might be used to flash the words FIRST and RETURN while waiting for input. Maybe a tune could play during VBLANK as an extra attraction.

Using these techniques, you can make a good educational program even better. A strong introduction to your program will give the kids a positive attitude about it right from the start.

Exciting Inputs

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
ME 10 DIM A$(10):OPEN #2,4,0,"K:"
MA 20 GRAPHICS 18:POSITION 6,1:? #6;"hELLO!":P
      OSITION 1,3:? #6;"please type in":POSITI
      ON 1,5:? #6;"YOUR FIRST NAME,":POSITION
      1,7
EA 30 ? #6;"then push RETURN.":SETCOLOR 0,12,8
      :SETCOLOR 1,15,12:SETCOLOR 2,1,6:SETCOLO
      R 3,3,6:SETCOLOR 4,6,4
PA 40 TRAP 20:A$="":POKE 764,255:POSITION 4,9:
      ? #6;"*":POSITION 15,9:? #6;"*":POSITION
       5,9
BP 50 GET #2,A:IF A=155 THEN TRAP 40000:GOTO 9
      0
DB 60 IF A=126 THEN A$=A$(1,LEN(A$)-1):POSITIO
      N 5+LEN(A$),9:? #6;" ":POKE 85,PEEK(91):
      POKE 84,9:GOTO 50
GJ 70 A$(LEN(A$)+1)=CHR$(A):SETCOLOR 0,3+2*LEN
      (A$),8:SOUND 0,200-10*LEN(A$),10,12:FOR
      T=1 TO 30:NEXT T:SOUND 0,0,0,0
PP 80 PUT #6,A:GOTO 50
IL 90 IF LEN(A$)<1 THEN 20
NE 100 FOR T=1 TO LEN(A$):IF (ASC(A$(T,T))<65
       AND ASC(A$(T,T))<>32) OR ASC(A$(T,T))>9
       0 THEN 20
CF 110 NEXT T
LB 120 REM REST OF YOUR PROGRAM
```

Chapter 3

Test Maker

Stephen Levy

Teachers will find this easy-to-use system for storing questions and printing tests to be an invaluable tool and an effective timesaver. Questions can be multiple-choice, fill in the blank, short answer, true/false, or any combination.

"Test Maker" was written for teachers. It allows you to print nicely formatted tests using questions that have been previously LISTed to disk or tape.

You may save numerous files of questions and pick and choose from all the files. Questions can be used in any order. Once you have selected the questions, the program will print out the test in a uniformly formatted style.

Midterms and finals become a snap. Just load up all test questions used throughout the term and pick the ones you want for the final. Save questions from year to year, and add to your list each year. Think of it—a simple, easy-to-use way to create different tests for each of your classes without having to retype the test. It's a teacher's dream come true.

When printing your tests, you can use virtually any type of master. For instance, I have used Test Maker with standard ditto masters with an Atari 825 printer (removing the tissue, of course) to create masters that have produced well over 100 copies.

Creating the Questions

Test Maker stores all your questions in DATA statements. If you create different files for each unit and do not use DATA statement line numbers more than once, you will be able to draw questions from numerous files. Keep track of the DATA statement line numbers. I find it best to print out a copy of the files so I know what is in each file.

Writing the actual questions in DATA statements requires that you follow a few simple rules. Here's how a sample multiple-choice question would look in final DATA statement form:

```
600 DATA 4
601 DATA The first President of the
602 DATA United States was
603 DATA Thomas Jefferson
```

```
604 DATA George Washington
605 DATA Jimmy Carter
606 DATA Richard Nixon
```

The first DATA statement (line 600) tells the program that this is a four-choice multiple-choice question, although you could have had as few as two choices. Put a 0 or 1 here if the question is not multiple-choice.

Lines 601 and 602 are the actual question. All questions must be broken into two DATA statements, with the break occurring between words. In that way, questions can exceed the three-physical-lines limitation imposed by Atari BASIC.

Finally, each of the four possible answers appears as a DATA statement.

The question could have been written this way:

```
600 DATA 4,The first President of the,United States was,Thomas Jefferson,George Washington,Jimmy Carter,Richard Nixon
```

However, I prefer the first method because it is easier to use months or years later when the questions aren't quite so fresh in my mind.

I could also have included an indication of the correct answer as a REM:

```
607 REM A is the correct answer
```

Printing Special Characters

Since the questions are held in DATA statements, it might be difficult to use a comma embedded in a question. The program would assume that the comma indicated the end of the DATA item. Whenever you need to include a comma in a question, press SHIFT = to get the vertical line character. The program will convert this to a comma when it prints a hard copy.

You can print any character from the Atari character set (within the capabilities of your printer, of course). Science and math teachers can even print exponents. Just use the characters that cause the printer to reverse linefeed one-half line, print the exponent, and then insert the characters which cause a half-linefeed forward. On an 825 printer, for example, the following keystrokes would produce an exponent of 2:

ESC, ESC, ESC, CTRL +, 2, ESC, ESC, ESC, CTRL –

Be sure that you do not use any line number below 571 for a DATA statement. In addition, it's best not to use the same line numbers in more than one file. If you have duplicate line numbers, then those questions cannot be used on the same test. You can avoid the problem completely if you start each question on a line divisible by 10, starting with 600, and if you never use a number twice.

Printing Tests

Once you have created your test questions, you must LIST them to disk or tape. Do not use the SAVE command. For tape, LIST "C:" and note where on the tape the file starts. For disk, LIST "D:*filename*".

Assuming you have saved a copy of Test Maker and LISTed copies of the DATA statements, you are ready to print a test. First, load Test Maker from disk or tape. Change line 20 so it contains your own directions. Then ENTER the files that include any questions you want to include. If you have used different line numbers for all your questions and have not used a line number below 571, you will end up with the questions as well as the driver program in memory.

Referring to your list of questions (or scanning the DATA statements), make a list of the first DATA statement line number for each question that you want to include. Also note the order in which you wish them to appear. You do not have to use all the questions that are held in memory.

Run the program. Answer the prompts as they appear. Enter one line number at a time, pressing the RETURN key after each one. When asked "How many questions on this page?" you'll have to estimate how many questions will fit on your page. It may be helpful to first print out the entire test on a continuous piece of paper, figure how many questions should be on each page, and then ask for another copy of the test. You can print additional copies of the test without entering all the numbers again.

Practice

When learning to use this program, it is a good idea to make a few practice runs. Create some questions and print some sample tests. Get the hang of it before you tackle the real thing.

After just a little practice, you'll find that this program is extremely easy to use. It even includes a simple routine to tell

you when you have entered DATA incorrectly. Also included are three sample questions (all lines from 600 up). These sample questions should not be typed in or included as part of Test Maker; they have only been included as examples.

Test Maker

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
ED 10  DIM YES$(1),A$(110),Q1$(125),Q2$(125),P$
       (250),BL$(11):BL$="{11 SPACES}"
AB 20  DATA DIRECTIONS: REPLACE THIS LINE WITH
       YOUR SPECIFIC DIRECTIONS FOR THE TEST.
MO 30  TRAP 30:PRINT CHR$(125);"HOW MANY QUESTI
       ONS ON THE TEST";:INPUT NUM
OP 35  IF NUM<2 THEN PRINT CHR$(253);CHR$(125);
       "YOU MUST HAVE AT LEAST 2 QUESTIONS":FOR
        I=1 TO 1500:NEXT I:GOTO 30
ND 37  DIM N(NUM)
LI 40  PRINT "ENTER THE DATA STATEMENT NUMBERS
       {6 SPACES}ONE AT A TIME.":G=60
JG 50  FOR I=1 TO NUM
ON 60  TRAP 70:INPUT D:GOTO 80
AC 70  PRINT CHR$(253);"ENTER A NUMBER PLEASE,
       TRY AGAIN.":FOR W=1 TO 500:NEXT W:GOTO G
NB 80  N(I)=D
PF 85  NEXT I
CJ 90  Z=1:P=1
FI 100 PRINT "Questions listed to the Screen o
       r Printer?";:INPUT YES$:CLOSE #1
DE 105 TRAP 70
KL 110 G=105:PRINT :PRINT "This is page ";P;"
       .How many":PRINT "questions on this pag
       e";:INPUT C:IF Z>1 THEN GOTO 250
HH 200 IF YES$="S" THEN OPEN #1,8,0,"E:":GOTO
       240
IG 210 TRAP 230:IF YES$="P" THEN OPEN #1,8,0,"
       P:":GOTO 240
PE 220 PRINT "Enter a P for Printer or S for S
       creen";CHR$(253):PRINT :INPUT YES$:CLOS
       E #1:GOTO 105
CL 230 PRINT CHR$(253);"TURN ON THE PRINTER!":
       PRINT :PRINT :GOTO 100
DD 240 RESTORE :READ P$:PRINT #1;"Name";:FOR I
       =1 TO 30:PRINT #1;"_";:NEXT I:PRINT #1:
       PRINT #1:GOSUB 500
JJ 250 P=P+1:TRAP 550:FOR I=1 TO C
LD 255 IF Z>NUM THEN GOTO 350
MP 260 D=N(Z):RESTORE D:PRINT #1
KB 270 READ TYPE,Q1$,Q2$:Y=0:CH=64:L=LEN(Q1$)
```

```
EP 280  P$=Q1$:P$(L+1,L+1)=" ":P$(L+2,LEN(Q2$)+
        1+L)=Q2$
AE 290  PRINT #1;"__________";Z;".";:GOSUB 500:
        Z=Z+1
NB 300  IF TYPE<2 THEN GOTO 340
LA 310  FOR ANS=1 TO TYPE:Y=0
KG 320  CH=CH+1:READ A$:PRINT #1;BL$;CHR$(CH);"
          ";:P$=A$:GOSUB 500
LH 330  NEXT ANS
PC 340  IF YES$="S" THEN FOR W=1 TO 500:NEXT W
CN 345  NEXT I:GOTO 105
AD 350  PRINT :PRINT "Print another copy";:INPU
        T YES$
NG 360  IF YES$="N" THEN 380
DM 370  GOTO 90
LJ 380  PRINT "ARE YOU SURE YOU ARE FINISHED";:
        INPUT YES$
BO 390  IF YES$<>"Y" THEN 350
GL 400  END
GN 500  FOR X=1 TO LEN(P$):IF P$(X,X)="|" THEN
        P$(X,X)=","
OO 510  Y=Y+1:IF P$(X,X)=" " AND Y>50 THEN PRIN
        T #1:PRINT #1;BL$;:Y=0
IP 520  PRINT #1;P$(X,X);
CP 530  NEXT X
MD 540  Y=0:P$="":PRINT #1:RETURN
CP 550  PRINT :PRINT "YOU HAVE AN ERROR IN YOUR
         DATA{8 SPACES}STATEMENTS. CHECK THE QU
        ESTIONS WHOSE DATA BEGIN ON THE"
NL 560  PRINT "FOLLOWING LINES":PRINT N(Z),N(Z-
        1)
HD 570  END
OB 600  DATA 1
FG 601  DATA What is the capital of the
MO 602  DATA United States?
OF 610  DATA 4
LP 611  DATA The first President of the
NL 612  DATA United States was
MC 613  DATA Thomas Jefferson
DA 614  DATA George Washington
BN 615  DATA Jimmy Carter
IA 616  DATA Richard Nixon
OG 620  DATA 4
FK 621  DATA The only President of the
GH 622  DATA United States to resign from offic
        e was
MD 623  DATA Thomas Jefferson
DB 624  DATA George Washington
BO 625  DATA Jimmy Carter
IB 626  DATA Richard Nixon
```

Chapter 4

Applications

Chapter 4

Shopping List

John E. Dombrow
and John Dombrow

"Shopping List" is a program that lets you create, update, and save to disk personalized shopping checklists. It combines menus, editing capabilities, parameter-driven printing, and error recovery to produce a remarkably practical shopper's aid.

"Shopping List" incorporates mixed graphics modes and display list interrupts. It also uses PLOT and DRAWTO in GRAPHICS 0, machine language subroutines, and a text window in GRAPHICS 0. Other features include multiple colors and luminances as well as keyboard INPUT without the ? prompt.

Shopping List was designed to be practical. It allows you to specify shopping categories in the order that you find them at your grocery store. If that order changes, it's a simple matter to change the sequence. Items and categories may also be added, deleted, or changed. When items are changed, the new name will be displayed on the screen where the old name appeared, allowing you to verify any changes that you make. Once a new function is performed, the changed names will automatically be sorted into their proper alphabetic sequence.

Shopping Menus

Main Menu. When the program is initially run, you'll get the Main Menu. It offers the following options:

View List. If no file is loaded in memory, this selection will display a directory of all .SHP files on disk and ask which one you want to load. However, if a file is loaded into memory, it displays the Categories screen instead. To override this action and load a new file, select Read File from the Main Menu when a file is currently in memory.

Print List. If no file is in memory, this displays a directory of .SHP files and asks which file to load for printing. It continues to a screen which lets you specify normal or condensed print, the number of shopping lists to print, the number of item columns to print, and whether you are using continuous forms or cut sheet paper (which would require a stop at the end of each page). This option is designed for use with an Epson MX-100 printer.

Read File. This lets you load a new file when one is already in memory. If the file in memory has not been saved on disk, you'll get a warning message with several options.

Save File. This will appear only if a file is currently in memory (variable C>0). It offers you the option of saving the file under its most recent name, saving it under a new name, or pressing RETURN to cancel. If you continue, the program will calculate the number of disk sectors necessary to save the current file. If sufficient space is not available, you'll be so notified and no attempt to save the file will be made. At that point, you may DELETE some other file to make room or use a different disk.

Create File. This option allows you to make up a new shopping list. If a file is in memory and has not been saved, you'll get a warning message with options; otherwise, you'll get a screen that lets you begin entering categories. A maximum of 38 categories (governed by variable MC in line 620) is allowed.

Delete File. This allows you to delete unwanted .SHP files. You'll get a directory and be asked which file to delete. If you don't want to delete a file, press RETURN. This option will not affect a file in memory and may be selected at any time.

Quit. This ends program execution. If a file exists in memory, you'll get a warning message with appropriate options.

Every effort has been made to insure that a file or updates to a file cannot be accidentally lost by inadvertent use of the above options. Variable C (number of categories) and variable FS (file saved) are always checked to see if a file exists or differs from the version saved on disk; if so, the program will allow you to recover. You always have the option to return from a selection that was inadvertently entered. To select an option, simply enter the associated letter without pressing RETURN.

Directory Menu. This displays a directory of up to 20 .SHP filenames per disk. It is also useful with Read, Save, or Delete. In the case of Save File, the directory is included as a convenience to show what filenames already exist.

Each filename is displayed with a unique alphabetic identifier. This identifier allows for selection with a single keystroke. All filenames are suffixed automatically with the extender .SHP; you do not need to type in the extender yourself.

After the filenames, the remaining sectors available on the disk are displayed.

Categories Menu. You'll get this menu if you want to view a list with a file in memory, view a list or read a file after a file is loaded, create a file, or use the Cats option from the Items Menu.

The Categories screen will display all categories entered, in any order determined by you. To select an option, simply enter the associated letter.

Add. This allows you to add categories to your shopping list. All additions will appear at the end of the list. When you enter the Add option, the text window will change to allow entry of the new categories. The format allows for up to 16 characters per category; any characters beyond will be truncated. To exit, press RETURN with a blank category name.

Insert. This option lets you insert a new category into an existing list. Enter the number that you want the new category to have and press RETURN. The text window will change to the Add mode for the entry of one new category. Then enter the new category and press RETURN. The Categories Menu will again be displayed with the new category inserted at its proper place.

View. This lets you view, enter, and update the items associated with each category. Enter the category number and press RETURN. The Items Menu will then be displayed.

Renum. This lets you rearrange the order of existing categories. Specify the old category and its new location, and the Categories screen will be redisplayed with the categories in the new order.

Change. This lets you change a category name. Enter the category number and press RETURN. The original name will be displayed for you to modify as desired; when you're through, press RETURN. The Categories Menu will be redisplayed with the change.

Delete. This lets you delete an unwanted category. Enter the number and press RETURN. The text window will display the category name selected and ask for verification. Enter a Y or N.

If you try to delete a category that still has items associated with it, further verification is requested. If you continue, all items associated with the category are deleted as well.

Main Menu. This returns you to the Main Menu. If you make a mistake while using any of these options, you'll get an appropriate error message and be returned to the Categories Menu.

Items Menu. This menu and display will show all items, if any, associated with the selected category. All items will be in alphabetical order according to the ASCII sequence. A list of the remaining entries is displayed in the text window during certain options and is governed by variable ME (maximum entries) in line 620, which defines the maximum number of items across all categories (initially set to 400). The keyboard is set to lowercase when entering items, and the program will capitalize the first character of the item if necessary.

To select an option, enter the appropriate letter without pressing RETURN. The following options are available.

Next Pg (Next page). This displays the next sequential screen of items, if more items exist.

Last Pg (Last page). This displays the previous screen of items, if you have advanced beyond the first screen.

Add. This lets you enter additional items in the selected category. The items will be displayed in the order entered until you exit the Add mode; at that time all items will be sorted and page 1 of the Items Menu will be redisplayed. To exit and return to the Items Menu, press RETURN with a blank item name.

Cats (Categories). This option will return you to the Categories Menu.

Change. This lets you modify an item name displayed on the current screen. Enter the appropriate item number and press RETURN. The item name will be displayed in the text window. Press RETURN after you have completed the change, and the new name will be displayed in place of the original.

The new name(s) will be sorted into correct sequence when you select one of the Add, Cats, Sort, or Pg 1 options. That allows you to easily update all items.

Delete. This deletes a specific item from the current screen. Enter the appropriate number and press RETURN. The item name will be displayed, along with the option to continue. Enter Y or N accordingly. If Y is entered, the item will be deleted and scrolled off the graphics portion of the display.

Sort. This forces a sort of all items associated with the selected category, if any updates have been performed. For ex-

ample, it may be used after making modifications with Change, to display all items in the correct sequence without having to exit the Items Menu or do additional Adds. After entering this option, the first page of the Items Menu will be displayed.

Pg 1 (Page 1). This option displays the first screen of the Items Menu/display. If the items have been modified, they will be re-sorted into ascending ASCII sequence.

Main Menu. Returns you to the Main Menu.

Print Menu. This menu lets you print your list. To select an option, enter the corresponding number without pressing RETURN. It offers the following options:

Normal/Condensed. This allows you to print your shopping list in normal or condensed print; the default is condensed print. The default may be changed by modifying variable MODE$ in line 720 in the program.

Number of Columns of Items. This lets you select the number of columns of items across the page of the shopping list. The default value is 5; it can range from 1 to 9. The default may be changed by modifying variable COL in line 720.

Number of Pages to Print. This lets you select the number of shopping lists to print. One list is considered a page. A page eject is issued after each list is printed. The default is 1; the value can range from 1 to 9. The default may be changed by modifying variable PAGES in line 720.

Any changes made to these parameters during program execution will remain in effect until other changes are made (or until the session is completed).

A Closer Look

The shopping list file is kept in memory as a sequential group of records in F$. All items sequentially follow their respective category. Array P contains a relative displacement to each category in F$. Variable C maintains a count of active categories, while variable E maintains a count of active items.

When a category is added, it is placed at the end of F$, and P(C) is updated to reflect the displacement. When a category is inserted, the proper position in F$ is calculated, the data to the right of this position is moved one entry to the right, and the new category is inserted. Array P is then updated to reflect the new displacements. Likewise, when a category is deleted, the category and any items are removed by

moving all data to the right of the category back to the left and updating array P.

The move left and move right subroutines are machine language subroutines. Machine language is needed to move data left because of BASIC's inability to move multiples of 256 characters.

The machine language subroutines implemented in Shopping List have been converted to string arrays.

With the exception of documenting the Epson MX-100 printer control characters, REMarks have been omitted from the program to increase speed and reduce size. For those who wish to analyze the program or make changes or enhancements, here is a line-by-line description.

Line(s)

20	Go to initialization code.
30–40	Solicit a reply and compare it to legal values in R$. If not valid, sound the keyboard speaker. If valid, R is set to the relative position of the response in R$.
50	Sound the keyboard speaker.
60–70	Display Categories/Items screen in a two-column format. Note in line 70 the use of PLOT and DRAWTO in GRAPHICS 0. First the COLOR is set to the value of the character you wish to propagate with the DRAWTO.
100–120	Solicit a category or item number and verify. If not valid, sound the keyboard speaker.
130	Display all categories in two-column format.
140–170	Display one screenful of items in two-column format.
180–200	Sort all items associated with the specific category and reset the changed flag.
210–250	Display an item for change, solicit the new value, and verify. If the first character is lowercase, change it to uppercase.
260	Set up a display list routine to alter the luminance of the characters in the GRAPHICS 0 graphics part of the screen and the luminance of the characters in the GRAPHICS 0 text window, and change the background luminance of the text window to one shade darker than the graphics part of the screen.
270–300	Display PRINT parameter menu with current settings.
310–340	Print horizontal lines in shopping list printout.
350–470	Display disk directory of .SHP files. Current option is displayed in GRAPHICS 2 mode at the top of the screen, and the directory is displayed in GRAPHICS 0. A text

window is used at the bottom to request selections. The filenames found on the disk are saved in array D$.

480–540 Request a file selection and validate. If invalid, sound keyboard speaker. If valid, build complete filename in FN$.

550–570 Set up mixed GRAPHICS 2/GRAPHICS 0 screen by modifying the display list. The current option is displayed in the GRAPHICS 2 portion.

580–590 Make a sound. The pitch of the sound is determined by the value passed in variable K.

600–720 Initialize constants and variables; set up machine language subroutines and printer parameter defaults.

730 Open the keyboard and screen editor for input.

740–860 Display Main Menu, get selection, and go to appropriate routine.

870–930 Quit option routine. Lines 880–900 are also used by other routines to display the FILE NOT SAVED CONT Y/N? option.

940–1030 Display Categories Menu, get selection, and go to appropriate routine.

1040–1220 Add or Insert category routine.

1230–1250 Change category routine.

1260–1350 Delete category routine.

1360–1480 Renumber category routine.

1490 View category.

1500–1610 Display Items Menu, get selection, and go to appropriate routine.

1620–1630 Sort items routine.

1640–1670 Next Pg option on Items Menu.

1680–1690 Last Pg option on Items Menu.

1700–1880 Add items routine.

1890–1940 Change item routine.

1950–2010 Delete item routine.

2020–2120 Load a file from disk for Main Menu Read option, or View/Print option and no file in memory. See write-up for lines 2640–2830 for file characteristics. If a disk error occurs that inhibits loading the complete file, responding to the RETRY Y/N message with an N will cause retention of data already read to help in the re-creation of the file. Before continuing, check the last category displayed, then resave and reread the partial file before rebuilding.

2130–2610 Print a shopping list based on PRINT parameters. The title is printed in the Epson enlarged character set if the number of columns is greater than 1. The PRINT routine uses Epson MX-100 printer control characters.

2620–2630 Read file initialization.
2640–2830 Save file routine. The file is a sequential file with items following their respective categories. Prior to writing a new category, a one-character binary value of the category number is written. This is used to detect each new category. Before writing the file to disk, a check is made to see if the file will fit. If a new filename is entered for the SAVE and the filename already exists on the disk, an OVERWRITE Y/N request message will be issued.
2840–2930 Delete file routine.
2940–2970 Error recovery subroutine for disk and printer errrors.

Shopping List

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
DB 20 GOTO 600
BP 30 GET #K5,ANS:FOR R=K1 TO LEN(R$):IF ANS=A
   SC(R$(R,R)) THEN RETURN
OK 40 NEXT R:GOSUB 50:GOTO 30
NF 50 FOR N=K15 TO K40:POKE 53279,K0:NEXT N:RE
   TURN
KN 60 POKE 752,K1:? #K6;CS$;:POSITION K18-LEN(
   R$)/K2,K0:? #6;"■";R$;"■";
AO 70 COLOR 124:PLOT K19,K1:DRAWTO K19,K19:RET
   URN
BA 80 U=USR(PK,656,K1,657,K2):? "I ENTER NUMBE
   R TO ";R$;" :{8 SPACES}I";
IB 90 U=USR(PK,752,K0,657,27):? ":";:INPUT #K4
   ,A$:POKE 752,K1:? "{Z}";:IF LEN(A$)<K8 T
   HEN A$(LEN(A$)+K1,K8)=BL$
EB 100 A$=A$(K1,K8):IF A$=BL$(33) THEN X=K0:RE
   TURN
HA 110 TRAP 120:X=VAL(A$):TRAP CLEAR:IF X>K0 A
   ND X<=K AND INT(X)=X THEN RETURN
AG 120 GOSUB 50:POKE 656,K1:GOTO 90
OO 130 FOR I=X TO C:X=P(I):J=I>K19:POSITION (I
   <K10)+K20*J,I-K19*J:? #6;STR$(I);" ";F$
   (X,X+Z);:NEXT I:RETURN
NF 140 R$=I$:GOSUB 60
HJ 150 X=F+EL+(PG-K1)*K38*EL:FOR I=K1 TO K38:I
   F X=T THEN 170
BM 160 J=I>K19:POSITION (I<K10)+K20*J,I-K19*J:
   ? #6;STR$(I);" ";F$(X,X+Z);:X=X+EL:NEXT
    I
CM 170 K=I-K1:RETURN
KC 180 X=(T-F)/EL:IF X<K3 THEN 200
JD 190 U=USR(PK,204,Z,205,EL,206,K0):U=USR(SRT
   ,ADR(F$(F+EL)),X-K1)
```

```
IE 200 CHNG=K0:RETURN
NB 210 U=USR(PK,656,K1,657,K2):? "I ENTER CHAN
       GE{3 SPACES}:";F$(J,J+Z);"I";:U=USR(PK,
       752,K0,657,K19)
EC 220 ? ":";:INPUT #K4,A$:POKE 752,K1:? "{Z}"
       ;:IF LEN(A$)<EL THEN A$(LEN(A$)+K1,EL)=
       BL$
MI 230 I=ASC(A$):IF I>96 AND I<123 THEN A$(K1,
       K1)=CHR$(I-32)
CA 240 A$=A$(K1,EL):IF A$=BL$(K1,EL) OR A$=F$(
       J,J+Z) THEN RETURN
PK 250 F$(J,J+Z)=A$:FS=K0:I=X>K19:POSITION K3+
       K20*I,X-K19*I:? #6;A$;:CHNG=K1:RETURN
JJ 260 DL=PEEK(560)+PEEK(561)*K256:U=ADR("H■
       {D}{■}{J}■{■}{X}■{J}{■}{W}■h@"):POKE U
       +K2,K:U=USR(PK,512,U,DL+24,130,54286,19
       2):RETURN
BL 270 POSITION K2,K6:? #K6;"1. PRINT NORMAL/C
       ONDENSED.........";MODE$
JP 280 ? #K6;"2. NUMBER OF COLUMNS OF ITEMS...
       ...";COL
OM 290 ? #K6;"3. NUMBER PAGES TO PRINT.......
       ...";PAGES
PP 300 ? #K6;"4. CONTINUOUS FORM/SINGLE SHEET.
       ...";TYPE$:RETURN
NC 310 ? #K1;" ";A$;:IF COL>K1 THEN FOR J=K1 T
       O COL-K1:? #K1;"-";A$;:NEXT J
AM 320 ? #K1;" ":RETURN
FA 330 ? #K1;"I";A$;:IF COL>K1 THEN FOR J=K1 T
       O COL-K1:? #K1;"-";A$;:NEXT J
IK 340 ? #K1;"I":RETURN
LC 350 GOSUB 550
KE 360 GOSUB 580:TRAP 540:OPEN #K1,K6,K0,"D1:*
       .SHP":D$=""
FH 370 TRAP 390:INPUT #K1,A$:TRAP CLEAR:IF LEN
       (A$)<K17 THEN 390
IJ 380 L=LEN(D$)+K1:D$(L)=A$(K3,K10):D$(L+K8)=
       A$(K15):GOTO 370
EJ 390 CLOSE #K1:POSITION K2,K1:? "{Q}{35 R}
       {E}":IF D$="" THEN 460
FL 400 T=LEN(D$)/K22:IF INT(T)<>T THEN D$(LEN(
       D$)+K1)=BL$(K1,K11):T=T+0.5
OM 410 FOR I=K1 TO T:R$=BL$(K5):FOR J=K0 TO K1
       :X=(I+J*T)*K11:L=K18*J+K1
HH 420 R$(L,L)="I":IF D$(X-K10,X-K10)=" " THEN
        450
IK 430 R$(L+K1,L+K4)="{@}  ":R$(L+K2,L+K2)=CHR$
       (64+X/K11):R$(L+K5,L+K12)=D$(X-K10,X-K3
       ):R$(L+K14,L+K16)=D$(X-K2,X)
```

```
PJ 440 FOR X=L+K13 TO L+K6 STEP -K1:IF R$(X,X)
       =" " THEN R$(X,X)=".":NEXT X
MO 450 NEXT J:? R$;"|":NEXT I
LG 460 ? "|";BL$(K1,35);"|":? "|(9 SPACES)";A$
       ;"(10 SPACES)|"
GB 470 ? "(Z)(35 R)(C)":? :? :? "(3 SPACES)ONL
       Y '.SHP' FILES ARE LISTED.":RETURN
IB 480 GOSUB 50
HC 490 POSITION K0,K20:? "(DEL LINE)";:POSITIO
       N K8,K20:? "  SELECTION OR 'RETURN'  ";
KE 500 GET #K5,X:IF X=155 THEN RETURN
IA 510 X=(X-64)*K11:TRAP 480:A$=D$(X-K10,X-K3)
       :TRAP CLEAR:IF A$=BL$(K1,K8) THEN 480
GA 520 FN$="D1:":FOR I=K1 TO K8:IF A$(I,I)<>"
       " THEN FN$(I+K3)=A$(I,I):NEXT I
EH 530 FN$(I+K3)=".SHP":RETURN
PE 540 GOSUB 2940:GOTO 360
MN 550 GRAPHICS K0:DL=PEEK(560)+PEEK(561)*K256
IP 560 U=USR(PK,752,K1,709,K8,710,66,711,182,7
       12,66,DL+K3,71,DL+K6,K7,DL+28,65,DL+29,
       DL,87,K2)
AH 570 ? #K6;"(3 SPACES)";R$;"  file":POKE 87,K
       0:RETURN
NE 580 K=K13
BG 590 FOR I=K15 TO K0 STEP -0.5:FOR J=K2 TO K
       0 STEP -K1:SOUND K0,K-J,K10,I:NEXT J:NE
       XT I:RETURN
EA 600 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K7=7
       :K8=8:K9=9:K10=10:K11=11:K12=12:K13=13:
       K14=14:K15=15
PB 610 K16=16:K17=17:K18=18:K19=19:K20=20:K21=
       21:K22=22:K38=38:K39=39:K40=40:K128=128
       :K256=256
MH 620 MC=K38:ME=400:MF=K20:EL=K16:Z=EL-K1:CLE
       AR=40000:FS=K1:C=K0
HG 630 DIM SRT$(126),ML$(39),MR$(47),PK$(25),M
       ODE$(K1),TYPE$(K1),F$((ME+MC)*EL),D$(MF
       *K11),FN$(K15),RN$(K15)
EE 640 DIM BL$(K40),A$(K40),R$(K40),I$(EL),BZ$
       (K1),CS$(K1),P(MC+K1)
IE 650 RESTORE 5000
MH 660 FOR I=1 TO 126:READ QQ:SRT$(I,I)=CHR$(Q
       Q):NEXT I
DL 670 FOR I=1 TO 39:READ QQ:ML$(I,I)=CHR$(QQ)
       :NEXT I
EB 680 FOR I=1 TO 47:READ QQ:MR$(I,I)=CHR$(QQ)
       :NEXT I
JI 690 FOR I=1 TO 25:READ QQ:PK$(I,I)=CHR$(QQ)
       :NEXT I:RESTORE
```

```
AB 700 SRT=ADR(SRT$):ML=ADR(ML$):MR=ADR(MR$):P
       K=ADR(PK$)
IF 710 BL$=" ":BL$(K40)=" ":BL$(K2)=BL$:FN$=""
       :RN$=FN$:BZ$=CHR$(253):CS$=CHR$(125)
PH 720 MODE$="C":COL=K5:PAGES=K1:TYPE$="S":P(K
       0)=K0:RS=K0
AF 730 OPEN #K5,K4,K0,"K:":OPEN #K4,K4,K0,"E:"
FF 740 GRAPHICS K2:U=USR(PK,710,K4,711,72,712,
       K4)
NO 750 POKE 201,K4:POSITION K3,K1
MA 760 ? #K6;"SHOPPING LIST":? #K6
OI 770 ? #K6,"[V]iew list"
FL 780 ? #K6,"[P]rint list"
MB 790 IF C THEN ? #K6,"[R]ead file":? #6,"[S]ave
       file"
HP 800 ? #K6,"[C]reate file"
HP 810 ? #K6,"[D]elete file"
NP 820 ? #K6,"[Q]uit":POKE 752,K1
AJ 830 ? CS$:? :? "INSERT DATA DISK AND ENTER
       SELECTION."
IF 840 R$="VPRSCDQ":IF  NOT C THEN R$(K3,K4)=R
       $
IA 850 GOSUB 30
CO 860 ON R GOTO 2020,2130,2620,2640,940,2850
JA 870 IF FS THEN 910
IJ 880 ? CS$:? :? "{3 SPACES}FILE NOT SAVED CO
       NTINUE [Y/N]";
AM 890 I=R:R$="NY":GOSUB 30:IF R=K1 THEN 830
AO 900 R=I:FS=K1:GOTO 860
NP 910 CLOSE #K4:CLOSE #K5:GRAPHICS K0:POKE 20
       1,K10
HM 920 POSITION K9,K6:? "[SHOPLIST TERMINATED]
       ";
HP 930 POSITION K2,K18:END
IF 940 IF  NOT FS THEN 880
PF 950 F$="":C=K0:E=K0
NB 960 GRAPHICS K0:U=USR(PK,709,K12,710,198,71
       2,50,703,K4):K=196:GOSUB 260
AP 970 R$="[CATEGORIES]":GOSUB 60:IF  NOT C THEN
        R=K1:GOTO 1070
AA 980 X=K1:GOSUB 130
JB 990 ? CS$;"{Q}{8 R}{W}{7 R}{W}{8 R}{W}{7 R}
       {E}"
BJ 1000 ? "| [A]DD{4 SPACES}| [V]IEW  | [C]HANGE |
       {3 SPACES}?{3 SPACES}|"
JK 1010 ? "| [I]NSERT | [R]ENUM | [D]ELETE | {5 R} |
       "
CE 1020 ? "{Z}{7 R}['M' FOR MAIN MENU]{6 R}
       {C}";
```

```
JG 1030 R$="AICDRVM":GOSUB 30:ON R GOTO 1070,1
        210,1230,1260,1360,1490,740
FI 1040 ? CS$;BZ$:? :? "■MAXIMUM NUMBER CATEGO
        RIES ENTERED■";:GOTO 1060
KO 1050 ? CS$;BZ$:? :? ,"■NO CATEGORIES TO ";
        R$;
BC 1060 FOR I=K1 TO K256:NEXT I:GOTO 990
FL 1070 IF C=MC THEN 1040
FO 1080 ? CS$;"{Q}{33 R}{E}":? "|{33 SPACES}|"
LK 1090 ? "{Z}{5 R}■'RETURN' FOR OPTIONS■{5 R}
        {C}";
FM 1100 IF R>K1 THEN K=C:GOSUB 80:IF  NOT X TH
        EN 990
HF 1110 ON R GOTO 1120,1120,1250,1280,1380,151
        0
AA 1120 U=USR(PK,656,K1,657,K2):? "| ENTER CAT
        EGORY :{16 SPACES}|  ";:U=USR(PK,752,K0
        ,657,K19)
HD 1130 ? ":";:INPUT #K4,A$:POKE 752,K1:? "
        {Z}";:IF LEN(A$)<EL THEN A$(LEN(A$)+K1
        ,EL)=BL$
MP 1140 A$=A$(K1,EL):IF A$=BL$(K1,EL) THEN 990
HA 1150 C=C+K1:FS=K0:K=LEN(F$)+K1:F$(K)=A$:IF
        R=K2 THEN 1190
EG 1160 P(C)=K:X=C
JO 1170 GOSUB 130:IF C<MC THEN ON R GOTO 1120,
        990
AG 1180 ? CS$:? :? "■MAXIMUM NUMBER CATEGORIES
        ■ENTERED■";:K=K40:GOSUB 590:GOTO 990
HD 1190 ? CS$;:I=P(X):J=ADR(F$(I)):U=USR(MR,J,
        J+EL,K-I-K1):F$(I,I+Z)=A$
KI 1200 FOR I=C TO X+K1 STEP -K1:P(I)=P(I-K1)+
        EL:NEXT I:GOTO 1170
MM 1210 IF C THEN R$="INSERT":GOTO 1070
AK 1220 R$="■INSERT■":GOTO 1050
KA 1230 IF C THEN R$="CHANGE":GOTO 1080
DN 1240 R$="■CHANGE■":GOTO 1050
DF 1250 J=P(X):GOSUB 210:FS=K0:GOTO 990
LA 1260 IF C THEN R$="DELETE":GOTO 1080
EN 1270 R$="■DELETE■":GOTO 1050
BK 1280 I=P(X):U=USR(PK,656,K1,657,K2):? "| DE
        LETE ";F$(I,I+Z);":■Y/N■ ? |":? "
        {DEL LINE}{Z}{33 R}{C}";
CM 1290 R$="NY":GOSUB 30:IF R=K1 THEN 990
JG 1300 L=LEN(F$)+K1:P(C+K1)=L:J=P(X+K1):K=J-I
        :IF K=EL THEN 1320
EE 1310 ? CS$;BZ$:? :? "ITEMS STILL ASSIGNED -
         CONTINUE ■Y/N■";:R$="NY":GOSUB 30:IF
        R=K1 THEN 990
```

```
PH 1320 ? CS$:C=C-K1:IF X>C THEN F$(I)="":GOTO
        1350
IO 1330 U=USR(ML,ADR(F$(J)),ADR(F$(I)),L-J):F$
        =F$(K1,L-K-K1)
ON 1340 FOR I=X TO C:P(I)=P(I+K1)-K:NEXT I:GOS
        UB 130
OF 1350 E=LEN(F$)/EL-C:J=C>K18:POSITION K20*J,
        C+K1-K19*J:? #6;BL$(K1,EL+K3);:FS=(C=K
        0):GOTO 990
DP 1360 IF C>K1 THEN R$="RENUM ":GOTO 1080
PL 1370 R$="RENUMBER ":GOTO 1050
HN 1380 I=X:R$="MAKE  ":GOSUB 80:IF  NOT X OR
        X=I THEN 990
MA 1390 ? CS$:P(C+K1)=LEN(F$)+K1:F=P(I):K=P(I+
        K1)-F:R=K:D$(MF*K11)="X"
BL 1400 IF I>X THEN F=P(X):GOTO 1450
II 1410 L=LEN(D$):IF L>K THEN L=K
AC 1420 T=P(X+K1)-L:J=ADR(F$(F)):U=USR(ML,J,AD
        R(D$),L):U=USR(ML,J+L,J,T-F):U=USR(ML,
        ADR(D$),ADR(F$(T)),L)
BN 1430 K=K-L:IF K THEN 1410
ND 1440 FOR J=I+K1 TO X:P(J)=P(J+K1)-R:NEXT J:
        X=I:GOSUB 130:GOTO 990
IM 1450 L=LEN(D$):IF L>K THEN L=K
KG 1460 T=P(I+K1)-L:U=USR(ML,ADR(F$(T)),ADR(D$
        ),L):J=ADR(F$(F)):U=USR(MR,J,J+L,T-F-K
        1):U=USR(ML,ADR(D$),J,L)
CF 1470 K=K-L:IF K THEN 1450
KE 1480 FOR J=I TO X+K1 STEP -K1:P(J)=P(J-K1)+
        R:NEXT J:GOSUB 130:GOTO 990
DN 1490 IF C THEN R$="VIEW  ":GOTO 1080
NB 1500 R$="VIEW ":GOTO 1050
LH 1510 GRAPHICS K0:U=USR(PK,709,K12,710,K6,71
        2,50,703,K4):K=K4:GOSUB 260
JB 1520 P(C+K1)=LEN(F$)+K1:L=X:F=P(L):T=P(L+K1
        ):PG=K1:CHNG=K0
CE 1530 I$=F$(F,F+Z):FOR I=EL TO K1 STEP -K1:I
        F I$(I,I)=" " THEN NEXT I
MN 1540 I$=I$(K1,I):FOR I=K1 TO LEN(I$):I$(I,I
        )=CHR$(ASC(I$(I))+K128):NEXT I
OA 1550 GOSUB 140
LL 1560 ? CS$;"{Q}{9 R}{W}{6 R}{W}{8 R}{W}{7 R}
        {E}"
EJ 1570 ? "| NEXT PG | ADD  | CHANGE | SORT  |
        "
CO 1580 ? "| LAST PG | CATS | DELETE | PG 1  |
        "
KC 1590 ? "{Z}{8 R} 'M' FOR MAIN MENU {6 R}
        {C}";
```

```
PP 1600 R$="NLTACDMPS":GOSUB 30:IF (R=3 OR R=7
        ) AND CHNG THEN GOSUB 180
CM 1610 ON R GOTO 1640,1680,960,1700,1910,1950
        ,740,1890
OP 1620 IF T-F>EL THEN 1890
EJ 1630 R$="SORT":GOTO 1930
AP 1640 IF (T-F)/EL-K1>PG*K38 THEN PG=PG+K1:GO
        TO 1550
KK 1650 ? CS$:? :? ,," NO NEXT";
GO 1660 ? " TO SHOW ";:GOSUB 50
EB 1670 FOR I=K1 TO K128:NEXT I:GOTO 1560
DA 1680 IF PG>K1 THEN PG=PG-K1:GOTO 1550
OD 1690 ? CS$:? :? ,," NO LAST";:GOTO 1660
OP 1700 K=KØ:X=KØ:IF E=ME THEN 1850
AJ 1710 R$=I$:GOSUB 60
FP 1720 ? CS$;"{Q}{33 R}{E}":? "!{33 SPACES}!"
FA 1730 ? "{Z}{5 R}'RETURN' FOR OPTIONS{5 R}
        {C}":? ,"{3 SPACES}REMAINING ENTRIES:
        ";ME-E;"{3 SPACES}";
IO 1740 IF R>K4 THEN GOSUB 80:IF  NOT X THEN 1
        560
AI 1750 J=F+(PG-K1)*K38*EL+X*EL:ON R-K4 GOTO 1
        940,1970
MH 1760 U=USR(PK,656,K1,657,K2):? "! ENTER ITE
        M --->:{16 SPACES}!  ";:U=USR(PK,702,KØ
        ,656,K3,657,28)
BD 1770 ? ME-E;"{3 SPACES}";:U=USR(PK,656,K1,7
        52,KØ,657,K19)
BG 1780 ? ":";:INPUT #K4,A$:POKE 752,K1:? "
        {Z}";:POKE 702,64:IF LEN(A$)<EL THEN A
        $(LEN(A$)+K1,EL)=BL$
AH 1790 A$=A$(K1,EL):IF A$=BL$(K1,EL) THEN 186
        Ø
PM 1800 I=ASC(A$):IF I>96 AND I<123 THEN A$(K1
        ,K1)=CHR$(I-32)
DL 1810 FS=KØ:E=E+K1:X=X+EL:K=K+K1:I=LEN(F$)+K
        1:F$(I)=A$
PO 1820 IF I-T THEN J=ADR(F$(T)):U=USR(MR,J,J+
        EL,I-T-K1):F$(T,T+Z)=A$
ON 1830 IF K>K38 THEN GOSUB 60:K=K1
JJ 1840 J=K>K19:POSITION (K<K1Ø)+K20*J,K-K19*J
        :? #6;STR$(K);" ";A$;:IF E<ME THEN ON
        (K=K1 AND X>EL)+K1 GOTO 1760,1720
NE 1850 ? CS$:? :? ," MAXIMUM ITEMS ENTERED
        ";:K=K40:GOSUB 590
KI 1860 IF  NOT X THEN 1890
OI 1870 CHNG=K1:IF E<ME THEN ? CS$;
EJ 1880 FOR I=L+K1 TO C:P(I)=P(I)+X:NEXT I:T=T
        +X
MJ 1890 IF CHNG THEN GOSUB 180
```

```
FI 1900 PG=K1:GOTO 1550
PJ 1910 IF T-F>EL THEN R$="CHANGE":GOTO 1720
GJ 1920 R$="CHANGE"
JI 1930 ? CS$;BZ$;:? :? ,,"  NO ITEMS TO ";R$;"
        ";:GOTO 1670
GB 1940 POKE 702,K0:GOSUB 210:POKE 702,64:GOTO
         1560
AK 1950 IF T-F>EL THEN R$="DELETE":GOTO 1720
LK 1960 R$="DELETE":GOTO 1930
GJ 1970 U=USR(PK,656,K1,657,K2):? "I DELETE ";
        F$(J,J+Z);": Y/N ? I":? "{DEL LINE}
        {Z}{33 R}{C}";
FM 1980 R$="NY":GOSUB 30:IF R=K1 THEN 1560
EJ 1990 ? CS$;:I=ADR(F$(J)):U=USR(ML,I+EL,I,LE
        N(F$)+K1-J-EL):FOR I=L+K1 TO C:P(I)=P(
        I)-EL:NEXT I:E=E-K1:FS=K0:T=T-EL
DF 2000 F$=F$(K1,LEN(F$)-EL):GOSUB 150:IF K<K3
        8 THEN J=K>K18:POSITION K20*J,I-K19*J:
        ? #6;BL$(K1,EL+K3);
MI 2010 GOTO 1560
GE 2020 IF C THEN 960
HH 2030 R$="View"
PI 2040 GOSUB 350:GOSUB 490:IF X=155 THEN 740
AL 2050 POKE 752,K1:POSITION K2,K20:? "
        {DEL LINE}{3 SPACES} LOADING FILE ";F
        N$
NP 2060 GOSUB 580:F$="":TRAP 2110:OPEN #K1,K4,
        K0,FN$:E=K0:C=K0:RN$=FN$
BB 2070 INPUT #K1,A$:IF LEN(A$)=K1 THEN C=C+K1
        :P(C)=LEN(F$)+K1:GOTO 2070
BN 2080 IF E<ME THEN F$(LEN(F$)+K1)=A$:E=E+K1:
        GOTO 2070
DM 2090 POSITION K3,K21:? BZ$;"MEMORY FILE TOO
         SMALL - ENTER  RET ";:R$=CHR$(155):GO
        SUB 30
GP 2100 CLOSE #K1:TRAP CLEAR:ON R GOTO 960,214
        0,960
KC 2110 E=E-C:IF PEEK(195)=136 THEN 2100
EP 2120 GOSUB 2940:GOTO 2060
JF 2130 IF  NOT C THEN R$="Print":GOTO 2040
AD 2140 R$="Print":GOSUB 550:U=USR(PK,709,K2,7
        10,248,711,116,712,152,703,K4,DL+22,13
        0)
PG 2150 U=ADR("H{}{}{M}{J}T{M}{X}P{N}{M}{W}Ph
        ?"):U=USR(PK,512,U,54286,192)
JF 2160 POSITION K11,K3:? #K6;" PRINTOUT OPTIO
        NS ":GOSUB 270
HH 2170 ? #K6:? #K6:? #K6;"{3 SPACES}ENTER '#
        1{3 SPACES} TO CHANGE OPTION"
```

```
IH 2180 ? #K6;"{3 SPACES}ENTER 'M'{3 SPACES}
        FOR MAIN MENU"
GD 2190 ? #K6;"{3 SPACES}ENTER 'RET' TO STAR
        T PRINT"
MC 2200 ? CS$;:U=USR(PK,656,K0,657,K12):? "EX
        TER SELECTION ";
KK 2210 R$="1234M":R$(K6)=CHR$(155):GOSUB 30:?
         CS$;:U=USR(PK,656,K0,657,K5)
HG 2220 CLOSE #K1:? CS$:U=USR(PK,656,K0,657,K5
        ):ON R GOTO 2240,2250,2270,2290,740,23
        10
PD 2230 GOSUB 270:SOUND K0,29,K10,K8:FOR I=K1
        TO K10:NEXT I:SOUND K0,K0,K0,K0:FOR I=
        K1 TO 100:NEXT I:GOTO 2200
PO 2240 MODE$=CHR$(145-ASC(MODE$)):? ," PRINT
        CHANGED TO ";CHR$(ASC(MODE$)+K128);"
        ";:GOTO 2230
AD 2250 ? "ENTER NUMBER COLUMNS (1-9): ";:R$=
        "123456789":GOSUB 30:COL=R
BN 2260 ? CS$;:POKE 656,K0:POKE 657,K9:? "COL
        UMNS CHANGED TO ";CHR$(COL+176);"";:G
        OTO 2230
CG 2270 ? "ENTER NUMBER PAGES{3 SPACES}(1-9):
        ";:R$="123456789":GOSUB 30:PAGES=R
AA 2280 ? CS$;:POKE 656,K0:POKE 657,K9:? "PAG
        ES CHANGED TO ";CHR$(PAGES+176);"";:G
        OTO 2230
HN 2290 TYPE$=CHR$(150-ASC(TYPE$)):? ,"  FORM
        CHANGED TO ";CHR$(ASC(TYPE$)+K128);"
        ";:GOTO 2230
GN 2300 U=USR(PK,559,34,657,K3):RS=K1:GOSUB 29
        40:IF RS THEN RS=K0:GOTO 2600
EB 2310 POKE 657,K3:? "READY PRINTER - ENTER
        SELECTION ";:R$="1234M":R$(K6)=CHR$(15
        5):GOSUB 30:IF R<K6 THEN 2220
IM 2320 ? CS$;:TRAP 2300:OPEN #K1,K8,K0,"P:"
AB 2330 ? #K1;CHR$(27);"O";:REM ** RELEASE SKI
        P-OVER PERFORATION
BE 2340 ? #K1;CHR$(27);"8";:REM ** DESELECT PA
        PER OUT DETECTOR
MN 2350 ? #K1;CHR$(K18-K3*(MODE$="C"));:REM **
         SET NORMAL/CONDENSED PRINT
FA 2360 ? #K1;CHR$(27);"A";CHR$(K6);:REM SET L
        INE SPACING TO 6/72
IG 2370 POKE 559,K0:IF F$(K1,K1)<CHR$(K128) TH
        EN FOR I=K1 TO C:X=P(I):F$(X,X)=CHR$(A
        SC(F$(X))+K128):NEXT I
DH 2380 A$="----------------------"
KI 2390 D$="|  ":D$((EL+K5)*COL)="  ":D$(K3)=D$(
        K2):D$(LEN(D$))="|"
```

```
FH 2400 X=INT((E+C+COL-K1)/COL):FOR L=K1 TO PA
        GES:GOSUB 3100:? #K1;D$:? #K1;D$
EM 2410 IF COL=K1 THEN T=INT((LEN(D$)-K13)/K2)
        +K1:D$(T,T+K12)="SHOPPING LIST":? #K1;
        D$:D$(T,T+K12)=BL$:GOTO 2430
AJ 2420 T=INT((LEN(D$)-26)/K2)+K1:D$(T,T+25)="
        {5 ,}{N}SHOPPING LIST{6 ,}{T}":? #K1;D
        $:D$(T,T+25)=BL$
FF 2430 ? #K1;D$:? #K1;D$:GOSUB 330
AJ 2440 FOR I=K1 TO X:FOR J=K0 TO COL-K1:K=(I+
        J*X)*EL
FG 2450 R$="|":R$(K2)=BL$(K1,EL):R$(EL+K2)="|
        {3 SPACES}":IF K>LEN(F$) THEN 2490
LE 2460 IF F$(K-Z,K-Z)<CHR$(K128) THEN R$(K2,E
        L+K1)=F$(K-Z,K):GOTO 2490
LD 2470 I$=F$(K-Z,K):I$(K1,K1)=CHR$(ASC(I$)-K1
        28):FOR T=EL TO K1 STEP -K1:IF I$(T,T)
        =" " THEN NEXT T
EG 2480 I$=I$(K1,T):F=INT((EL+K6-T)/K2)+K1:R$(
        EL+K2,EL+K2)=" ":R$(F,F+T-K1)=I$:R$(K2
        ,K3)="**":R$(EL+K4)="**"
KL 2490 ? #K1;R$;:NEXT J:? #K1;"|":GOSUB 330:N
        EXT I:D$(K2,K7)="NOTES:":? #K1;D$:D$(K
        2,K7)=BL$
KC 2500 FOR I=K1 TO K9:? #K1;D$:NEXT I:GOSUB 3
        10
KA 2510 ? #K1;CHR$(K12):REM HOME PAPER
BH 2520 IF TYPE$="C" OR L=PAGES THEN 2550
IP 2530 POKE 559,34:U=USR(PK,656,K0,657,K3):?
        " READY PRINTER - DEPRESS 'RETURN' ";
GP 2540 R$="1234M":R$(K6)=CHR$(155):GOSUB 30:I
        F R<K6 THEN 2220
AM 2550 POKE 559,K0:NEXT L:IF TYPE$="S" THEN 2
        590
KP 2560 ? #K1;CHR$(27);"N";CHR$(K6);:REM ** SE
        T SKIP-OVER PERFORATION 6 LINES
DN 2570 ? #K1;CHR$(27);"9";:REM SELECT PAPER O
        UT DETECTOR
FE 2580 ? #K1;CHR$(27);"2":REM SET NORMAL LINE
         SPACING
LN 2590 CLOSE #K1:TRAP CLEAR
ND 2600 IF F$(K1,K1)>CHR$(K128) THEN FOR I=K1
        TO C:X=P(I):F$(X,X)=CHR$(ASC(F$(X))-K1
        28):NEXT I
EE 2610 GRAPHICS K2:POKE 559,34:GOTO 740
LC 2620 IF  NOT FS THEN 880
JH 2630 R$="read":GOTO 2040
GG 2640 OV=K0:R$="save":GOSUB 350:IF RN$="" TH
        EN 2660
```

```
JI 2650 POSITION K4,K20:? "OK TO SAVE ";RN$;"
        Y/N ";:R$="NY":GOSUB 30:IF R=K2 THEN
        FN$=RN$:OV=K1:GOTO 2700
LI 2660 POSITION K2,K20:POKE 752,K0:? #K4;"
        {DEL LINE}ENTER FILENAME OR 'RET'
        :";:INPUT #K4,I$:POKE 752,K1:?
BG 2670 IF I$="" THEN 740
AK 2680 IF LEN(I$)>K8 OR I$(K1,K1)<"A" OR I$(K
        1,K1)>"Z" THEN ? BZ$;:GOTO 2660
JB 2690 FN$="D1:":FN$(K4)=I$:FN$(LEN(FN$)+K1)=
        ".SHP"
FK 2700 X=K0:TRAP 2830:OPEN #K1,K6,K0,FN$
AN 2710 TRAP 2830:INPUT #K1,A$:TRAP CLEAR:IF L
        EN(A$)=K17 THEN X=X+VAL(A$(K15)):GOTO
        2710
BO 2720 X=X+VAL(A$(K1,K3)):CLOSE #K1:IF X>INT(
        (C*2+E*(EL+K1)+K1+124)/125) THEN 2760
PN 2730 POSITION K2,K20:? "{DEL LINE}INSUFFICI
        ENT ROOM ON DISK - HIT 'RET'";BZ$;
BK 2740 GET #K5,ANS:IF ANS<>155 THEN 2740
KC 2750 GOTO 740
AC 2760 IF OV=K1 OR X=VAL(A$(K1,K3)) THEN 2780
HJ 2770 POSITION K2,K20:? "{DEL LINE}
        {6 SPACES}OVERWRITE ";I$;" Y/N ";:R$=
        "NY":GOSUB 30:IF R=K1 THEN 2660
OL 2780 POSITION K2,K20:? "{DEL LINE}
        {4 SPACES}SAVING FILE ";FN$;
LK 2790 GOSUB 580:TRAP 2800:XIO 36,#K1,K0,K0,F
        N$
FH 2800 P(C+K1)=LEN(F$)+K1:TRAP 2840:OPEN #K1,
        K8,K0,FN$
LB 2810 FOR I=K1 TO C:? #K1;CHR$(I):FOR J=P(I)
         TO P(I+K1)-EL STEP EL:? #K1;F$(J,J+Z)
        :NEXT J:NEXT I
NF 2820 CLOSE #K1:TRAP CLEAR:XIO 35,#K1,K0,K0,
        FN$:FS=K1:RN$=FN$:GOTO 740
FI 2830 GOSUB 2940:GOTO 2700
GC 2840 GOSUB 2940:GOTO 2790
IL 2850 R$="delete":GOSUB 350
LA 2860 GOSUB 490:IF X=155 THEN 740
GF 2870 POSITION K3,K20:? "OK TO DELETE ";FN$;
        " Y/N ";:R$="NY"
II 2880 GOSUB 30:IF R=K1 THEN 2860
HB 2890 POSITION K2,K20:? "{DEL LINE}
        {4 SPACES}DELETING FILE ";FN$;
BM 2900 GOSUB 580:TRAP 2930:XIO 36,#K1,K0,K0,F
        N$:XIO 33,#K1,K0,K0,FN$:TRAP CLEAR
JE 2910 IF FN$=RN$ THEN FS=K0
KB 2920 GOTO 740
```

```
FL 2930 GOSUB 2940:GOTO 2900
MM 2940 CLOSE #K1:POSITION K3,K21:? BZ$;"ERROR
        ";PEEK(195);" ENCOUNTERED - RETRY [Y/
        N]";
ME 2950 I=R:R$="NY":GOSUB 30:POSITION K2,K21:?
         "{DEL LINE}";:IF R=K2 THEN R=I:RS=K0:
        RETURN
BE 2960 IF RS THEN RETURN
MP 2970 POP :GOTO 740
AH 5000 DATA 104,104,133,217,104,133,216
BF 5010 DATA 104,133,209,104,133,208,169
EF 5020 DATA 0,133,218,133,207,162,1
BK 5030 DATA 165,216,133,214,165,217,133
NK 5040 DATA 215,24,165,214,133,212,101
BC 5050 DATA 205,133,214,165,215,133,213
KD 5060 DATA 105,0,133,215,164,203,165
OD 5070 DATA 206,240,10,177,214,209,212
JA 5080 DATA 144,44,240,12,176,19,177
HM 5090 DATA 214,209,212,144,13,240,2
OC 5100 DATA 176,30,200,196,204,240,227
IE 5110 DATA 176,23,144,223,169,1,133
PA 5120 DATA 218,164,205,136,177,214,72
BE 5130 DATA 177,212,145,214,104,145,212
EI 5140 DATA 192,0,208,241,232,224,0
LH 5150 DATA 208,2,230,207,228,208,208
DA 5160 DATA 172,165,209,197,207,208,166
IL 5170 DATA 165,218,201,0,208,144,96
AM 5180 DATA 104,104,133,215,104,133,214
BB 5190 DATA 104,133,217,104,133,216,104
KD 5200 DATA 133,218,104,170,160,0,177
KJ 5210 DATA 214,145,216,200,208,4,230
BK 5220 DATA 215,230,217,202,208,242,198
HG 5230 DATA 218,16,238,96
BB 5240 DATA 104,104,133,255,104,133,254
AD 5250 DATA 104,133,253,104,133,252,104
OK 5260 DATA 170,24,101,255,133,255,138
OH 5270 DATA 24,101,253,133,253,104,168
DA 5280 DATA 177,254,145,252,136,192,255
DH 5290 DATA 208,247,198,253,198,255,202
GM 5300 DATA 224,255,208,238,96
HG 5310 DATA 104,74,170,160,0,104,133
KD 5320 DATA 255,104,133,254,104,240,4
BI 5330 DATA 200,145,254,136,104,145,254
KD 5340 DATA 202,208,237,96
```

Chapter 4

Coupon File

Stan Silverman

"Coupon File" is a practical coupon-sorting program. It might even help you save some money. Requires at least 32K and a disk drive.

"That's the most ridiculous thing I've ever heard," she said. "A computer program to keep track of store coupons? You remember the hours we spent trying to use the computer to balance our checkbook, don't you? And you expect me to think that this will be different?"

I flinched. She did conjure up images of those endless sessions in front of the screen, with the incessant whirring of the disk drive in the background, as we tried to use the computer for a task better done with pencil, paper, and calculator.

But "Coupon File" is different. It is a practical coupon-sorting program, with features that make it extremely useful, and we've found it to be a valuable money-saving tool.

Program Requirements

Coupon File requires at least 32K of memory and one disk drive. That gives it a maximum capacity of 400 coupons. In a 40K (or 48K) system, the capacity is 600 coupons. The program checks the size of installed memory and adjusts for it accordingly.

Before using the program, you will have to write a reference number on each of your coupons. That lets you identify the coupons when you want to take them to the store. The number has no meaning to the program, but it does check to make sure that you don't try to use a number more than once. The allowable range of reference numbers allowed is 0–9999, so you should be able to use the program for many years without worrying about running out of numbers.

Data Entry

Every time the program needs information from you, it will display a rectangle into which your keyboard response will go. You will be able to see how much space remains for your use in each information field. The information fields that you will use are described below.

Reference Number. Up to four numerals will be accepted. If

you don't want to enter all four digits, press RETURN to end the entry.

Description. Up to 17 characters of any kind will be accepted. Use RETURN to terminate descriptions less than 17 characters.

Amount. Up to four characters, including the decimal point (period) will be accepted. Press RETURN to complete entries of less than four characters.

Dates. These entries are in the form of MM/DD/YY. All six numerals must be entered. It is not necessary to enter the slashes, as the computer will place them in the rectangle for you.

After you have completed each field, you can verify that the displayed information is correct by typing a Y. If you made an error, entering N will clear out the rectangle and let you reenter the information.

The Menu

After the program has been initialized and all of the coupon data has been loaded from disk, the program will display the number of coupons that is on file and the highest reference number in use. Next, the menu will be displayed. The program will return to this menu at the end of every operation. You may return to the menu at any time by pressing the ESC (Escape) key. You don't have to worry about confusing the program if you abort an operation with the ESC key. The program performs its operation only upon receiving a final verification from you and will not be left in limbo if you change your mind about what you are doing.

The menu will give you these choices:

1 Add Coupon
2 Delete Coupon
3 Sort by Date
4 Sort by Description
5 List All Coupons
6 List Expired Coupons
7 List by Date Range
8 List by Description Range
9 End Session

Press the number corresponding to the choice you want.

Add Coupon. You will be asked to enter a reference number, which will be checked to make sure that a coupon of the same number has not been filed before. Next, you will be asked to enter the description, the amount that the coupon is worth, and its expiration date. If the coupon does not have an expiration date, simply enter a date like 12/31/99 to indicate unlimited validity. After all of the information has been entered, a facsimile of the coupon is displayed for final verification. If you wish, you can reject the coupon at that point and create a new one.

Delete Coupon. You will be asked to enter the reference number of the coupon you wish deleted. If there is no coupon on file with that reference number, you will be notified of that fact. If the coupon is in the file, a facsimile of it will be displayed. You will be asked for verification before it is deleted from the file.

Sort by Date. This operation (as well as Sort by Description) is included to make for more useful coupon listings. It is not required. No other operation is dependent upon the coupons being sorted, and the program will give you all the information you ask of it whether or not the date is sorted.

Unfortunately, the sorts are slow. Sorting 300 coupons can take a half-hour. An onscreen notice will inform you that the screen will go dark during sorts. This is done to improve the sort speed. To wake you up, the built-in speaker will sound at the completion of the sort.

Sort by Description. Of the two sorts, this one is probably the most useful.

List All Coupons. The file of coupons will be listed to the screen in the order in which you entered them, unless you have subsequently sorted them. When the screen is filled, you can either continue the listing or return to the menu.

List Expired Coupons. You will be asked to enter the current date. The program will then list all expired coupons (those with earlier expiration dates) on the screen. Make a note of their reference numbers if you wish to delete them later.

List by Date Range. You will be asked to enter a starting date and then an ending date. The program will list all coupons whose dates fall within that range.

List by Description Range. You will be asked to enter a starting description and an ending description. The program

will list all coupons whose descriptions fall within that range. For example, if you want to list all coupons whose descriptions begin with C, you should enter C as the starting description and D as the ending description. Similarly, if you want to list all coupons whose descriptions begin with CEREAL, enter CEREAL as the starting description and CEREALA as the ending description. Play with this feature for a few minutes and you will quickly learn how to use it.

End Session. If you have made any additions or deletions to the file or if you have sorted the coupons during the session, the disk will be updated to reflect these changes. You will be asked not to turn off the computer until the disk has stopped. No disk operation will occur if you have only listed coupons to the screen.

File Initialization

The first time you use Coupon File, it will create a data file containing one dummy coupon. You may delete this coupon anytime after you have added one of your own. The program needs at least one coupon in the file in order to operate properly.

Typing the Program

In order to provide memory space for as many coupons as possible, several memory-saving techniques are used to reduce the amount of memory required by the program itself. Techniques include the use of strings to store most of the numeric information, the use of variables for frequently used constants and line numbers, and the use of multiple-statement lines.

To get the most from multiple-statement lines, the abbreviations for BASIC's reserved words are often used to pack instructions into the three physical lines allowed for a logical program line. *This means that as you type in the program you may have to use abbreviations for the BASIC keywords as well as eliminating spaces wherever possible.* For example, if you were to see a statement like this

```
300 FOR I=1 TO 1000:NEXT I:RETURN
```

you would type the following:

```
300F.I=K1TOK1000:N.I:RET.
```

Obviously, for a short line such as this, it would not be necessary to use abbreviations. For longer lines, however,

abbreviations can make a significant difference. Using abbreviations will cause us to have to sacrifice the advantages of using "The Automatic Proofreader." The checksums for lines with abbreviations will not match up. If you use the technique described in the Automatic Proofreader article for lines with abbreviations, with program lines of more than three physical lines, you run the risk of losing the ends of the program lines.

Coupon File

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
FE 1  READ 0,K1,K2,K3,K4,K5,K6,K7,K8,K11,K14,K1
      7,K27,K78,K89,K100,K1000,K10000:OPEN #K1,
      K4,0,"K:":GOSUB K10000:GOTO K10000*K2
0G 2  FOR I=0 TO ICOUNT-K1:FOR J=I TO ICOUNT-K1
      :DESBUF$=DES$(VREF(I)*K17+K1):IN$=DES$(VR
      EF(J)*K17+K1)
KI 3  IF IN$<DESBUF$ THEN A=VREF(I):VREF(I)=VRE
      F(J):VREF(J)=A
LH 4  NEXT J:NEXT I:GOTO 4010
OI 6  FOR I=0 TO ICOUNT-K1:FOR J=I TO ICOUNT-K1
      :DBUF$=DAT$(VREF(I)*K6+K1):IN$=DAT$(VREF(
      J)*K6+K1)
BE 7  IF IN$<DBUF$ THEN A=VREF(I):VREF(I)=VREF(
      J):VREF(J)=A
LL 8  NEXT J:NEXT I:GOTO 4010
NF 11  FOR J=0 TO ICOUNT-K1:IF VAL(IN$)=REF(J)
       THEN 10470
AG 12  NEXT J:RETURN
FL 27  POSITION I-K2,K8:? "■■{⊟}■■":POSITION I-
       K2,K6:? "■■{⊟}■■":RETURN
DK 100  POKE K78+K4,K6:GOSUB K10000:? :? :? "1
        ";M1$:? "2  ";M2$:? "3  Sort by Date":
        ? "4  Sort by Description"
AI 110  ? "5  List All Coupons":? "6  List Expi
        red Coupons":? "7  ";M7$:? "8  ";M8$:?
        "9  End Session":POSITION K7+K6,K8+K8
DB 120  ? "PLEASE CHOOSE"
KD 130  GET #K1,A:A=A-K6*K8:IF A<K1 OR A>K8+K1
        THEN 130
EH 140  GOSUB K10000:POKE K78+K4,K2:ON A GOTO K
        1000,K2*K1000,K3*K1000,K4*K1000,K5*K100
        0,K6*K1000,K7*K1000,K8*K1000,9000
BI 200  IN$="":FOR I=ISTART TO K5*K4
BH 201  GOSUB K27:POSITION K11+K7,K1:GET #K1,A:
        ? A-A;"{BACK S}0":IF A=126 THEN 10230
HJ 202  IF A=155 THEN FOR L=I TO K5*K4:IN$(L-IS
        TART+K1)=" ":NEXT L:GOTO 210
```

```
JK 203 IF A=K27 THEN GOTO K100
OA 204 IF RFLAG THEN IF A<46 OR A>57 OR A=47 O
       R (RFLAG=K1 AND A=46) THEN 201
NL 205 POSITION I,K7:? CHR$(A):IN$(I-ISTART+K1
       )=CHR$(A):NEXT I
CA 210 GOSUB 10020:RETURN
KI 220 DBUF$="":FOR I=K11+K2 TO K5*K4:GOSUB K1
       0000+K100:NEXT I:GOSUB 10020:RETURN
GB 400 DESBUF$=DES$(I*K17+K1):IF DESBUF$>=LODE
       S$ AND DESBUF$<=IN$ THEN 5100
JI 410 GOTO 5050
HI 1000 POSITION K14,K2:? M1$:GOSUB 10400:GOSU
        B K100000+K1000:GOSUB K11:A=ICOUNT:REF(
        A)=VAL(IN$):VREF(A)=A:GOSUB 10040
CJ 1050 GOSUB 1300:POSITION K6,K11:? "
        {3 SPACES}Please Enter Description  ":
        GOSUB K100*K2
KJ 1060 GET #K1,A:IF A=K78 THEN 1050
NB 1065 IF A=K27 THEN GOTO K100
ME 1070 IF A<>K89 THEN 1060
FF 1090 DES$(ICOUNT*K17+K1)=IN$:GOSUB 10040:PO
        SITION K2,K6:? B$(K6);BL$(K5*K3):? B$(
        K6);"  $";B$(K14):? B$(K6);BL$(K5*K3)
CB 1100 RFLAG=K2:POSITION K8+K1,K11:? " Please
        Enter Amount{4 SPACES}":ISTART=K17:GO
        SUB K100*K2
EH 1110 GET #K1,A:IF A=K78 THEN POSITION K17,K
        7:? "{4 SPACES}":GOSUB 10040:GOTO K100
        *K11
MN 1115 IF A=K27 THEN GOTO K100
LM 1120 IF A<>K89 THEN 1110
GJ 1140 RFLAG=0:AMT$(ICOUNT*K4+K1)=IN$
FH 1150 GOSUB 10040:GOSUB 10030:POSITION K6,K1
        1:? "Please Enter Expiration Date":GOS
        UB 220
EB 1160 GET #K1,A:IF A=K78 THEN GOSUB 10030:GO
        SUB 10040:GOTO 1150
NC 1165 IF A=K27 THEN GOTO K100
MG 1170 IF A<>K89 THEN 1160
LG 1180 L=ICOUNT:DAT$(L*K6+K1)=DBUF$(K4+K1):DA
        T$(L*K6+K3)=DBUF$(K1,K4):POSITION K7+K
        7,K2:? M1$:GOSUB 10300
CC 1190 GET #K1,A:IF A=K78 THEN GOSUB K100000:G
        OTO K1000
MK 1200 IF A=K89 THEN ICOUNT=ICOUNT+K1:DISKFLA
        G=K1:GOTO K100
MN 1205 IF A=K27 THEN GOTO K100
MI 1210 GOTO 1190
OA 1300 GOSUB 10040:RFLAG=0:POSITION K2,K6:? B
        L$:? BLS$;B$;BLS$:? BL$:ISTART=K4:RETU
        RN
```

```
OM 2000 POSITION K7+K6,K2:? M2$:GOSUB 10400:GO
        SUB K10000+K1000:FOR L=0 TO ICOUNT-K1:
        IF VAL(IN$)<>REF(L) THEN NEXT L:GOTO 1
        00500
OG 2010 GOSUB 10040:GOSUB 10300
MC 2020 GET #K1,A:IF A=K78 THEN GOSUB K10000:G
        OTO K2*K1000
MK 2030 IF A=K27 THEN GOTO K100
LP 2040 IF A<>K89 THEN 2020
JA 2042 POSITION K6+K7,K7:? "* DELETING *":POS
        ITION K11,K17:? B$
FF 2045 IF L=ICOUNT-K1 THEN 2060
NK 2050 DES$(L*K17+K1)=DES$(L*K17+K11+K7):DAT$
        (L*K6+K1)=DAT$(L*K6+K7):AMT$(L*K4+K1)=
        AMT$(L*K4+K5)
HC 2060 FOR J=L TO ICOUNT-K2:REF(J)=REF(J+K1):
        NEXT J
EL 2070 FOR J=0 TO ICOUNT-K1:IF VREF(J)=L THEN
         FOR K=J TO ICOUNT-K2:VREF(K)=VREF(K+K
        1):NEXT K
DL 2080 NEXT J:FOR J=0 TO ICOUNT-K2:IF VREF(J)
        >L THEN VREF(J)=VREF(J)-K1
JP 2090 NEXT J:ICOUNT=ICOUNT-K1:DISKFLAG=K1:GO
        TO K100
PL 3000 GOSUB 3100:GOTO K6
MM 3100 POSITION K3,K8:? "NOTE: Screen is dark
         during Sorts":POSITION K8+K1,K4:? "Pr
        ess Y to begin Sort"
EK 3110 GET #K1,A:IF A=K89 THEN POKE 559,0:RET
        URN
AE 3120 POP :GOTO K100
PI 4000 GOSUB 3100:GOTO K2
KC 4010 DISKFLAG=K1:POKE K11*K7,0:? "{BELL}":P
        OKE 559,K27+K7:GOTO K100
DD 5000 PRFLAG=K5
IG 5010 GOSUB 10050:FOR K=0 TO ICOUNT-K1:I=VRE
        F(K):DBUF$=DAT$(I*K6+K1)
LP 5020 IF PRFLAG=K8 THEN GOTO K100*K4
PP 5030 IF PRFLAG=K5 OR (PRFLAG=K6 AND NOW$>DB
        UF$) OR (PRFLAG=K7 AND (DBUF$>=LODA$ A
        ND DBUF$<=HIDA$)) THEN 5100
AL 5050 NEXT K:GOSUB 10060:GOSUB 10075:GOTO K1
        00
AP 5100 POSITION K2,J:? REF(I):POSITION K7,J:?
         DES$(I*K17+K1,I*K17+K17):POSITION K27
        -K2,J
IM 5110 ? DBUF$(K3,K4);"/";DBUF$(K5);"/";DBUF$
        (K1,K2);" ";AMT$(I*K4+K1,I*K4+K4):J=J+
        K1
```

```
IN 5120 IF J=K27-K6 THEN GOSUB 10070:GOSUB K10
       000:GOSUB 10050
MM 5130 GOTO 5050
FF 6000 PRFLAG=K6:GOSUB 10030:POSITION K8,K11:
       ? "Please Enter Today's Date":GOSUB 22
       0
MJ 6010 GET #K1,A:IF A=K78 THEN GOSUB K10000:G
       OTO K6*K1000
ME 6020 IF A<>K89 THEN 6010
AG 6030 NOW$=DBUF$(K5):NOW$(K3)=DBUF$:GOSUB K1
       0000:GOTO 5010
GP 7000 POSITION K11,K2:? M7$:GOSUB 10030:POSI
       TION K7,K11:? "Please Enter Beginning
       Date":GOSUB 10160:GOSUB 220
DC 7010 GET #K1,A:IF A=K78 THEN GOSUB 10030:GO
       SUB 10040:GOTO K7*K1000
MO 7020 IF A=K27 THEN GOTO K100
MH 7030 IF A<>K89 THEN 7010
DB 7040 LODA$=DBUF$(K5):LODA$(K3)=DBUF$
FH 7050 GOSUB 10030:GOSUB 10040:POSITION 20,K1
       1:? "Ending Date{3 SPACES}":GOSUB 220
EL 7060 GET #K1,A:IF A=K78 THEN GOSUB 10030:GO
       SUB 10040:GOTO 7050
ND 7070 IF A=K27 THEN GOTO K100

NB 7080 IF A<>K89 THEN 7060
AG 7090 HIDA$=DBUF$(K5):HIDA$(K3)=DBUF$:PRFLAG
       =K7:GOSUB K10000:GOTO 5010
AP 8000 POSITION K7,K2:? M8$:GOSUB 1300:POSITI
       ON K2,K11:? "Please Enter Starting Des
       cription":GOSUB K100*K2
NH 8010 GET #K1,A:IF A=K78 THEN GOTO K8*K1000
MP 8020 IF A=K27 THEN GOTO K100
MJ 8030 IF A<>K89 THEN 8010
FP 8040 LODES$=IN$
OD 8050 GOSUB 1300:POSITION K11+K4,K11:? "Endi
       ng Description  ":GOSUB K100*K2
LH 8060 GET #K1,A:IF A=K78 THEN 8050
NE 8070 IF A=K27 THEN GOTO K100
ND 8080 IF A<>K89 THEN 8060
HM 8090 PRFLAG=K8:GOSUB K10000:GOTO 5010
HC 9000 CLOSE #K1:IF DISKFLAG=0 THEN 9020
OF 9005 ? :? :? "Please wait until Disk Drive
       stops":? "before turning the system of
       f.":OPEN #K2,K8,0,"D:DAT":? #K2;ICOUNT
KJ 9010 FOR J=0 TO ICOUNT-K1:I=VREF(J):? #K2;R
       EF(I):? #K2;DES$(I*K17+K1,I*K17+K17):?
        #K2;DAT$(I*K6+K1,I*K6+K6)
GE 9015 ? #K2;AMT$(I*K4+K1,I*K4+K4):NEXT J
```

```
ME 9020 CLOSE #K2:GRAPHICS 0:CLR :? :? :? "It'
       s O.K. to shut down now.":END
HM 10000 GRAPHICS 0:POKE K8+K8,K8*K8:POKE 5377
         4,K8*K8:POKE 709,K7*K4:POKE 710,K11+K
         7:POKE 752,K1:POKE 712,K7+K11:POSITIO
         N K14,K1
ID 10010 ? "COUPON FILE":RETURN
NL 10020 POSITION K6*K4,K7:? COR$:RETURN
KB 10030 POSITION K11,K6:? BL$(K8+K2):POSITION
          K11,K7:? BLS$;D$;BLS$:POSITION K11,K
         8:? BL$(K8+K2):RETURN
AL 10040 POSITION K6*K4,K7:? B$(K4):RETURN
JP 10050 J=K4:? :? "Ref.{=}{3 SPACES}Descripti
         on{3 SPACES}{=}Expires {=}Amt.":RETUR
         N
MD 10060 POSITION K6+K7,J:? "* End of List *":
         RETURN
HA 10070 GOSUB 10160
AG 10075 POSITION K6,K11*K2:? "Press Space Bar
          to Continue"
GH 10080 GET #K1,A:IF A=K8*K4 THEN RETURN
AD 10085 IF A=K27 THEN GOTO K100
CM 10090 GOTO 10080
IE 10100 IF I=K11+K4 OR I=K11+K7 THEN I=I+K1
FF 10105 GOSUB K27:GET #K1,A:IF A=126 THEN 101
         30
PO 10107 IF A=K27 THEN GOTO K100
NP 10110 IF A<K8*K6 OR A>57 THEN 10105
EC 10120 POSITION I,K7:? CHR$(A):DBUF$(LEN(DBU
         F$)+K1)=CHR$(A):RETURN
CA 10130 IF I=K11+K2 THEN I=I-K1:RETURN
BO 10135 IF I=K14 THEN DBUF$="":GOTO 10150
HN 10140 DBUF$=DBUF$(K1,LEN(DBUF$)-K1):IF I=K8
         +K8 OR I=K11+K8 THEN I=I-K1
II 10150 I=I-K1:POSITION I,K7:? " ":GOTO K1000
         0+K100
JE 10160 POSITION K6,K27-K4:? MESC$;:RETURN
JG 10230 IF LEN(IN$)<K2 THEN IN$="":I=ISTART:G
         OTO 10250
OJ 10240 I=I-K1:IN$=IN$(K1,LEN(IN$)-K1)
CH 10250 POSITION I,K7:? " ":GOTO 201
AJ 10300 GOSUB K10000:POKE K78+K4,K7:POSITION
         K7,K6
LP 10305 ? "{Q}{23 R}{E}":? MT$:? "I
         {6 SPACES}Ref. # ";REF(L):POSITION K2
         7+K4,K8:? "I"
PB 10310 ? MT$:? "I";DES$(L*K17+K1,L*K17+K17);
         " $";AMT$(L*K4+K1,L*K4+K4);"I":? MT$:
         A=L*K6+K6
```

```
GG 10320 ? "|{3 SPACES}Expires ";DAT$(A-K3,A-K
         2);"/";DAT$(A-K1,A);"/";DAT$(A-K5,A-K
         4);"{4 SPACES}|":? MT$
NL 10330 ? "{Z}{23 R}{C}":POKE K78+K4,K2:POSIT
         ION K11+K2,K17:? COR$:GOSUB 10160:RET
         URN
IP 10400 POSITION K11+K4,K6:? BL$(K14):POSITIO
         N K11+K4,K7:? BLS$;B$(K14);BLS$:POSIT
         ION K11+K4,K8:? BL$(K14)
PN 10410 POSITION K2,K11:? "{4 SPACES}Please E
         nter Reference Number  ":GOSUB 10160:
         ISTART=K17:RFLAG=K1:GOSUB K100*K2
PA 10420 GET #K1,A:IF A=K78 THEN GOSUB 10040:G
         OTO 10400
PO 10440 IF A=K27 THEN GOTO K100
CG 10450 IF A<>K89 THEN 10420
NL 10460 RETURN
CO 10470 GOSUB 10040:POSITION K2,K11:? "THIS R
         EFERENCE NUMBER HAS BEEN USED":FOR J=
         0 TO K10000/K27:NEXT J:POP :GOTO K100
         0
LH 10500 GOSUB 10040:POSITION K6,K11:? "THIS N
         UMBER IS NOT IN THE FILE":FOR I=0 TO
         K10000/K27:NEXT I:GOTO K2*K1000
IP 11000 POSITION K6*K4,K7:? "Please Wait...":
         RETURN
DL 20000 DIM DESBUF$(K17),LODES$(K17),MT$(K5*K
         5),IN$(K17):C=PEEK(K100+K6)*K5*K5/K4-
         K4*K100
GC 20010 DIM REF(C-K1),VREF(C-K1),DES$(C*K17),
         DAT$(C*K6),AMT$(C*K4),BL$(K17+K4),BLS
         $(K2),D$(K8),COR$(K14),DBUF$(K6)
IL 20020 DIM B$(K17),NOW$(K6),MESC$(K27),M1$(K
         5+K5),M2$(K6+K7),M7$(K17+K1),M8$(K5*K
         5),LODA$(K6),HIDA$(K6),AMTBUF$(K4)
HM 20030 MT$="|{23 SPACES}|":MESC$="Press ESC t
         o Return to Menu":M1$="Add Coupon":M2
         $="Delete Coupon"
IK 20050 M7$="List by Date Range":M8$="List by
          Description Range":D$="  /  /  ":BL$
         ="{21 SPACES}":BLS$=BL$
MN 20060 COR$="CORRECT? (Y/N)":B$="{17 SPACES}"
         :TRAP 21000:OPEN #K2,K4,0,"D:DAT":TRA
         P K10000*K4
EG 20070 INPUT #K2;ICOUNT:FOR I=0 TO ICOUNT-K1
         :INPUT #K2;A,DESBUF$,DBUF$,AMTBUF$:VR
         EF(I)=I:REF(I)=A:IF A>HIREF THEN HIRE
         F=A
```

```
OD 20080 DES$(I*K17+K1)=DESBUF$:DAT$(I*K6+K1)=
         DBUF$:AMT$(I*K4+K1)=AMTBUF$:NEXT I
IJ 20090 CLOSE #K2:POSITION K6,K8:? "The highe
         st Reference Number":POSITION K11+K2,
         K8+K1:? "used is ";HIREF;".":POSITION
          K6,K6
FB 20100 ? "There are ";ICOUNT;" Coupons filed
         .":GOSUB 10075:GOTO K100
BN 21000 CLOSE #K2:IF PEEK(195)<>170 THEN RUN
IA 21010 OPEN #K2,K8,0,"D:DAT":? #K2;K1:? #K2;
         K1:? #K2;"SEVENTEEN LETTERS":? #K2;"9
         91231":? #K2;"9.99":CLOSE #K2:RUN
II 30000 DATA 0,1,2,3,4,5,6,7,8,11,14,17,27,78
         ,89,100,1000,10000
```

Chapter 4

Investment Tracker

John L. Nuss

If you invest in the stock market, this program will help you follow the progress of your portfolio. It also demonstrates some techniques for using Atari's "Return Key Mode."

"Investment Tracker" was designed to help investors follow the stock market and to quickly determine the overall value of a portfolio. Many investors sit down with their Sunday papers and review the performance of their stock portfolio, and I wrote this program to facilitate that process.

After current stock prices are entered, it will compute the market value of each holding, the gain or loss on each investment, and the dividend yield if applicable. That information is summarized for the entire portfolio, and provision is made to review the details of each holding as well as to consolidate multiple holdings of a given stock. Then all that has to be done is to sit back while the computer calculates and displays the results.

By using Atari's well-documented dynamic keyboard feature, I've made it easy for users to update their portfolios and enter the current prices. It is possible to get information directly from the screen without input prompts for every data field. A screen is displayed with columns for each field and a row for each holding or stock. Information already in the DATA statement files is appropriately displayed, and the user is free to edit as needed (for instance, to update current stock prices).

New entries are added by positioning the cursor at the next blank line on the screen and typing in the appropriate information in each column field. Entries may also be deleted, and a routine is available to keep the files sorted in alphabetical order. Suitable menus and prompts are included to make the procedures self-explanatory.

Two DATA Files

Two separate DATA statement files are used. The first stores the portfolio data and has fields for the stock name, the purchase price, and the acquisition data of each holding. The second contains the latest price and dividend information and the date the information was current.

It might seem unnecessarily complicated to have two files, since the current dividend and price for each stock could just as easily have been included in the portfolio file. However, using separate files makes it easier to handle situations in which the portfolio contains several separate holdings of a single stock. That way, the current price needs to be entered only once for each stock in order for the program to have data to calculate the value of each individual holding.

Subroutines have been included for dollar and cents formatting and to convert fractional stock prices to decimal values with which the computer can work. The program will handle up to 98 holdings of as many as 98 individual stocks.

Using the Program

While program operation should be self-explanatory, some detailed explanations may still be helpful.

After loading and running the program you will be asked to type in the current date. The program expects six digits, so preface single-digit months and dates with zero. The date you enter should correspond to the date of the price quotations you will be using to update the current price and dividend records. If you make a mistake while typing in the date, just type any nonnumerical key to start over.

After entering the date, you will be presented with the main program menu. If this is your initial run, you will first want to enter your portfolio data. Type 1 to go to the REVISE PORTFOLIO routine.

The portfolio holdings screen and its command menu will be drawn on the screen. You should type an A to begin entering your holdings. The cursor will go to the first open line on the screen, where you can enter the name of the stock, the number of shares that make up the holding, the purchase price per share, and the acquisition date. Use the TAB key to move from one column to the next. You may ignore the RE or Reference Number field at this point. It will be filled automatically and is used by the program if a record must be edited or deleted.

You have 13 spaces for the name of the stock. If this is not room enough to type the full names of some companies, use the same abbreviations that appear in the newspaper financial pages. You may shift to lowercase letters where

needed. An alternative would be to use the trading symbol abbreviation for the stock.

Type the number of shares making up the holding in the SHRS field. The purchase price may be entered as a fraction (for example, 35 1/8) or as a decimal value, whichever way the stock or mutual fund is normally quoted. There is not much space left for the acquisition date, so you must enter it as six unseparated digits.

The tab stops have been programmed so you can tab from one field to the next without having to resort to the cursor control keys. If you should, however, happen to space over and consequently erase one of the vertical lines separating the fields, don't worry. They're there only for cosmetic purposes.

If you make a mistake, use the cursor keys to correct it. Then, when you're satisfied with your entry, hit the RETURN key and the program will create a DATA statement containing the information you have just entered. The lower six lines of the screen will flash as the Return Key Mode is utilized to read the DATA statement into the program. The upper portion of the screen containing your portfolio information will be unaffected except for the insertion of a reference number.

Continue in this manner until all of your stock holdings have been entered. Remember, if you have more than one holding of a stock, a record must be entered for each purchase so that separate gains or losses can be calculated.

Sorting Your Stocks

A simple bubble sort routine, which puts the portfolio in alphabetical order, can be accessed from the main menu. The sort routine will blank the screen to cut processing time; then, when the menu returns, the sort is complete. The stocks will show up sorted as you requested the next time you look at your portfolio. Actually, this routine is rather slow, so I recommend that you attempt to enter your initial portfolio information in alphabetical order and rely on the sort routine only to put new holdings in order when you add them later on.

Should you find that you have made any mistakes, they can be fixed by typing E to access the editing function. That allows you to move the cursor to the offending field, correct it, and rewrite the DATA statement with a press of the RETURN key. You can also delete a holding (after a sale, for instance)

by typing D. The program will ask for the reference number of the holding to be deleted. All you need do is type in the number and hit RETURN. Finally, when your portfolio information is up-to-date, type R to return to the main menu.

Current Prices

The next step is to enter or update the current prices. Type 2 to call up this routine. This price update routine works exactly as the portfolio revision does, except that you are supplying information for each stock and not each individual holding. Type A to enter new stocks to the file.

The fields to be filled for each stock record are the stock name, which must be entered in the identical manner which it appears in the portfolio file; the current price; the current annual dividend; and the date this information was obtained. The date field for any record you add or update will be automatically updated to the date you entered when the run commenced.

Enter the stock name, price, and dividend just as you did the portfolio holding information. You will have to enter data for each unique stock you own, but you do not have to enter the data more than once if you hold more than one block of a particular stock. Corrections and/or price and dividend changes are made by typing U for the update routine. This routine functions almost like the edit routine, except that after each change is recorded (by hitting RETURN) the cursor will move to the next stock to permit you to continue to update prices. Respond with a Y or N when the program asks if you have more to update.

The D and R routines function as previously described. Both the portfolio file and the price file update routines will allow you to continue entering data on a new screen, should you be unable to find enough room on the initial one.

Evaluation

Once all of the relevant data has been loaded into the program, you may proceed with computing and summarizing the value and gain of your stock holdings. You have two routines to choose from. Return to the main menu and type 3 for a summary of the entire portfolio status. The program will display each holding on the screen, along with its current value, gain (or loss), and the total annual dividend. The overall totals for the portfolio will be accumulated at the bottom of the

screen. If your portfolio won't fit on the screen, you can review the first portion of it and then continue the listing on a new screen. The totals at the bottom will include only the holdings already listed, so you must list all of the portfolio to see the grand total.

Your other option is selection number 4, which will allow you to review all of your holdings of a specific stock in more detail. If you type 4, a list of your stocks will appear on the screen, and you will be prompted to type S to select a stock. Any other response will return you to the main menu.

Having entered an S, you will be prompted to type in the name of the stock you want to review. Do so and hit RETURN. Be sure to type in the name exactly as it was entered in your data field, being careful to use lowercase characters if applicable. (Actually, you needn't enter the entire name, just the first unique character string. If the only stock you own that begins with an A is Allied Corporation, then an A plus RETURN will suffice. If you also own ATT then you must type in at least Al to look at the Allied.)

The program will then compile all of the information it has for that stock and display it on the screen. The particulars on each holding will appear, with room for up to three holdings on the screen at once. If you have more than three blocks of a stock, you will be told that there is more to see. You can continue to review the holdings of that stock three blocks at a time.

The information displayed will include the name of the stock, its current price and dividend, and (for each holding) the number of shares, their purchase price, the current value, gain or loss, and dividend. At the bottom of the screen will be total value, dividend, and gain or loss for all of the holdings listed so far, plus the dividend yield for the stock.

Your other main menu options are the file sorting routines, a routine to save the program along with the latest data, and a routine to delete all of the data should you want to start over or begin another file for a separate portfolio.

This is a lengthy program, but it is well worth the effort to type it in. I'm sure you'll find it useful if you're a stock market investor and haven't yet purchased a more sophisticated commercial stock-tracking program.

Saving the File

The program uses DATA statements to store the data. When you select the SAVE option, the program will save out the whole program to tape. If you prefer to save to a disk, change the following few lines:

```
6000 REM SAVE DATA ON DISK
6060 REM
6070 REM
6080 REM
6100 SAVE "D:TRACKER":END
6110 PRINT "CANNOT SAVE DATA": STOP
6120 REM
```

The program will be saved with the filename TRACKER.

Save a backup copy of the program just in case the worst happens.

Investment Tracker

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
ND 50 REM *** Stock Portfolio Tracker ***
NK 110 GOSUB 8000
IN 120 GOTO 1000
PN 200 REM -CLEAR SCREEN & WRITE TITLE
KN 210 ? "{CLEAR}":SETCOLOR 2,11,0:SETCOLOR 4,
      11,0:POKE 752,1:TRAP 1000
KB 220 POSITION 5,0:? TITLE$;CDATE$(1,2);SL$;C
      DATE$(3,4);SL$;CDATE$(5,6)
HL 290 RETURN
NL 300 REM -FORCED READ PART 1
EE 310 POSITION 0,19:? CBS$
OK 320 POSITION 0,19:RETURN
OB 350 REM -FORCED READ PART 2
FA 360 ? "CONT"
NE 370 POSITION 0,18
HI 380 POKE 842,13:STOP
GA 390 POKE 842,12:GOSUB 310:RETURN
PP 400 REM -INTERPRET PRICE STRINGS
AC 410 REM -CONVERT FRACTIONS TO DECIMAL
BA 420 FOR I=1 TO LEN(PRC$)
AG 430 IF PRC$(I,I)=SL$ THEN 450
CP 440 NEXT I:GOTO 480
PG 450 PRC=VAL(PRC$(I-1,I-1))/VAL(PRC$(I+1,I+1
      ))
BF 460 IF I>=3 THEN PRC=PRC+VAL(PRC$(1,I-3))
HB 470 GOTO 490
PL 480 PRC=VAL(PRC$)
```

```
HG 490  I=LEN(PRC$):RETURN
HN 500  REM -DOLLAR AND CENTS FORMAT
DN 510  IF ABS(AMT)>=10^PWR THEN AMT$=STR$(AMT)
        :RETURN
CJ 520  AMT$="":AMT$(1,PWR+3)=BL$:AMT$(PWR+1,PW
        R+1)=".":B=PWR+1-LEN(STR$(INT(AMT)))+(A
        MT<0)
HN 530  IF AMT<0 THEN AMT=-AMT:AMT$(B-1,B-1)="-
        "
CA 540  AMT$(B,PWR)=STR$(INT(AMT))
HJ 550  AMT$(PWR+1,PWR+3)=STR$(100+INT((AMT-INT
        (AMT))*100+0.5)):AMT$(PWR+1,PWR+1)="."
CN 560  IF VAL(AMT$)<0 THEN AMT=-AMT
HM 570  RETURN
FC 600  REM -DATA FOR DATE, NUMBER OF HOLDINGS
        AND STOCKS
IJ 610  DATA 000000,0,0
BK 700  REM -HOLDINGS DATA
II 799  DATA [E]
DG 800  REM -PRICE DATA
IJ 899  DATA [E]
IB 900  REM -CLEAR TAB STOPS
JK 910  ? "{CLEAR}"
OL 940  FOR I=1 TO 6:? "{TAB}{CLR TAB}";:NEXT I
HO 950  RETURN
CM 1000 REM -MAIN MENU
NE 1010 GOSUB 200
BN 1020 POSITION 3,3:? "PLEASE SELECT ONE:"
HI 1030 POSITION 6,6:? "1 - REVISE PORTFOLIO"
HJ 1040 POSITION 6,8:? "2 - UPDATE PRICES"
BD 1050 POSITION 6,10:? "3 - SUMMARIZE PORTFOL
        IO VALUE"
LB 1060 POSITION 6,12:? "4 - REVIEW INDIVIDUAL
         STOCKS"
IF 1070 POSITION 6,14:? "5 - SORT DATA"
FL 1080 POSITION 6,16:? "6 - SAVE PROGRAM AND
        DATA"
LF 1090 POSITION 6,18:? "7 - ERASE DATA"
HH 1120 GET #2,R
NN 1130 IF R<49 OR R>55 THEN 1120
OC 1140 ON VAL(CHR$(R)) GOTO 5000,2000,3000,40
        00,7400,6000,7000
HL 1300 REM -ELIMINATE COMMAS FROM DATA
PB 1310 FOR I=1 TO LEN(IN$)
PC 1320 IF IN$(I,I)="," THEN IN$(I,I)="[■]"
EP 1330 NEXT I
KI 1340 RETURN
IM 1390 POP :GOTO RN+200
DE 1400 REM -ADD TO PORTFOLIO  OR PRICE FILE
```

```
MI 1410 IF CNT=13 THEN CNT=0:SCR=SCR+1:POP :GO
        TO RN+20
MN 1415 IF CNT=13 THEN POP :GOTO RN+100
HI 1420 POSITION 1,19:? CBS$
HI 1430 POSITION 3,18:? "ENTER THE APPROPRIATE
         DATA IN EACH{3 SPACES}COLUMN AND PRES
        S RETURN "
LI 1440 POKE 752,0:POSITION 3,CNT+4:? "|";
MO 1450 TRAP 1390
HJ 1460 INPUT #1,IN$:POKE 752,1:GOSUB 1310
KM 1470 RETURN
NI 1490 CNT=CNT+1:ITM=ITM+1
EB 1500 POSITION 1+(ITM<10),CNT+3:? ITM
FN 1510 GOTO RN+200
AG 2000 REM -PRICES AND DIVIDENDS
CM 2010 SCR=1:CNT=0:ITM=0:RESTORE 800
BL 2015 GOSUB 910:? "{4 SPACES}{SET TAB}
        {14 SPACES}{SET TAB}{8 SPACES}{SET TAB}
        {6 SPACES}{SET TAB}"
NG 2020 GOSUB 200
PP 2030 POSITION 0,1:? OL$
CA 2040 POSITION 0,2:? "{V}RE|{4 SPACES}STOCK
        {4 SPACES}|C.PRICE|DIVND| DATE {B}"
DN 2050 POSITION 0,3:? "{V}{2 M}|{13 M}|{7 M}|
        {5 M}|{6 M}{B}"
CA 2060 FOR I=1 TO 13:POSITION 0,I+3:? PR$:NEX
        T I
EA 2070 POSITION 0,17:? UL$
ML 2100 IF CNT=13 THEN 2200
AO 2110 READ STK$:IF STK$="E" THEN 2200
MP 2120 CNT=CNT+1:ITM=ITM+1
FK 2130 READ CP$,DV$,CD$
EC 2140 POSITION 1+(ITM<10),CNT+3:? ITM
GP 2150 POSITION 4,CNT+3:? STK$
EG 2160 POSITION 18,CNT+3:? CP$
EN 2170 POSITION 26,CNT+3:? DV$
DI 2180 POSITION 32,CNT+3:? CD$
MI 2190 GOTO 2100
KP 2200 POSITION 1,18:? CBS$;
JG 2210 ? "{3 SPACES}ENTER AN INSTRUCTION TO P
        ROCEED:"
CP 2220 ? "{6 SPACES}A - ADD A NEW STOCK"
DD 2230 ? "{6 SPACES}D - DELETE A STOCK"
NG 2240 ? "{6 SPACES}U - UPDATE INFORMATION"
KI 2250 ? "{6 SPACES}R - RETURN TO MAIN MENU"
EM 2260 IF CNT=13 AND NUMS>13*(SCR) THEN POSIT
        ION 6,23:? "C - CONTINUE LISTING";
IA 2290 GET #2,R
OJ 2300 IF R=65 OR R=97 THEN 2370
AM 2310 IF R=68 OR R=100 THEN 2710
```

```
BD 2320 IF R=85 OR R=117 THEN 2520
AH 2330 IF R=82 OR R=114 THEN 1010
PN 2340 IF R=67 OR R=99 THEN RN=2000:GOTO 1410
NA 2350 GOTO 2290
OL 2370 RN=2000:GOSUB 1410
ID 2470 STK$=IN$(1,13):CP$=IN$(15,21):DV$=IN$(
        23,27):CD$=IN$(29,34)
OF 2480 GOSUB 310:? "{DOWN}";800+ITM+1;D$;STK$
        ;CM$;CP$;CM$;DV$;CM$;CD$:GOSUB 360
CP 2490 NUMS=NUMS+1:GOTO 1490
HJ 2520 POSITION 1,18:? CBS$
JJ 2530 POSITION 3,19:? "MOVE CURSOR TO LINE T
        O BE UPDATED,{3 SPACES}MAKE CHANGES AN
        D PRESS [RETURN]"
JO 2540 POKE 752,0:POSITION 0,18:? " ";:TRAP 1
        390
BG 2550 INPUT #1,IN$:GOSUB 1310
OO 2580 RE=VAL(IN$(2,3)):STK$=IN$(5,17):CP$=IN
        $(19,25):DV$=IN$(27,31):CD$=CDATE$
DI 2590 GOSUB 310:? "{DOWN}";800+RE;D$;STK$;CM
        $;CP$;CM$;DV$;CM$;CD$:GOSUB 360
FL 2595 POSITION 32,RE+3-13*(SCR-1):? CDATE$;
MO 2600 POSITION 1,22:? "MORE TO UPDATE?";
HO 2630 GET #2,R
KL 2640 IF R=78 OR R=110 THEN POKE 752,1:GOTO
        2200
MG 2650 IF R<>89 AND R<>121 THEN 2630
HM 2655 IF RE/13=INT(RE/13) THEN RN=2000:GOSUB
         1410
FN 2660 POSITION 1,22:? "{DEL LINE}"
BE 2670 POSITION 0,RE+4-13*(SCR-1):? "{V}
        {LEFT}";
NF 2680 GOTO 2550
DN 2700 REM -DELETE STOCK INFO
HK 2710 POSITION 1,18:? CBS$
LK 2720 POSITION 3,19:? "ENTER REFERENCE NUMBE
        R OF INFO TO BE DELETED, PRESS [RETURN
        ]":POKE 752,0
EL 2730 INPUT RE:POKE 752,1:RESTORE 800+RE+1:I
        F RE=NUMS THEN 2780
BF 2740 FOR I=RE TO NUMS-1
KE 2750 READ STK$,CP$,DV$,CD$
OJ 2760 GOSUB 310:? "{DOWN}";800+I;D$;STK$;CM$
        ;CP$;CM$;DV$;CM$;CD$:GOSUB 360
FI 2770 NEXT I
FD 2780 GOSUB 310:? "{DOWN}";800+NUMS:GOSUB 36
        0
CJ 2790 NUMS=NUMS-1:GOTO 2010
MA 3000 REM -PORTFOLIO SUMMARY
AE 3010 SCR=0:ITM=0:CNT=0:TV=0:NG=0:DV=0
```

```
NH 3020 GOSUB 200
AB 3030 POSITION 1,1:? OL$
JA 3040 POSITION 1,2:? "{V}{3 SPACES}STOCK
        {3 SPACES}|  VALUE |NET GAIN| DIVND
        {B}"
NA 3050 POSITION 1,3:? "{V}{11 M}|{8 M}|{8 M}|
        {7 M}{B}"
NA 3060 FOR I=4 TO 19:POSITION 1,I:? SR$:NEXT
        I
AB 3070 POSITION 1,20:? "{V}{11 M}|{8 M}|{8 M}
        |{7 M}{B}"
EC 3080 POSITION 1,21:? SR$
IE 3090 POSITION 8,21:? "TOTAL";
HD 3100 POSITION 1,22:? UL$;
HK 3110 ITM=ITM+1:RESTORE 700+ITM:TRAP 1390
BH 3120 READ STK$:IF STK$="E" THEN 3800
GH 3130 READ SH$,PP$,AD$
IE 3140 RESTORE 800
PN 3150 READ SEL$:IF SEL$="E" THEN 3700
GC 3155 READ CP$,DV$,CD$
OJ 3160 IF SEL$<>STK$ THEN 3150
IB 3170 POSITION 2,CNT+4:? STK$(1,11)
GG 3180 PRC$=CP$:GOSUB 400:AMT=VAL(SH$)*PRC:PW
        R=5:GOSUB 500
JG 3190 POSITION 14,CNT+4:? AMT$
GD 3200 TV=TV+AMT
NH 3210 PRC$=PP$:GOSUB 400:AMT=AMT-VAL(SH$)*PR
        C:GOSUB 500
JA 3220 POSITION 23,CNT+4:? AMT$
DM 3230 NG=NG+AMT
OK 3240 AMT=VAL(DV$)*VAL(SH$):PWR=4:GOSUB 500
JD 3250 POSITION 32,CNT+4:? AMT$
EJ 3260 DV=DV+AMT
HL 3270 CNT=CNT+1:IF CNT=16 THEN 3800
ML 3280 GOTO 3110
PK 3700 POSITION 1,23:? "NO INFO FOR ";STK$;
GB 3710 FOR I=1 TO 300:NEXT I
MK 3720 GOTO 3110
II 3800 AMT=TV:PWR=5:GOSUB 500
LE 3810 POSITION 14,21:? AMT$
NA 3820 AMT=NG:GOSUB 500
LG 3830 POSITION 23,21:? AMT$
HL 3840 AMT=DV:PWR=4:GOSUB 500
LI 3850 POSITION 32,21:? AMT$
PN 3860 IF ITM>=NUMH THEN 3950
KN 3870 POSITION 1,23:? "MORE STOCKS ... PRESS
        C TO CONTINUE ";
IN 3900 GET #2,R:IF R<>67 AND R<>99 THEN 3900
HD 3910 IF ITM<NUMH THEN CNT=0:GOTO 3020
MI 3920 GOTO 1000
```

```
PI 3950 POSITION 1,23:? "{8 SPACES}PRESS C TO
        CONTINUE";
NH 3960 GOTO 3900
GH 4000 REM -REVIEW INDIVIDUAL STOCKS
KD 4010 SCR=1:ITM=1:RESTORE 800
NI 4020 GOSUB 200
JD 4030 POSITION 14,2:? "STOCKS"
NO 4040 READ STK$:IF STK$="E" THEN ITM=0:GOTO
        4110
NN 4050 IF ITM>17 THEN 4110
FO 4060 READ CP$,DV$,CD$
DI 4070 ITM=ITM+1
BF 4080 POSITION 10,2+ITM
CC 4090 ? STK$
MG 4100 GOTO 4040
IN 4110 POSITION 1,21:? "{3 DEL LINE}"
KD 4120 ? "{4 SPACES}ENTER S TO SELECT A STOC
        K"
OB 4130 IF ITM<18 THEN 4170
AC 4140 POSITION 1,23:? "  ENTER C TO LIST MO
        RE STOCKS";
HP 4170 GET #2,R
BD 4180 IF R=83 OR R=115 THEN 4210
AL 4190 IF R=67 OR R=99 THEN ITM=1:GOSUB 200:G
        OTO 4060
MA 4200 GOTO 1000
IN 4210 POSITION 1,21:? "{3 DEL LINE}"
OO 4220 POSITION 1,21:? "WHICH STOCK?";:POKE
         752,0
OB 4230 INPUT #1,SEL$
IA 4240 I=0
IF 4250 I=I+1:IF I>NUMS THEN 4900
PM 4260 RESTORE 800+I
EA 4270 READ STK$:IF STK$(1,LEN(SEL$))<>SEL$ T
        HEN 4250
GC 4280 READ CP$,DV$,CD$
OB 4290 GOSUB 200
FL 4300 TV=0:NG=0:DV=0
KE 4350 POSITION 9,2:? "STOCK: ";STK$
FD 4360 POSITION 2,4:? "CURRENT PRICE: ";CP$
EF 4365 POSITION 25,4:? "DIVIDEND: ";DV$
AO 4370 POSITION 1,6:? OL$
EK 4380 POSITION 1,7:? "{V}  DATA{4 SPACES}| H
        OLDING| HOLDING| HOLDING{B}"
IE 4390 POSITION 1,8:? "{V}{10 SPACES}|  NO.
        {3 SPACES}|  NO.{3 SPACES}|  NO.
        {3 SPACES}{B}"
NG 4400 POSITION 1,9:? "{V}{10 M}|{8 M}|{8 M}|
        {8 M}{B}"
OL 4410 FOR I=10 TO 16:POSITION 1,I:? DR$:NEXT
         I
```

```
IB 4420 POSITION 2,10:? "SHARES"
NO 4430 POSITION 2,11:? "ACQ. DATE"
FH 4440 POSITION 2,12:? "PUR. PRICE"
FG 4450 POSITION 2,13:? "CUR. VALUE"
FC 4460 POSITION 2,14:? "GAIN/LOSS"
AM 4470 POSITION 2,15:? "DIVIDEND"
EI 4480 POSITION 1,17:? UL$
IG 4490 POSITION 1,19:? "TOTAL VALUE:
        {10 SPACES}DIVIDEND:"
EB 4500 POSITION 1,21:? "NET GAIN/LOSS:
        {11 SPACES}YIELD:{6 SPACES}%"
DM 4510 ITM=0:CNT=0:SCR=0
EC 4520 ITM=ITM+1:RESTORE 700+ITM
CD 4530 READ STK$:IF STK$="[E]" THEN 4850
NE 4540 IF STK$(1,LEN(SEL$))<>SEL$ THEN 4520
GO 4550 READ SH$,PP$,AD$
EM 4560 CNT=CNT+1:IF CNT=4 THEN 4800
EP 4565 POSITION 10+CNT*9,8:? CNT+SCR*3
LE 4570 POSITION 7+CNT*9,10:? SH$
AN 4580 POSITION 4+CNT*9,11:? AD$(1,2);SL$;AD$
        (3,4);SL$;AD$(5,6)
LL 4590 POSITION 5+CNT*9,12:? PP$
GH 4600 PRC$=CP$:GOSUB 420:AMT=PRC*VAL(SH$):PW
        R=5:GOSUB 510
PG 4610 POSITION 4+CNT*9,13:? AMT$
DP 4620 TV=TV+AMT:YD=100*VAL(DV$)/PRC
OA 4630 PRC$=PP$:GOSUB 410:AMT=AMT-VAL(SH$)*PR
        C:GOSUB 510
PK 4640 POSITION 4+CNT*9,14:? AMT$
ED 4650 NG=NG+AMT
EO 4660 AMT=VAL(SH$)*VAL(DV$):GOSUB 510
PO 4670 POSITION 4+CNT*9,15:? AMT$
FA 4680 DV=DV+AMT
OM 4690 AMT=TV:GOSUB 510
LK 4700 POSITION 14,19:? AMT$
NA 4710 AMT=NG:GOSUB 510
LH 4720 POSITION 16,21:? AMT$
HL 4730 AMT=DV:PWR=4:GOSUB 510
LO 4740 POSITION 32,19:? AMT$
HO 4750 AMT=YD:PWR=2:GOSUB 510
LK 4760 POSITION 33,21:? AMT$
NG 4770 GOTO 4520
KB 4800 SCR=SCR+1:CNT=0:LN=4560
KA 4810 POSITION 1,23:? "MORE HOLDINGS, TYPE
        C TO CONTINUE";
NH 4820 GOTO 4930
OB 4850 POSITION 1,23:? "NO MORE HOLDINGS, TY
        PE C TO CONTINUE";
KN 4860 LN=1000:GOTO 4930
```

```
PI 4900 POSITION 1,23:? "■STOCK NOT FOUND, TYP
        E C TO CONTINUE■";:LN=1000
ID 4930 GET #2,R
AA 4940 IF R=67 OR R=99 THEN GOTO LN
NL 4950 GOTO 4930
GC 5000 REM -REVISE PORTFOLIO
CD 5010 SCR=1:CNT=0:ITM=0:RESTORE 700
BO 5015 GOSUB 910:? "{4 SPACES}{SET TAB}
        {14 SPACES}{SET TAB}{6 SPACES}{SET TAB}
        {8 SPACES}{SET TAB}"
NJ 5020 GOSUB 200
AC 5030 POSITION 0,1:? OL$
GK 5040 POSITION 0,2:? "{V}RE|{4 SPACES}STOCK
        {4 SPACES}| SHRS|P.PRICE|A.DATE{B}"
EA 5050 POSITION 0,3:? "{V}{2 M}|{13 M}|{5 M}|
        {7 M}|{6 M}{B}"
BL 5060 FOR I=1 TO 13:POSITION 0,I+3:? HR$:NEX
        T I
ED 5070 POSITION 0,17:? UL$
NB 5100 IF CNT=13 THEN 5200
BE 5110 READ STK$:IF STK$="■" THEN 5200
NC 5120 CNT=CNT+1:ITM=ITM+1
GJ 5130 READ SH$,PP$,AD$
EF 5140 POSITION 1+(ITM<10),CNT+3:? ITM
HC 5150 POSITION 4,CNT+3:? STK$
FB 5160 POSITION 18,CNT+3:? SH$
FE 5170 POSITION 24,CNT+3:? PP$
DJ 5180 POSITION 32,CNT+3:? AD$
MO 5190 GOTO 5100
HG 5200 POSITION 0,18:? CBS$
BB 5210 POSITION 4,18:? "ENTER AN INSTRUCTION
        TO PROCEED:"
OE 5220 ? "{5 SPACES}A - ADD TO PORTFOLIO"
GD 5230 ? "{5 SPACES}D - DELETE FROM PORTFOLIO
        "
KE 5240 ? "{5 SPACES}E - EDIT PORTFOLIO"
KL 5250 ? "{5 SPACES}R - RETURN TO MAIN MENU"
ED 5260 IF CNT=13 AND NUMH>13*(SCR) THEN POSIT
        ION 5,23:? "C - CONTINUE LISTING";
ID 5290 GET #2,R
OP 5300 IF R=65 OR R=97 THEN 5370
BB 5310 IF R=68 OR R=100 THEN 5610
BF 5320 IF R=69 OR R=101 THEN 5530
AK 5330 IF R=82 OR R=114 THEN 1010
AD 5340 IF R=67 OR R=99 THEN RN=5000:GOTO 1410
NG 5350 GOTO 5290
PB 5370 RN=5000:GOSUB 1410
JH 5470 STK$=IN$(1,13):SH$=IN$(15,19):PP$=IN$(
        21,27):AD$=IN$(29,34)
```

```
PD 5480 GOSUB 310:? "{DOWN}";700+ITM+1;D$;STK$
        ;CM$;SH$;CM$;PP$;CM$;AD$:GOSUB 360
BM 5490 NUMH=NUMH+1:GOTO 1490
HN 5530 POSITION 1,18:? CBS$
EF 5540 POSITION 3,19:? "MOVE CURSOR TO LINE T
        O BE EDITED,{4 SPACES}MAKE CHANGES AND
         PRESS RETURN"
KA 5560 POKE 752,0:POSITION 3,18:? " ";:TRAP 5
        200
HP 5570 INPUT #1,IN$:POKE 752,1:GOSUB 1310
HN 5580 RE=VAL(IN$(2,3)):STK$=IN$(5,17):SH$=IN
        $(19,23):PP$=IN$(25,31):AD$=IN$(33,38)
IA 5590 GOSUB 310:? "{DOWN}";700+RE;D$;STK$;CM
        $;SH$;CM$;PP$;CM$;AD$:GOSUB 360:GOTO 5
        200
IB 5600 REM -DELETE FROM PORTFOLIO
HM 5610 POSITION 1,18:? CBS$
JF 5620 POSITION 3,19:? "ENTER REFERENCE NUMBE
        R OF HOLDING TO BE DELETED, PRESS RET
        URN":POKE 752,0
ED 5630 INPUT RE:POKE 752,1:RESTORE 700+RE+1:I
        F RE=NUMH THEN 5680
AM 5640 FOR I=RE TO NUMH-1
LC 5650 READ STK$,SH$,PP$,AD$
PG 5660 GOSUB 310:? "{DOWN}";700+I;D$;STK$;CM$
        ;SH$;CM$;PP$;CM$;AD$:GOSUB 360
FK 5670 NEXT I
EJ 5680 GOSUB 310:? "{DOWN}";700+NUMH:GOSUB 36
        0
BI 5690 NUMH=NUMH-1:GOTO 5010
OH 6000 REM -SAVE DATA ON TAPE
NJ 6010 GOSUB 200
JP 6020 POSITION 14,2:? "SAVING DATA"
NN 6030 GOSUB 310
GO 6040 ? "{DOWN}";610;D$;CDATE$;CM$;NUMH;CM$;
        NUMS
OE 6050 GOSUB 360
GO 6060 POSITION 4,5:? "1 - TURN OVER CASSETTE
         AND REWIND"
MK 6070 POSITION 4,7:? "2 - PRESS PLAY AND
        RECORD"
KG 6080 POSITION 4,9:? "3 - PRESS RETURN"
MO 6090 TRAP 6110
KA 6100 LPRINT
DK 6110 CSAVE
BB 6120 POSITION 4,11:? "4 - REWIND TAPE"
KB 6130 END
DL 7000 REM -ERASE ALL DATA
NK 7010 GOSUB 200
```

```
JN 7020 POSITION 4,8:? "DO YOU REALLY WANT TO
        ERASE ALL OF{13 SPACES}{6 M}{27 SPACES}Y
        OUR DATA?"
HP 7050 GET #2,R
IJ 7060 IF R<>89 THEN 1000
OA 7070 GOSUB 200
KB 7080 POSITION 15,10:? "ERASING DATA..."
EJ 7090 FOR I=1 TO NUMH
EP 7100 GOSUB 310:? "{DOWN}";700+I:GOSUB 360
FB 7110 NEXT I
EO 7120 FOR I=1 TO NUMS
FD 7130 GOSUB 310:? "{DOWN}";800+I:GOSUB 360
FI 7135 NEXT I
FL 7140 NUMS=0:NUMH=0
MA 7150 POSITION 15,10:? "DATA ERASED
        {4 SPACES}"
GE 7160 FOR I=1 TO 300:NEXT I
MJ 7170 GOTO 1000
DO 7400 REM - SORT DATA
NO 7410 GOSUB 200
EI 7420 POSITION 12,2:? "DATA SORT ROUTINE"
PC 7430 POSITION 8,6:? "ENTER 1 TO SORT PORTFO
        LIO"
ML 7440 POSITION 8,8:? "ENTER 2 TO SORT PRICE
        FILE"
ID 7450 GET #2,R
HE 7460 IF R=49 THEN ID=700:ND=NUMH:GOTO 7490
HJ 7470 IF R=50 THEN ID=800:ND=NUMS:GOTO 7490
NM 7480 GOTO 7450
CG 7490 SA=PEEK(559):POKE 559,0
DB 7500 SCHK=0:IN$=BL$:RESTORE ID
BA 7510 FOR I=ID+1 TO ID+ND
MP 7520 IF ID=700 THEN READ STK$,SH$,PP$,AD$
MF 7530 IF ID=800 THEN READ STK$,CP$,DV$,CD$
IC 7540 IF STK$<IN$(1,13) THEN 7590
PB 7550 IN$(1,13)=STK$
FO 7560 IF ID=700 THEN IN$(15,19)=SH$:IN$(21,2
        7)=PP$:IN$(29,34)=AD$
EP 7570 IF ID=800 THEN IN$(15,21)=CP$:IN$(23,2
        7)=DV$:IN$(29,34)=CD$
OB 7580 GOTO 7670
IE 7590 SCHK=I
LL 7600 GOSUB 310:? "{DOWN}";I-1;D$;STK$;CM$;
KM 7610 IF ID=700 THEN ? SH$;CM$;PP$;CM$;AD$
KC 7620 IF ID=800 THEN ? CP$;CM$;DV$;CM$;CD$
BM 7630 GOSUB 360:? "{DOWN}";I;D$;IN$(1,13);CM
        $;
KA 7640 IF ID=700 THEN ? IN$(15,19);CM$;IN$(21
        ,27);
JN 7650 IF ID=800 THEN ? IN$(15,21);CM$;IN$(23
        ,27);
```

```
FO 7660 ? CM$;IN$(29,34):GOSUB 360
FM 7670 NEXT I
PG 7680 IF SCHK>0 THEN 7500
JN 7690 POKE 559,SA:GOTO 1010
EE 8000 REM -DIMENSION AND INITIALIZE
MK 8010 OPEN #1,12,0,"E:":OPEN #2,4,0,"K:":POK
        E 82,0
NE 8020 DIM TITLE$(22),CDATE$(6),SL$(1),CM$(1)
        ,D$(6),CBS$(6)
OO 8030 DIM STK$(13),SH$(5),PP$(7),AD$(6),CP$(
        7),CD$(6),DV$(5)
BL 8040 DIM SEL$(13),IN$(39),AMT$(9),PRC$(7),B
        L$(39)
HM 8050 DIM OL$(39),UL$(39),HR$(39),PR$(39),DR
        $(39),SR$(39)
FM 8110 TITLE$="STOCK PORTFOLIO AS OF "
DO 8120 SL$="/"
CN 8130 CM$=","
NA 8140 D$=" DATA "
DA 8150 OL$="{39 N}"
BA 8160 UL$="{39 M}"
BH 8170 HR$="{V}   |{13 SPACES}|{5 SPACES}|
        {7 SPACES}|{6 SPACES}{B}"
CA 8180 PR$="{V}   |{13 SPACES}|{7 SPACES}|
        {5 SPACES}|{6 SPACES}{B}"
JJ 8190 DR$="{V}{10 SPACES}|{8 SPACES}|
        {8 SPACES}|{8 SPACES}{B}"
KA 8200 SR$="{V}{11 SPACES}|{8 SPACES}|
        {8 SPACES}|{7 SPACES}{B}"
PO 8210 BL$="{39 SPACES}"
LF 8220 SEL$=BL$:STK$=BL$:SH$=BL$:PP$=BL$:AD$=
        BL$:IN$=BL$:CP$=BL$:DV$=BL$:AMT$=BL$:P
        RC$=BL$
PC 8230 CBS$="{6 DEL LINE}"
AB 8900 REM -READ KEY DATA
EG 8910 READ CDATE$,NUMH,NUMS
HG 9000 REM -ENTER DATE
NM 9010 GOSUB 200
PO 9020 POSITION 2,3:? "ENTER CURRENT DATE (MM
        /DD/YY)."
OB 9030 POSITION 22,5:? "■■/■■/■■"
EF 9050 FOR I=1 TO 6
ID 9070 GET #2,R
GB 9080 IF R<48 OR R>57 THEN CDATE$="
        {6 SPACES}":I=6:GOTO 9030
OL 9090 POSITION 21+I+(I>2)+(I>4),5
KN 9100 ? CHR$(R)
EA 9110 CDATE$(I,I)=CHR$(R)
FE 9120 NEXT I
KO 9140 RETURN
```

Chapter 4

Horizon: A Celestial Coordinates Calculator

Russell A. Grokett, Jr.

Among your Atari's many talents is the ability to precisely locate the planets and stars. With this program, astronomy and photography buffs will be able to pinpoint celestial bodies with remarkable accuracy.

Remember when you got that telescope for Christmas, and how you ran out to set up your new equipment, only to discover how hard it was to find anything more difficult than the moon or a few stars?

Now your Atari comes to the rescue. With the aid of "Horizon," your computer, and a star atlas or almanac, you can find the altitude and azimuth, in degrees, of any celestial object, at any time, whether it's rising, setting, or high in the sky. Then, with the use of a compass, you can position your camera or telescope in just the right direction, ready to begin observation.

Using Horizon

In order to calculate the altitude and azimuth of an object, the program will ask for the date (month, day, year) and universal time (UT), in hours and minutes, of the event. It will also ask for your latitude and longitude (in degrees and minutes of arc) at the time of the event, as well as the right ascension (RA) (in hours and minutes) and the declination (DEC) (in degrees and minutes) of the object, as published in a star atlas or celestial almanac.

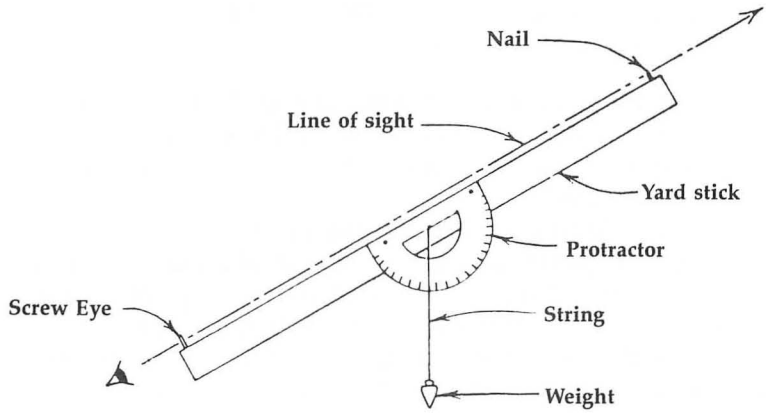
The program will then print out the altitude and azimuth of the object for the specified time and location. Note that if an object is below your horizon at the time, the altitude angle will be a negative number.

If you want to calculate the azimuth angle for a rising or setting object, you will need only your latitude and the object's declination. The output will then be the azimuth angle of the object.

With that information, set up your camera or telescope. Use a compass to position your camera the number of degrees from true north specified by the azimuth angle. If the altitude angle of your camera needs to be set, use a device like that

shown in the figure to tilt your camera the required number of degrees. Lock everything down, and wait for the specified time to arrive!

A Simple Elevation-Only Tracking Device



How Horizon Works

Lines 250–370 calculate your local sidereal time for the event. Lines 390–480 gather information concerning your position and the object's position. Lines 490–540 convert everything to radians and calculate the altitude and azimuth of the object, at the specified time, and lines 850–990 calculate the object's rising or setting azimuth.

Lines 1110–1230 calculate the Julian day for the month, day, and year that you entered, in order to determine your local sidereal time. If you wish, you can modify lines 330, 440, and 860 to print your longitude and latitude.

Horizon: A Celestial Coordinates Calculator

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
HC 150 DIM N(12)
CM 160 LET RADIAN=0.0174532
EG 170 LET DEGREE=57.295778
NA 180 OPEN #1,4,0,"K:"
FB 190 FOR I=1 TO 12:READ N:N(I)=N:NEXT I
```

```
LK 200 DATA 0,31,59,90,120,151,181,212,243,273
       ,304,334
GF 210 GOTO 720
KB 220 REM ** CALCULATE LST **
PE 230 SETCOLOR 2,6,4
LP 240 POKE 752,0
MH 250 ? "{CLEAR}{DOWN}Input Month,Day,Year (i
       .e.,12,7,78)":INPUT MO,DA,YR
KG 260 YEAR=YR+1900
NL 270 GOSUB 1100
BA 280 N=N(MO)+DA
JK 290 ? "{2 DOWN}UNIVERSAL TIME (UT) of Event
        (Hr.,Min)"
FK 300 INPUT T1,T2
EB 310 T=T1+(T2/60)
LL 320 ? "{2 DOWN}Input your Longitude (Deg,Mi
       n.)"
KO 330 ? "{DOWN}JAX, FL = 81, 39.74"
MH 340 ? :INPUT L1,L2
CN 350 L=L1+(L2/60)
LJ 360 LST=K+(0.0657*N)+(1.0027*T)-(L/15)
KF 370 IF LST>24 THEN LST=LST-24
FN 380 REM ** CAL. ALT & AZIM **
NM 390 ? "{2 DOWN}Input R.A. of Object (Hr,Min
       )":INPUT R1,R2
HM 400 RA=R1+(R2/60)
AO 410 HA=LST-RA
HF 420 HA=HA*15
JN 430 ? "{CLEAR}{2 DOWN}Input your Latitude (
       Deg,Min)"
KJ 440 ? "{DOWN}JAX, FL = 30, 19.75"
FA 450 INPUT L1,L2
CP 460 L=L1+(L2/60)
IA 470 ? "{2 DOWN}Declination of Object (Deg,M
       in)":INPUT D1,D2
KB 480 DEC=D1+(D2/60)
MI 490 REM ** CONVERT ALL TO RADIANS **
LN 500 HA=HA*RADIAN
EE 510 L=L*RADIAN
EF 520 DEC=DEC*RADIAN
EK 530 Y=(SIN(DEC)*SIN(L))+(COS(DEC)*COS(L)*CO
       S(HA))
GF 540 E=ATN(Y/SQR(1-Y^2))
PN 550 ALT=E
IO 560 Y=(SIN(DEC)-SIN(L)*SIN(E))/(COS(L)*COS(
       E))
EL 570 IF Y<0 THEN AZ=3.1415927+ATN(SQR(1-Y^2)
       /Y):GOTO 590
LP 580 AZ=ATN(SQR(1-Y^2)/Y)
HL 590 REM ** CONVERT BACK TO DEG
```

```
GL 600 ALT=ALT*DEGREE
OA 610 AZ=AZ*DEGREE
LP 620 IF SGN(HA)=1 THEN AZ=360-AZ
KK 630 REM ** PRINT OUT **
PJ 640 SETCOLOR 2,7,3
ME 650 ? "{CLEAR}{DOWN}** HORIZON COORDINATES
       **"
AF 660 ? "{2 DOWN}DATE: ";DA,MO,YR+1900
DJ 670 T=INT(T*100)/100
GG 680 ? "{DOWN}UT= ";T;" HR's"
MO 690 ? "{3 DOWN}Altitude of object= ";INT(AL
       T*100)/100;" Deg."
AG 700 ? "{3 DOWN}Azimuth of Object= ";INT(AZ*
       100)/100;" Deg."
JC 710 GOTO 1000
LD 720 REM ** START **
PG 730 SETCOLOR 2,3,4
MF 740 POKE 752,1
KP 750 ? "{CLEAR}{DOWN}** HORIZON COORDINATES
       CALCULATOR **"
FI 760 ? "{3 DOWN}1. Cal. Altitude & Azimuth"
DO 770 ? "2. Cal. Angle of Rising Object"
LC 780 ? "3. Cal. Angle of Setting Object"
LP 790 ? "{2 DOWN}Input one of above."
DJ 800 GET #1,A:IF A<49 OR A>51 THEN 800
NG 810 IF A=49 THEN 220
JG 820 IF A=51 THEN SET=1
DI 830 REM * CAL. ANGLE *
MF 840 POKE 752,0
GK 850 SETCOLOR 2,13,4:? "{CLEAR}{2 DOWN}Input
        your Latitude (Deg,Min)"
HM 860 ? "{DOWN}JAX,FL. is 30, 19.75"
MP 870 ? :INPUT L1,L2
DF 880 L=L1+(L2/60)
NF 890 ? "{3 DOWN}Input Object's Declination (
       D,M)":INPUT D1,D2
JO 900 DEC=D1+(D2/60)
EI 910 L=L*RADIAN
EJ 920 DEC=DEC*RADIAN
OK 930 Y=SIN(DEC)/COS(L)
LD 940 AN=ATN(Y/SQR(1-Y^2))
MP 950 AN=AN*DEGREE
OK 960 IF SGN(AN)=-1 THEN AN=ABS(AN)+90:GOTO 9
       80
JB 970 AN=90-AN
PP 980 IF SET=1 THEN AN=ABS(AN)+180
FE 990 ? "{2 DOWN}Object's Azimuth Angle= ";IN
       T(AN*100)/100;" DEG."
KN 1000 REM ** AGAIN **
OM 1010 POKE 752,1
```

```
BP 1020 ? "{2 DOWN}Another calculation?"
GF 1030 GET #1,A
GH 1040 IF A=89 THEN RUN
HC 1050 IF A=78 THEN GOTO 1070
ME 1060 GOTO 1030
PB 1070 POKE 752,0
EK 1080 GRAPHICS 0
KB 1090 END
OP 1100 REM ** JULIAN DAY **
MD 1110 A=INT((7*YEAR)/4)
IL 1120 B=(367*YEAR)-A+30
FH 1130 MJD=B-678987+0.5
ON 1140 JD=MJD+2400000
JI 1150 J1=JD-2415020
AI 1160 T=J1/36525
HB 1170 J2=8640184.54*T+0.0929*T^2
PI 1180 J3=J2/3600
AC 1190 J3=J3+6.6460656
JA 1200 J4=J3/24
OF 1210 J5=J4-INT(J4)
FM 1220 K=J5*24
KG 1230 RETURN
```

Chapter 4

Invisible Music

Paul Gentieu

Using the simple routine described here, you can add sophisticated music to your BASIC programs—without affecting execution speed.

If you've written a program that includes music, you probably noticed that playing that music requires quite a bit of processing time. The reason is simple: The sound registers must be constantly updated. As a result, it is difficult to do any complex calculations or graphics manipulations while your music routine is playing.

Having run into this problem, I decided to write a small machine language routine that plays music "in the background." That music is invisible, as far as BASIC is concerned, and it frees your program to do more important things. It can be of great value, particularly in games or other applications where it would be nice to add music without affecting execution speed.

This routine interfaces with BASIC via the USR function. Simply make one call to the routine and forget it. The tune will immediately begin playing and will not affect the execution speed of any BASIC program. You can use up to four voices.

Once you have decided how many voices you want, POKE the audio control registers (53761, 53763, 53765, and 53767) with 160 for a pure tone plus the volume (from 1 to 15) that you wish to use. The voice parameter is passed to machine language, along with the address of the string holding your music data, by the statement A=USR(1536,ADR(A$),VOICES). When setting up the string, the first number is the duration; it is followed by the notes themselves. The table gives a listing of note values.

An example is helpful. A typical statement might be A=USR(1536,ADR (A$),2). In this case, the string A$ would be made up of a duration value, then the values for two notes to be played simultaneously, then another duration, then two more notes, and so on.

The duration is measured in sixtieths of a second. Since a string can hold only individual values from 0 to 255, the duration can range from 1/60 second to 4-1/4 seconds. That should be a wide enough range for most applications. Note

that when you use a zero for the duration, the routine will start the music over from the beginning.

Note Values

	Note	Value
High notes	C	29
	B	31
	A#	33
	A	35
	G	40
	F#	42
	F	45
	E	47
	D#	50
	D	53
	C#	57
	C	60
	B	64
	A#	68
	A	72
	G#	76
	G	81
	F#	85
	F	91
	E	96
	D#	102
	D	108
	C#	114
Middle C	C	121
	B	128
	A#	136
	A	144
	G#	153
	G	162
	F#	173
	F	182
	E	193
	D#	204
	D	217
	C#	230
Low Notes	C	243

modified to work on strings of any length, but 256 bytes should be enough for most tunes.

Caution

One thing to watch for: When the routine is running (and it will continue to run if you press the BREAK key to stop the program), you should not type in any program lines or cause the program to modify itself in any way. Nor should you type in anything in immediate mode, as that may cause the string holding the music to be moved around in memory and result in incorrect notes.

A Simple Example

A sample BASIC program with a demonstration tune is included to show just how easy the routine is to set up and use.

The routine works using the interrupt generated by the second system software timer. I chose to use the timer interrupt over the vertical blank because the music routine is short and many excellent utilities already use the vertical blank interrupt. The second timer is one of two that generate interrupts. Timer 1 was not used, because it is used to time input/output and serial bus events.

Duration is very easily implemented using timers. The second timer is started by storing a clock value in $21A (the timer 2 value address). This value is decremented during each vertical blank interrupt (once every 1/60 second). Once the timer hits zero, the computer interrupts what it is doing and performs an indirect JSR through $228 and $229 (the timer 2 interrupt vector).

The music routine is set up by placing its beginning address in these vector locations. Once that's done and the timer has been started, the routine will execute without slowing down BASIC operations. To implement duration, all that must be done is to store different clock values in $21A, controlling how frequently the routine is called (and how often the sound registers are updated). The routine ends with RTS since it was called with JSR.

Invisible Music

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
PA 5 M=1
JM 10 POKE 53761,168:POKE 53763,168:REM POKE A
      UDIO CONTROL REGISTERS WITH A PURE TONE
      AND VOLUME OF 8
AK 20 DIM A$(256):REM MUSIC HOLDING STRING
PP 30 TRAP 50
LH 40 READ D:POKE 1536+T,D:T=T+1:GOTO 40:REM R
      EAD ROUTINE DATA
CD 50 TRAP 200
GK 60 READ D:A$(M,M)=CHR$(D):M=M+1:GOTO 60:REM
       READ MUSIC DATA
IH 70 REM ***ROUTINE DATA***
IP 80 DATA 104,169,0,141,254,6,104,133,205,104
      ,133,204,104,104,10,233,1,141,255,6,169,
      36,141,40,2,169,6,141,41,2,169
KI 81 DATA 1,141,26,2,96,172,254,6,162,0,177,2
      04,240,29,141,253,6,200,177,204,157,0,21
      0,232,232,236,255,6,48,243,240
EG 82 DATA 241,200,140,254,6,173,253,6,141,26,
      2,96,169,0,141,254,6,169,10,141,26,2,96,
      -1
OE 90 REM ***MUSIC DATA***
AE 100 DATA 9,81,121,9,96,143,9,121,81,9,96,12
       1,9,81,143,9,96,81,9,121,121,9,96,143,9
       ,81,81,9,60,121,9,60,143,9,64,81
CP 101 DATA 9,72,121,9,91,143,9,108,81,9,121,1
       21,9,108,143,9,91,81,9,108,121,9,121,14
       3,9,108,81,9,68,121,9,68,143
AK 102 DATA 9,72,81,9,68,121,9,81,143,9,96,81,
       9,121,121,9,96,143,9,81,81,9,96,121,9,1
       21,143,9,96,81,9,60,121,9,60,143
EK 103 DATA 9,64,81,9,72,121,9,91,143,9,108,81
JA 104 DATA 9,121,121,9,108,143,9,91,81,9,108,
       121,9,72,143,9,91,81,9,53,121,9,53,143,
       9,60,81,9,64,121,9,60,143,9,81
LN 105 DATA 81,9,47,121,9,60,143,9,40,81,9,40,
       121,9,53,143,9,64,81,9,64,121,0,-1
IB 200 REM ***START MUSIC***
EC 210 A=USR(1536,ADR(A$),2)
AK 220 GOTO 220:REM YOUR PROGRAM HERE
```

Chapter 5

Tape and Disk Utilities

Chapter 5

Atari Tape Enhancer

Jordan Powell

If you've ever been frustrated by the lack of file handling on the Atari program recorder, then the two short programs described here are for you. The article also includes information on string handling and program compaction with Atari BASIC.

Looking for files on the Atari program recorder can be a real chore. You can write down the file description and its location, but the paper could get misplaced. Loading files one after the other to get to the right one isn't my idea of fun either.

Taking a lesson from the way the VIC-20 handles its tape files, I have written two programs to help make life easier for Atari tape users. You use them as follows: Start out by loading a tape and zeroing the tape counter, then advance the tape to a reading of 20 on the counter. Next, store up to six files on the tape, being careful to note the filename you want (up to 16 characters), the location of the beginning of the file, and the command used to save the file.

When you're done, rewind the tape and run Program 1. It will ask you for the name you would like to give the cassette; respond with a tape name of up to 16 characters (for example, GAME TAPE #1). The program will then ask for the names, locations, and commands used to save up to six files on the tape. You respond with the filename, tape counter reading for the beginning of the file, and the first letter of the command used to save the file. If you have less than six files, respond to the filename prompt after the last file by pressing ESCape twice followed by RETURN. The program will stop prompting you and store what I call the system tape file on the cassette. It contains the information you just typed in, and all the information you need to locate and load all of the files on the tape is safely stored on the cassette itself.

Program 2 reads the system file, writes a menu to the screen, and asks you to select the number of the file you want to load. After you select the appropriate file, it tells you the counter reading at which it will be found. Advance the tape to that location and press RETURN to load the file. If you make a mistake locating the file, the program gives you another chance.

Atari Strings

The key to these programs lies in an understanding of Atari BASIC string handling. A string is a sequence of one or more characters. In statement 40 of Program 1, the dimension statement defines the string variables used in the program. CBUF$ is the string variable which will be put into the cassette buffer and subsequently written to the tape. The cassette buffer is an area in RAM from which the Atari writes to the program recorder.

The rest of the string variables will hold inputs from the keyboard. As filenames, locations, and commands are entered, they are added to CBUF$ one after the other. S is used as a space-saving measure. Every time a constant is used in Atari BASIC, it takes up seven bytes. Using a variable causes it to be stored once when it is defined, and all other references take up only one byte.

Adding characters to CBUF$ as in line 110 is done by using the following form of expression: CBUF$(start,end)= TN$, where start and end are the starting and ending positions in the string CBUF$. TN$ contains the character string to be placed into CBUF$ at the positions indicated by start and end. By manipulating the starting and ending positions of data within CBUF$, the string is filled with file data one piece at a time. With this explanation and the Atari BASIC reference manual you should be able to decipher the rest of the program.

To speed up the loading of these programs, you can make them smaller so there is less to load. This can be done by removing REM statements, substituting variables for constants as explained above, putting two lines of code on one logical line (a logical line is one starting with a line number) separated by colons, and by substituting ? for the word PRINT in PRINT statements.

Program 1. Tape File Maker

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
AI 10  REM CREATE SYSTEM TAPE FILE
KH 20  PRINT (CLEAR)
HL 30  REM DEFINE VARIABLES AND CONSTANTS
LO 40  DIM CBUF$(128),FN$(12),CT$(3),TN$(16),SM
       $(1):S=16
JN 50  REM FILL CASSETTE BUFFER WITH BLANKS
```

```
CG 60 FOR N=1 TO 128:CBUF$(N,N)=" ":NEXT N
BN 70 REM INPUT TAPE VOLUME LABEL
CD 80 PRINT "INPUT TAPE NAME(16 CHARS MAX)"
LP 90 INPUT TN$
GM 100 REM PUT TAPE LABEL INTO BUFFER
NK 110 CBUF$(1,S)=TN$
GF 120 REM INPUT INFORMATION FOR UP TO 6 FILES
BA 130 FOR N=1 TO 6
DP 140 PRINT "INPUT FILE ";N;" NAME"
NO 150 INPUT FN$
JG 160 IF FN$(1,1)="{ESC}" THEN GOTO 270:REM
       {T} WHEN ESCAPE HIT WE ARE DONE ENTERIN
       G FILE INFO
BA 170 REM WHAT IS TAPE COUNTER READING
NA 180 CBUF$(N*S+1,N*S+13)=FN$
HF 190 PRINT "TAPE COUNTER READING FOR THIS FI
       LE? (3 DIGITS)"
IG 200 INPUT CT$:CBUF$(N*S+13,N*S+15)=CT$
PO 210 REM HOW IS TAPE FILE SAVED{6 SPACES}
BJ 220 PRINT "WHICH COMMAND USED TO SAVE ?"
NP 230 PRINT "CSAVE/SAVE C/LIST C -(C,S,L)"
OK 240 INPUT SM$
OL 250 CBUF$(N*S+S,N*S+S)=SM$
CF 260 NEXT N
AM 270 REM WRITE SYSTEM FILE TO CASSETTE
CJ 280 PRINT "PRESS PLAY/RECORD-RETURN"
MO 290 OPEN #1,8,0,"C:"
IN 300 PRINT "WRITING SYSTEM FILE"
PE 310 PRINT #1;CBUF$
```

Program 2. Tape File Reader

```
LP 10 REM DISPLAY SYSTEM TAPE FILE MENU
AJ 20 PRINT "{CLEAR}":REM {T} CLEAR SCREEN
HL 30 REM DEFINE VARIABLES AND CONSTANTS
LO 40 DIM CBUF$(128),FN$(12),CT$(3),TN$(16),SM
      $(1):S=16
BH 50 REM OPEN CASSETTE
OO 60 PRINT "PRESS PLAY THEN HIT RETURN"
JG 70 OPEN #1,4,0,"C:"
AA 80 REM FILL CASSETTE BUFFER WITH BLANKS
CI 90 FOR N=1 TO 128:CBUF$(N,N)=" ":NEXT N:REM
       READ IN AND PRINT TAPE LABEL TO THE SCR
      EEN
FN 100 PRINT "{CLEAR}";"READING SYSTEM FILE"
HN 110 INPUT #1;CBUF$:PRINT "{CLEAR}"
NN 120 PRINT CBUF$(1,15):PRINT :REM {T} SPACE
LH 130 REM PRINT FILE MENU ON SCREEN
BB 140 FOR N=1 TO 6
```

```
MN 150 FN$=CBUF$(N*S+1,N*S+13)
AG 160 CT$=CBUF$(N*S+13,N*S+15)
OM 170 SM$=CBUF$(N*S+S,N*S+S)
NJ 180 PRINT N;" ";FN$;"  ";CT$;"  ";SM$
CH 190 PRINT
BP 200 NEXT N
EL 210 REM PROMPT FOR PROGRAM TO BE LOADED
BI 220 PRINT :PRINT "ENTER NUMBER OF PGM YOU W
       ANT RUN"
HD 230 INPUT N
MN 240 FN$=CBUF$(N*S+1,N*S+13)
AG 250 CT$=CBUF$(N*S+13,N*S+15)
MB 260 PRINT "ADVANCE TAPE TO ";CT$;" FEET"
GI 270 TRAP 350
KN 280 REM DECIDE WHICH COMMAND TO LOAD WITH
DP 290 IF CBUF$(N*S+S,N*S+S)="C" THEN GOTO 330
DK 300 IF CBUF$(N*S+S,N*S+S)="E" THEN GOTO 340
NA 310 REM LOAD PROGRAM
EG 320 LOAD "C:":PRINT "HIT RETURN"
MJ 330 CLOAD :PRINT "HIT RETURN"
NG 340 ENTER "C:"
EE 350 PRINT "REPOSITION TAPE TO ";CT$;" FEET
       AND RETRY"
GN 360 GOTO 290
```

Chapter 5

Disk Catalog Utility

Andrew Genser

For many computer users, one of the most time-consuming tasks is searching through disks to find a particular program. This program gives you an alternative.

Wouldn't it be nice if you could have an index of all your disks, stored on one disk? That way, you would never have to switch disks—and with a few enhancements like a search capability and printout, you would have an extremely useful utility.

The problem with many disk cataloging systems is that they do much more than is really necessary. The time spent in keeping one up-to-date usually negates its usefulness. Thus, many home computer users end up with a list of programs that includes all kinds of unwanted information. However, this disk housekeeping utility forgoes the unneeded frills to make it much simpler to keep track of your disks.

One Large Directory

"Disk Catalog Utility" (DCU) is really just a mass directory. It creates separate files, on one designated catalog disk, of all your disks' directories. This means that you can get a directory of any disk without going to DOS and switching disks around.

Three other features make DCU even more useful. First, you have the option of doing a disk directory of all of your disks, one of your disks, or all disks in a certain range. Second, DCU makes full use of Atari's wild card. This is a symbol in the filename of a program that allows you to view everything with a specified string in its name. For example, if you want to see files ending with the letters .LST, type *.LST in response to the prompt. The asterisk is the most common wild card symbol, but you may choose your own when using DCU.

A file specification doesn't have to have a wild card. If you know exactly which filename you are looking for, type that in, and DCU will identify all the disks that contain that filename.

Finally, for those who own a printer, DCU will also give you a hard copy of the directory catalog. It channels whatever is printed on the screen directly to your printer, using the LPRINT statement.

Creating the Directory

First, you must number each disk, starting with disk number 1. Using a felt-tip pen, write the disk number on the label (not on the disk cover).

Then run the program. You'll be asked to specify the desired wild card symbol. If you hit return, DCU will automatically use the asterisk (*) as your wild card.

At that point, you are ready to catalog your disks. Run DCU, and type C for catalog disk. You will be asked which disk you want to be cataloged. Type in the appropriate number and press RETURN. Then insert disk number one and press RETURN. DCU will read in the directory of disk 1. Then insert the data disk (the disk with DCU and all of your DCU files on it) and press RETURN. DCU will then save that disk's directory as a data file called DISK1.

Repeat this procedure until all of your disks have been cataloged. Be sure to type in the correct disk number when cataloging.

Searching for a File

To get a complete directory, type D to get to the directory mode; then insert the data disk and press RETURN. You will then have several options. If you have a printer and want a printout of your directories, type Y when asked if you want a printout. If you don't have a printer, press N (or just hit RETURN, and DCU will default to nonprintout).

Next, you will be asked if you want to search a particular range of disks. Type Y to specify a range. For example, if you type Y followed by 3,8 then DCU will search through directories 3, 4, 5, 6, 7, and 8 for whatever filename you specify. If you want DCU to start with disk 1 and go on until it can't find any more files, simply press RETURN.

You can also specify the file spec (name of the file) you are looking for. This is where you use the wild card symbol. Type in your chosen file spec; alternately, hit RETURN to display all files from the specified disk range. DCU will display each disk number as it searches, followed by all files that match the file spec. When DCU has searched through all catalogs, or has finished the specified range, it will display the number of files found and RETURN you to the main screen.

Disk Catalog Utility

For error-free program, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
HO 10 GRAPHICS 0:SETCOLOR 2,12,4:SETCOLOR 1,0,
      15:DIM D(1000),SPEC$(17),F$(17),FN$(15):
      DSK=1
JN 15 OPEN #1,4,0,"K:"
OD 17 ? "D I S K  C A T A L O G  U T I L I T Y
      ":? "WILDCARD symbol ?":GET #1,WILD:IF W
      ILD=155 THEN WILD=ASC("*")
AO 19 FOR I=1 TO 38:? CHR$(WILD);:NEXT I
ID 20 CLOSE #2:? :? "DISK CATALOG UTILITY [DCU
      ]":? "BY ANDREW GENSER":? :? "CATALOG DI
      SK":?
BM 22 ? "DIRECTORY"
AF 40 ? "Your choice=>":GET #1,K:IF CHR$(K)<>"
      C" AND CHR$(K)<>"D" THEN ? "{CLEAR}
      {BELL}D OR C":GOTO 40
EH 50 IF CHR$(K)="C" THEN 3000
IF 60 REM DIRECTORY FUNCTION
CG 65 ? "{CLEAR}{DOWN}[DCU] DIRECTORY":FLS=0:S
      ETCOLOR 2,15,4
IJ 70 ? "Insert data disk,then hit RETURN":GET
       #1,K
AF 75 ? "Screen output to printer (y/n)":GET #
      1,K:IF CHR$(K)="Y" THEN FLAG=1:GOTO 80
PF 77 FLAG=0
MN 80 ? "Disk range specs (y/n)":GET #1,K:IF C
      HR$(K)="Y" THEN ? "Disk range ";:INPUT S
      TART,EN:GOTO 100
IM 90 START=1:EN=99999
JD 100 ? "File spec ";:INPUT SPEC$
DL 105 IF SPEC$="" THEN SPEC$(1)=CHR$(WILD):SP
      EC$(2)=".":SPEC$(3)=CHR$(WILD)
EL 110 F$=SPEC$:SPEC$(1,2)="  ":SPEC$(3)=F$:F$
      =""
FN 120 FOR DSK=START TO EN
HN 130 FN$="D:DISK":FN$(7)=STR$(DSK)
KN 140 TRAP 605:OPEN #2,4,0,FN$:? "{BELL}Disk
      ";DSK:TRAP 605:IF FLAG THEN LPRINT "DIS
      K ";DSK
BD 150 INPUT #2;F$:I=3
LN 152 REM FILENAME COMPARISON/SEARCH
PK 155 IF F$(1,1)<>" " AND F$(1,1)<>"*" THEN 6
      00
HL 160 IF ASC(SPEC$(I,I))=WILD THEN I=I+1:GOTO
       500
PI 170 IF SPEC$(I,I)="." THEN 500
DO 180 IF SPEC$(I,I)<>F$(I,I) THEN 150
```

```
IJ 200 IF I<LEN(SPEC$) THEN I=I+1:GOTO 160
CO 210 FLS=FLS+1:? F$:IF FLAG THEN LPRINT F$
GH 215 GOTO 150
MK 499 REM EXTENDER COMPARISON ROUTINE
IA 500 I=I+1:L=LEN(F$)-6
NK 510 IF ASC(SPEC$(I,I))=WILD THEN 210
EC 520 IF SPEC$(I,I)<>F$(L,L) THEN 150
PJ 530 IF I<LEN(SPEC$) THEN I=I+1:L=L+1:GOTO 5
       10
DE 540 FLS=FLS+1:? F$:IF FLAG THEN LPRINT F$
GK 542 GOTO 150
JP 600 ? F$:CLOSE #2:NEXT DSK
NJ 605 ? :? FLS;" File(s) found":IF FLAG THEN
       LPRINT :LPRINT FLS;" File(s) found"
DC 610 GOTO 20
EC 2999 REM CATALOG FUNCTION
IA 3000 ? "{CLEAR}":SETCOLOR 2,1,4:SETCOLOR 1,
        0,15:? "[DCU]CATALOG":? :? "DISK # TO
        BE CATALOGED ";:INPUT DSK
LC 3015 FN$="D:DISK":FN$(7)=STR$(DSK)
HA 3020 ? :? "INSERT DISK #";DSK;:? " THEN PRE
        SS RETURN":GET #1,A
HB 3045 OPEN #2,6,0,"D:*.*":I=0
KN 3050 TRAP 3500:GET #2,K
NA 3052 D(I)=K:I=I+1:GOTO 3050
OG 3500 D(I+1)=0:CLOSE #2:? :? "INSERT DATA DI
        SK-HIT RETURN":GET #1,A:OPEN #2,8,0,FN
        $:I=0
KI 3505 TRAP 3600:? CHR$(D(I));:PUT #2,D(I):I=
        I+1
JL 3510 IF D(I)=0 THEN 3600
NE 3515 GOTO 3505
HH 3600 ? "CATALOG COMPLETE{BELL}":CLOSE #2:GO
        TO 20
```

Chapter 5

Diskovery

John C. Waugh

Would you like to know what's really hidden beneath the dull brown exterior of your disks? "Diskovery" will help you find out. Requires a disk drive and Atari DOS 2.0S.

What's on an Atari disk? One way to find out is to use the DOS directory. That won't tell you much, though, just names, lengths, and free sectors. And it won't tell you anything if it's not a DOS-made disk. To *really* find out what's on it, you'll have to *diskover* it—and this program will give you the tools you need. It will provide you with the ability to change disk memory directly.

Be careful using this program. Be sure to have a backup of any important programs on another disk before trying the examples in this article.

The Problem

The problem was simple to begin with. A friend had brought me a disk with a problem: There were two versions of a program on it, but both had the same name. Both appeared on the DOS directory listing, although with different lengths. COPY and LOAD would get only the first and older version of the two. And even the *rename* option of DOS would rename both programs simultaneously, to the same new name. We didn't try the *delete file* option, for fear of losing both. So, knowing something about Atari's direct disk access, I wrote a short and simple program to fix the problem. That was the kernel from which the current program grew.

Atari DOS accesses disks a sector at a time. However, that can also be done independently of DOS by setting up certain pointers in the disk control block (DCB) section of memory and calling the operating system (OS) to do the dirty work. The DCB and OS do not need the DOS loaded in with this method. Look at lines 8010 to 8100 in the program. This is the subroutine to do direct disk access. Here's how it works.

A buffer area must be set up to hold the contents read from a disk sector, or to be written to a disk sector. This buffer should be 128 bytes long since there are 128 bytes per sector. BUF$ is used in this program. Line 8010 identifies the beginning of the DCB, which occupies memory locations 768–779. POKEing a 1 into the second byte (BLK+1) specifies access to

drive 1. The variable FUN in line 8020 is set by the calling routine; 82 decimal means a read sector command, and 87 will cause a sector to be written to the disk.

In order to read into the buffer, or write from it, the address of the buffer area must be placed in BLK+4 and BLK+5 in low byte/high byte format. This means that BLK+5 will contain the *high* part of the address, the number of times that 256 will go into the address. This is expressed as INT(ADR(BUF$)/256). BLK+4 should have the *low* part of the address, the remainder from division by 256, given by ADR(BUF$)−256*(high part). This low byte/high byte address format is standard on the Atari and other 6502-based microcomputers.

Next, the sector number to be accessed must be broken up into low/high form and placed in BLK+10 and BLK+11. The sectors are numbered from 1 to 720. Some of the sectors are used for the directory, boot-up, and so on, so DOS will show only 707 free sectors on an empty disk, although all sectors can be accessed with this program.

Now it's time to call the operating system. Located at hexadecimal address $E453 (decimal 58451) is the subroutine that will either write the buffer to the specified disk sector or read the sector contents into the buffer. To call this subroutine you need to do an assembly language JSR, so a short machine language routine is in order. Line 8080 initiates the short machine language program that was put in DSKINV$ by lines 205 and 210. Essentially all this does is JSR to $E453, then return to BASIC with an RTS. After return, if there was a problem reading or writing a sector, an error code is placed in BLK+3 (the value 1 indicates no error). Line 8080 also puts this code in the variable NR for use in lines 8086 and 8090.

Looking at the Disk

Now that you know how direct disk access can be achieved, look at the main program. When you have the program entered, save it before running it. Put into your drive a disk that has a lot of varied DOS files on it—perhaps an AUTORUN.SYS, a machine language object program, a LIST/ENTER program, and some SAVE/LOAD programs. Of course the DOS.SYS and DUP.SYS files themselves would be nice to look at too.

Run the program, and a menu will appear. Push the SELECT button. If all is well, you should be able to toggle the

screen background between normal blue and green (or blue and darker blue, depending on your TV). When the screen is green (or darker blue), whatever is on the screen will be dumped to a printer, assuming that a printer is on-line. At any point in the program run, regardless of what it is doing, you can toggle between printing and nonprinting modes. That will not interrupt a screen dump already in progress, but any normal keyboard key will. The toggle is done using a *vertical blank interrupt*. More on that later.

Select a normal blue screen (for nonprinting) for now. Choose option 2 and press RETURN. This option allows you to look at the contents of a sector in a special way; the program will assume that you are looking at one of the sectors from 361 to 368, and will display the data accordingly. In response to the sector number prompt, enter 361, to look at the first *directory sector*. You will hear the beep from the TV speaker as the sector is read, and then the bytes in the sector are displayed as characters on the screen. Directory sectors are arranged in sets of 16 bytes, each set being the information about one file listed in the DOS directory. Probably DOS.SYS and DUP.SYS will be among those shown on your screen.

DOS refers to programs or file entries by number, the first one being 0 up to a maximum of 64 files (note that the maximum file number is thus 63). The display is arranged in 8 lines of 16 bytes each, so that each directory entry appears on a separate line. Notice the first byte of a typical directory entry. An uppercase B indicates a normal file that is ready to use, so a first byte of 66 (the ATASCII value for B) tells DOS that this file is OK and ready to go. A locked file has bit 5 of the first byte set (a 1, or on bit); locked files will appear on the directory dump as a lowercase b. If bit 7, the topmost bit, of the first byte is set, then the file is considered deleted from the directory by DOS. Its directory space is up for grabs, and will be overwritten by the next new file saved on the disk. If a file has been deleted using the D option of DOS, as long as no new files have been saved over its directory entry, the file can be undeleted and gotten back.

Undeleting

Let's try an example of that, to see if you've really entered the program without any errors. Write some dummy program, like 10 REM, and save it on disk. Go to DOS and list the directory.

Now use the D option to delete the dummy file; a directory listing will no longer show it. Go back to BASIC and load in the "Diskovery" program. Run it, and choose option 2 again. Look at sector 361. If there are many files on the disk, you may have to look at more sectors, perhaps up to sector 368, to see the dummy program; by pressing RETURN a couple of times, you'll get back to the main menu to select the next sector. Do this until you find the sector with the dummy program—it will show up even though it was deleted.

Now press RETURN once more to get the *change sector?* prompt. Answer Y, then type C for the next prompt, so you can insert bytes into the sector as characters rather than as numbers. Type 1 for the number of characters to change. For the starting byte prompt, determine which byte is the first one in the dummy file. This will be an inverse heart (CHR$(128)) indicating a deleted file. The byte number range for each entry is given on the left side of the screen.

Enter the number of the starting byte. The program will ask for one character to insert. Type a B, then type Y for the next prompt, to actually carry out the change. You will hear a clunk sound as the modified sector is written out to the disk, and a beep as it is read back in and redisplayed. There should be a B as the first byte of the dummy file entry. The dummy has been undeleted. To check, try loading the dummy, or go back to DOS and look at a directory listing.

Protecting the File

Although the dummy file is now available for loading and saving, its sectors have not been protected in the usual DOS way. Thus, future programs might save over them. To protect it for now, save it under a new name (or under the same name). DOS will mark the sectors as in use.

A similar method was used to solve the original problem that led to this program. Two files had exactly the same directory name, so I simply modified the name of the first to be different from the second. That way, DOS would recognize them as different files. *Voilà!* The lost file was recovered.

Invisible Files

As you can see in the directory, the filename extender is simply the characters in the last three spaces allocated for the file entry. You can change the characters in a filename with

interesting results. For example, run option 2 again and choose the change sector option. Choose N this time, for inserting numbers. Type 1 to insert just one number. Now use the dummy file again, and use one of the bytes in its name for the byte to change. Change it to value 125, then type Y to modify the sector. ATASCII value 125 is the clear screen character, displayed in the sector dump as a bent arrow. If you were to go to DOS and request a disk directory now, when the dummy filename goes on the screen, it would clear the screen.

You can't load the dummy file with this modified name because CHR$(125) isn't a legal filename character. However, DOS will not consider the dummy file sectors to be available, so they are protected from being overwritten. You have created a file that cannot be loaded or run without modification of its name. With the proper choice of control characters (for instance, CTRL-back arrow), you can name a file so it doesn't appear on the listing at all.

Here's something else to try. Make another short dummy and save it as the first program on a blank formatted disk. Then save a few more programs (or copies of the dummy) under other names. Run the Diskovery program, use option 2 to look at sector 361, and use the change option with number insertion eight times to insert the following numbers at the beginning of the dummy file entry area (start at the beginning, not at the filename): 66,1,0,4,0,88,156,155. Then change the remaining eight bytes of that directory entry to spaces. You can use the number insertion option, with multiple-byte insertion, since they are all the same number. The number 32 represents the ATASCII character for blank space.

Now to the DOS. List the directory; nothing appears at all. You can still load, run, and save programs normally—but only if you know the right names.

If you've followed all this, you might wonder why, when the sector contents are displayed onscreen, the CHR$(125) is displayed as a bent arrow, rather than clearing the screen. There is a POKE to make the computer display command characters instead of performing their commands. It is found in line 1047 (POKE 766,1). POKE 766,0 will return you to the normal mode. Try POKE 766,1 in direct mode and try editing the screen with the CTRL arrows.

Using option 2, directory dumps also show more information about a file. Stored in the bytes between the first byte

and the first name byte are four bytes that contain the starting sector number of the file and the length of the file in sectors. This information is stored in low/high format, as previously explained. With option 2, the interpretation of these bytes is displayed on the right side of the screen.

Other Options

Let's have some fun with the other options. Examine the directory sector that contains the DUP.SYS file entry, the Disk Utilities Package. Note what sector it starts with. Now go back to the main menu and choose option 1. Specify the sector that DUP.SYS starts with. Enter the choice for character dump, and you will see the first DUP.SYS sector displayed in character form. Somewhere in this sector you will see the text that is shown at the top of the screen when DOS gets control. Using the sector-changing options, you could put in your own string of characters so that whenever you go to DOS, your own customized message would be displayed.

Also note that the computer beeps several times during the display. This is because there are several bytes in the file with value 155, the code for the RETURN key being pressed. These have no character representation, even with POKE 766,1, so they would normally disrupt the screen display with line returns. To avoid this, the program detects them, beeps the speaker to let you know about them, and puts an inverse asterisk onscreen to mark their location. This is done by line 1067. It is necessary to POKE 766 back to normal to get the computer to beep, then POKE 766 again to display control characters for the rest of the dump.

Each sector on the disk is 128 bytes long. However, if the sector has been filled by the Atari DOS, only 125 of those sectors are used for actual data in the file. The other three bytes contain housekeeping information. Byte 126 has, in six of its bits, the file number that the sector is associated with. Thus all sectors that are a part of the second directory entry file will have the number 1 (remember that file references start at file 0, not file 1).

Stored in the other two bits of byte 126, and in eight bits of byte 127, is the forward pointer. This tells which sector is next in line in that particular DOS file. Sectors associated with a certain DOS entry do not have to be next to each other

(contiguous) on the disk. Each one contains a reference to the next one in line. We'll see an example of that in a moment.

Byte 128 contains a count of how many bytes are unused in that sector. Thus, if a sector is the last one in a particular file, it may not be full. DOS needs to know how many bytes are unused, so it doesn't load the remaining garbage into the computer as part of the file. Bytes 125–128, then, are used in this program to compute the information given at the bottom of the sector dumps: file number, forward pointer, and extra sector bytes.

Option 3 on the main menu will automatically trace through a specified DOS file, sector by sector, following the forward pointers. Find the starting sector of the DUP.SYS file. Run the trace option with its first sector as the starting one. Now a menu appears with either *auto trace* or *pause* as an option. *Pause* means the computer will wait for a keypress after each sector is displayed before going on. With *auto trace*, the sectors will go by automatically until the end of the file. If you have a printer on-line, each sector dump will be printed before going to the next one.

For this example, choose the *pause* option. Choose *character dump* next. You will see the sectors of the DUP.SYS file successively displayed. Some of the information, such as the DOS menu, will be intelligible. *Trace* could also be used to look through stored programs. If a program has been saved, much of it will look like garbage since it is in tokenized form. However, if a program has been LISTed to disk, it will be readable since it is in straight ATASCII form. Looking at a LISTed program, you will see the regular program lines, followed by a beep, and the inverse asterisk character. This is the return code (155) for the return keypress after each physical program line.

Option 6 will give you a table of all the sectors used by a given file, if you supply the starting sector. In many cases, the sectors will not be sequential.

Similar to the file trace is main menu option 4. This is used to look at a block of contiguous sectors by inputting the starting sector and the number of sectors. This will not follow the linked list method of the *trace*; it will ignore forward pointers.

Since DOS-created sectors all contain the number of their

associated directory entry, it is possible to make a map of the disk, marking each sector with the file that it is linked to. This is just what option 5 does. Run it, with a disk that has a lot of files on it and has been well used with resaving and reloading. It takes a few minutes to map all 720 sectors. If you have a printer, you might want to toggle into print mode to save a copy of the map for future reference.

The file ID number of each sector is read in turn, and a single character is put on the map: a 0 for file 0, a 1 for file 1 (the second entry), and so on. After file 9, uppercase alphabet characters are put in, A–Z, then lowercase a–z. Since there are 64 possible filenames per disk, there could still be two files after z. These will be the next ATASCII characters, numbers 123 and 124 on Appendix page C-3 in the Atari manual. Also, if a bad sector is found, it is displayed as an asterisk. Bad sectors are often used as a protection device on commercial disks.

Most sectors on a partly full disk will probably be zeros. This doesn't mean they are part of file number 0—they just don't have anything in them. To find out which of those sectors are really part of file 0, use option 3 or 6.

Since file sectors for a given program do not have to be contiguous, it may be that a much-modified program is spread all over the disk. For example, suppose you write and save programs A, B, and C in that order. They are stored sequentially on the disk, with so many sectors per file (say 10 each). Then you go back and modify program A, adding some lines so that it takes up 15 sectors. When it is saved, DOS will still use the first 10 sectors—but it will also take 5 more, the first 5 available after the sectors for program C. So the file linkage map becomes A-B-C-A.

Then you might modify program C to give A-B-C-A-C, and so on. With a heavily modified disk, programs are here, there, and everywhere. The sector linkage map will show this. All this splitting up makes the disk more prone to LOAD and SAVE errors, since the driver has to jump all over the disk for a given file. It is also slower. Duplicating the disk will still retain the same organization on the new one, but using the COPY option of DOS, with filename *.*, file sectors will all be placed contiguously on the new disk, making it more reliable.

DOS Sector Map

Since file sectors can be dispersed on a disk, you might wonder how DOS keeps track of what sectors are in use (and thus should be protected from overwriting). It would be impractical to check all sectors for each SAVE. Instead, DOS maintains a sector map of the disk, updated with each new modifying operation. This map, a bitmap, is stored in sector 360. Thus sector 360 is a special reserved sector, created during formatting. 360 was chosen since it is near the middle of the disk, and so has a short average access time. That is important, since it is accessed during each data saving operation.

Run main menu option 1. Type in 360 for the sector. Choose option 4 (which only appears for a sector 360 choice), a hex map of sector 360. It takes a few seconds for the conversion of numbers to hexadecimal, so be patient. You will then see the contents of sector 360 displayed in hex. Probably this will be mostly F's and 0's.

Zeros indicate sectors that are in use (or thought to be in use by the DOS). Actually each 0 marks four sectors in use, since this is a bitmap. Each set of two hex digits represents one byte. A byte is eight bits, so each two hex digits represent eight sectors on the map. F is the hex digit for decimal 15, which in binary is 1111. Free sectors on a disk are shown as 1's, so FF means that there are eight free sectors. C in hex equals 12 decimal, which is binary 1100. Thus a C on the map means two free sectors followed by two in-use (locked) sectors. In-use sectors are locked out from use by the DOS, whether or not this locking was done by the DOS or by some other method (for example, this program). Thus, you can reserve space which will never be touched by DOS.

It may take a while to get used to reading the sector map if you are not familiar with hexadecimal numbers. To help, decimal numbers that show what sector is represented by the last bit of the last number on that line are listed on the right side of the screen. You can count backwards to identify the other sectors.

Locked Sectors

Format a new disk and leave it blank. Load and run Diskovery, then put the blank disk back in. Choose main menu option 9, lock/unlock sectors. Start at sector 700 and

specify four sectors. Specify *lock*. When the beeps stop, go back to the main menu and choose option 1; then select option 4 for a hex map of sector 360. Notice that not all of sector 360 is used for the hex map. Also notice that the map does not start at byte 1 of sector 360. Although the sector identifier on the right of the map goes up to sector 751, that is just a reference number. Actual disk sectors end with 720.

Sector 360 also contains this information; bytes 4 and 5 have the free sector count that is used on the DOS display. It is stored in low/high form. Take the value of byte 5 times 256, add the value of byte 4 (do a decimal dump of sector 360 if you're uncertain about hex), and the result is the free sector count of the disk. This number is updated as needed by the DOS, and also by the Diskovery program.

Note the values of bytes 4 and 5 from the map of your blank test disk with the four sectors reserved. It should come out to 707−4, or 703. Go back to the main menu and run option 8. Leave the same disk in as both source and destination disk.

Specify sector 690 as the starting point. You will hear various beeps and clunks. When it's done, run a disk map of sector 360 to see the lockouts. What has been done is that sectors 360–368, the DOS directory sectors, have been copied and protected, starting at sector 690. Run a sector dump of 690 to verify this—it should be a copy of the disk bitmap.

If an error is made saving a sector in an individual file to the disk, only that file is lost. However, if an error is made saving a directory entry or the bitmap, it may make all programs on a disk inaccessible. If you have a copy of these sectors, though, you could put them back aright, making the disk healthy again. Thus option 8 can be used to give some measure of protection for important disks, by saving the directory sectors elsewhere on the disk in case they are needed. You could also save directories from a number of disks on a separate directory backup disk. Although the sectors might not be absolutely current if a lot of changes have been made, they will usually allow many otherwise lost programs to be recovered.

Copying Sectors

Since the sector 360 bitmap is resaved every time a SAVE is done, it is the most likely one to go bad. Check for garbage in sector 360 on a problem disk. You may be able to modify it,

or to copy sector 360 from a full disk to protect all sectors. Option 7 on the main menu allows you to copy single sectors from anywhere to anywhere, on the same disk or to other disks. The bit map will also be updated with this option.

The last option on the main menu, 10, will allow you to use the various features of Diskovery to examine the computer's main memory, instead of disk storage. For example, select this option and give 42240 for the start of memory to look at. You may then enter the number of sequential 128-byte blocks to dump. Memory will be displayed in 128-byte blocks, since this option uses the same routines as the sector dumps and each sector is 128 bytes long.

Next indicate either automatic sequential dumps or pause between blocks. Finally, specify the type of display. Remember that a hex display will take longer to do the conversions than the others. With a starting location of 42240, and a character dump, you will see some of the BASIC keyword identifiers, since you are looking into the brains of the BASIC cartridge. Somewhere in main memory, you could also find the actual lines for the Diskovery program since it is currently loaded in. Look at memory starting at 1536 to see the machine language routine to toggle the print mode. Those numbers in decimal should be the same as the DATA items on lines 230 and 231 in the program listing.

Use this program with caution. Misuse can really scramble a disk. But proper use will allow you to rejuvenate sick disks and to perform nice non-DOS tricks. Either way, you'll learn a lot in the doing.

Diskovery Variables List

BIT$	Lookup table of decimal numbers with successive bits set.
BLA$	String of blanks for erasing screen text.
BUF$	Buffer area used to store information from a sector, or to be sent to a sector.
DSKINV$	Calls the OS subroutine for disk access.
G$	Holds Y or N for user response.
H$	Stores BUF$ contents after hex conversion.
HEX$	Lookup table of hex digits for conversion.
KNOT$	Machine language routine to logically NOT a number.
ND$	Logically ANDs a number with another number.
OAR$	Logically ORs a number with another number.

VB$	Machine language routine to initialize the vertical blank routine set up on page 6.
AP	Decision for auto trace or pause.
B	Temporary variable for reading data into strings.
BIT	ATASCII value of BIT$(BITMOD).
BITMOD	Index to BIT$.
BLK	Start of disk control block in memory.
BN	Bytes of same number to insert into BUF$.
CN	Number of characters to insert into BUF$.
FN	File number from sector data.
FP	Forward pointer to next file sector.
FUN	Holds function number for disk access routine.
H	ATASCII value of BUF$ characters to convert to hex.
HH	High byte of H.
HL	Low byte of H.
I	Loop index variable.
J	Temporary storage and loop index.
K	Byte from screen in printer dump routine.
LK	Choice variable for lock/unlock options.
M	Temporary storage of PEEK values.
MEM	Pointer to memory block for dump.
N	Index variable to H$.
NB	New byte value to insert into BUF$.
NFRE	Number of free sectors from sector 360.
NFREH	High byte of NFRE.
NFREL	Low byte of NFRE.
NR	Error number for bad sector read.
NSL	Number of sectors to lock/unlock.
NUM	Number of sectors or blocks to look at.
P	Byte from keyboard to continue at keypress.
PK	Variable for temporary PEEK value storage.
Q	USR function dummy variable.
RSLT	Result of logical operation subroutine.
SEC	Sector to access.
SECHI	High byte of SEC.
SECLO	Low byte of SEC.
SS	Starting sector for operations.
SSH	High byte of SS.
SSL	Low byte of SS.
TY	Type of dump requested.
WH	Main menu choice.

Diskovery Subroutines List

100–260 Initialization.
300–770 Main menu and submenus.
660–670 Look at memory instead of disk.
780–1000 Handling some menu choices.
1040–1105 Character dump.
1106–1160 Change a sector.
1200 Erase text.
1400–1440 Copy sector option.
1600–1640 Backup directory option.
1700–1799 Lock and unlock sectors on bitmap, and update free sector count.
1800–1840 Single file map options.
2010–2200 Sector linkage map option.
3000–3060 Screen dump to printer.
4040–4060 Hex conversion.
4205–4240 Hex dump.
5015–5060 Decimal dump.
7010–7100 Determine file number and forward pointer.
8010–8100 Read/write file sector.

Diskovery

Caution: The misuse of this program can destroy valuable programs; please read the accompanying article before using Diskovery.

For error-free program entry, read "The Automatic Proofreader" in Chapter 1 before typing in this program.

```
JA 10 REM LISTING 1
DO 100 REM DISKOVER.Y *** BY JOHN C. WAUGH --
      4/83 ***
ON 105 CLOSE #2:OPEN #2,4,0,"K:":CLOSE #4:OPEN
       #4,12,0,"S:":POKE 766,0
MI 200 DIM BUF$(130),DSKINV$(10),G$(1),ND$(12)
      ,HEX$(16),H$(1155),BLA$(38),VB$(11),BIT
      $(8),OAR$(12),KNOT$(12)
GP 203 HEX$="0123456789ABCDEF":BLA$="
      {37 SPACES}"
NB 205 DATA 104,32,83,228,96
HE 210 RESTORE 205:FOR I=1 TO 5:READ B:DSKINV$
      (I)=CHR$(B):NEXT I:REM PLA,JSR $53$E4 (
      DSKINV),RTS
FH 215 DATA 104,104,104,133,209,104,104,37,209
      ,133,209,96
```

```
JE 220 RESTORE 215:FOR I=1 TO 12:READ B:ND$(I,
       I)=CHR$(B):NEXT I:REM PLA,PLA,PLA,STA20
       9,PLA,PLA,AND209,STA209,RTS
AP 225 OAR$=ND$:OAR$(8)=CHR$(5):KNOT$=ND$:KNOT
       $(8)=CHR$(69):REM CHANGE AND TO OR AND
       EOR FUNCT.
AC 230 DATA 72,169,8,141,31,208,173,31,208,201
       ,5,208,8,173,255,6,73,255,141,255,6
AG 231 DATA 173,255,6,201,255,240,7,169,148,14
       1,198,2,208,5,169,162,141,198,2,104,76,
       95,228
MH 235 RESTORE 230:FOR I=1 TO 44:READ B:POKE 1
       535+I,B:NEXT I:REM SEE ASSEMBLER LISTIN
       G & TEXT
EE 240 DATA 104,162,6,160,0,169,6,32,92,228,96
ON 245 RESTORE 240:FOR I=1 TO 11:READ B:VB$(I)
       =CHR$(B):NEXT I
FE 250 Q=USR(ADR(VB$))
EL 260 J=128:FOR I=1 TO 8:BIT$(I)=CHR$(J):J=J/
       2:NEXT I
CE 300 LM=23:POSITION 0,0:? CHR$(125);"(ON A S
       TANDARD DOS DISK":? "SECTORS 361-368 AR
       E THE DIRECTORY)":? :?
BP 310 ? "1=LOOK AT SECTOR"
EJ 315 ? "2=LOOK AT SECTOR AS DIRECTORY SEC"
MN 320 ? "3=TRACE FILE BY HAND"
HL 325 ? "4=LOOK AT BLOCK OF SECTORS"
PE 330 ? "5=SECTOR LINKAGE MAP OF DISK"
FB 335 ? "6=SINGLE FILE MAP"
JJ 340 ? "7=COPY SECTOR"
PP 345 ? "8=BACKUP DIRECTORY"
NI 350 ? "9=LOCK/UNLOCK SECTORS"
FN 351 ? "10=LOOK AT MEMORY"
EF 355 ? :INPUT WH
GM 360 ON WH GOTO 400,400,450,480,2010,1800,14
       00,1600,1700,375,1900
PJ 375 ? CHR$(125):? :? :? "STARTING MEMORY LO
       CATION";:INPUT MEM
EL 376 ? :? "# OF 128 BYTE BLOCKS TO LOOK AT";
       :INPUT NUM:GOTO 500
FE 400 ? CHR$(125):? :? :? "(DIRECTORY SECTORS
        = 361 - 368)":? :? "WHAT SECTOR";:INPU
       T SEC
IK 430 IF WH=2 THEN TY=1:GOTO 720
GH 440 GOTO 600
PN 450 ? CHR$(125):? :? :? "WHAT SECTOR TO STA
       RT TRACE";:INPUT SEC:GOTO 500
IK 480 ? CHR$(125):? :? :? "NUMBER OF SECTORS"
       ;:INPUT SEC2:? :? "STARTING SECTOR";:IN
       PUT SEC1
```

```
AB 500  ? CHR$(125):? :? :? "1=AUTO TRACE":? "2
        =PAUSE":? :INPUT AP
BF 600  ? CHR$(125):? :? :? "TYPE OF DISPLAY:":
        ? :? "1=CHARACTER":? :? "2=HEXADECIMAL"
        :? :? "3=DECIMAL"
AL 605  IF SEC<>360 THEN ? :INPUT TY:GOTO 650
CA 610  ? :? "4=SECTOR 360 HEX MAP":? :INPUT TY
GK 650  IF WH<>10 THEN 700
IJ 660  LM=12:? CHR$(125):SEC=1:FOR M=MEM TO ME
        M+128*(NUM-1) STEP 128:POSITION 2,0:? "
        MEMORY STARTING AT ";M
IO 670  FOR PK=1 TO 128:BUF$(PK)=CHR$(PEEK(M+PK
        -1)):NEXT PK:GOTO 735
LF 700  IF WH=4 THEN FOR SEC=SEC1 TO SEC1+SEC2-
        1
JP 720  BUF$(128)=" ":FUN=82:REM GET SECTOR
OD 730  GOSUB 8010
HN 735  REM HOLD PLACE FOR FUTURE DISASSEMBLER
        LINK
AJ 740  ON TY GOSUB 1040,4040,5015,4040
DJ 765  IF WH=10 THEN 840
DP 770  IF WH<>2 THEN 800
IL 780  ? :? "DIRECTORY ENTRY BYTES:":? :? "STA
        RT+0=FLAG":? "(66/NORMAL;128/DELETED;98
        /LOCKED;etc)"
DJ 790  ? "START+1,2=SECTOR COUNT LO/HI":? "STA
        RT+3,4=STARTING SECTOR LO/HI"
HI 800  GOSUB 7010:REM GET FILE # AND FORWARD P
        OINTER (FN AND FP)
CJ 810  POSITION 2,20:? "FILE NO. = ";FN:POSITI
        ON 2,21:? "FORWARD POINTER TO SECTOR ";
        FP
LL 820  POSITION 2,22:? "EXTRA SECTOR BYTES = "
        ;125-ASC(BUF$(128))
LO 840  IF PEEK(1791)=255 THEN GOSUB 3000
DC 845  IF AP=1 THEN GOTO 920
BC 850  POSITION 2,23:? "ANY KEY TO CONTINUE";:
        GET #2,P:GOSUB 1105
PD 920  IF WH=3 AND FP=0 THEN POSITION 2,23:? B
        LA$;:POSITION 2,23:? "END OF TRACE";:AP
        =2:WH=1:GOTO 850
PB 930  IF AP<>1 THEN POSITION 2,21:? BLA$:POSI
        TION 2,21:? "PRESS ANY KEY TO CONTINUE"
        ;:GET #2,P
CI 935  IF WH=10 THEN NEXT M
CE 940  IF WH=3 THEN SEC=FP:GOTO 720
IG 950  IF WH=4 THEN NEXT SEC
IN 1000 GOTO 300
IM 1040 IF WH=10 THEN 1047
```

```
JC 1045 POSITION 0,0:? CHR$(125);"SECTOR #";SE
        C;"  (BEEP=[RETURN] CODE AT [K])":?
PL 1047 POKE 766,1
MJ 1060 ? "BYTE","0 2 4 6 8 A C E(3 SPACES)SST
         LTH":? :FOR I=1 TO 8
NM 1065 ? 16*(I-1);"-";16*I-1,:FOR J=0 TO 15:K
        =16*(I-1)+1+J
DL 1067 IF ASC(BUF$(K,K))=155 THEN POKE 766,0:
        ? CHR$(253);"[K]";:POKE 766,1:GOTO 1070
FD 1069 ? BUF$(K,K);
FB 1070 NEXT J
BC 1071 IF SEC<361 OR SEC>368 THEN ? :GOTO 107
        5
KF 1072 POSITION 30,3+I:? 256*ASC(BUF$(16*(I-1
        )+5))+ASC(BUF$(16*(I-1)+4))
KH 1073 POSITION 35,3+I:? 256*ASC(BUF$(16*(I-1
        )+3))+ASC(BUF$(16*(I-1)+2))
LN 1075 NEXT I:POKE 766,0
LA 1078 RETURN
OH 1103 IF PEEK(1791)=255 THEN GOSUB 3000
MF 1105 IF WH=10 THEN POP :GOTO 935
IH 1106 FOR I=0 TO 3:POSITION 2,20+I:? BLA$;:N
        EXT I
DA 1107 GOSUB 1200:? "CHANGE SECTOR (Y/N)";:IN
        PUT G$:IF G$<>"Y" THEN RETURN
NP 1110 GOSUB 1200:? "INSERT # (N) OR CHARACTE
        RS (C)";:INPUT G$
IC 1112 IF G$="N" THEN 1130
LE 1114 IF G$<>"C" THEN 1110
NC 1116 H$=" ":GOSUB 1200:? "HOW MANY CHARACTE
        RS TO CHANGE";:INPUT CN
MO 1118 GOSUB 1200:? "START AT WHICH BYTE";:IN
        PUT BYTE
EL 1120 GOSUB 1200:? "TYPE ";CN;" CHARACTERS T
        O BE INSERTED..":INPUT H$
OE 1122 BYTE=BYTE+1:BUF$(BYTE,BYTE+CN-1)=H$:GO
        TO 1150
PA 1130 GOSUB 1200:? "HOW MANY BYTES TO SAME #
         ";:INPUT BN
LH 1142 GOSUB 1200:? "BYTE # TO BEGIN MOD. ";:
        INPUT BYTE:B=BYTE+1
KI 1144 GOSUB 1200:? "OLD BYTE ";BYTE;"=";ASC(
        BUF$(B,B));
CF 1146 ? " || NEW BYTE ";BYTE;"=";:INPUT NB
KG 1148 FOR I=B TO B+BN-1:BUF$(I,I)=CHR$(NB):N
        EXT I
MC 1150 G$=" ":GOSUB 1200:? "TYPE 'Y' TO MODIF
        Y SECTOR";:INPUT G$:IF G$<>"Y" THEN 11
        05
BF 1160 FUN=87:GOSUB 8010:POP :GOTO 720
```

```
OP 12ØØ POSITION 2,21:? BLA$:POSITION 2,21:RET
       URN
OB 14ØØ POSITION Ø,Ø:? CHR$(125);:? :? "COPY W
       HICH SECTOR";:INPUT SEC:? :? "INSERT S
       OURCE DISK, HIT ANY KEY"
DH 141Ø GET #2,P:BUF$(128)=" ":FUN=82:GOSUB 8Ø
       1Ø:SECT=SEC:? :? "SAVE TO WHAT DEST. S
       ECTOR";:INPUT SEC
JA 143Ø ? :? "INSERT DEST. DISK, HIT ANY KEY":
       GET #2,P:FUN=87:GOSUB 8Ø1Ø:LK=1:SS=SEC
       :NSL=1:GOSUB 171Ø
MM 144Ø ? :? "OLD SECTOR ";SECT;:? " SAVED AS
       NEW SEC ";SEC:? :? "ANY KEY FOR MENU":
       GET #2,P:GOTO 3ØØ
DK 16ØØ POSITION Ø,Ø:? CHR$(125);:? :? "INSERT
        DISK W/DIR. TO SAVE":? "AND HIT ANY K
       EY TO CONTINUE":GET #2,P
EH 161Ø FUN=82:BUF$(128)=" ":FOR I=Ø TO 8:SEC=
       36Ø+I:GOSUB 8Ø1Ø:N=128*I+1:H$(N,N+127)
       =BUF$:NEXT I
NM 162Ø ? :? "STARTING SECTOR TO SAVE DIR. AT"
       ;:INPUT SS:? :? "INSERT DEST. DISK, PU
       SH ANY KEY":GET #2,P
BC 163Ø FUN=87:FOR I=Ø TO 8:SEC=SS+I:N=128*I+1
       :BUF$=H$(N,N+127):GOSUB 8Ø1Ø:NEXT I:LK
       =1:NSL=9:GOSUB 171Ø
OM 164Ø ? :? "DIR SAVED ON SEC ";SS;" TO ";SS+
       8:? :? "ANY KEY FOR MENU":GET #2,P:GOT
       O 3ØØ
FC 17ØØ POSITION Ø,Ø:? CHR$(125):? "1=LOCK":?
       "2=UNLOCK":? :INPUT LK
EO 1705 ? :? :? "START AT SECTOR";:INPUT SS:?
       :? "HOW MANY SECTORS";:INPUT NSL
DE 171Ø SEC=36Ø:FOR I=Ø TO NSL-1:BUF$(128)=" "
       :FUN=82:GOSUB 8Ø1Ø
LN 172Ø S=SS+I:SSH=INT(S/8):SSL=S-8*SSH:BYTE=A
       SC(BUF$(11+SSH)):BITMOD=SSL+1:BIT=ASC(
       BIT$(BITMOD))
CK 1722 Q=USR(ADR(ND$),BIT,BYTE):IF (LK=2 AND
       PEEK(2Ø9)>Ø) OR (LK=1 AND PEEK(2Ø9)=Ø)
        THEN 1795:REM ALREADY RIGHT BIT
BF 1725 IF LK=2 THEN Q=USR(ADR(OAR$),BIT,BYTE)
       :RSLT=PEEK(2Ø9):GOTO 1735
FL 173Ø Q=USR(ADR(KNOT$),BIT,255):RSLT=PEEK(2Ø
       9):Q=USR(ADR(ND$),RSLT,BYTE):RSLT=PEEK
       (2Ø9)
JE 1735 BUF$(11+SSH)=CHR$(RSLT):NFRE=ASC(BUF$(
       4))+256*ASC(BUF$(5)):IF LK=1 THEN NFRE
       =NFRE-1:GOTO 1745
```

```
LL 1740 NFRE=NFRE+1
DE 1745 NFREH=INT(NFRE/256):NFREL=NFRE-256*NFR
        EH
BM 1750 BUF$(4)=CHR$(NFREL):BUF$(5)=CHR$(NFREH
        )
OJ 1790 FUN=87:GOSUB 8010
PO 1795 NEXT I:IF WH=7 OR WH=8 THEN RETURN
KG 1799 GOTO 300
ND 1800 BUF$(128)=" ":N=0:POSITION 0,0:? CHR$(
        125):? "FILE STARTS AT SECTOR";:INPUT
        SEC:POKE 82,1:POKE 201,5:?
ND 1810 FUN=82:GOSUB 8010:GOSUB 7010:N=N+1:IF
        N=9 THEN N=1:? :? "{UP}";
PL 1820 ? SEC,:SEC=FP:IF FP<>0 THEN 1810
CL 1830 POKE 82,2:POKE 201,10:? :? :? "END OF
        MAP...":IF PEEK(1791)=255 THEN GOSUB 3
        000
LP 1840 POSITION 2,23:? "ANY KEY";:GET #2,P:GO
        TO 300
GG 2010 ? CHR$(125):SEC=0:FUN=82:BUF$(128)=" "
        :POSITION 1,23:? "ANY KEY TO TERMINATE
        ";
EN 2020 POKE 764,255:POKE 766,1:FOR J=0 TO 23:
        POSITION 1,J:? J*30+1;"-";(J+1)*30;
FE 2025 POSITION 9,J:FOR I=1 TO 30:SEC=SEC+1
HO 2030 IF PEEK(764)=255 THEN GOSUB 8010:GOSUB
         7010:GOTO 2038
OP 2035 POKE 764,255:POP :POP :GOTO 2150
PE 2038 IF NR<>1 THEN ? "*";:GOTO 2100
OC 2040 IF FN<10 THEN ? CHR$(FN+48);:GOTO 2100
LC 2050 IF FN>9 THEN IF FN<36 THEN ? CHR$(FN+5
        5);:GOTO 2100
LC 2060 IF FN>35 THEN ? CHR$(FN+61);
AO 2100 NEXT I:NEXT J
OK 2150 IF PEEK(1791)=255 THEN GOSUB 3000
OC 2160 POSITION 2,23:? BLA$;:POSITION 1,23:?
        "ANY KEY";
OD 2200 POKE 766,0:GET #2,P:GOTO 300
IA 3000 POKE 764,255:POKE 766,1:TRAP 3060:CLOS
        E #3:OPEN #3,8,0,"P:"
IF 3020 FOR J=0 TO LM:FOR I=0 TO 39
OC 3030 POSITION I,J:GET #4,K
EH 3035 IF K>127 THEN K=K-128
PB 3036 IF K<32 OR K=127 THEN ? #3;"*";:GOTO 3
        045
GP 3040 ? #3;CHR$(K);
BD 3045 NEXT I:LPRINT :IF PEEK(764)=255 THEN N
        EXT J:GOTO 3050
MK 3047 POKE 764,255:POP :GOTO 3050
MH 3050 LPRINT :LPRINT :LPRINT
```

```
HG 3060 POKE 766,0:TRAP 40000:RETURN
EB 4040 FOR I=1 TO 382 STEP 3:H=ASC(BUF$(INT(I
        /3)+1)):HH=INT(H/16):HL=H-16*HH:HH=HH+
        1:HL=HL+1

DB 4050 H$(I,I)=HEX$(HH,HH):H$(I+1,I+1)=HEX$(H
        L,HL):H$(I+2,I+2)=" ":NEXT I
DO 4060 IF WH=10 THEN ? :GOTO 4210
CD 4205 ? CHR$(125):? :? "SECTOR ";SEC;" HEX D
        UMP":?
MG 4210 FOR I=1 TO 16:? 8*(I-1);"-";8*I-1,H$(2
        4*(I-1)+1,24*I):NEXT I
GJ 4220 IF TY=4 THEN J=47:FOR I=5 TO 16:POSITI
        ON 36,I:? J;:J=J+64:NEXT I
KK 4240 RETURN
IK 5015 POKE 82,1:POKE 201,5:REM COLUMN WIDTH
MN 5017 IF WH=10 THEN GOTO 5025
DG 5020 POSITION 2,0:? CHR$(125);"SECTOR ";SEC
        ;" DECIMAL DUMP"
PD 5025 ? :? ,:FOR I=0 TO 7:? I,CHR$(30);:NEXT
         I:? :?
GP 5030 FOR I=1 TO 16:? 8*(I-1),
FN 5040 FOR J=1 TO 8:? ASC(BUF$(8*(I-1)+J)),CH
        R$(30);:NEXT J:?
AM 5050 NEXT I:POKE 201,10:POKE 82,2
KL 5060 RETURN
IH 7010 FP=ASC(BUF$(126)):Q=USR(ADR(ND$),FP,3)
        :FP=PEEK(209)*256:FP=FP+ASC(BUF$(127))
GE 7050 FN=ASC(BUF$(126)):Q=USR(ADR(ND$),FN,25
        2):FN=INT(FN/4)
KI 7100 RETURN
HM 8010 BLK=768:REM START OF DISK CONTROL BLOC
        K
DP 8020 POKE BLK+1,1:POKE BLK+2,FUN
PG 8040 ABUF=ADR(BUF$):AHI=INT(ABUF/256):ALO=A
        BUF-256*AHI
NM 8050 POKE BLK+4,ALO:POKE BLK+5,AHI
IC 8060 SECHI=INT(SEC/256):SECLO=SEC-256*SECHI
GM 8070 POKE BLK+10,SECLO:POKE BLK+11,SECHI
II 8080 Q=USR(ADR(DSKINV$)):NR=PEEK(BLK+3)
GN 8085 IF WH=5 THEN 8100
GL 8086 IF NR=1 THEN 8100
NN 8090 POSITION 0,0:? CHR$(125):? :? :? "SECT
        OR ";SEC;" = BAD SECTOR":IF AP=1 THEN
        8100
MF 8095 ? :? "ANY KEY TO CONTINUE":GET #2,P:PO
        P :GOTO 300
KJ 8100 RETURN
```

Appendix

How to Type In Programs

Appendix

How to Type In Programs

In order to make special characters, inverse video, and cursor characters easy to type in, we use the following listing conventions for all the programs in this book. Please refer to the table and explanations if you come across an unusual symbol in a program listing.

Conventions

Characters in inverse video will appear like: INVERSE VIDEO
Enter these characters with the Atari key.

When you see	Type	See
{CLEAR}	ESC SHIFT <	Clear Screen
{UP}	ESC CTRL -	Cursor Up
{DOWN}	ESC CTRL =	Cursor Down
{LEFT}	ESC CTRL +	Cursor Left
{RIGHT}	ESC CTRL *	Cursor Right
{BACK S}	ESC DELETE	Backspace
{DELETE}	ESC CTRL DELETE	Delete Character
{INSERT}	ESC CTRL INSERT	Insert Character
{DEL LINE}	ESC SHIFT DELETE	Delete Line
{INS LINE}	ESC SHIFT INSERT	Insert Line
{TAB}	ESC TAB	TAB key
{CLR TAB}	ESC CTRL TAB	Clear TAB
{SET TAB}	ESC SHIFT TAB	Set TAB stop
{BELL}	ESC CTRL 2	Ring Buzzer
{ESC}	ESC ESC	ESCape key

Graphics characters, such as CTRL-T, the ball character, will appear as the normal letter enclosed in braces, {T}.

A series of identical control characters, such as 10 spaces, 3 cursor lefts, or 20 CTRL-Rs, will appear as {10 SPACES}, {3 LEFT}, {20 R}, etc. If the character in braces is in inverse video, that character or characters should be entered with the Atari key.

Program entry can be mistake-proof if you use "The Automatic Proofreader" by Charles Brannon; see Chapter 1.

Index

Notes

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!** Magazine. Use this form to order your subscription to **COMPUTE!**.

For Fastest Service
Call Our **Toll-Free** US Order Line
800-334-0868
In NC call 919-275-9809

COMPUTE!
P.O. Box 5406
Greensboro, NC 27403

My computer is:
☐ Commodore 64 ☐ TI-99/4A ☐ Timex/Sinclair ☐ VIC-20 ☐ PET
☐ Radio Shack Color Computer ☐ Apple ☐ Atari ☐ Other _____
☐ Don't yet have one...

☐ $24 One Year US Subscription
☐ $45 Two Year US Subscription
☐ $65 Three Year US Subscription

Subscription rates outside the US:

☐ $30 Canada
☐ $42 Europe, Australia, New Zeland/Air Delivery
☐ $52 Middle East, North Africa, Central America/Air Mail
☐ $72 Elsewhere/Air Mail
☐ $30 International Surface Mail (lengthy, unreliable delivery)

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Payment must be in US funds drawn on a US bank, international money order, or charge card.

☐ Payment Enclosed ☐ Visa
☐ MasterCard ☐ American Express

Acct. No. _____ Expires ____/____

Your subscription will begin with the next available issue. Please allow 4–6 weeks for delivery of first issue. Subscription prices subject to change at any time.

751199

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **800-334-0868** (in NC 919-275-9809) or write COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403.

Quantity	Title	Price*	Total
_____	Machine Language for Beginners	**$14.95**	_____
_____	The Second Book of Machine Language	**$14.95**	_____
_____	COMPUTE!'s Guide to Adventure Games	**$12.95**	_____
_____	Computing Together: A Parents & Teachers Guide to Computing with Young Children	**$12.95**	_____
_____	Personal Telecomputing	**$12.95**	_____
_____	BASIC Programs for Small Computers	**$12.95**	_____
_____	Programmer's Reference Guide to the Color Computer	**$12.95**	_____
_____	Home Energy Applications	**$14.95**	_____
_____	Creating Arcade Games on the Timex/Sinclair	**$12.95**	_____
_____	COMPUTE!'s First Book of PET/CBM	**$12.95**	_____
_____	Commodore Peripherals: A User's Guide	**$ 9.95**	_____
_____	All About the Commodore 64, Volume 1	**$12.95**	_____
_____	COMPUTE!'s Commodore Collection, Volume 1	**$12.95**	_____
_____	Every Kid's First Book of Robots and Computers	**$ 4.95†**	_____
_____	The Beginner's Guide to Buying a Personal Computer	**$ 3.95†**	_____

* Add $2.00 per book for shipping and handling.
† Add $1.00 per book for shipping and handling.
Outside US add $5.00 air mail or $2.00 surface mail.

Shipping & handling: $2.00/book _____
Total payment _____

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.
NC residents add 4.5% sales tax.
☐ Payment enclosed.
Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No._____ Exp. Date_____

Name_____

Address_____

City_____ State _____ Zip_____

*Allow 4–5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

COMPUTE! Books

P.O. Box 5406 Greensboro, NC 27403

Ask your retailer for these **COMPUTE! Books**. If he or she has sold out, order directly from **COMPUTE!**.

For Fastest Service
Call Our **TOLL FREE US Order Line**
800-334-0868
In NC call 919-275-9809

Quantity	Title	Price	Total
_____	COMPUTE!'s First Book of Atari	**$12.95**	_____
_____	COMPUTE!'s Second Book of Atari	**$12.95**	_____
_____	COMPUTE!'s Third Book of Atari	**$12.95**	_____
_____	COMPUTE!'s First Book of Atari Graphics	**$12.95**	_____
_____	COMPUTE!'s Second Book of Atari Graphics	**$12.95**	_____
_____	Mapping the Atari	**$14.95**	_____
_____	COMPUTE!'s First Book of Atari Games	**$12.95**	_____
_____	The Atari BASIC Source Book	**$12.95**	_____
_____	Inside Atari DOS	**$19.95**	_____
_____	Machine Language for Beginners	**$14.95**	_____
_____	Second Book of Machine Language	**$14.95**	_____
_____	Computing Together: A Parent and Teacher's Guide to Using Computers with Young Children	**$12.95**	_____
_____	Personal Telecomputing	**$12.95**	_____
_____	Home Energy Applications on Your Personal Computer	**$14.95**	_____

Add $2.00 shipping and handling. Outside US add $5.00 air mail or $2.00 surface mail.

Please add shipping & handling for each book ordered. _____

Total enclosed or to be charged _____

All orders must be prepaid (money order, check, or charge). All payments must be in US funds. NC residents add 4½% sales tax.
☐ Payment enclosed Please charge my: ☐ Visa ☐ MasterCard
☐ American Express Acct. No. Expires

_____

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Allow 4–5 weeks for delivery.
Prices and availability subject to change without notice.

Whether you've had an Atari for years, or just bought one of the newer XL models, you'll find *COMPUTE!'s Atari Collection, Volume 1* full of useful, never-before-published articles and programs.

There are games and educational programs to entertain and challenge, tutorials that illustrate BASIC programming techniques, and applications that turn your computer into a management tool.

The editors of *COMPUTE!* have collected the best original programs and articles from more than two dozen experienced Atari programmers. High-quality programs and clearly written articles—both hallmarks of COMPUTE! Publications—will show you just what your Atari can do.

Here's a sample of what's inside:

- "Nessie," a nonviolent game where you try to photograph the elusive Loch Ness monster.
- Utilities which make it easy to catalog disks or locate programs on tape.
- "Reversi" and "Memory Match," thinking games that probe your logic while you have fun.
- "Diskovery," to help you explore your disks, sector by sector.
- Programs for children that teach and entertain.
- Editors to help you create sounds and songs for that special program.
- "Investment Tracker," a data base program that stores and analyzes your investments.
- Joystick routines you can add to any program.
- An introduction to PEEK and POKE, with numerous practical examples.
- And "The Automatic Proofreader," which helps you correctly type in programs.

It's been almost a year since COMPUTE! Publications released a new Atari book. Now, with its all-original programs and articles, and with something for both experienced and beginning Atari users, *COMPUTE!'s Atari Collection, Volume 1* offers a new and detailed look at your favorite computer's power and flexibility.

ISBN 0-942386-79-5