THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS

ANALOG COMPUTING

T.M.

JUNE 1989
ISSUE 73

DISK VERSION $12.95

SPECIAL ADVENTURE ISSUE

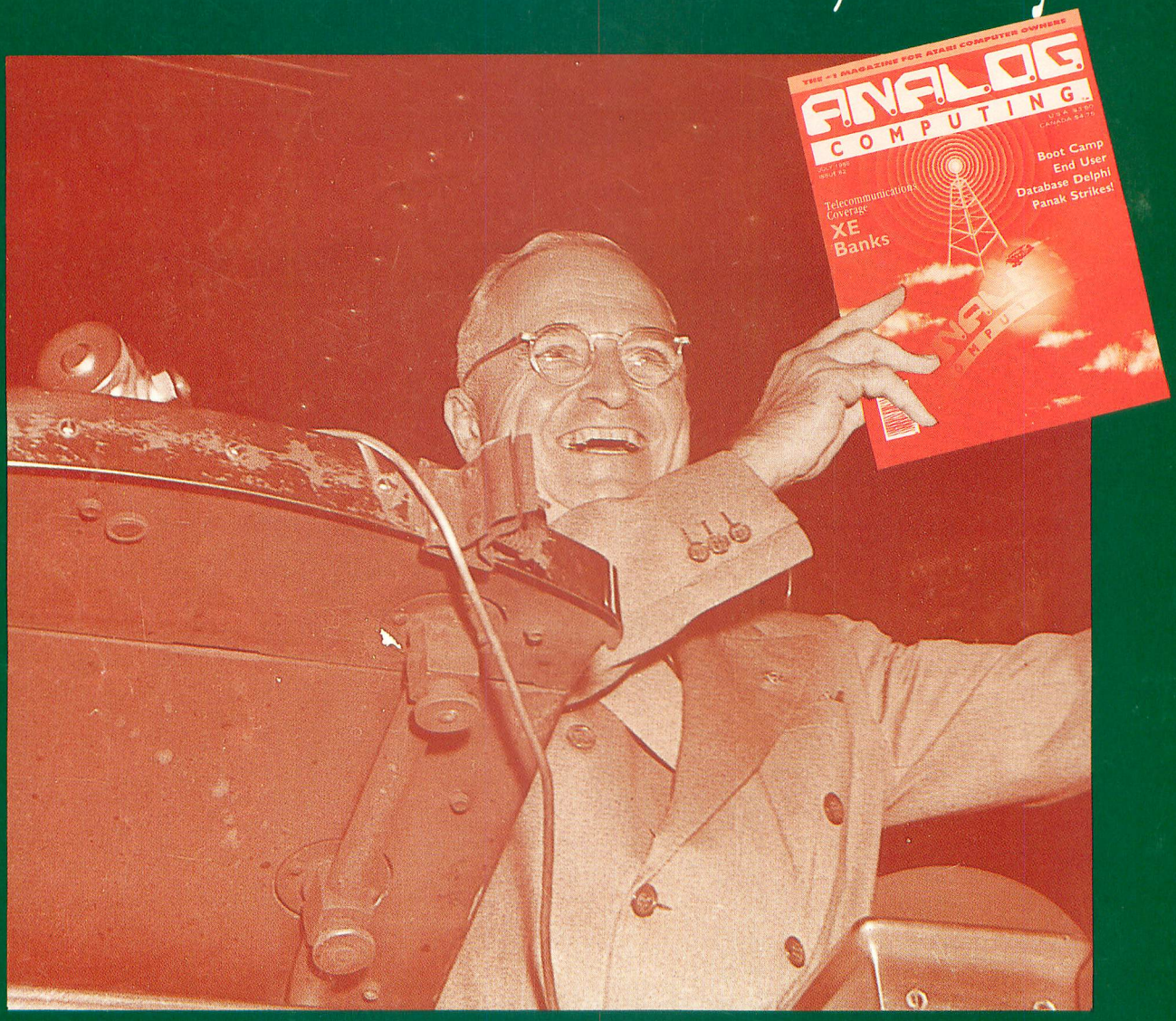
REVIEWS:
MARIO BROS.
DESERT FALCON

ADVENTURE GAME SHOWDOWN
SECRET AGENT
ACCESSING HIDDEN XL MEMORY

Type-in Software Inside!

06
74369 50077 8

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



Two Historic Facts:

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

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

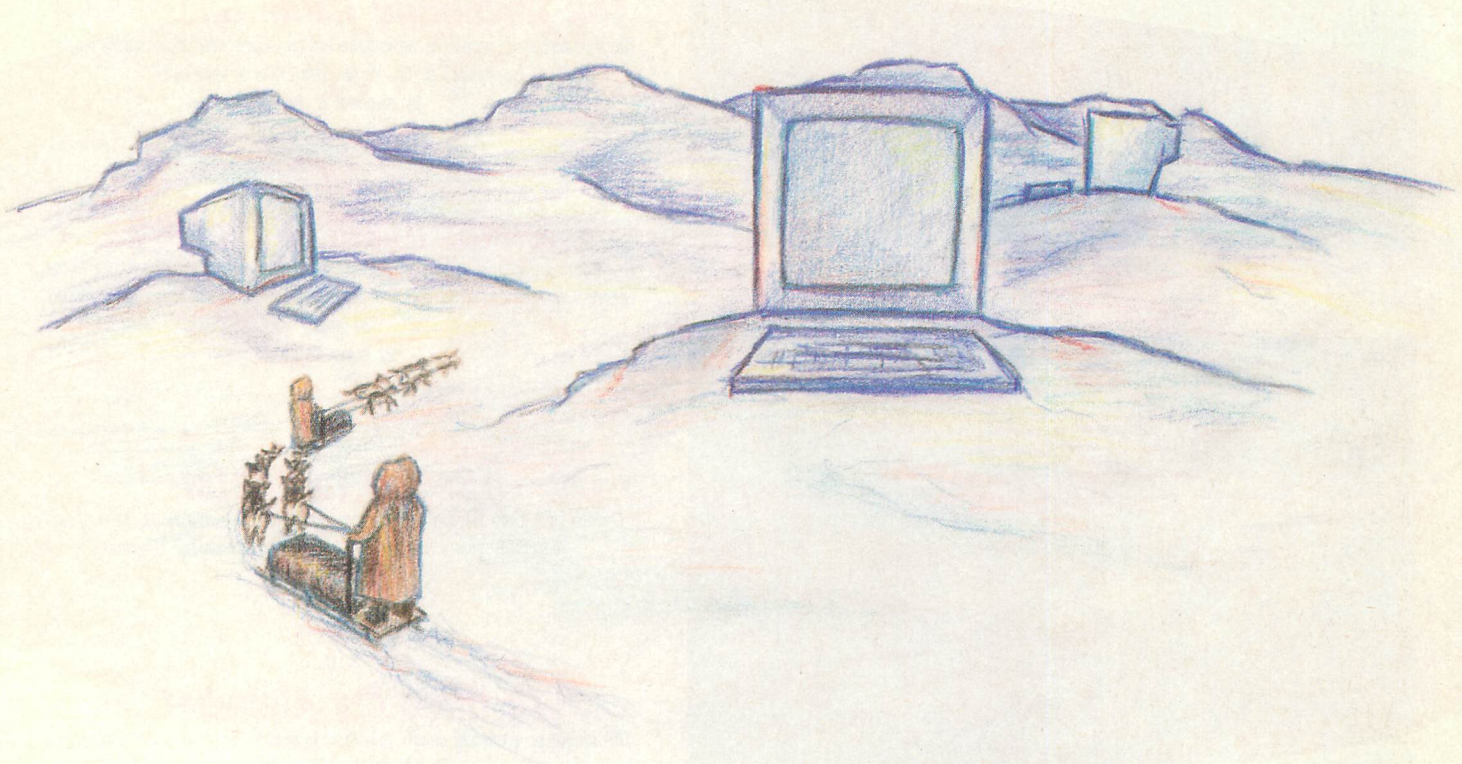
1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE

Editorial



by Clayton Walnum

Adventure games have been with us for a long time, and over the years they have gone through many changes. While some of those changes have increased the quality of the adventure experience, others have been the subject of constant debate and controversy.

In the beginning, there were text adventures that accepted only simple two-word, verb/noun commands. Soon, however, clever programmers expanded the intelligence of adventure parsers (the section of the program that translates for the computer the text typed in by the player) so that they could understand full sentences. Instead of typing a series of commands like "GET KNIFE," "CUT ROPE" and "GO NORTH," suddenly we had the capability to enter "GET THE KNIFE, CUT THE ROPE AND GO NORTH."

But even with that kind of flexibility, players were not satisfied. They wanted not only words, but pictures too. Soon, text adventures were developed that, though they played exactly like the games that came previously, included a graphics window above the text, displaying a static picture of the current "room."

The key word in the previous sentence is "static." Except for perhaps providing a clue or two, the graphics were not an interactive part of the adventure; all the player could do was look. But you can't stop progress, and it wasn't long before games employed graphics in a more sophisticated way, allowing players to "pick up" certain objects on the screen.

Still, adventure creators and players remained unsatisfied. Luckily, computer animation techniques had become very sophisticated; player-controlled "movies" could now be created, which gave birth to the animated adventure games we're so familiar with today, most of which incorporate little text.

Has the adventure-game industry moved forward? Or are the modern, animated graphics adventures just more cheap entertainment for a lazy society brought up on TV and fast food? It's an interesting question and one that is tackled in this issue by adventure and science fiction author Michael A. Banks. Fans of any type of adventure game will enjoy Mike's thought-provoking article "The Adventure Game Showdown."

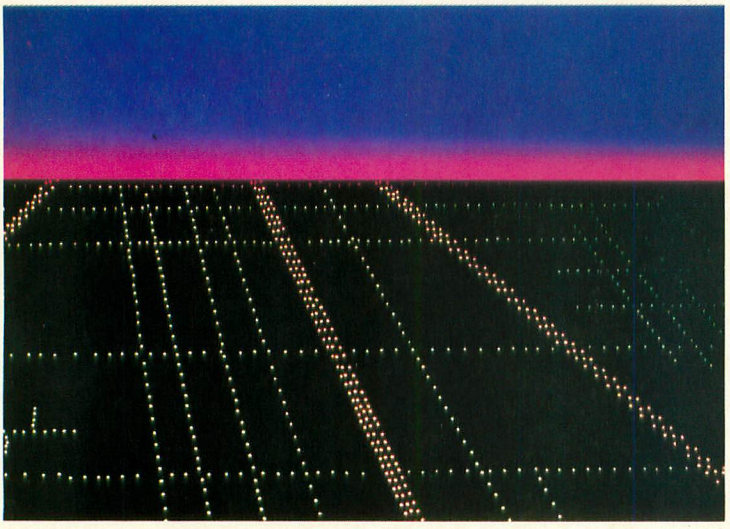
Also in this issue, you'll find the first ANALOG text adventure that will accept complete sentences as input. Don't miss Barry Kolbe's 100% machine-language spy story, "Secret Agent: Mission 1."

Of course, this issue is not dedicated only to adventure games, but to the entire Atari computing adventure. For example, fans of brainteasers are sure to be delighted with Earl Hill's "Marble Magic," a computer version of a popular puzzle game. Craig J. Stadler's "Disk Directory Alphabetizer" is a handy utility that'll take the drudgery out of finding a particular file on a crowded disk.

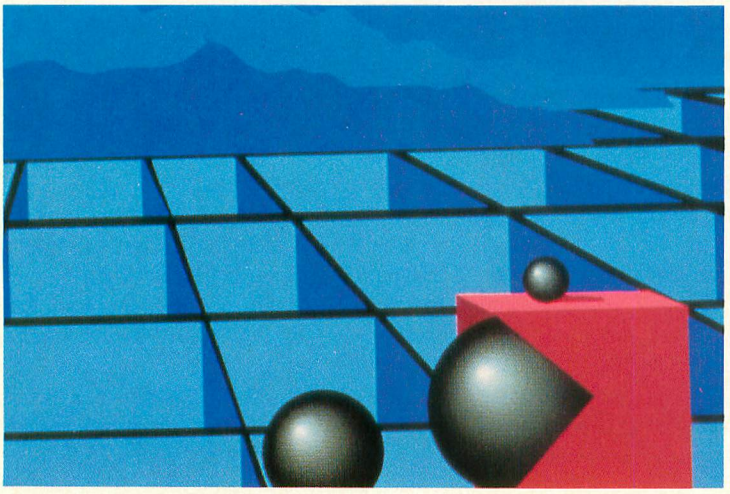
Adventures in programming continue this month with Kevin T. Pate's "Accessing Atari XL Hidden Memory," as well as with ANALOG's regular columns *Boot Camp* and *BASIC Training*.

As usual, we have something for everyone. Just turn the page and see.

on page 10



on page 18



on page 74

Cont

FEATURES

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. U.S. subscription: $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscription: Add $10 per year. Single copy $3.50 (add $1 for postage). Change of address: Six weeks advance notice, and both old and new addresses are needed. POSTMASTER: Send change of address to **ANALOG Computing** Magazine, P.O. Box 16927, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

e n t s

Screen photo on cover: Desert Falcon

JUNE 1989

ISSUE 73

Where to Write

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

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

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

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

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

Advertising Sales

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

Permissions

Subscriptions

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

Authors

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

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

ANALOG COMPUTING STAFF

READER COMMENT

Panak Stricken

I feel that the 8-bit Ataris are the finest computers around for the paltry sum asked, and the best support for the Atari is ANALOG Computing. It's unfortunate then that such a winning combination should continue to support the drivel spouted by Steve Panak every month.

For example, he gets himself in trouble with Micromiser and, when his errors are pointed out, proclaims that his remarks do not control the entire industry. But we're not talking about the entire industry, are we? We're talking about a small slice of the industry which *is* influenced by Mr. Panak's gloom-and-doom column. I, for one, find his contempt for the "lowly" 8-bit line completely tasteless.

I suggest that, if Mr. Panak wishes to continue writing, he write for magazines other than the one that supports the 8-bit Atari. Perhaps he should avoid the Atari completely—assuming he finds the ST as backward and contemptible.

I'm sure there are dozens, if not hundreds, of readers who would not only buy their software, but review it in an objective manner without criticizing the hardware on which it runs.

—Mark Gamber
Lancaster, PA

Ahem. I don't suppose you'll be sending Steve a Christmas card this year. Seriously, though, Steve certainly holds no contempt for Atari computers. After all, he's been writing for the Atari community for many years and has, in fact, reviewed more software for ANALOG *than any other single contributor.*

But just as people have very differing opinions about which computer is best, so do they have differing opinions about which computers are best for which applications. If Steve prefers a different computer for his business applications, that's not so much a reflection on the hardware, but on the software that's available for the machine. The programs that Steve likes for business are not available on the Atari; it's as simple as that. I don't believe Steve has ever called an Atari computer contemptible.

By the way, Steve does own an ST and is currently reviewing many games on that machine. Why would he run out and buy an ST if he thought Atari machines were "backward"?

Snowplow Clarification

I am a recent subscriber to ANALOG, and I have an 800XL with an XF551 disk drive. I enjoyed your programs *Snowplow* and *Snowplow Editor* from the September and October 1988 issues. I made two extra maps with the editor and named these SMAP.2 and SMAP.3. I then added them to the back-up disk containing SNOWPLOW.OBJ and SMAP.1 from the October disk.

However, now when I play *Snowplow*, it cycles through the original game map, SMAP.1, and SMAP.2, but it skips SMAP.3 and reruns the original map, SMAP.1. What is the problem?

—John E. Basich
Greendale, WI

Bryan Schappel, co-author of Snowplow *has informed us that you need to use a two-digit extension on your map files. You should rename SMAP.2 to SMAP.002, SMAP.3 to SMAP.003, and so on. Sorry for the confusion.*

I would like to start an Atari farmer's and gardener's users' group. Anyone who is using an Atari 8-bit computer to help them with their gardening/farming is welcome to join. If there is enough interest, we will issue a disk of useful programs for gardeners, once we've accumulated them.

At the moment, the Atari 8-bit is the most cost-effective computer around. Let's get on the ball and see if we can apply it to this rewarding hobby. If anyone has software they have written and would like to share with us, or if anyone would like to help us get this project rolling, drop us a letter with a self-addressed, stamped envelope, and we'll let you know how it's going.

We are particularly interested in artificial intelligence applications for sorting plant nutrient requirements, programs to track nutrient usage and anything that would help with the task of growing food. Thank you very much for publishing this letter.

—Lee Jones
Gromor-Systems
Rt. 1, Box 76-B
Pleasantville, TN 37147-9801

A House Divided

A few months ago I purchased a used Atari 800XL with a disk drive. Having no previous Atari experience and no manuals for the computer, I turned to magazines as my only resource. I was very excited to see your magazine. I have recently sent in my subscription and am looking forward to reading your magazine every month. It has truly been a big help.

I've been following the plight of Atari for a much longer time. I have all of Atari's game machines except the XEGS (owning an XL computer covers that). It seems to me that the troubles Atari is undergoing are

rooted deep within Atari itself. Sure, there are many reasons why they've had trouble—their small ST sales in the U.S., for example. However, Atari seems to have a lot of internal problems, as well. Your editorial in the January '89 issue talks of the loss of Neil Harris. I've also heard of Atari losing other top people. Maybe Atari should work on fixing their internal troubles first; a house divided cannot stand. Atari has good enough equipment out there. They don't need to introduce anything new just yet.

Switching gears, I have a problem I was hoping you could help me with. I have a Brother EP-22 typewriter/printer with a standard 25-pin RS-232 interface. I'd like to use it with my 800XL, but I'm unsure how to do this. Does an Atari 850 interface have an RS-232 port that I could use, or will I need some other interface?

Thank you for your support, and keep up the good work. It's the best Atari 8-bit coverage! —Scott M. Standiford
Post Falls, ID

I doubt Atari is in any serious trouble. Their "problems" are perceptual, the result of a poor image in the U.S., not with the financial health of the company. You are right in your observation that Atari goes through a lot of executives. In fact, this phenomenon has become so well known it was dubbed the "Atari rotating-door syndrome." We at ANALOG *suspect that the reason behind this is the fact that Atari is a "family" company—never heard of a Tramiel being fired, have you?*

You are wrong, however, when you say that Atari doesn't have to introduce new products. In the highly volatile computer market, if you don't keep up, you're doomed.

As for your problem with the Brother printer: Yes, the Atari 850 interface has an RS-232 port—in fact, it has four *of them. But you'll have to find software that'll enable you to direct your output to the RS-232 port rather than to the standard parallel printer port. Want to use a telecommunications program as a word processor?*

Send your letters to:
ANALOG Computing
Reader Comment
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

Is a Picture Worth a Thousand Words? Text vs. Graphics: THE ADVENTURE GAME SHOWDOWN

by Michael A. Banks

Have you read a good book lately? If you're under 25, the probability is better than 50% that you haven't. It's a sad-but-true fact that literacy is on the decline in our country. Sad, because literacy—the ability to access information as a solo effort, with minimal graphic support and without dedicated technological enhancement (a/k/a *reading*)—is one of the cornerstones of Western culture. Unfortunately, most students leave the American public school system with minimal reading skills. And many otherwise-intelligent people who *can* read simply will not read a book because it takes too much effort.

While the electronic media is not solely responsible for this sad state of affairs, it can be argued that television, movies and computer gaming have helped lower the level of literacy by discouraging the reading of books. The connection is simple: Television and films are so much *easier* to use than books. Novels force you to create your own images, to concentrate on what's going on—to *think*. (So do radio dramas, incidentally.) TV, films and the vast majority of computer games, on the other hand, require nothing more than open eyes and ears (and with games, a little finger and wrist movement). Open wide and let someone else's images pour in. And, being human, most of us tend to take the path of least resistance—even in seeking entertainment—perhaps ignorant of the fact that with greater effort come greater rewards.

This is not to say I don't like television or movies or arcade games (although, as a writer, I have a vested interest in the print media). I'm a card-carrying member of the so-called television generation, counting myself among that first group of millions of children to grow up watching TV. (I'm 38, and some of my earliest memories are of watching westerns on an old black-and-white set in 1954). My collection of movies on videotape—ranging from *Casablanca* to *Beverly Hills Cop* to *2001: A Space Odyssey*—is prolific and eclectic, to say the least. And, like most of you, I've wasted hundreds of hours at coin-op and computer arcade games.

My point—and it's an entirely philosophical, though pragmatic, point—is this: Though

television has its subjective and objective merits, there must be a balancing influence. Call it "moderation in all things—even in moderation," as Father Perrault taught in James Hilton's *The Lost Horizon*. As a culture, we're not achieving that moderation. We're pigging out on the easy information sources, the kind that force us to let others make our pictures (and, in effect, do our thinking) for us.

The effects are adverse, to say the least.

Saying It With Words

What, you may wonder, has all this to do with adventure gaming? Quite a lot. There are more than a few parallels between the movement of adventure gaming toward graphics and the displacement of books and radio by television and movies.

In the beginning, text adventure games used words—and words only. Early (*circa* 1980) personal computers such as the TRS-80 Model I and the Apple II couldn't handle anything else decently. They had their RAMs full, running a program that could parse simple verb/noun input; decide on, search out, and deliver canned responses from databases; and juggle behind-the-scenes flags as appropriate. Besides, those machines had poor graphics capabilities, and the addition of then

continued on page 80

SECRET AGENT: *Mission 1*

by Barry Kolbe

You glance at your watch for the sixth time. It is just past 1:30 a.m. Time to start work. Just two days ago was the first you'd heard of Dr. Moore. Nice guy, designer of a killer biological organism. CONTROL called and briefed you. There was not much information. Moore had been trailed to a lab—the lab you're standing in front of right now. Hmmm, suspicious, only one entrance just west of you. The report said there was a vial containing the organism and three forms of data relating to the organism. One form was most likely a formula. As to the others, it's anyone's guess. Your job: Steal it before Moore can sneak it out of the country and sell it to our enemies.

There doesn't seem to be an alarm system, and the front door was easy to pick. Too easy. You wonder what it's like inside. They must have some security measures. It looks very high tech. Well, here goes. The door is open now. You're going inside. The door is swinging shut—slowly. What's that? *Wham*! A steel panel dropped from the ceiling and is blocking the exit. Now how do you get out? Well, that can wait. First you'd better find that organism.

Typing in Secret Agent

Listing 1 is the data used to create your copy of *Secret Agent*. Please refer to the M/L Editor, on page 61 for typing instructions. *Secret Agent* is a machine-language game and is loaded by using the binary load option of your DOS.

How to Play Secret Agent

To carry out your mission, you type in commands. There are two types of commands: single letters and sentences. The single letter commands are: N, S, E, W, U and D. Use these commands to move from room to room. Other single-letter commands include "L" to look at a room, "I" to list the objects you are carrying, "X" to save the game at your current location and "R" to load a saved game.

Sentences may be typed in the format shown in Figure 1. The parentheses indicate optional words and the square brackets indicate a prepositional phrase, which is needed at times. It is up to you to discover the verbs, nouns and prepositions. No spaces are allowed at the beginning of a command.

Some command examples:

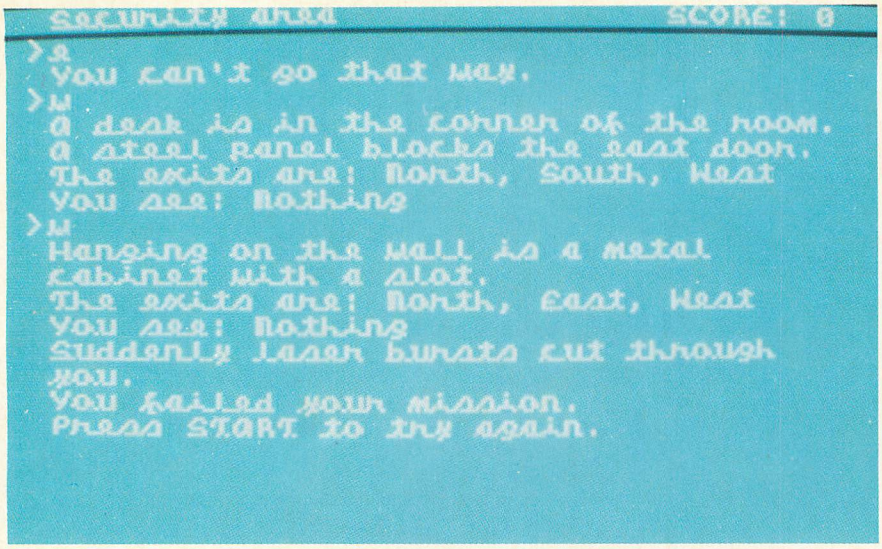
```
S
OPEN THE DOOR
OPEN THE DOOR WITH THE FANCY
KEY
```

Besides the regular alpha keys, only three other keys are accepted: CapsLock, Shift-Clear and Delete-Backspace. Caps allows you to toggle between lower- and upper-case. Do not press Shift with Caps! Shift-Clear clears the screen. Delete-Backspace is the only input editing key. One additional note: Only the first three letters of nouns and verbs are used in interpreting your command. Other words like "with" and "the" must be typed completely or an error will be given.

If your command is not valid (mostly because the verb or noun is not in the dictionary), or there is a misspelled word, the message "What?" will appear. Other error messages include "Not yet," which means you are on the right train of thought; "You can't do that," which indicates an action that can never be done, or at least not in the room you're in; and "You haven't got that," which means you are missing the object you tried to use.

The name of each room that you enter is displayed on the top line of the screen. At the extreme right is your score. When it is 4, it is time to leave the lab. Every time you enter a room, you will be given a description of the room, a list of exits and a list of visible objects. Only the visible objects can be picked up (example: GET KEY). However,

Illustration by · ALAN HUNTER



you may be able to do something with things mentioned in the description.

Technical Notes

This program has a nice text-compression routine that Bryan Schappel and I developed for use in adventure games or possibly for spelling dictionaries. First each text character is coded into a number from 0 to 31. Because there are more than 32 letters and punctuation marks there is a second set of 32 codes. The first code set is:

0 = end of message flag
1-26 = a-z (lowercase)
27 = end of text flag
28 = space
29 =
30 = capital letter flag
31 = flag for second set of 32

The second set of codes is:

0-18 = "?!,-:/()0123456789

Since the coded text uses only five of the eight bits available in a byte, it can be compressed. The coded text is slightly larger than the pure text because of capital letters taking two bytes for one. For example "A" is coded as 31,1. The compression routine then takes eight bytes of data and compresses them into five bytes. This results in quite a savings of memory. The text for *Secret Agent* was about 2800 bytes. It compressed to a little less than 1900 bytes. Of course, some of that gain was lost by having to incorporate the uncompression and decoding routines into the program.

The assembly listings (Listings 2 through 6) for *Secret Agent* do not contain the decompression or decoding routines. That is a separate file which I used from the DDT debugger in MAC/65. Getting the magazine version going was quite a chore. I had three separate binary files to interface: the character set data, the compressed text and the game itself.

First, I assembled the revised program with the decompressor and decoder built in. Then I BLOADed it into MAC/65. Next, I BSAVEd the game portion and the character set as separate files. I loaded in the compressor program and ran it to compress the text. Then the text had to be BSAVEd. So now I had the three separate files, plus I had to add a run address.

But how to get all four files together? I used DOS 2 to append the files to one another. But, whenever I tried to load in the whole file, the computer locked up. I finally unified the file and then it worked. I still don't know why it wouldn't work before unifying it.

What Is Next?

There are some obvious refinements that can be made to this type of adventure game. One is to allow multiple commands for input. Sound effects would be nice also. Adding more prepositions like "in," "on," "to," etc., would make for more interesting games.

Secret Agent was originally written in BASIC, and in that form it was 196 sectors long. It is now in machine language (minus sound effects) and only 61 sectors! The text compressor was a valuable tool for bringing down the size of this text adventure. I hope you enjoy playing it.

Barry Kolbe is a high school math teacher. He has been programming since 1981 on an Atari 800, and for the last four years, he has programmed mostly in 6502 assembly language. He is now studying GFA BASIC for his new ST, while perusing the C and 68000 assembly books at the bookstore.

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,128,255,129,0,0,0,
,0,0,0,0,0,0,24,4710
1010 DATA 24,24,24,0,24,0,0,102,102,10
02,0,0,0,0,0,102,5660
1020 DATA 255,102,102,255,102,0,24,62,
,96,60,6,124,24,0,0,102,8941
1030 DATA 102,24,48,102,70,0,28,54,28,
,56,111,102,59,0,0,24,7118
1040 DATA 24,24,0,0,0,0,0,14,28,24,24,
,28,14,0,0,112,4290
1050 DATA 56,24,24,56,112,0,0,102,60,2
255,60,102,0,0,0,24,8184
1060 DATA 24,126,24,24,0,0,0,0,0,0,0,2
24,24,48,0,0,2776
1070 DATA 0,126,0,0,0,0,0,0,0,0,0,24,2
24,0,0,6,2018
1080 DATA 12,24,48,96,64,0,0,60,102,11
10,118,102,60,0,0,24,8172
1090 DATA 56,24,24,24,126,0,0,60,102,1
12,24,48,126,0,0,126,8004
1100 DATA 12,24,12,102,60,0,0,12,28,60
0,108,126,12,0,0,126,7724
1110 DATA 96,124,6,102,60,0,0,60,96,12
24,102,102,60,0,0,126,9906
1120 DATA 6,12,24,48,48,0,0,60,102,60,
,102,102,60,0,0,60,7738
1130 DATA 102,62,6,12,56,0,0,0,24,24,0
0,24,24,0,0,0,2758
1140 DATA 24,24,0,24,24,48,6,12,24,48,
,24,12,6,0,0,0,3036
1150 DATA 126,0,0,126,0,0,96,48,24,12,
,24,48,96,0,0,60,6220
1160 DATA 102,12,24,0,24,0,0,60,102,11
10,110,96,62,0,0,30,7624
1170 DATA 55,103,103,111,59,0,0,30,51,
,115,126,115,127,0,0,30,9225
1180 DATA 51,96,96,112,63,0,0,60,102,9
99,99,99,126,0,0,30,9257
1190 DATA 51,96,124,96,127,0,0,30,51,9
96,124,96,96,0,0,30,8727
1200 DATA 51,96,110,99,62,0,0,99,99,99
9,127,99,99,0,0,127,1056
1210 DATA 24,24,24,24,127,0,0,3,3,3,11
15,54,60,0,0,103,6507
1220 DATA 110,124,124,108,111,0,0,112,
,112,96,96,99,127,0,0,99,1280
1230 DATA 99,119,127,107,99,0,0,124,11
18,118,118,118,119,0,0,28,814
1240 DATA 54,99,99,54,28,0,0,30,51,51,
,62,48,48,0,0,28,5684
1250 DATA 54,99,99,111,62,3,0,60,54,54
4,62,51,51,0,0,30,6514
1260 DATA 51,96,62,3,127,0,0,63,108,10
08,12,12,27,0,0,51,6335
1270 DATA 51,51,51,99,62,0,0,99,99,99,
,54,60,24,0,0,99,8165
1280 DATA 99,107,127,119,99,0,0,99,102
2,60,28,54,99,0,0,99,9082
1290 DATA 99,54,30,12,24,0,0,63,102,12
2,24,51,126,0,0,30,6291
1300 DATA 24,24,24,24,30,0,0,64,96,48,
,24,12,6,0,0,120,5952
1310 DATA 24,24,24,24,120,0,0,8,28,54,
,99,0,0,0,0,0,4095
1320 DATA 0,0,0,0,255,0,0,131,255,131,
,0,24,60,126,126,60,2930
1330 DATA 24,0,0,0,30,54,118,118,159,0
0,0,96,96,124,102,102,2327
1340 DATA 253,0,0,0,60,112,96,112,223,
,0,0,6,6,62,102,102,320
1350 DATA 255,0,0,0,28,54,54,28,247,0,
,0,28,56,48,62,123,9528
1360 DATA 217,28,0,0,30,51,115,222,135
5,60,0,96,96,96,124,102,3721
1370 DATA 231,0,0,24,0,24,24,126,195,0
0,0,28,0,28,60,111,8176
1380 DATA 205,60,0,48,48,62,54,60,247,
,0,0,24,24,24,24,24,7270
1390 DATA 231,0,0,0,51,127,127,219,219
9,0,0,0,124,102,102,102,3452
1400 DATA 231,0,0,0,60,102,103,231,60,
,0,0,0,124,102,102,124,2206
1410 DATA 231,96,0,0,62,102,102,126,14
43,14,0,0,96,126,102,102,2078
1420 DATA 195,0,0,0,14,27,51,99,223,0,
,0,12,63,12,12,28,6762
1430 DATA 247,0,0,0,115,51,51,51,223,0
0,0,0,54,54,54,127,9630
1440 DATA 205,0,0,0,99,99,107,127,221,
,0,0,0,51,126,110,219,4069
1450 DATA 177,0,0,0,27,27,31,54,207,24
4,0,0,0,102,91,219,973
1460 DATA 131,62,0,24,60,126,126,24,60
0,0,24,24,24,24,24,24,6425
```

```
1470 DATA 24,24,0,126,112,124,110,102,
,6,0,8,24,56,120,56,24,8998
1480 DATA 8,0,16,24,28,30,28,24,16,0,0
0,96,41,103,243,235,3016
1490 DATA 73,164,133,7,138,234,72,46,2
25,65,233,20,101,133,18,247,6359
1500 DATA 116,50,40,65,225,189,198,44,
,138,225,151,178,123,218,15,45,8978
1510 DATA 52,26,10,224,122,101,29,100,
,154,103,161,145,66,15,23,16,2355
1520 DATA 44,146,210,186,157,147,222,2
208,122,116,124,130,114,246,79,123,137
76
1530 DATA 65,224,185,45,11,59,39,189,1
160,245,203,58,116,174,39,94,7824
1540 DATA 110,160,44,101,195,144,121,1
1,99,46,28,131,195,9,139,66,4233
1550 DATA 201,33,7,167,64,166,87,43,24
4,15,60,198,72,202,15,13,2374
1560 DATA 5,106,70,22,119,204,8,129,23
36,124,146,14,123,230,4,64,6023
1570 DATA 240,222,216,86,133,151,124,1
192,136,30,130,31,71,247,204,8,8116
1580 DATA 129,226,176,163,164,158,228,
,142,125,243,2,32,122,109,11,25,5479
1590 DATA 217,61,237,7,169,87,56,172,7
7,134,19,148,30,115,229,68,5714
1600 DATA 124,125,7,166,250,209,31,31,
,65,226,134,116,248,250,15,92,9760
1610 DATA 179,167,199,208,122,176,248,
,250,15,17,247,119,199,208,123,47,1500
0
1620 DATA 175,80,27,23,229,239,156,250
0,33,46,63,65,236,190,189,153,1040
1630 DATA 75,242,247,215,64,105,240,13
31,196,123,139,192,121,52,248,39,940
1640 DATA 215,62,157,65,100,94,116,30,
,160,86,87,115,160,246,95,94,7916
1650 DATA 145,238,248,41,212,6,197,237
7,16,26,115,160,241,36,248,64,9520
1660 DATA 164,231,65,236,190,189,24,93
3,240,83,163,12,165,158,133,217,1172
1670 DATA 107,228,94,116,30,203,235,20
09,133,223,5,58,119,246,136,13,9081
1680 DATA 59,112,231,157,7,168,130,244
4,184,77,39,208,200,191,47,65,9559
1690 DATA 236,190,189,153,75,242,244,3
30,203,235,208,178,69,9,51,212,1041
1700 DATA 6,197,237,16,26,115,160,246,
,95,94,204,165,235,159,68,37,9695
1710 DATA 199,236,224,81,164,44,231,65
5,228,211,224,159,95,2,187,157,1712
1720 DATA 7,156,250,119,37,167,58,15,3
38,157,187,221,240,83,166,77,9603
1730 DATA 57,208,123,47,175,67,34,246,
,229,12,146,227,246,136,47,102,193
1740 DATA 58,136,61,57,67,46,116,30,11
15,232,132,184,253,64,97,2,6831
1750 DATA 186,124,232,61,151,215,163,1
11,190,10,116,143,237,16,26,115,6735
1760 DATA 160,248,61,68,23,162,101,75,
,209,86,148,123,187,74,201,211,968
1770 DATA 235,204,110,209,5,235,3,50,2
202,124,248,58,15,81,5,237,7113
1780 DATA 19,70,23,169,159,203,223,81,
,5,239,138,99,20,116,235,205,736
1790 DATA 223,9,47,249,60,19,149,212,1
159,94,238,249,161,107,66,201,2095
1800 DATA 243,160,242,105,217,135,51,2
249,123,240,122,137,58,40,87,118,8753
1810 DATA 101,29,100,154,103,175,119,6
66,198,116,204,123,229,62,124,29,8642
1820 DATA 7,178,250,244,97,119,193,78,
,200,161,39,104,128,211,157,7,9255
1830 DATA 178,250,244,97,119,193,78,15
56,180,237,16,26,124,93,7,178,7520
1840 DATA 250,244,97,119,193,78,190,5,
,119,104,128,211,157,7,174,128,8996
1850 DATA 211,166,124,190,30,131,156,2
239,134,196,141,124,231,65,234,32,1605
5
1860 DATA 189,152,76,94,190,5,116,249,
,208,122,136,44,139,240,79,161,785
1870 DATA 119,124,255,59,230,243,252,1
111,27,179,186,104,52,117,238,234,3453
3
1880 DATA 105,206,131,217,125,122,25,2
23,183,40,100,151,31,180,65,122,6183
1890 DATA 112,207,173,12,215,206,131,2
217,125,122,212,206,101,39,58,15,8065
1900 DATA 7,172,12,203,46,137,129,157,
,58,105,61,189,162,11,217,60,7296
1910 DATA 79,167,67,71,181,58,44,251,1
151,215,226,232,61,68,23,178,94
1920 DATA 120,159,78,153,50,44,254,62,
,138,180,235,83,57,150,124,232,1028
1930 DATA 61,68,23,161,129,122,57,86,1
104,44,251,151,214,66,123,139,9028
1940 DATA 62,115,190,28,130,53,243,157
7,7,168,130,244,48,47,83,62,6459
1950 DATA 141,1,145,210,227,244,52,238
8,95,95,139,160,245,16,94,153,756
1960 DATA 143,124,186,153,244,172,40,2
233,36,153,37,39,59,237,107,73,7504
1970 DATA 167,57,208,123,47,175,70,135
5,172,189,12,251,7,166,111,113,8692
1980 DATA 73,211,134,125,153,75,9,245,
,46,163,251,151,214,93,101,92,303
1990 DATA 121,252,93,7,136,80,145,153,
,236,130,68,134,137,123,186,210,1754
2000 DATA 178,147,238,95,89,116,204,44
4,209,206,131,217,125,122,96,165,1886
2010 DATA 133,39,114,250,203,173,76,23
30,151,187,157,7,178,250,244,179,5366
2020 DATA 24,96,82,116,210,123,123,68,
,23,172,8,185,208,122,117,33,7069
2030 DATA 10,230,103,172,12,203,46,138
8,178,157,39,209,214,157,162,36,296
2040 DATA 250,157,29,203,235,206,131,2
217,125,101,214,166,115,75,221,219,441
12
2050 DATA 134,125,157,70,50,206,102,17
71,57,208,122,225,161,101,211,49,844
2060 DATA 239,36,251,68,23,163,123,96,
,25,81,165,117,57,208,123,47,6622
2070 DATA 175,73,39,221,221,75,187,68,
,23,173,172,151,158,220,52,44,9059
2080 DATA 167,206,9,149,23,57,208,122,
,101,28,139,78,248,39,43,169,6944
2090 DATA 242,247,205,76,230,151,187,1
191,87,65,222,252,227,250,59,245,6793
2100 DATA 126,95,143,227,209,103,190,1
16,101,44,247,203,123,4,94,131,8537
2110 DATA 208,145,103,62,250,126,167,1
131,233,122,157,163,251,73,103,161,279
94
2120 DATA 56,82,238,116,30,154,30,250,
,116,52,237,208,26,124,61,7,6570
2130 DATA 168,130,200,191,4,250,231,24
46,79,123,121,208,122,136,47,75,190
2140 DATA 49,134,9,113,251,100,214,83,
,232,105,64,141,115,238,95,89,163
2150 DATA 116,79,38,121,208,120,61,157
7,10,86,118,1,113,89,209,49,5359
2160 DATA 227,92,251,68,23,165,12,233,
,210,61,242,231,65,228,211,168,4473
2170 DATA 12,250,30,158,65,178,105,158
8,141,238,164,158,198,20,157,157,1866
2180 DATA 211,65,163,157,7,168,130,244
4,174,164,130,225,151,180,127,104,2931
1
2190 DATA 130,247,200,11,159,98,203,22
28,245,50,194,203,178,44,202,25,1721
2200 DATA 13,29,48,70,150,38,153,235,1
18,89,246,143,237,16,94,220,333
2210 DATA 179,167,58,15,7,164,44,215,2
212,207,169,119,104,130,244,111,1877
2220 DATA 147,139,46,188,221,162,11,21
17,61,237,231,65,234,32,178,47,577
2230 DATA 83,62,133,221,125,130,206,20
08,34,97,123,116,209,29,26,2,5797
2240 DATA 153,79,157,7,131,211,61,228,
,232,194,36,184,180,234,103,215,4455
2250 DATA 187,180,65,122,80,206,157,18
84,88,206,116,30,64,92,116,184,8979
2260 DATA 253,123,187,68,23,183,11,25,
,212,207,161,235,75,64,179,163,815
2270 DATA 8,146,226,211,183,77,17,208,
,246,108,125,57,208,122,69,39,8676
2280 DATA 66,226,116,76,169,122,42,210
0,143,116,250,25,23,174,40,101,7449
2290 DATA 218,32,189,35,223,46,116,30,
,154,10,203,22,125,104,86,94,5580
2300 DATA 214,29,162,11,217,190,180,71
1,110,22,51,157,7,130,238,133,8419
2310 DATA 130,75,159,208,192,189,76,25
50,151,116,61,24,78,94,116,30,6553
2320 DATA 162,11,34,244,50,47,104,17,4
48,179,248,250,52,5,50,159,6968
2330 DATA 199,208,184,157,15,108,87,17
7,46,63,90,17,161,46,47,58,2201
2340 DATA 15,7,164,40,223,32,209,54,47
7,96,192,186,157,154,105,62,8714
2350 DATA 165,221,162,11,209,190,78,44
4,185,208,122,136,44,139,212,207,2299
2360 DATA 161,234,220,241,117,238,237,
,16,94,220,44,103,58,15,7,179,7112
```

continued on page 20

Database DELPHI

by Michael A. Banks

Though I've never asked the management, I think that the most frequently used feature of the Atari SIG (after the databases) is the realtime Conference. It's more than a gut feeling; I can log on to DELPHI at any time, day or night, and find someone in a conference, somewhere, and I've certainly spent my share of time getting to know people in Conference, people who've become friends offline, as well.

And, as I've frequently mentioned in these pages, the Atari SIG hosts a realtime conference each Tuesday at 10 p.m., EST. You'll find the conferences an excellent venue for sharing information about Atari computers, getting answers to questions, making new friends, etc. And it's quite a kick to be able to chat with people from all around the country—kind of like a telephone conference call, but cheaper and far more entertaining.

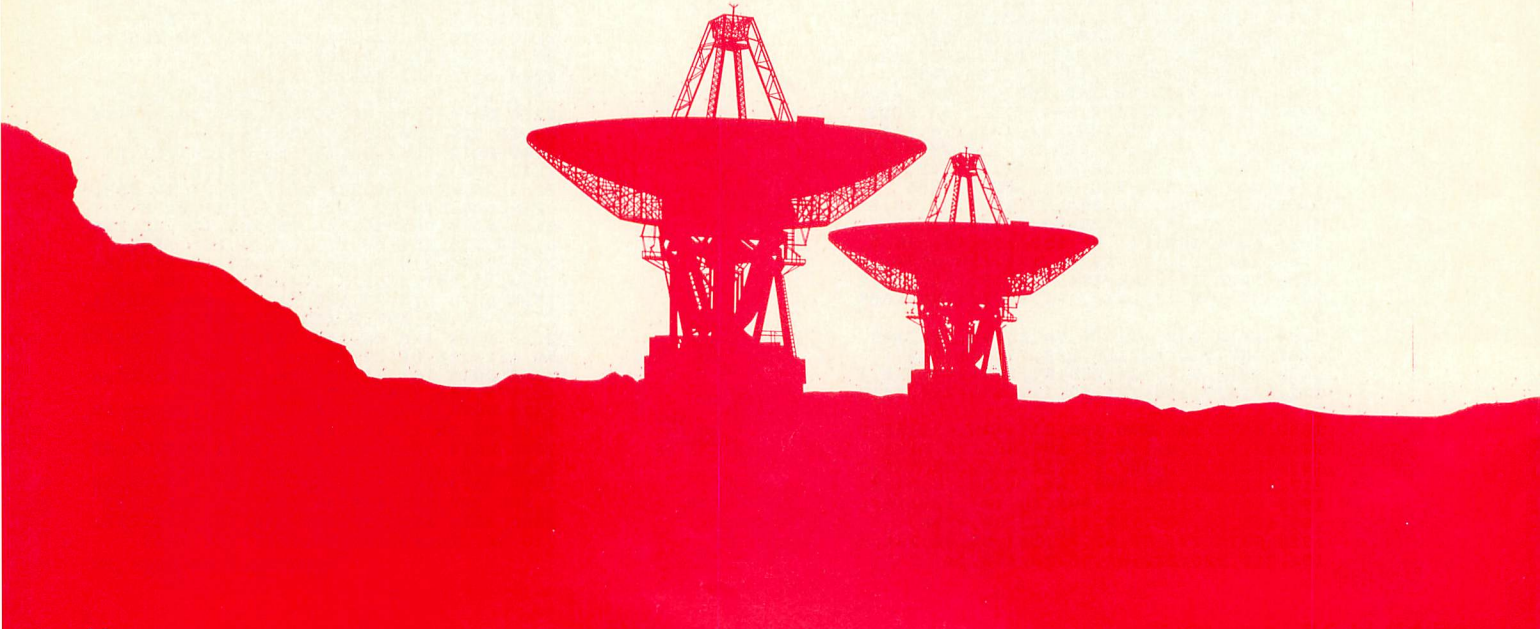
With that in mind, I'm going to devote this column to Conference. I covered the basics of using the ATARI SIG's realtime conference about a year ago. But some of you may have missed that installment of the column, and some may well need a refresher in conference commands.

I'll cover the basics first, and then discuss the "bells and whistles" of which even many veteran conference users are unaware (judging by my observations in the weekly conferences).

Conference Basics

If you haven't yet used Conference on DELPHI, read this quick guide, and keep it handy the first couple of times you're in a conference. (If you are comfortable in Conference, skip ahead to the "Tricks and tips. . ." section.)

Getting There. To get the Atari SIG Conference area, type CONFERENCE (or just CO) at the Atari SIG main menu. You'll see the Conference Menu:



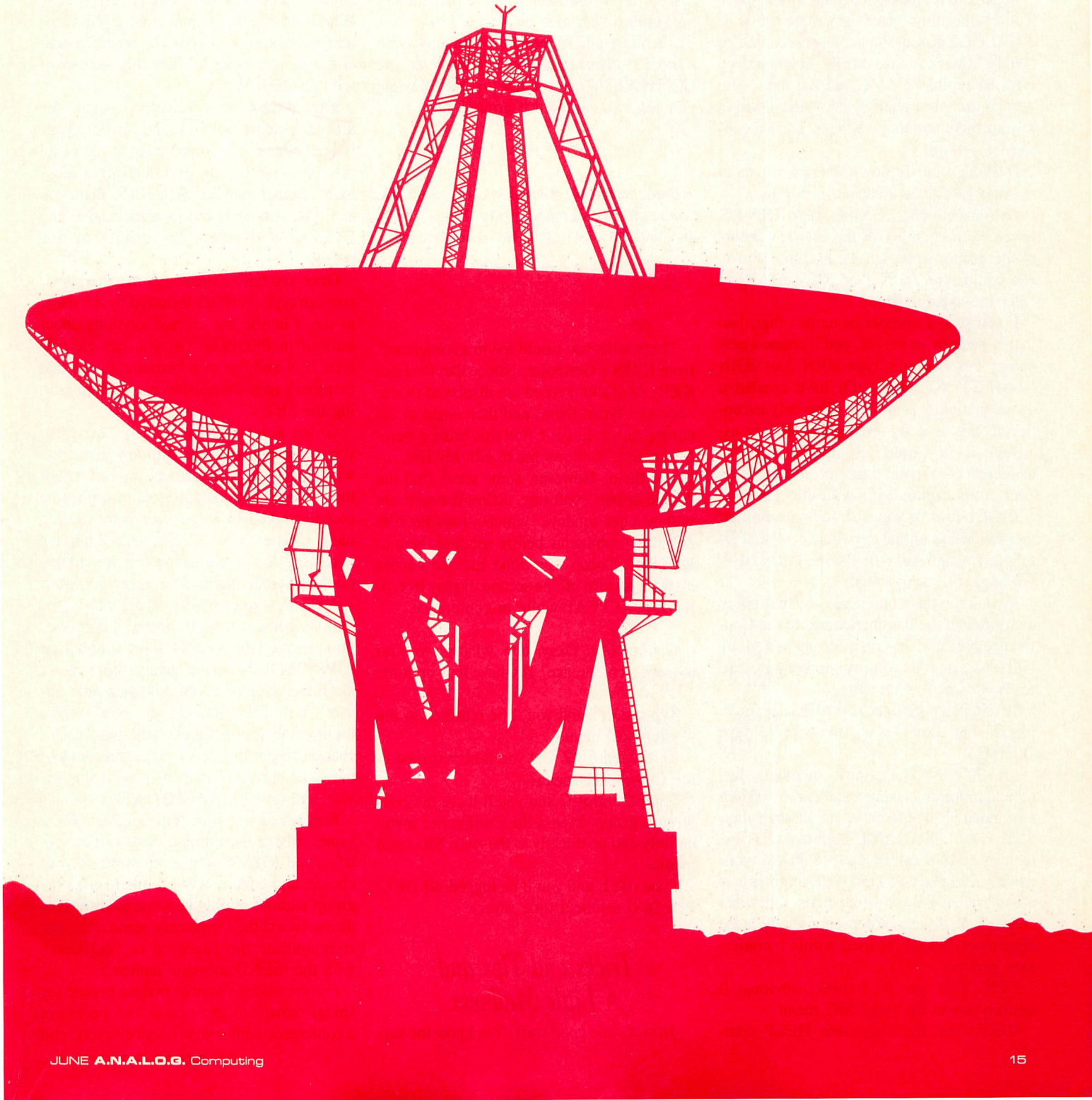
```
Welcome to the ANALOG's ATARI SIG
Conference System
Conference Menu:
WHO (list groups) PAGE a user
JOIN a group      NAME nickname
EXIT
CONFERENCE>
```

Let's examine each of the commands in turn.

WHO (list groups.) If you type WHO at the CONFERENCE> prompt, you'll see a list of everyone currently in the Atari SIG, similar to the list you see when you type WHO (or /WHO) at other SIG prompts. Members who are in the conference area will have parentheses around their names.

In addition to the list of membernames, you'll see any active conference groups listed. (Yes, more than one group can exist in the Atari SIG Conference area.) Each listing will have an identifying number and—usually—a name. The names or assumed nicknames of the participants will be listed, as well, along with the total number of people currently in the SIG. Here's an example:

Nicknames. Incidentally, you'll notice that two of the membernames are enclosed in parentheses; this indicates that these mem-



```
GROUP LIST:   02:58:50
18)  Atari 8-bit books
     KZIN, Andy
 - idle
AVAILABLE LIST: () = in conf
   (KZIN), (ANALOG2), LOUISWU, ANALOG4,
KIP, DAN, PEABO, ELLENSKI
------ [8 in this area]
```

bers are in the Conference area, just as it does on the list you see at any other SIG prompt. However, while the membername KZIN appears both on the "Available List" and in the Conference group, ANALOG2 doesn't show up in the group, even though the parentheses indicates that he's in the Conference area.

So, where is ANALOG2, and who's Andy? Those questions have a simple answer: ANALOG2 has assumed the nickname of "Andy." (You can take on any nickname you wish by typing /NAME and the name you wish to use. Your name is automatically reset to your membername when you leave a conference group.)

JOIN a group. Typing JOIN puts you in an existing group, or creates a new one.

To join a conference group, type JOIN followed by the number or the first few letters of the group name. (In the example above, you could type JOIN 1 or JOIN ATARI to enter the existing group.)

If you type a number or name other than that of a group on the list, you'll create a new group. For example, type JOIN 3 or JOIN COMPUTING, and you'll have created a group that others can join. (The only exception to this is if the number is already assigned to a group in another SIG or Conference area; you'll then be informed that your group cannot begin with that number.)

PAGE a user. If you wish to invite another member who's online anywhere on DELPHI to join your group, type PAGE, followed by the name of the member.

If you're in the Conference area but not in a group, and the member comes to the Conference area and accepts your page, a group will be created for you, its number automatically assigned as its name.

To cancel a page, type /CANCEL. If you're paged and don't wish to accept, type /REJECT.

NAME nickname. Some members feel more comfortable using a nickname (or their real name) in conference, rather than their usernames. The NAME command, as explained earlier, can be used to change your name (as displayed before your comments in a conference group) to whatever you wish. (There is no difference between NAME at the menu and /NAME as used within a conference group.)

EXIT. This one is, I think, obvious. It returns you to the Atari SIG menu.

HELP. Yes, the ubiquitous HELP command, though not listed on the menu, is available in Conference. Type HELP at the Conference menu for a listing of available commands. Type /HELP while in a conference group for the same list. In either case, if you follow the command with the name of a command from the list, you'll get detailed information on how to use the command in question.

#

Note that any of the commands just discussed can be used when you're in a conference group. They must be preceded by a slash (/), as must all conference commands. For more information, type HELP at the CONFERENCE> prompt, or /HELP while you are in a conference group.

#

Now, let's look at what you can do while you're in a conference group. First, a look at what it looks like to be in conference:

```
Hi, Andy.  How's the new job going?
KZIN> Hi, Andy.  How's the new job going?
Great--glad to hear it!
KZIN> Great--glad to hear it!
```

That's what you would see in a conference, from KZIN's viewpoint. Notice that the lines KZIN (that's me) typed are displayed twice: once as I typed them, and then again to the conference at large. I find this to be a helpful feature; it allows me to edit my lines as I type them. However, some users find the double display confusing, and/or use software with a "chat window," which allows them to edit their comments before sending them to DELPHI. Those who don't like the double display can turn off the second display by typing /NOREPEAT while in conference.

Getting Out. Okay, now what do you do to leave a conference? You have four options:

Type /EXIT and you'll be returned to the Conference menu.

Enter Control-Z and you'll be returned to the Conference menu.

Enter Control-C and you'll leave the conference group, bypass the Conference menu entirely and go directly to the Atari SIG main menu.

Type /BYE and you'll be logged off DELPHI (a convenient time saver).

Tricks and Tips and A Little Madness

Now for the good stuff. You know the various ways to get into and out of a conference, and the basics of what you can do while out or in. Here are some of the more useful and fun options available to you in a conference group.

Who's really who. I've already explained how you can adopt a nickname in Conference. But, you may wonder, is there any way to find out who's behind a nickname? Indeed there is; DELPHI provides several ways to "unmask" people in conference.

If you want to match all conference group member's nicknames to their real membernames, you can do so by typing /SHOWRN. When you do this, each nickname will automatically be preceded by the membername of the person using it. Turn this feature off with /NOSHOWRN.

The double-name display makes for difficult reading, however, and you don't really have to use it if all you want to see is the membername of one person using a nickname. Rather than set /SHOWRN, then, you can type /RN followed by a nickname, and DELPHI will reveal the identify of that person.

Once you have a member's membername, you can type /WHOIS followed by the name to see if he/she has entered anything about himself in PEOPLE. Or, you can cleverly type /EN and the membername to see that member's name as he/she entered it on joining the SIG.

Group names. As previously described, you can create a group with any name you wish. Once you're in a group, you can change the name to anything you wish by typing the command /GNAME followed by the desired name. (There's a 32-character limit on names.) I don't recommend that you do this on the Tuesday night conferences, however.

Making a group private. There may be occasions when you wish to keep a conversation private. This is easily done; simply type /GPRIVATE, or create/change the name of a conference group to one with the word "private" in it. This locks out anyone else from joining the group unless you page them, change the group name to something without the word "private" in it, or reverse the Private setting with /NOGPRIVATE.

Password protection. You can also make a group private by setting a password for entry. Type /GPASS and a password of your choice, and nobody will be able to enter the group without setting the same password for themselves at the Conference menu. (You can communicate the password to "outsiders" with the /SEND message feature.)

Private asides. Want to make a private (or snide) remark to just one or two people in a conference with you, without everyone else

seeing it? Use the /SEND command (type /SEND <membername(s)> <message>. Separate multiple membernames with commas).

Make sure there are no spaces or characters preceding the /SEND, by the way; if the "/" isn't at the beginning of the line, your "private" comment will be displayed for all to see. It's great entertainment for others in the group, but quite embarrassing for you! (I generally type / alone to make sure the line is clear, as it were, before typing a /SEND.)

Ignoring the obnoxious. Speaking of those private asides, you'll be glad to know that you can turn 'em off if you wish. . .

As a matter of fact, you can turn off another member completely, at least from your viewpoint. Let's face it: you'll occasionally run into an obnoxious person—even on DELPHI. If this happens to you, you don't have to see that person's remarks. Simply type /SQUELCH and the name of the bothersome user, and you will see neither comments nor /SENDs from that person. (You will, in effect, have squelched the user; that's why it's called SQUELCH.) To remove squelches, type /NOSQUELCH, alone or followed by a membername.

Seeing what's in your workspace. Want to check your Workspace directory while you're in conference? It's easily done—just type /DIR *.* to see a listing of every file in your Workspace. Alternatively, you can replace one of the wildcards with a filename or extension. The directory is displayed to you only.,

Displaying a file. And why would you want to view a Workspace directory? Well, to see what you can show off, for one thing. An interesting but little-known Conference feature is the ability to display a text file from your Workspace. (Brief files recommended!) Simply type /DISPLAY and the name of the file, and the file will be "typed" to the screen one line at a time.

To pause or stop the display, type /NODISPLAY.

For the record. If you'd like to keep a record of a conference, you can, of course, capture it on disk as you go (provided your software accommodates this). However, DELPHI can handle this chore for you; type /LOG with or without a file name, and everything that is typed in the conference (even private /SENDS to and from you) will be recorded in a Workspace file. This is useful if you want to E-Mail a log of the conference to someone else, or simply to avoid delays during the conference while your software writes to disk (you can download the log from your Workspace).

To terminate logging, type /NOLOG, or exit Conference.

Keeping it quiet. You'll notice that your computer beeps a lot when you're in a conference group. Whenever anyone enters or leaves, when someone /SENDs you a private line, or at certain other times, DELPHI sends a Control-G to your computer—to which almost every system responds by beeping its speaker. This can get quite annoying, but is easily overcome: simply have your communications software filter out Control-G.

Switching groups. Once you're comfortable with talking in conference, you might want to try talking in more than one group at a time. It's quite a challenge, more than a little fun, and easy to do. If you see two conference groups when you type /WHO, you can join the one you're not in by typing /J and the number of that group; you'll exit your current group and pop up in the other group.

Two at once. However, if you want to keep your place in the first group, you can do so by typing /T and the number of the second group. Whatever you type will be displayed in the second group, but, while those in the first group won't be able to see what you type, you will be able to see what they type! It gets a little crazy, but, fortunately, everyone's comments are preceded by the number of the group in which they're currently residing.

To switch back to your original group, type /T and its number, and your comments will be displayed there. (And you'll still be able to see what's said in the second group.)

To end the madness and resume with one group, use the /T command to make sure you're in the group you don't want to be in, then type /EXIT or Control-Z; you'll resume with the other group—only.

Conference Games

Dice. Are you a high roller? If so, type /ROLL next time you're in a conference group; you'll be pleasantly surprised. Then type /HELP /ROLL for full instructions on how to specify how many multiple-sided dies to roll, and more.

SCRAMBLE. You may have seen some allusions to something called SCRAMBLE when logging on or off DELPHI. If you're wondering what it is, you might drop into DELPHI's main Conference area, or the Conference area in the Science Fiction, Color Computer, C*SIX, or PC SIG to see what it's about.

SCRAMBLE is a fun, fast and fascinating word game that tests your eye for picking words out of a block of letters. Here's a sample:

A H E Y
R A L F
E K I M
D E E S

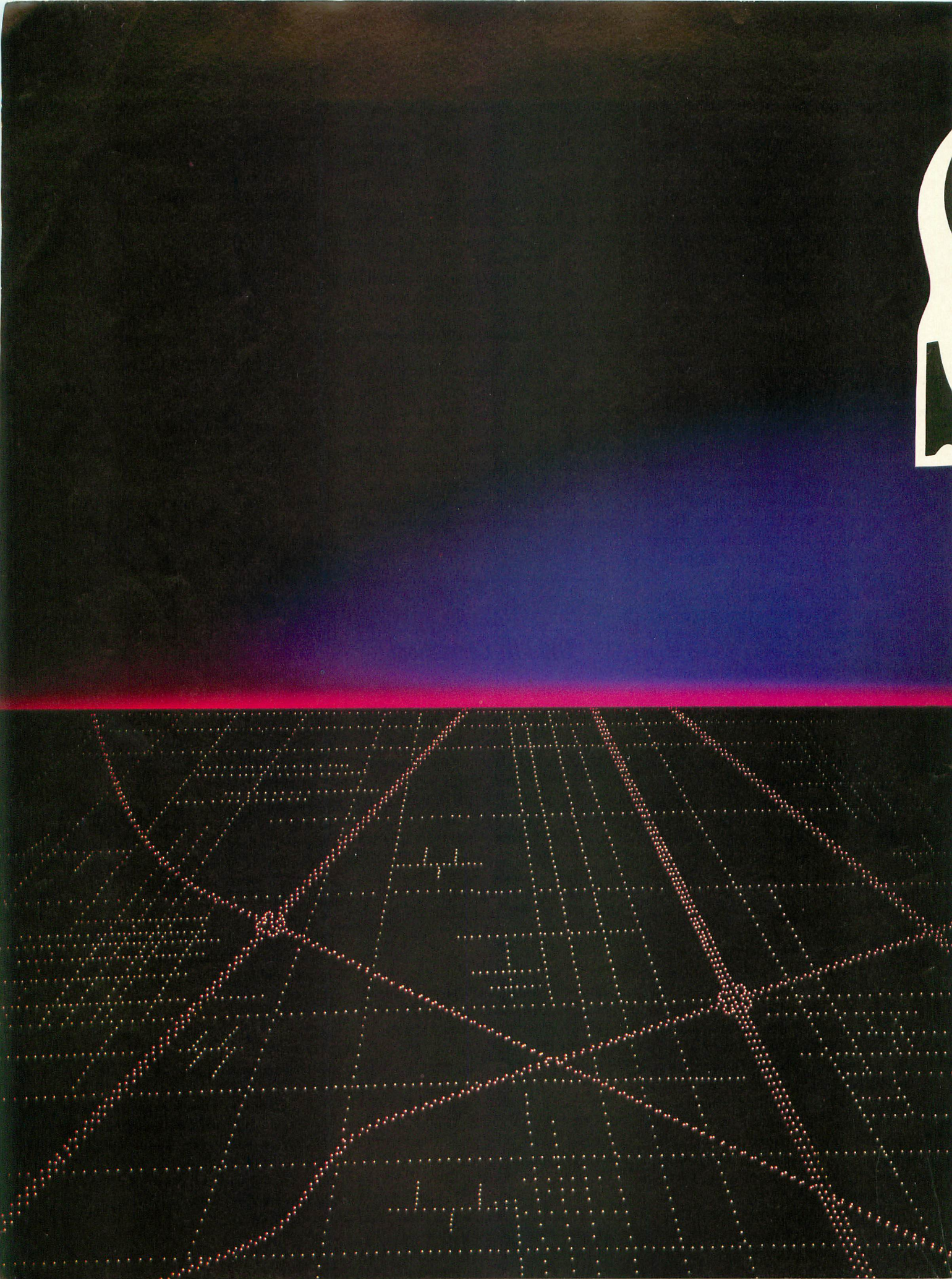
How many words can you pick out? Quite a few? Try limiting yourself to 90 seconds and having to type the words in (with maybe a competitor or two online with you), and you'll have some idea of what SCRAMBLE is like.

After you enter a conference area that hosts SCRAMBLE, type JOIN SCRAMBLE. Full directions are displayed when you enter the game, and high scorers' membernames are posted.

#

That's it for now. See you in Conference!

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



SECTOR TO PRINTER MAPPING

by David Hill

Have you ever been so frustrated by a computer adventure game that you actually sat down and mapped the dungeon by hand? If you have, this program may be just what you need.

After playing computer fantasy games for hours on end and getting nowhere, I set out to see exactly how the data for the maps the games used was stored on the disks. The program began as a simple machine-language routine that loaded a sector from the disk, coupled with a BASIC routine that loaded the sectors one by one into a string, and then printed the entire lump of data out together. That program, after much hair-pulling and gnashing of teeth, evolved into the demo program you see here. I used a disk editor to find the likely location of the data on the disk and by trial and error produced the filter data used to make the maps legible.

Once the filter began to work properly, I realized that a vast majority of the runtime was taken up by the then-BASIC filtering routine. To correct that large problem, I resorted to a short assembly language program that alters the sectors in memory immediately after they are loaded, and then stores the new data in a string for later use.

For this example, I have chosen the program *Ultima III* by Lord British and Origin Systems, as the maps it uses lend themselves very well to this method of printing. With that information out of the way, let us examine the program itself.

Running the Program

The program requires one disk drive, virtually any printer and the player disk from Ultima III. Once you have typed the program (using BASIC Editor II on page 54 to check your work), run it and the program will begin by initializing the extensive machine-language data for all the subroutines. After initialization the program will ask you to insert the player disk you wish to print from, and once satisfied, it will read the first sector of the disk to determine if the disk is in proper format.

Then a series of easy-to-follow menus will appear and direct you through the printing process. The program will allow you to print any of the cities, the two continents, the dungeons or a handy legend for interpretation purposes. The dungeon section only prints one level at a time, so after printing a level you must type a "1" to move to the next level and print it. Since most of the maps—the dungeons in particular—look better in emphasized or double-strike print, I suggest that you insert a control code for your own printer into the program, either at the beginning or on the line I left in the dungeon section.

Program Explanation

Lines 0 through 85 serve to set up the various strings and load the required data. Lines 1000 through 1070 contain the main menu and choice entry code. Lines 2000 through 2095 contain the city menu, and after checking the entry in Lines 2100 and 2105, the program calculates the starting sector number in Line 2110 and begins the loading process. Once each sector has been loaded, it is filtered, copied into CITY$, and then the next one is loaded. The print section outputs 64 characters at a time since the cities and continents are 64 x 64 squares. Upon completion of the printing, the program returns to the city menu to begin anew.

Lines 3000 through 3065 hold the dungeon menu, and Line 3070 calculates the sector lo-

continued on page 64

SECRET AGENT: *Mission 1*

```
2370 DATA 25,92,145,246,9,29,105,34,24
45,1,113,231,215,187,180,65,607
2380 DATA 122,80,206,157,184,88,206,13
33,196,232,123,64,137,133,235,18,1523
2390 DATA 89,246,143,237,16,94,220,179
9,167,58,15,13,5,106,70,22,3319
2400 DATA 79,172,73,122,247,118,136,47
7,104,17,48,179,231,125,68,22,6708
2410 DATA 69,248,39,208,246,119,77,6,1
142,139,61,162,11,210,61,242,9608
2420 DATA 231,65,230,61,235,159,72,23,
,28,178,125,103,212,187,168,44,9773
2430 DATA 139,206,131,212,65,100,94,16
66,125,15,80,246,23,174,40,101,7191
2440 DATA 218,32,189,185,103,78,145,23
39,151,58,15,81,5,145,122,153,7388
2450 DATA 244,61,27,195,78,160,46,58,9
92,126,189,221,15,80,247,175,452
2460 DATA 157,7,168,130,200,189,76,250
0,23,119,193,245,60,31,75,201,336
2470 DATA 239,213,246,125,62,199,139,2
209,189,176,173,11,47,143,164,146,1427
7
2480 DATA 108,94,133,196,236,36,151,80
0,178,234,11,34,243,160,240,123,2490
2490 DATA 4,62,143,249,10,230,6,71,44,
,187,34,206,147,235,221,208,3039
2500 DATA 245,129,23,104,17,48,188,232
2,60,30,145,239,151,82,238,209,2361
2510 DATA 5,233,67,58,118,225,99,57,20
08,122,136,47,77,89,57,52,5589
2520 DATA 172,139,209,189,211,76,233,6
62,188,221,15,72,89,175,161,113,752
2530 DATA 58,30,140,34,75,139,78,116,3
30,15,100,241,62,157,61,67,5290
2540 DATA 34,79,180,65,122,92,38,147,2
231,65,224,247,236,251,250,151,7041
2550 DATA 13,29,66,78,142,140,34,75,13
39,78,166,125,75,187,68,23,5857
2560 DATA 163,124,156,89,115,160,243,3
31,73,245,230,236,139,52,206,133,3501
2570 DATA 148,250,23,19,163,66,97,62,1
141,149,165,11,46,209,5,237,7846
2580 DATA 2,38,22,124,232,61,68,22,69,
,234,103,208,244,76,169,122,711
2590 DATA 42,210,143,119,69,158,209,5,
,233,30,249,115,160,240,122,208,3867
2600 DATA 184,236,41,59,39,137,244,235
5,18,89,245,238,237,16,94,153,1986
2610 DATA 143,124,185,208,120,61,33,66
6,78,134,5,235,18,89,245,238,9993
2620 DATA 237,16,94,153,143,124,185,20
08,122,136,44,139,212,207,161,236,4656
6
2630 DATA 194,98,245,5,145,121,208,108
8,0,48,64,67,32,91,61,32,2651
2640 DATA 194,65,76,186,50,112,112,96,
,66,90,48,16,66,0,140,2,1614
2650 DATA 2,2,2,2,2,2,2,2,2,2,2,2,2,2,
,2,2,2922
2660 DATA 2,2,2,2,2,65,9,48,0,40,80,12
20,160,200,240,24,5111
2670 DATA 64,104,144,184,224,8,48,88,1
128,168,208,248,32,72,112,0,7518
2680 DATA 0,0,0,0,0,0,1,1,1,1,1,1,2,2,
,2,2,2853
2690 DATA 2,2,2,3,3,3,0,0,0,0,0,0,0,0,
,0,0,2747
2700 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,0,0,2700
2710 DATA 0,0,0,0,51,35,47,50,37,26,0,
,16,0,0,64,0,5649
2720 DATA 32,96,63,21,18,58,42,56,61,5
57,13,1,5,0,37,35,6851
2730 DATA 8,10,47,40,62,45,11,16,46,22
2,43,23,33,34,32,52,7444
2740 DATA 12,118,32,46,44,126,155,125,
,72,169,0,133,20,165,20,240,6973
2750 DATA 252,104,96,160,0,152,153,0,1
140,153,0,141,153,0,142,153,7170
2760 DATA 0,143,200,208,241,32,151,51,
,96,164,84,185,44,48,24,105,6272
2770 DATA 0,133,182,185,67,48,105,140,
,133,183,96,165,84,201,17,144,9328
2780 DATA 58,169,0,133,176,169,140,133
3,177,169,40,133,178,169,140,133,1873
2790 DATA 179,162,0,160,39,177,178,145
5,176,136,16,249,165,179,133,177,3182
2800 DATA 165,178,133,176,24,105,40,13
33,178,165,179,105,0,133,179,232,1258
2810 DATA 224,23,144,223,198,84,198,20
02,76,223,48,96,32,41,49,29,5683
2820 DATA 130,48,166,187,96,72,42,42,4
42,42,41,3,134,187,170,104,5693
2830 DATA 41,159,96,173,252,2,201,255,
,240,249,162,255,142,252,2,162,6376
2840 DATA 25,221,134,48,240,24,202,16,
,248,162,5,221,160,48,240,5,9778
2850 DATA 202,16,248,48,17,189,166,48,
,76,129,49,76,55,49,138,24,4065
2860 DATA 109,190,2,76,129,49,201,60,2
208,241,173,190,2,201,97,240,3085
2870 DATA 8,169,97,141,190,2,76,55,49,
,169,65,208,246,160,127,140,930
2880 DATA 31,208,162,8,202,208,253,136
6,16,245,96,134,180,132,181,160,3683
2890 DATA 0,177,180,201,255,208,1,96,3
32,205,48,160,0,132,200,177,352
2900 DATA 180,153,0,133,200,201,0,208,
,4,230,200,208,5,192,38,208,1471
2910 DATA 238,136,177,180,201,32,240,5
5,136,208,247,160,37,200,132,189,3814
2920 DATA 169,0,153,0,133,152,24,101,1
180,133,180,144,2,230,181,160,1280
2930 DATA 2,185,254,132,240,11,32,32,4
49,145,182,32,172,48,200,208,9851
2940 DATA 240,32,243,49,165,200,208,3,
,76,156,49,32,223,48,96,230,9532
2950 DATA 84,169,2,133,85,96,32,205,48
8,164,85,136,169,30,145,182,9118
2960 DATA 162,0,134,186,169,80,133,201
1,165,85,133,203,165,84,133,202,2914
2970 DATA 32,205,48,32,176,50,32,55,49
9,201,155,240,36,201,126,240,1576
2980 DATA 53,201,125,240,43,166,186,22
28,201,176,235,72,32,143,50,166,1949
2990 DATA 186,104,201,96,144,3,56,233,
,32,157,0,132,230,186,76,26,7957
3000 DATA 50,166,186,157,0,132,32,133,
,50,32,243,49,32,223,48,96,6473
3010 DATA 32,183,48,76,250,49,165,85,1
197,203,208,18,165,84,197,202,3050
3020 DATA 240,28,32,133,50,198,84,169,
,39,133,85,76,119,50,32,133,5705
3030 DATA 50,198,85,32,205,48,169,63,1
164,85,145,182,198,186,76,26,9698
3040 DATA 50,32,205,48,164,85,169,0,14
45,182,96,72,32,205,48,104,7189
3050 DATA 32,32,49,164,85,145,182,230,
,85,165,85,201,40,208,9,169,391
3060 DATA 2,133,85,230,84,32,223,48,32
2,176,50,96,32,205,48,169,7520
3070 DATA 63,164,85,145,182,96,169,97,
,141,190,2,169,128,141,244,2,290
3080 DATA 32,183,48,169,9,141,48,2,169
9,48,141,49,2,169,62,141,5259
3090 DATA 47,2,169,0,141,45,62,141,46,
,62,133,204,32,151,51,162,7017
3100 DATA 116,160,117,32,143,49,162,14
41,160,117,32,143,49,230,84,32,7593
3110 DATA 160,51,32,250,49,169,1,141,3
31,63,173,1,132,201,155,208,9869
3120 DATA 107,173,0,132,162,5,221,203,
,61,240,5,202,16,248,48,49,8724
3130 DATA 134,199,32,97,60,172,45,62,1
185,34,63,24,101,199,168,185,8854
3140 DATA 71,62,16,10,162,159,160,113,
,32,143,49,76,246,50,141,45,7113
3150 DATA 62,201,33,208,3,76,55,61,32,
,160,51,32,195,60,76,246,7173
3160 DATA 50,201,73,208,6,32,223,59,76
6,246,50,201,76,208,6,32,7526
3170 DATA 204,51,76,246,50,201,88,208,
,6,32,229,64,76,246,50,201,483
3180 DATA 82,208,9,32,73,65,32,160,51,
,76,246,50,32,163,52,165,6735
3190 DATA 196,240,10,162,65,160,113,32
2,143,49,76,246,50,165,192,10,8441
3200 DATA 170,189,203,66,141,146,51,18
89,204,66,141,147,51,32,255,255,2898
3210 DATA 76,246,50,169,1,133,84,133,8
85,230,85,96,160,24,169,0,7162
3220 DATA 153,90,48,136,16,250,173,45,
,62,10,170,189,223,61,133,182,848
3230 DATA 189,224,61,133,183,160,0,177
7,182,240,9,32,32,49,153,92,7263
3240 DATA 48,200,208,243,32,204,51,96,
,32,115,53,32,74,52,32,214,5792
3250 DATA 51,96,162,0,189,56,113,157,0
0,132,232,224,9,208,245,160,3131
3260 DATA 0,185,48,62,205,45,62,240,49
9,200,192,23,208,243,224,9,2110
3270 DATA 240,18,169,32,157,254,131,16
69,0,157,255,131,162,0,160,132,1324
3280 DATA 32,143,49,96,160,0,185,47,11
13,157,0,132,240,4,232,200,627
```

```
3290 DATA 208,244,162,0,160,132,32,143
3,49,96,152,72,10,168,185,184,9570
3300 DATA 64,133,183,185,183,64,133,18
82,160,0,177,182,240,7,157,0,9749
3310 DATA 132,232,200,208,245,169,44,1
157,0,132,169,32,232,157,0,132,30
3320 DATA 232,104,168,76,237,51,162,0,
,189,183,113,157,0,132,232,224,2763
3330 DATA 15,208,245,172,45,62,185,34,
,63,168,169,0,133,186,132,199,951
3340 DATA 185,71,62,48,29,132,199,165,
,186,10,168,185,33,62,133,182,9741
3350 DATA 185,34,62,133,183,160,0,177,
,182,157,0,132,240,4,232,200,2260
3360 DATA 208,245,164,199,230,186,200,
,165,186,201,6,208,209,169,32,157,4653
3
3370 DATA 254,131,169,0,157,255,131,16
62,0,160,132,32,143,49,96,162,8934
3380 DATA 0,134,190,134,192,134,193,13
34,197,134,198,134,196,169,69,133,3917
7
3390 DATA 182,169,63,133,183,166,190,1
160,0,177,182,221,0,132,208,8,672
3400 DATA 232,200,192,3,208,243,240,22
2,230,192,165,182,24,105,3,133,918
3410 DATA 182,144,2,230,183,165,192,20
01,18,208,218,230,196,96,189,0,3790
3420 DATA 132,201,155,240,246,201,32,2
240,3,232,208,242,32,40,53,32,9781
3430 DATA 53,53,32,40,53,32,184,53,176
6,225,165,193,133,194,189,0,1259
3440 DATA 132,201,155,208,1,96,201,32,
,240,3,232,208,241,232,134,190,6184
3450 DATA 32,40,53,32,72,53,32,40,53,3
32,53,53,32,184,53,165,3514
3460 DATA 193,133,195,96,189,0,132,201
1,32,208,3,232,208,246,134,190,4748
3470 DATA 96,160,0,185,219,61,221,0,13
32,208,8,232,200,192,4,208,2450
3480 DATA 242,134,190,96,166,190,160,0
0,189,0,132,217,209,61,208,11,658
3490 DATA 232,200,192,5,208,242,230,19
5,32,143,49,173,10,210,201,180,2247
4350 DATA 144,8,162,218,160,115,32,143
3,49,96,169,0,76,74,60,201,8126
4360 DATA 9,240,3,76,89,55,162,36,160,
,5,32,8,59,173,29,63,3929
4370 DATA 240,3,76,250,55,173,17,63,24
40,174,173,10,210,201,165,144,4026
4380 DATA 8,169,0,141,17,63,76,118,55,
,162,161,160,115,32,143,49,7907
4390 DATA 173,10,210,201,150,176,8,162
2,29,160,116,32,143,49,96,169,9385
4400 DATA 1,76,74,60,228,194,208,16,19
96,195,208,17,185,48,62,201,2332
4410 DATA 128,208,23,165,198,240,6,96,
,104,104,76,82,59,104,104,173,9270
4420 DATA 31,63,240,3,76,250,55,76,89,
,55,104,104,76,17,57,173,6774
4430 DATA 45,62,201,13,208,24,165,194,
,201,38,208,18,173,16,63,240,1096
4440 DATA 13,73,1,141,16,63,169,13,141
1,65,62,76,118,55,162,57,6070
4450 DATA 160,115,32,143,49,96,173,45,
,62,201,15,208,40,165,194,201,2085
4460 DATA 41,240,3,76,89,55,173,21,63,
,240,18,73,1,141,21,63,4799
4470 DATA 169,147,141,13,67,169,122,14
41,14,67,76,118,55,162,71,160,8395
4480 DATA 113,32,143,49,96,201,27,208,
,218,165,194,201,27,208,212,173,6190
4490 DATA 23,63,240,233,73,1,141,23,63
3,169,29,141,238,62,76,118,9091
4500 DATA 55,173,45,62,201,7,208,25,16
65,194,201,39,208,8,169,0,9442
4510 DATA 141,14,63,76,224,57,201,40,2
240,3,76,89,55,169,1,208,8879
4520 DATA 239,201,29,208,245,165,194,2
201,40,208,239,169,30,141,247,62,5419
4530 DATA 169,255,141,245,62,32,224,57
7,76,118,55,162,0,189,37,113,8560
4540 DATA 157,0,132,232,224,10,208,245
5,160,0,185,48,62,201,128,208,3536
4550 DATA 41,152,72,10,168,185,183,64,
,133,182,185,184,64,133,183,160,4153
4560 DATA 0,177,182,240,7,157,0,132,23
32,200,208,245,169,44,157,0,2937
4570 DATA 132,232,169,32,157,0,132,232
2,104,168,200,192,23,208,203,224,6326
4580 DATA 10,208,17,160,0,185,47,113,1
157,0,132,232,200,192,10,208,2455
4590 DATA 244,76,66,60,169,0,157,255,1
131,169,32,157,254,131,162,0,2079
4600 DATA 160,132,32,143,49,96,10,170,
,189,50,67,168,189,49,67,170,9765
4610 DATA 32,143,49,162,179,160,116,32
2,143,49,76,138,61,173,45,62,7797
4620 DATA 201,26,208,46,173,30,63,208,
,1,96,32,163,52,165,196,208,1362
4630 DATA 9,165,192,201,10,208,3,76,81
1,58,173,10,210,201,200,144,2456
4640 DATA 5,169,0,76,74,60,162,218,160
0,115,32,143,49,104,104,76,8422
4650 DATA 246,50,201,9,208,40,173,29,6
63,208,35,173,17,63,240,30,8649
4660 DATA 32,163,52,165,196,208,9,165,
,192,201,10,208,3,76,81,58,9035
4670 DATA 173,10,210,201,130,176,7,104
4,104,169,1,76,74,60,96,173,8443
4680 DATA 45,62,201,5,208,11,173,14,63
3,240,5,169,5,76,74,60,6150
4690 DATA 96,201,23,208,10,173,24,63,2
240,246,169,2,76,74,60,201,492
4700 DATA 17,208,10,173,27,63,208,232,
,169,4,76,74,60,201,18,208,157
4710 DATA 10,173,28,63,240,218,169,3,7
76,74,60,201,30,208,209,173,2818
4720 DATA 25,63,240,204,169,6,76,74,60
0,169,0,141,47,62,160,4,6277
4730 DATA 185,58,62,201,128,240,22,136
6,16,246,173,60,62,201,128,208,3438
4740 DATA 3,206,47,62,173,47,62,9,16,1
141,127,48,96,238,47,62,7001
4750 DATA 76,27,61,162,205,160,116,32,
,143,49,230,84,173,47,62,201,1132
4760 DATA 4,208,10,162,16,160,117,32,1
143,49,76,78,61,162,179,160,9828
3920 DATA 168,185,48,62,201,128,240,8,
,162,101,160,113,32,143,49,96,8640
3930 DATA 165,194,201,3,240,16,201,9,2
240,20,201,12,208,24,162,204,1322
3940 DATA 160,114,32,143,49,96,162,111
1,160,114,32,143,49,96,162,152,9330
3950 DATA 160,114,32,143,49,96,162,248
8,160,114,32,143,49,96,173,45,8889
3960 DATA 62,201,5,208,26,162,2,160,30
0,32,220,56,169,5,141,53,6579
3970 DATA 62,169,255,141,50,62,206,46,
,62,32,129,59,76,118,55,201,7817
3980 DATA 11,208,20,162,0,160,30,32,22
20,56,169,11,141,54,62,169,7295
3990 DATA 255,141,48,62,76,90,56,201,1
12,208,20,162,16,160,34,32,5661
4000 DATA 220,56,169,12,141,59,62,169,
,255,141,64,62,76,90,56,201,9189
4010 DATA 18,208,20,162,15,160,44,32,2
220,56,169,255,141,63,62,169,559
4020 DATA 21,141,182,62,76,90,56,201,2
20,208,20,162,4,160,43,32,5910
4030 DATA 220,56,169,20,141,62,62,169,
,255,141,52,62,76,90,56,201,9137
4040 DATA 21,208,18,162,12,160,42,32,2
220,56,169,21,141,61,62,141,7273
4050 DATA 60,62,76,90,56,76,89,55,228,
,194,208,16,196,195,208,12,1683
4060 DATA 189,48,62,201,128,208,15,165
5,197,240,16,96,162,25,160,114,829
4070 DATA 32,143,49,104,104,96,104,104
4,76,17,57,104,104,76,250,55,7382
4080 DATA 165,194,201,18,208,28,173,66
6,62,201,128,240,8,162,101,160,1558
4090 DATA 113,32,143,49,96,169,0,141,2
25,63,162,40,160,114,32,143,7075
4100 DATA 49,96,201,8,208,20,173,56,62
2,201,128,208,224,169,0,141,1801
4110 DATA 28,63,162,130,160,115,32,143
3,49,96,76,89,55,173,45,62,6237
4120 DATA 201,19,208,17,165,194,201,24
4,240,3,76,89,55,169,20,141,8370
4130 DATA 188,62,76,118,55,201,24,208,
,17,165,194,201,23,208,235,173,4308
4140 DATA 24,63,73,1,141,24,63,76,224,
,57,201,14,240,249,201,10,1157
4150 DATA 240,245,201,9,208,212,165,19
94,201,23,208,206,169,0,141,29,2113
4160 DATA 63,32,224,57,169,32,141,126,
,62,76,118,55,173,45,62,201,8520
4170 DATA 15,208,55,173,21,63,208,6,16
65,194,201,25,240,3,76,50,8483
4180 DATA 58,173,67,62,201,128,240,3,7
76,17,57,173,22,63,208,3,6403
4190 DATA 76,129,59,73,1,141,22,63,169
9,15,141,63,62,162,80,160,7314
4200 DATA 115,32,143,49,32,224,57,76,1
118,55,201,16,208,208,165,194,2725
```

SECRET AGENT: *Mission 1*

```
4210 DATA 201,25,208,202,173,27,63,73,
,1,141,27,63,162,68,160,115,7715
4220 DATA 32,143,49,96,165,194,201,19,
,208,15,173,67,62,201,128,208,2214
4230 DATA 19,162,97,160,115,32,143,49,
,96,201,6,208,18,173,54,62,7558
4240 DATA 201,128,240,3,76,17,57,162,4
49,160,118,32,143,49,96,162,7906
4250 DATA 233,160,113,32,143,49,96,166
6,194,189,48,62,201,128,240,3,1240
4260 DATA 76,17,57,224,6,208,7,173,45,
,62,201,23,240,8,162,73,8490
4270 DATA 160,114,32,143,49,96,173,26,
,63,73,1,141,26,63,240,8,5514
4280 DATA 169,255,141,211,62,76,118,55
5,169,16,208,246,96,169,0,141,1049
4290 DATA 31,63,165,194,201,26,208,11,
,169,1,133,198,169,5,133,195,1175
4300 DATA 76,121,58,201,36,240,241,165
5,194,201,5,208,8,162,195,160,4387
4310 DATA 117,32,143,49,96,173,45,62,2
201,26,208,67,162,26,160,5,7556
4320 DATA 32,8,59,173,30,63,208,3,76,8
82,59,173,10,210,201,100,9159
4330 DATA 176,18,169,0,141,30,63,169,8
80,141,35,67,169,122,141,36,7642
4340 DATA 67,76,118,55,162,161,160,115
97,134,190,96,189,0,132,217,214,5353
3500 DATA 61,208,11,232,200,192,5,208,
,242,230,198,76,92,53,96,173,2503
3510 DATA 45,62,10,170,189,240,66,168,
,189,239,66,170,32,143,49,173,1358
3520 DATA 45,62,201,1,208,8,162,12,160
0,118,32,143,49,96,201,9,6442
3530 DATA 208,33,173,29,63,240,8,162,1
140,160,119,32,143,49,96,173,8852
3540 DATA 17,63,240,8,162,29,160,116,3
32,143,49,96,162,116,160,122,8958
3550 DATA 32,143,49,96,169,123,133,182
2,169,63,133,183,162,0,134,193,1383
3560 DATA 166,190,160,0,177,182,221,0,
,132,208,8,232,200,192,3,208,2911
3570 DATA 243,240,21,230,193,165,182,2
24,105,3,133,182,144,2,230,183,1597
3580 DATA 165,193,201,46,208,218,56,96
6,24,96,173,46,62,201,6,176,8583
3590 DATA 57,166,194,224,45,240,67,224
4,23,176,55,189,48,62,205,45,9510
3600 DATA 62,240,20,201,128,208,8,162,
,209,160,113,32,143,49,96,162,9931
3610 DATA 77,160,113,32,143,49,96,169,
,128,157,48,62,238,46,62,162,8761
3620 DATA 93,160,113,32,143,49,32,13,6
61,96,162,133,160,113,32,143,7154
3630 DATA 49,96,162,14,160,115,32,143,
,49,96,160,0,185,48,62,205,7719
3640 DATA 45,62,240,10,192,23,240,3,20
00,208,241,76,129,59,174,46,663
3650 DATA 62,224,6,176,213,169,128,153
3,48,62,238,46,62,152,72,32,7829
3660 DATA 35,54,104,168,76,64,54,230,2
205,208,14,169,0,133,205,160,1373
3670 DATA 0,185,48,62,205,45,62,240,23
38,200,192,23,208,243,165,205,6472
3680 DATA 201,6,144,8,162,211,160,117,
,32,143,49,96,166,194,224,10,292
3690 DATA 208,3,76,22,55,224,45,240,55
5,189,48,62,201,128,240,8,9864
3700 DATA 162,101,160,113,32,143,49,96
6,173,45,62,157,48,62,224,18,6838
3710 DATA 208,7,169,1,141,25,63,208,9,
,224,8,208,5,169,1,141,7010
3720 DATA 28,63,206,46,62,162,123,160,
,113,32,143,49,32,13,61,96,4646
3730 DATA 160,0,185,48,62,201,128,240,
,8,200,192,23,208,244,76,79,1953
3740 DATA 54,165,205,201,6,240,157,192
2,10,240,39,201,8,208,5,169,774
3750 DATA 1,141,28,63,201,18,208,5,169
9,1,141,25,63,230,205,206,770
3760 DATA 46,62,152,72,32,201,54,104,1
168,173,45,62,153,48,62,76,6538
3770 DATA 221,54,104,104,169,7,76,74,6
60,76,32,55,165,194,201,7,7138
3780 DATA 208,32,173,55,62,201,128,240
0,3,76,17,57,173,20,63,208,7583
3790 DATA 3,76,19,58,73,1,141,20,63,17
73,45,62,141,56,62,76,4051
3800 DATA 118,55,201,28,208,108,173,45
5,62,201,1,240,16,201,2,240,353
3810 DATA 45,201,25,240,65,162,91,160,
,114,32,143,49,96,173,18,63,6961
3820 DATA 240,20,73,1,141,18,63,169,1,
,141,48,62,141,49,62,32,3581
3830 DATA 122,55,32,204,51,96,162,3,16
60,114,32,143,49,96,162,28,6470
3840 DATA 160,1,32,8,59,173,13,63,240,
,236,73,1,141,13,63,169,7057
3850 DATA 2,141,50,62,208,217,162,28,1
160,21,32,8,59,173,33,63,5022
3860 DATA 240,212,73,1,141,33,63,169,2
25,141,64,62,141,70,62,76,5485
3870 DATA 115,55,201,27,208,159,173,45
5,62,201,25,208,25,162,27,160,9268
3880 DATA 17,32,8,59,173,19,63,240,173
3,73,1,141,19,63,169,25,5615
3890 DATA 141,69,62,76,115,55,201,4,24
40,3,76,89,55,162,27,160,7045
3900 DATA 22,32,8,59,173,32,63,240,141
1,73,1,141,32,63,169,4,5263
3910 DATA 141,58,62,76,115,55,162,15,1
160,114,32,143,49,96,165,194,9024
4770 DATA 116,32,143,49,76,138,61,169,
,45,133,184,169,62,133,185,169,2496
4780 DATA 0,133,180,169,134,133,181,16
60,0,177,184,145,180,230,180,208,7399
4790 DATA 2,230,181,230,184,208,2,230,
,185,165,185,201,63,208,232,165,8350
4800 DATA 184,201,34,208,226,96,162,16
68,160,117,32,143,49,173,31,208,2034
4810 DATA 201,6,240,2,208,247,169,45,1
133,184,169,62,133,185,169,0,2310
4820 DATA 133,180,169,134,133,181,160,
,0,177,180,145,184,230,180,208,2,5085
4830 DATA 230,181,230,184,208,2,230,18
85,165,185,201,63,208,232,165,184,8663
3
4840 DATA 201,34,208,226,76,186,50,78,
,83,69,87,85,68,73,78,84,6941
4850 DATA 79,32,87,73,84,72,32,84,72,6
69,32,0,112,8,112,17,2504
4860 DATA 112,32,112,48,112,56,112,70,
,112,86,112,86,112,99,112,111,8234
4870 DATA 112,131,112,139,112,149,112,
,149,112,159,112,166,112,179,112,56,19
926
4880 DATA 112,149,112,189,112,202,112,
,86,112,56,112,149,112,159,112,56,472
4890 DATA 112,86,112,212,112,228,112,2
239,112,246,112,246,112,251,112,3,5332
2
4900 DATA 113,11,113,18,113,25,113,30,
,113,78,84,79,255,255,255,3,1619
4910 DATA 32,255,255,8,255,6,255,255,1
15,255,255,255,255,255,22,28,7598
4920 DATA 28,255,255,255,255,255,1,255
5,255,2,3,255,5,255,255,255,9043
4930 DATA 1,255,255,255,255,1,7,255,25
55,255,255,255,5,255,12,255,9201
4940 DATA 255,4,255,1,11,255,255,255,8
8,255,9,255,255,3,255,255,8425
4950 DATA 255,255,255,6,255,255,255,25
55,255,10,255,6,255,255,255,13,9324
4960 DATA 9,11,255,255,255,12,255,5,10
0,255,255,255,11,4,255,255,6557
4970 DATA 255,255,10,255,255,14,24,255
5,255,23,255,255,13,23,22,255,3643
4980 DATA 255,255,255,255,255,17,23,25
55,255,255,18,255,16,255,255,17,7086
4990 DATA 19,255,255,255,255,18,255,25
55,255,255,7,255,255,19,255,255,1980
5000 DATA 255,16,255,18,255,255,255,15
5,255,255,255,255,255,255,15,255,2734
5010 DATA 14,255,255,255,26,255,255,13
3,255,255,28,255,255,255,255,24,175
5020 DATA 27,28,255,255,255,26,255,255
5,255,255,255,25,255,255,26,255,1449
5030 DATA 255,27,255,255,255,255,255,2
255,255,33,255,255,255,255,255,255,703
34
5040 DATA 255,255,255,9,255,255,255,25
55,255,1,1,1,1,1,1,1,5622
5050 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,
,0,6,5251
5060 DATA 12,18,24,30,36,42,48,54,60,6
66,72,78,84,90,96,102,4852
5070 DATA 108,114,120,126,132,138,144,
,150,156,162,168,174,180,186,192,198,7
7918
5080 DATA 204,71,69,84,68,82,79,79,80,
,69,82,69,65,73,78,83,5491
5090 DATA 87,69,65,84,85,82,70,76,73,6
69,88,65,87,65,86,83,5615
5100 DATA 72,79,84,65,75,76,73,70,77,7
79,86,80,85,83,85,78,5923
5110 DATA 76,80,82,69,80,85,84,66,65,6
```

```
68,78,65,73,67,65,82,4971
5120 DATA 80,65,68,68,73,83,71,85,78,6
66,65,84,66,79,88,71,5351
5130 DATA 65,83,70,79,76,86,73,65,70,7
79,82,77,69,77,77,73,5358
5140 DATA 67,80,82,79,82,79,68,67,79,7
73,83,77,65,83,67,85,5475
5150 DATA 69,76,69,70,76,65,76,65,82,6
67,79,77,75,78,79,72,5285
5160 DATA 79,79,83,87,73,82,79,66,67,6
65,66,68,69,83,83,65,5071
5170 DATA 70,83,76,79,67,65,71,76,69,8
86,76,79,67,86,69,78,5403
5180 DATA 77,65,67,65,80,69,68,79,79,8
80,76,65,82,69,68,66,5005
5190 DATA 76,85,80,73,67,69,78,76,68,8
82,73,72,79,76,65,76,5252
5200 DATA 76,98,97,100,103,101,0,110,9
97,105,108,102,105,108,101,0,6891
5210 DATA 99,97,114,100,0,112,97,100,0
0,100,105,115,107,0,103,117,6739
5220 DATA 110,0,98,97,116,111,110,0,98
8,111,120,0,103,97,115,32,6274
5230 DATA 109,97,115,107,0,102,111,108
8,100,101,114,0,118,105,97,108,7910
5240 DATA 0,102,111,114,109,117,108,97
7,0,109,101,109,111,0,109,105,7279
5250 DATA 99,114,111,100,111,116,0,112
2,114,111,103,114,97,109,0,114,7705
5260 DATA 111,100,0,99,111,105,110,0,1
115,109,97,108,108,32,107,101,7483
5270 DATA 121,0,115,99,117,98,97,32,10
03,101,97,114,0,101,108,101,7262
5280 DATA 99,116,114,111,110,105,99,32
2,100,101,118,105,99,101,0,102,7327
5290 DATA 108,97,115,104,105,110,103,3
32,98,97,108,108,0,108,97,114,7642
5300 DATA 103,101,32,107,101,121,0,99,
,111,109,98,105,110,97,116,105,8787
5310 DATA 111,110,0,5,64,11,64,20,64,2
25,64,29,64,34,64,38,1409
5320 DATA 64,44,64,48,64,57,64,64,64,6
69,64,77,64,82,64,91,4768
5330 DATA 64,99,64,103,64,108,64,118,6
64,129,64,147,64,161,64,171,9672
5340 DATA 64,32,62,65,169,3,162,16,157
7,66,3,169,8,157,74,3,5633
5350 DATA 169,171,157,68,3,169,65,157,
,69,3,169,0,157,75,3,32,5502
5360 DATA 86,228,48,43,169,11,162,16,1
157,66,3,169,45,157,68,3,6376
5370 DATA 169,62,157,69,3,169,246,157,
,72,3,169,0,157,73,3,32,6574
5380 DATA 86,228,48,11,162,71,160,113,
,32,143,49,32,62,65,96,32,5679
5390 DATA 62,65,162,181,160,65,32,143,
,49,96,162,16,169,12,157,66,8501
5400 DATA 3,32,86,228,96,32,62,65,169,
,3,162,16,157,66,3,169,7502
5410 DATA 171,157,68,3,169,65,157,69,3
3,169,0,157,75,3,169,4,6214
5420 DATA 157,74,3,32,86,228,48,199,16
69,7,162,16,157,66,3,169,8867
5430 DATA 45,157,68,3,169,62,157,69,3,
,169,246,157,72,3,169,0,8693
5440 DATA 157,73,3,32,86,228,48,222,17
73,30,63,240,13,169,181,141,2726
5450 DATA 35,67,169,121,141,36,67,76,4
40,65,169,80,141,35,67,169,8469
5460 DATA 122,141,36,67,76,40,65,68,58
8,83,80,89,46,68,65,84,5028
5470 DATA 155,70,105,108,101,32,101,11
14,114,111,114,46,32,0,32,78,4914
5480 DATA 66,32,222,65,96,46,32,39,63,
,33,44,45,58,47,40,41,2417
5490 DATA 48,49,50,51,52,53,54,55,56,5
57,169,0,133,206,169,80,8747
5500 DATA 133,207,169,0,133,208,169,11
12,133,209,160,0,177,206,240,32,4890
5510 DATA 201,31,240,46,201,30,240,32,
,201,27,208,5,169,255,145,208,5495
5520 DATA 96,144,10,56,233,28,170,189,
,201,65,76,20,66,24,105,96,8033
5530 DATA 145,208,32,70,66,76,238,65,3
32,56,66,177,206,24,105,64,8750
5540 DATA 208,238,32,56,66,177,206,170
0,189,203,65,76,20,66,24,169,344
5550 DATA 255,145,208,96,230,206,208,2
2,230,207,96,230,208,208,2,230,8243
5560 DATA 209,96,32,56,66,32,63,66,96,
,120,169,0,133,206,169,96,379
5570 DATA 133,207,169,0,133,208,169,80
0,133,209,160,7,169,0,153,192,3055
5580 DATA 3,136,16,250,160,4,177,206,1
153,208,3,136,16,248,162,7,1958
5590 DATA 169,0,160,6,110,208,3,110,20
09,3,110,210,3,110,211,3,9395
5600 DATA 110,212,3,106,136,192,1,24,2
208,234,106,106,106,157,192,3,1752
5610 DATA 202,16,223,160,7,185,192,3,1
145,208,136,16,248,160,7,185,3268
5620 DATA 192,3,201,27,240,32,136,16,2
246,165,206,24,105,5,133,206,2145
5630 DATA 165,207,105,0,133,207,165,20
08,24,105,8,133,208,165,209,105,4029
5640 DATA 0,133,209,76,104,66,96,238,5
53,111,54,32,55,2,56,66,5533
5650 DATA 56,4,57,65,57,144,57,232,57,
,27,58,81,58,238,53,51,7639
5660 DATA 59,90,59,165,59,32,55,165,59
9,66,56,86,118,157,118,157,9781
5670 DATA 118,195,118,232,118,13,119,1
157,118,66,119,107,119,3,120,3,7969
5680 DATA 120,13,119,169,119,219,119,3
3,120,33,120,104,120,164,120,190,2299
5690 DATA 120,227,120,7,121,66,121,170
0,122,106,121,3,120,132,121,181,1504
5700 DATA 121,208,121,250,121,43,122,1
170,122,170,122,170,122,174,115,248,65
572
5710 DATA 115,58,116,93,116,143,116,23
32,116,77,117,229,117,226,2,227,4963
5720 DATA 2,0,48,0,0,0,0,0,0,0,0,0,0,0
0,0,0,5866
```

LISTING 2: ASSEMBLY

```
0100 ; SAVE#D:SPYPT1.M65
0110 ;
0120 ;------------------;
0130 ;   Secret Agent   ;
0140 ;     (c) 1988     ;
0150 ;  by Barry Kolbe  ;
0160 ;------------------;
0170      .OPT NO LIST
0180 ;
0190      .MACRO PRINT
0200      LDX # <%1
0210      LDY # >%1
0220      JSR PRINTE
0230      .ENDM
0240 ;
0250      *=  $B0
0260 SCR .DS 2          ;screen scroll
0270 SCR2 .DS 2         ;ditto
0280 ML   .DS 2         ;more indirects
0290 SL   .DS 2
0300 J    .DS 2
0310 X1   .DS 1         ;save X & Y
0320 X2   .DS 1
0330 Y1   .DS 1
0340 Y2   .DS 1
0350 IX   .DS 1         ;char counter
0360 XF   .DS 1
0370 VP   .DS 1         ;verb counter
0380 NP   .DS 1         ;noun counter
0390 DO   .DS 1         ;direct object
0400 PO   .DS 1         ;prep. object
0410 ERFLG .DS 1        ;error flag
0420 INTOF .DS 1        ;'INTO' flag
0430 WITHF .DS 1        ;'WITH' flag
0440 TEMP .DS 1
0450 EOLF .DS 1         ;in printing
0460 MAXLEN .DS 1       ;max input
0470 ROWX .DS 1         ;save rowcrs
0480 COLX .DS 1         ;save colcrs
0490 NOEND .DS 1
0495 INROOM .DS 1       ;objects in room
0500 ;
0510 GR0 =   $8C00      ;screen memory
0520 HOLD =  $8600      ;data backup
0530 CHSET = $8000      ;character set
0540 IBUF =  $8400      ;input buffer
0550 PBUF =  $8500      ;print buffer
0560 ;
0570 ROWCRS = $54       ;cursor row
0580 COLCRS = $55       ;& column
0590 RTCLOK = $14       ;clock
0600 CH  =   $02FC      ;key
0610 CAPS =  $02BE      ;caps flag
0620 CONSOL = $D01F     ;consol button
0630 RANDOM = $D20A     ;random number
0640 EOL =   $9B        ;end of line
```

SECRET AGENT: *Mission 1*

```
0650 BOTLIN = 17      ;scroll value
0660 ENLIN = $26      ;wrap around val
0670 NUMOBJ = 46      ;total # objects
0680 NUMSHO = 23      ;# obj can show
0690 NUMV =  18       ;number of verbs
0700 ;
0710     .INCLUDE #D:SPYPT3.M65
0720 ;
0730     *=  $6000
0740 ;
0750 START JSR KEEP   ;back up data
0760     JMP BEGIN
0770 ;
0780     .INCLUDE #D:SPYPT6.M65
0790 ;
0800 ;start things going
0810 ;
0820 BEGIN LDA #$61   ;lowercase
0830     STA CAPS
0840     LDA # >CHSET ;install
0850     STA $02F4    ;character set
0860     JSR CLRGR0   ;clear screen
0870     LDA # <GDLST ;install
0880     STA $0230    ;Display List
0890     LDA # >GDLST
0900     STA $0231
0910     LDA #$3E     ;enbale DMA
0920     STA $022F
0930     LDA #0       ;initialize
0940     STA WHERE    ;outside
0950     STA CARRY    ;carry nothing
0960     STA NOEND
0970     JSR HOME     ;home cursor
0980      PRINT  M47 ;print credits
0990      PRINT  M48
1000     INC ROWCRS  ;next line
1010     JSR SHOLOC  ;show room info
1020 ;
1030 ;Main loop
1040 ;
1050 DOIN JSR INPUT  ;get input
1060     LDA #1      ;shoot
1070     STA FK      ;shoot flag
1080     LDA IBUF+1  ;check for
1090     CMP #EOL    ;single letter
1100     BNE TRYCMD  ;command
1110     LDA IBUF
1120 ;
1130 ;Move from room to room
1140 ;
1150     LDX #5      ;a direction?
1160 CML1 CMP SINGLE,X
1170     BEQ GSING   ;yes
1180     DEX         ;try again
1190     BPL CML1
1200     BMI TRYSPC  ;try X,I,R,L
1210 GSING STX TEMP  ;save direction
1220     JSR APEROB  ;ape or robot?
1230     LDY WHERE   ;present loc
1240     LDA EXOF,Y  ;get offset
1250     CLC         ;add direction
1260     ADC TEMP    ;0-5
1270     TAY
1280     LDA EXITAB,Y ;get exit
1290     BPL GOOO    ;value:+ =ok
1300      PRINT  M8  ;'Can't go'
1310     JMP DOIN    ;more input
1320 GOOO STA WHERE  ;new room
1330     CMP #$21    ;escape?
1340     BNE GOOP    ;no
1350     JMP ENDGAM  ;see if won
1360 GOOP JSR SHOLOC ;show new room
1370     JSR CKDEAD  ;deadly room?
1380     JMP DOIN    ;more input
1390 ;
1400 ;Check other single letters
1410 ;
1420 TRYSPC CMP #'I  ;inventory?
1430     BNE TLOK
1440     JSR INVENT
1450     JMP DOIN
1460 ;
1470 TLOK CMP #'L    ;look at room?
1480     BNE TSAV
1490     JSR LOOK
1500     JMP DOIN
1510 ;
1520 TSAV CMP #'X    ;save data
1530     BNE TRYLOA  ;to disk?
1540     JSR SVEGAM
1550     JMP DOIN
1560 ;
1570 TRYLOA CMP #'R  ;retrieve data
1580     BNE TRYCMD
1590     JSR LOADGAM
1600     JSR SHOLOC  ;show room
1610     JMP DOIN
1620 ;
1630 ;Interpret sentence
1640 ;
1650 TRYCMD JSR INTPRET ;parse
1660     LDA ERFLG   ;error?
1670     BEQ AOK
1680      PRINT  M1  ;'What?'
1690     JMP DOIN    ;get input
1700 AOK LDA VP      ;get verb number
1710     ASL A       ;x2 for
1720     TAX         ;look up table
1730     LDA CMDTBL,X
1740     STA DOCM+1
1750     LDA CMDTBL+1,X
1760     STA DOCM+2
1770 DOCM JSR $FFFF  ;do verb
1780     JMP DOIN    ;more input
1790 ;
1800 ;Set the cursor at 2,1
1810 ;
1820 HOME LDA #1
1830     STA ROWCRS
1840     STA COLCRS
1850     INC COLCRS
1860     RTS
1870 ;
1880 ;Show the name of the room
1890 ;in the status line
1900 ;
1910 SHOLOC LDY #24  ;clear out
1920     LDA #0      line
1930 SC1 STA PLACE,Y
1940     DEY
1950     BPL SC1
1960     LDA WHERE   ;get room
1970     ASL A       ;x2 for table
1980     TAX
1990     LDA ROOMTAB,X ;use
2000     STA SL      ;indirect for
2010     LDA ROOMTAB+1,X ;for move
2020     STA SL+1
2030     LDY #0
2040 SC2 LDA (SL),Y  ;get byte
2050     BEQ SXT     ;done if 0
2060     JSR ASC2IC  ;Internal Code
2070     STA PLACE+2,Y ;on status
2080     INY         ;line
2090     BNE SC2
2100 SXT JSR LOOK    ;look at room
2110     RTS
2120 ;
2130 ;Look at a Room
2140 ;
2150 LOOK JSR DESCRIB ;describe it
2160     JSR SHOEXT  ;show exits
2170     JSR SHOOBJ  ;sho objects
2180     RTS
2190 ;
2200 ;Show objects in Room
2210 ;
2220 SHOOBJ LDX #0   ;'You see:'
2230 SHS LDA YOUS,X
2240     STA IBUF,X
2250     INX
2260     CPX #9
2270     BNE SHS     ;X = 9
2280     LDY #0      ;scan objects
2290 NN1 LDA OBJTBL,Y
2300     CMP WHERE   ;is it here?
2310     BEQ SOB     ;yes
2320 SOC INY         ;next object
2330     CPY #NUMSHO ;that all?
2340     BNE NN1     ;no
2350     CPX #9
2360     BEQ NOOB    ;you see nothing
2370     LDA #$20    ;zap last ','
2380     STA IBUF-2,X
2390     LDA #0      ;'EOL' marker
2400     STA IBUF-1,X
2410      PRINT  IBUF ;on screen
2420     RTS
2430 ;
2440 NOOB LDY #0     ;'Nothing'
2450 NOI LDA NOTH,Y
2460     STA IBUF,X
2470     BEQ NRT
2480     INX
2490     INY
2500     BNE NOI
2510 NRT  PRINT  IBUF ;print it
2520     RTS
```

```
2530 ;
2540 SOB TYA
2550     PHA          ;save Y
2560     ASL A        ;x2
2570     TAY          ;point to
2580     LDA OBJNAM+1,Y ;object's
2590     STA SL+1     ;name
2600     LDA OBJNAM,Y
2610     STA SL
2620     LDY #0       ;move name into
2630 ODM LDA (SL),Y   ;buffer
2640     BEQ ODN
2650     STA IBUF,X
2660     INX
2670     INY
2680     BNE ODM
2690 ODN LDA #',      ;add in ', '
2700     STA IBUF,X
2710     LDA #$20
2720     INX
2730     STA IBUF,X
2740     INX
2750     PLA          ;get Y back
2760     TAY
2770     JMP SOC      ;do more
2780 ;
2790 ;Show exits
2800 ;
2810 SHOEXT LDX #0    ;'The exits
2820 SK1 LDA M9,X     ;are:'
2830     STA IBUF,X
2840     INX
2850     CPX #15
2860     BNE SK1
2870     LDY WHERE    ;room #
2880     LDA EXOF,Y   ;offset into
2890     TAY          ;EXITAB
2900     LDA #0       ;ibuf counter
2910     STA X1       ;direction cntr
2920 SXX STY TEMP
2930     LDA EXITAB,Y ;get exits
2940     BMI ELQ      ;$FF = no
2950     STY TEMP
2960     LDA X1       ;get dir name
2970     ASL A        ;x2
2980     TAY
2990     LDA DIRECT,Y ;point to
3000     STA SL       ;name
3010     LDA DIRECT+1,Y
3020     STA SL+1
3030     LDY #0       ;move name
3040 ELP LDA (SL),Y   ;to buffer
3050     STA IBUF,X
3060     BEQ ELQ
3070     INX
3080     INY
3090     BNE ELP
3100 ELQ LDY TEMP
3110 NOEX INC X1      ;try next direct
3120     INY
3130     LDA X1
3140     CMP #6       ;is that all?
3150     BNE SXX
3160     LDA #$20     ;zap last ','
3170     STA IBUF-2,X
3180     LDA #0       ;set 'EOL'
3190     STA IBUF-1,X
3200      PRINT  IBUF ;print
3210     RTS
3220 ;
3230 ;Find the VERB & NOUN
3240 ;
3250 ;input format:
3260 ;  verb (the) object [with/into]
3270 ;         (the) object]
3280 ; (the) and [..] are optional
3290 ;
3300 ;if verb or first object are not
3310 ;recognized an error is returned
3320 ;
3330 ;spaces and 'the' are skipped
3340 ;
3350 INTPRET LDX #0   ;set variables
3360     STX IX       ;char counter
3370     STX VP       ;verb number
3380     STX NP       ;noun number
3390     STX INTOF    ;into and with
3400     STX WITHF    ;flags
3410     STX ERFLG    ;error flag
3420 ;
3430 ;search
3440 ;
3450     LDA # <VERBT ;point to
3460     STA SL       ;verb table
3470     LDA # >VERBT
3480     STA SL+1
3490 SV2 LDX IX       ;get counter
3500     LDY #0
3510 SV5 LDA (SL),Y   ;do the first
3520     CMP IBUF,X   ;3 letters
3530     BNE SV1      ;match?
3540     INX
3550     INY
3560     CPY #3
3570     BNE SV5      ;not done
3580     BEQ GOTV     ;got a verb
3590 SV1 INC VP       ;next verb
3600     LDA SL       ;add 3 to
3610     CLC          ;pointer
3620     ADC #3
3630     STA SL
3640     BCC SS1
3650     INC SL+1
3660 SS1 LDA VP       ;end of verbs?
3670     CMP #NUMV
3680     BNE SV2
3690 SV3 INC ERFLG    ;error
3700     RTS
3710 ;
3720 ;got verb
3730 ;
3740 GOTV LDA IBUF,X ;look for
3750     CMP #EOL     ;EOL or
3760     BEQ SV3      ;first space
3770     CMP #$20
3780     BEQ GOTSP
3790     INX
3800     BNE GOTV
3810 GOTSP JSR NSPACE ;find next char
3820     JSR SKPTHE   ;skip 'the'
3830     JSR NSPACE   ;find next char
3840 SH1 JSR FINDN    ;get noun
3850     BCS SV3      ;error
3860 SH2 LDA NP       ;save as
3870     STA DO       ;direct object
3880 ;
3890 ;find EOL or space
3900 ;
3910 SH3 LDA IBUF,X   ;if EOL done
3920     CMP #EOL
3930     BNE SH9
3940     RTS
3950 SH9 CMP #$20     ;find end of
3960     BEQ SH4      ;noun- i.e.
3970 SV8 INX          ;a space char
3980     BNE SH3
3990 SH4 INX
4000     STX IX
4010     JSR NSPACE   ;find non space
4020     JSR PREP     ;a preposition?
4030     JSR NSPACE   ;find non space
4040     JSR SKPTHE   ;skipping...
4050     JSR FINDN    ;last noun?
4060     LDA NP       ;as prep. object
4070     STA PO
4080     RTS
4090 ;
4100 ;Find a 'non' SPACE character
4110 ;
4120 NSPACE LDA IBUF,X
4130     CMP #$20
4140     BNE NS
4150     INX
4160     BNE NSPACE
4170 NS  STX IX       ;char counter
4180     RTS
4190 ;
4200 ;Skip the word 'the'
4210 ;
4220 SKPTHE LDY #0
4230 ST1 LDA THE,Y
4240     CMP IBUF,X
4250     BNE THER
4260     INX
4270     INY
4280     CPY #4
4290     BNE ST1
4300     STX IX
4310 THER RTS
4320 ;
4330 ;Check for 'INTO' or 'WITH'
4340 ;if found set flags
4350 ;
4360 PREP LDX IX
4370     LDY #0
4380 P2  LDA IBUF,X  ;'INTO'?
4390     CMP INTOB,Y
4400     BNE PREP1
```

```
4410      INX
4420      INY
4430      CPY #5
4440      BNE P2
4450      INC INTOF    ;flag
4460 P4   STX IX
4470      RTS          ;ok
4480 PREP1 LDA IBUF,X ;'WITH'?
4490      CMP WITHB,Y
4500      BNE P3
4510      INX
4520      INY
4530      CPY #5
4540      BNE PREP1
4550      INC WITHF    ;set flag
4560      JMP P4
4570 P3   RTS
4580 ;
4590 ;Get a description of a room
4600 ;
4610 DESCRIB LDA WHERE ;this room
4620      ASL A        ;x2
4630      TAX
4640      LDA DESTAB+1,X ;point to
4650      TAY          ;description
4660      LDA DESTAB,X
4670      TAX
4680      JSR PRINTE   ;print it
4690      LDA WHERE    ;room 1?
4700      CMP #1
4710      BNE D5S      ;special msg
4720       PRINT  M53
4730      RTS
4740 ;
4750 D5S  CMP #9       ;ape room?
4760      BNE DEOU
4770      LDA FI       ;in cage
4780      BEQ APER     ;yes
4790       PRINT  D9   ;'In cage'
4800      RTS
4810 APER LDA F6       ;ape alive?
4820      BEQ APE5
4830       PRINT  M37 ;'Charging'
4840      RTS
4850 APE5  PRINT  D20 ;'Dead ape'
4860 DEOU RTS
4870 ;
4880 ;Find the NOUN
4890 ;
4900 FINDN LDA # <NOUNT ;point to
4910      STA SL       ;noun table
4920      LDA # >NOUNT
4930      STA SL+1
4940      LDX #0       ;noun counter
4950      STX NP
4960 SN2  LDX IX       ;char counter
4970      LDY #0
4980 SN7  LDA (SL),Y   ;compare 3
4990      CMP IBUF,X   ;letters
5000      BNE SN1
5010      INX
5020      INY
5030      CPY #3       ;match 3?
5040      BNE SN7
5050      BEQ GOTN     ;yes
5060 SN1  INC NP       ;next noun
5070      LDA SL       ;add 3 to the
5080      CLC          ;pointer
5090      ADC #3
5100      STA SL
5110      BCC SN5
5120      INC SL+1
5130 SN5  LDA NP       ;at end of
5140      CMP #NUMOBJ ;nouns?
5150      BNE SN2      ;no
5160      SEC          ;error
5170      RTS
5180 GOTN CLC          ;got one
5190      RTS
5200 ;
5210 ;Beginning of Verb Handlers
5220 ; DO = direct object
5230 ; PO = prepositional object
5240 ;
5250 ;Get an object
5260 ;
5270 DGET LDA CARRY    ;carrying the
5280      CMP #6       ;maximum
5290      BCS NOGT     ;yes
5300      LDX DO       ;object number
5310      CPX #$2D     ;get all
5320      BEQ GETAL    ;yes
5330      CPX #$17     ;over maximum
5340      BCS GT       ;possible to
5350      LDA OBJTBL,X ;get
5360      CMP WHERE    ;is it here?
5370      BEQ IH       ;yes
5380      CMP #$80     ;you have it?
5390      BNE NTH      ;no
5400       PRINT  M11 ;'Have already'
5410      RTS
5420 NTH   PRINT  M3   ;'Not here'
5430      RTS
5440 IH   LDA #$80     ;take it
5450      STA OBJTBL,X
5460      INC CARRY    ;add 1
5470 TOOK  PRINT  M4   ;'Taken'
5480      JSR DOSCOR   ;check score
5490      RTS
5500 NOGT  PRINT  M7  ;'Can't carry'
5510      RTS
5520 GT    PRINT  M24 ;'Can't get'
5530      RTS
5540 ;
5550 ;Get all objects
5560 ;
5570 GETAL LDY #0      ;check if
5580 GET  LDA OBJTBL,Y ;here
5590      CMP WHERE
5600      BEQ TAK      ;yes take
5610      CPY #NUMSHO ;at end?
5620      BEQ TK1      ;yes
5630      INY
5640      BNE GET
5650 TK1  JMP PDON     ;'Done'
5660 TAK  LDX CARRY    ;carry more?
5670      CPX #$06
5680      BCS NOGT     ;no
5690      LDA #$80     ;take it
5700      STA OBJTBL,Y
5710      INC CARRY    ;add 1
5720      TYA          ;save Y
5730      PHA
5740      JSR TOOK     ;'Taken'
5750      PLA          ;get Y back
5760      TAY
5770      JMP GET      ;do more
5780 ;
5790 ;Drop object(s)
5800 ;
5810 INJ  INC INROOM   ;add 1
5820      BNE INK
5830 ;
5840 DDRO LDA #0       ;zero counter
5850      STA INROOM
5860      LDY #0       ;count objects
5870 INN  LDA OBJTBL,Y ;in room
5880      CMP WHERE
5890      BEQ INJ      ;here's one
5900 INK  INY          ;next object
5910      CPY #NUMSHO ;done?
5920      BNE INN
5930      LDA INROOM   ;at 6 or more?
5940      CMP #6
5950      BCC INL      ;ok
5960 INM   PRINT  M51 ;'No room'
5970      RTS
5980 INL  LDX DO       ;is it the
5990      CPX #$0A     ;dreaded vial?
6000      BNE PP1      ;no - whew
6010      JMP VDED     ;a goner
6020 PP1  CPX #$2D     ;drop all?
6030      BEQ DPAL     ;yup
6040      LDA OBJTBL,X ;have it?
6050      CMP #$80
6060      BEQ DH       ;yes
6070       PRINT  M5  ;'Don't have'
6080      RTS
6090 DH   LDA WHERE    ;put in room
6100      STA OBJTBL,X
6110      CPX #$12     ;was it scuba?
6120      BNE DDY
6130      LDA #1       ;reset flag
6140      STA FE
6150      BNE DDX
6160 DDY  CPX #$08     ;is it the mask?
6170      BNE DDX
6180      LDA #1       ;reset flag
6190      STA FH
6200 DDX  DEC CARRY    ;1 less to carry
6210 DP4   PRINT  M6   ;'Dropped'
6220      JSR DOSCOR   ;change score?
6230      RTS
6240 ;
6250 ;Drop all
```

```
6260 ;
6270 DPAL LDY #0      ;scan objects
6280 DP1 LDA OBJTBL,Y
6290     CMP #$80     ;have this 1?
6300     BEQ DP2      ;yes
6310 DP3 INY          ;next object
6320     CPY #NUMSHO  ;done?
6330     BNE DP1
6340     JMP TK1      ;'done'
6350 DP2 LDA INROOM   ;is there room?
6360     CMP #6       ;6 maximum
6370     BEQ INM      ;no room
6380     CPY #$0A     ;vial?
6390     BEQ VDED     ;dead!
6400     CMP #$08     ;gas mask?
6410     BNE NOGS
6420     LDA #1       ;flip flag
6430     STA FH
6440 NOGS CMP #$12    ;scuba gear
6450     BNE NOSC
6460     LDA #1       ;flip flag
6470     STA FE
6480 ;
6490 NOSC INC INROOM ;1 more in room
6500     DEC CARRY    ;carry 1 less
6510     TYA          ;save Y
6520     PHA
6530     JSR DP4      ;'Dropped'
6540     PLA
6550     TAY          ;get Y
6560     LDA WHERE    ;put in room
6570     STA OBJTBL,Y
6580     JMP DP3      ;more
6590 ;
6600 VDED PLA         ;virus dropped
6610     PLA
6620     LDA #7
6630     JMP DEAD
6640 ;
6650 ;
6660 ;Unlock is the same as OPEN
6670 ;
6680 DUNL JMP DOPE
6690 ;
6700 ;Open
6710 ;
6720 DOPE LDA DO      ;the box?
6730     CMP #7
6740     BNE DDF
6750     LDA OBJTBL+7 ;have it?
6760     CMP #$80
6770     BEQ GSS
6780     JMP HAVNT    ;'Haven't got'
6790 GSS LDA F9       ;if opned before
6800     BNE GD1
6810     JMP EE2      ;see 'Nothing'
6820 GD1 EOR #1       ;else set flag
6830     STA F9       ;so no reopen
6840 GGS LDA WHERE    ;gas mask here
6850     STA OBJTBL+8
6860     JMP LKSEE    ;show it
6870 ;
6880 DDF CMP #$1C     ;open desk?
6890     BNE OCC
6900     LDA WHERE    ;which room?
6910     CMP #1       ;entrance?
6920     BEQ OD1
6930     CMP #2       ;reception?
6940     BEQ OD2
6950     CMP #$19     ;office?
6960     BEQ OD9
6970 NOCAN  PRINT  M19 ;'Can't '
6980     RTS
6990 ;
7000 OD1 LDA F7       ;entrance desk
7010     BEQ ITOPN    ;already open?
7020     EOR #1
7030     STA F7
7040     LDA #1       ;badge &
7050     STA OBJTBL   ;nailfile
7060     STA OBJTBL+1
7070 OE1 JSR ITOPN    ;'Open'
7080 LKSEE JSR LOOK   ;show room
7090     RTS
7100 ;
7110 ITOPN  PRINT  M13 ;'It's open'
7120     RTS
7130 OD2 LDX #$1C     ;reception desk
7140     LDY #1       ;use nailfile
7150     JSR CKWITH   ;say 'WITH'
7160     LDA F1       ;opened flag
7170     BEQ ITOPN
7180     EOR #1
7190     STA F1
7200     LDA #2       ;show card
7210     STA OBJTBL+2
7220     BNE OE1
7230 ;
7240 OD9 LDX #$1C     ;office desk
7250     LDY #$15     ;large key
7260     JSR CKWITH   ;'WITH'?
7270     LDA FN       ;open flag
7280     BEQ ITOPN
7290     EOR #1
7300     STA FN
7310     LDA #$19     ;show:
7320     STA OBJTBL+$10 ;coin
7330     STA OBJTBL+$16 ;combination
7340     JMP OE1
7350 ;
7360 OCC CMP #$1B     ;cabinet
7370     BNE NOCAN
7380     LDA WHERE
7390     CMP #$19     ;office?
7400     BNE OCK
7410     LDX #$1B     ;cabinet
7420     LDY #$11     ;small key
7430     JSR CKWITH   ;'WITH'
7440     LDA F8       ;open flag
7450     BEQ ITOPN
7460     EOR #1
7470     STA F8
7480     LDA #$19
7490     STA OBJTBL+$15 ;large key
7500     JMP OE1
7510 ;
7520 OCK CMP #4       ;kitchen?
7530     BEQ KOK
7540     JMP NOCAN    ;'Can't'
7550 KOK LDX #$1B     ;cabinet
7560     LDY #$16     ;combination
7570     JSR CKWITH   ;'WITH'
7580     LDA FM       ;open flag
7590     BEQ ITOPN
7600     EOR #1
7610     STA FM
7620     LDA #4
7630     STA OBJTBL+$0A ;vial
7640     JMP OE1
7650 ;
7660 NOTYT  PRINT  M14 ;'Not yet!'
7670     RTS
7680 ;
7690 ;Read
7700 ;
7710 DREA LDA DO      ;what shall
7720     TAY          ;we read?
7730     LDA OBJTBL,Y
7740     CMP #$80     ;have it?
7750     BEQ DORE
7760      PRINT  M5  ;'Can't read'
7770     RTS
7780 DORE LDA DO
7790     CMP #3       ;pad
7800     BEQ RPD
7810     CMP #9       ;folder
7820     BEQ RFO
7830     CMP #$0C     ;memo
7840     BNE NORE
7850      PRINT  M22 ;'Tighten
7860     RTS          ;security'
7870 RPD  PRINT  M20 ;'Push button'
7880     RTS
7890 RFO  PRINT  M21 ;'Experiments'
7900     RTS
7910 NORE  PRINT  M23 ;'Can't read'
7920     RTS
7930 ;
7940 ;Insert (Put) into
7950 ;
7960 DINS LDA WHERE
7970     CMP #5       ;security?
7980     BNE IRB
7990     LDX #2       ;card
8000     LDY #$1E     ;slot
8010     JSR CKINTO   ;say 'INTO'
8020     LDA #5
8030     STA OBJTBL+5 ;got a gun
8040     LDA #$FF     ;lose card
8050     STA OBJTBL+2
8060 DINR DEC CARRY  ;1 less
8070     JSR PDON     ;'Done'
8080     JMP LKSEE    ;show
8090 ;
8100 IRB CMP #$0B     ;hallway
8110     BNE IRC
```

SECRET AGENT: *Mission 1*

```
8120      LDX #0       ;badge
8130      LDY #$1E     ;slot
8140      JSR CKINTO   ;'INTO'
8150      LDA #$0B
8160      STA OBJTBL+6 ;baton
8170      LDA #$FF
8180      STA OBJTBL   ;lose badge
8190      JMP DINR
8200 ;
8210 IRC  CMP #$0C     ;cafeteria
8220      BNE IRR      ;try rod
8230      LDX #$10     ;coin
8240      LDY #$22     ;vending
8250      JSR CKINTO   ;'INTO'
8260      LDA #$0C
8270      STA OBJTBL+$0B ;formula
8280      LDA #$FF
8290      STA OBJTBL+$10 ;lose coin
8300      JMP DINR
8310 ;
8320 IRR  CMP #$12     ;security
8330      BNE IRD      ;try disk
8340      LDX #$0F     ;rod
8350      LDY #$2C     ;hole
8360      JSR CKINTO   ;'INTO'
8370      LDA #$FF
8380      STA OBJTBL+$0F ;lose rod
8390      LDA #$15     ;gain exit
8400      STA EXITAB+111
8410      JMP DINR
8420 ;
8430 IRD  CMP #$14     ;computer room
8440      BNE IRM      ;try memo
8450      LDX #4       ;disk
8460      LDY #$2B     ;drive
8470      JSR CKINTO   ;'INTO'
8480      LDA #$14     ;program
8490      STA OBJTBL+$0E
8500      LDA #$FF     ;no disk
8510      STA OBJTBL+4
8520      JMP DINR
8530 ;
8540 IRM  CMP #$15     ;photo lab
8550      BNE IRO
8560      LDX #$0C     ;memo
8570      LDY #$2A     ;enlarger
8580      JSR CKINTO   ;'INTO'
8590      LDA #$15
8600      STA OBJTBL+$0D ;microdot
8610      STA OBJTBL+$0C ;memo
8620      JMP DINR
8630 IRO  JMP NOCAN    ;'Can't'
8640 ;
8650 ;Check DO, PO , ownershp
8660 ;and for the word 'INTO'
8670 ; Is X= DO, Y = PO?
8680 ;
8690 CKINTO CPX DO     ;X = DO?
8700      BNE NTFIT
8710      CPY PO       ;Y = PO?
8720      BNE NTFIT
8730      LDA OBJTBL,X ;have X?
8740      CMP #$80
8750      BNE UNV
8760      LDA INTOF    ;said 'INTO'?
8770      BEQ WRA
8780      RTS          ;sure go ahead
8790 NTFIT  PRINT  M16 ;'Won't fit'
8800      PLA
8810      PLA          ;pull return
8820      RTS
8830 UNV  PLA          ;pull return
8840      PLA
8850      JMP HAVNT    ;'Don't have it'
8860 WRA  PLA          ;pull return
8870      PLA
8880      JMP NOTYT    ;'Not yet'
8890 ;
8900      .INCLUDE #D:SPYPT2.M65
8910      .INCLUDE #D:SPYPT4.M65
8920      .INCLUDE #D:SPYPT5.M65
8930      *=  $02E0
8940      .WORD START
```

LISTING 3: ASSEMBLY

```
0100 ;SAVE#D:SPYPT2.M65
0110 ;
0120 ;-------------------------;
0130 ;  Secret Agent  (c) 1988  ;
0140 ;  By: Barry Kolbe         ;
0150 ;  Second half of Main     ;
0160 ;  program                 ;
0170 ;-------------------------;
0180 ;
0190 ;Wear scuba gear or gas mask
0200 ;
0210 DWEA LDA DO
0220      CMP #$12     ;scuba?
0230      BNE WTA      ;no
0240      LDA OBJTBL+$12 ;have it?
0250      CMP #$80
0260      BEQ WTB      ;yes
0270 HAVNT  PRINT  M5 ;'haven't got'
0280      RTS
0290 ;
0300 WTB  LDA #0       ;set flag
0310      STA FE
0320       PRINT  M17 ;'Wearing ..'
0330      RTS
0340 ;
0350 WTA  CMP #$08     ;gas mask?
0360      BNE WTC      ;no
0370      LDA OBJTBL+$08 ;own it?
0380      CMP #$80
0390      BNE HAVNT    ;no
0400      LDA #0       ;set wear flag
0410      STA FH
0420       PRINT  M31 ;'Wearing...'
0430      RTS
0440 WTC  JMP NOCAN    ;'Can't do that'
0450 ;
0460 ;Turn
0470 ;
0480 DTUR LDA WHERE    ;room?
0490      CMP #$13     ;stairwell
0500      BNE TR1      ;nope
0510      LDA DO
0520      CMP #$18     ;hook?
0530      BEQ TR2      ;yes
0540 TR3  JMP NOCAN    ;'Can't'
0550 TR2  LDA #$14     ;new exit west
0560      STA EXITAB+117
0570      JMP LKSEE    ;show
0580 ;
0590 TR1  CMP #$18     ;stairwell
0600      BNE TR4      ;no
0610      LDA DO
0620      CMP #$17     ;knob
0630      BNE TR3      ;'Can't'
0640      LDA FD       ;flip flag
0650      EOR #1       ;electric floor
0660      STA FD
0670 TR5  JMP PCLIK    ;say 'Click'
0680 TR4  CMP #$0E     ;stairwell
0690      BEQ TR5      ;just 'Click'
0700      CMP #$0A     ;west end?
0710      BEQ TR5      ;yes
0720 TR6  CMP #9       ;ape room?
0730      BNE TR3      ;no
0740      LDA DO
0750      CMP #$17     ;knob
0760      BNE TR3      ;'Can't'
0770      LDA #0       ;loose ape!
0780      STA FI
0790      JSR PCLIK    ;'Click'
0800      LDA #$20     ;south exit
0810      STA EXITAB+55 ;now
0820      JMP LKSEE
0830 ;
0840 ;flip switch
0850 ;
0860 DFLI LDA WHERE    ;room?
0870      CMP #$0F     ;office?
0880      BNE FCH      ;no
0890      LDA FA       ;is safe visible
0900      BNE FEV      ;yes
0910      LDA DO
0920      CMP #$19     ;switch?
0930      BEQ FES      ;yes
0940 FEV  JMP NOSPEC   ;'Nothng happns
0950 FES  LDA OBJTBL+$13 ;have it
0960      CMP #$80
0970      BEQ FET      ;yes
0980      JMP HAVNT
0990 FET  LDA FB       ;opened before?
1000      BNE FEU      ;no
1010      JMP PDON     ;'Done'
1020 FEU  EOR #1       ;set flag
1030      STA FB
1040      LDA #$0F     ;show rod
1050      STA OBJTBL+$0F
```

```
1060      PRINT  M29 ;'Safe open'
1070     JSR PCLIK    ;'Click'
1080     JMP LK5EE
1090 ;
1100 FCH CMP #$10     ;chem lab?
1110     BNE FEV      ;no
1120     LDA DO
1130     CMP #$19     ;switch?
1140     BNE FEV      ;no
1150     LDA FG       ;X-ray machine
1160     EOR #1
1170     STA FG       ;fall through
1180 PCLIK  PRINT  M28 ;'.Click..'
1190     RTS
1200 ;
1210 ;Examine something
1220 ;
1230 DEXA LDA DO
1240     CMP #$13     ;electronic
1250     BNE EE1      ;device?
1260     LDA OBJTBL+$13
1270     CMP #$80
1280     BNE EVB
1290      PRINT  M30 ;'On/Off switch'
1300     RTS
1310 EE1 CMP #$06     ;baton?
1320     BNE EE2
1330     LDA OBJTBL+6
1340     CMP #$80
1350     BEQ EE3
1360 EVB JMP HAVNT    ;'Haven't got'
1370 EE3  PRINT  M54 ;'Gravity
1380     RTS          ;switch'
1390 EE2  PRINT  M12 ;'Nothing '
1400     RTS
1410 ;
1420 ;Wave an object
1430 ;
1440 DWAV LDX DO      ;have it?
1450     LDA OBJTBL,X
1460     CMP #$80
1470     BEQ WA1      ;yes
1480     JMP HAVNT    ;'Don't have'
1490 WA1 CPX #6       ;baton?
1500     BNE NOSPEC   ;'Nothing'
1510     LDA WHERE
1520     CMP #$17     ;in right room?
1530     BEQ WA4
1540 NOSPEC  PRINT  M18 ;'Nothing
1550     RTS          ;special'
1560 WA4 LDA FF       ;flip flag
1570     EOR #1
1580     STA FF
1590     BEQ WA2
1600     LDA #$FF     ;close exit
1610 WA3 STA EXITAB+140
1620     JMP LK5EE
1630 WA2 LDA #$10     ;open east
1640     BNE WA3      ;exit
1650     RTS
1660 ;
1670 ;Shoot the ___ with the gun
1680 ;
1690 DSHO LDA #0      ;message flag
1700     STA FK       ;for CKWITH
1710     LDA DO
1720     CMP #$1A     ;shoot robot?
1730     BNE GAP      ;no
1740 SF2 LDA #1       ;set flags
1750     STA WITHF    ;auto WITH
1760     LDA #5       ;gun
1770     STA PO       ;as prep object
1780     JMP SF1      ;check it out
1790 GAP CMP #$24     ;shoot ape?
1800     BEQ SF2      ;yes
1810     LDA DO       ;shoot gun?
1820     CMP #$05
1830     BNE SF1
1840      PRINT  M50 ;'At what?'
1850     RTS
1860 SF1 LDA WHERE
1870     CMP #$1A     ;in robot room?
1880     BNE SAPE     ;no
1890     LDX #$1A     ;robot=DO
1900     LDY #$05     ;gun =PO
1910     JSR CKWITH   ;check it
1920     LDA FJ       ;robot alive?
1930     BNE SS3      ;yes
1940 SS5 JMP WHAFOR  ;'What for?'
1950 SS3 LDA RANDOM  ;see if hit
1960     CMP #100
1970     BCS SS2      ;Miss
1980     LDA #0       ;a hit
1990     STA FJ       ;robot dead
2000     LDA # <D1F   ;change message
2010     STA ROBROM
2020     LDA # >D1F
2030     STA ROBROM+1
2040     JMP LK5EE    ;show it
2050 ;
2060 SS2  PRINT  M33 ;'You missed'
2070     LDA RANDOM   ;see if robot
2080     CMP #180     ;hits
2090     BCC SS4
2100      PRINT  M35 ;robot missed
2110     RTS
2120 SS4 LDA #0       ;gotcha!
2130     JMP DEAD
2140 ;
2150 SAPE CMP #9      ;ape room?
2160     BEQ SAP1     ;yes
2170     JMP NOCAN    ;'Can't do that'
2180 SAP1 LDX #$24    ;ape =DO
2190     LDY #5       ;gun =PO
2200     JSR CKWITH   ;check mostly
2210     LDA FI       ;for ownership
2220     BEQ SJ1      ;is ape out?
2230     JMP NOTYT    ;no->'Not yet'
2240 SJ1 LDA F6       ;alive ape?
2250     BEQ SS5      ;no
2260     LDA RANDOM   ;hit or miss?
2270     CMP #165
2280     BCC SA2
2290     LDA #0       ;one dead ape
2300     STA F6
2310     JMP LK5EE    ;tell us all
2320 SA2  PRINT  M33 ;'You missed'
2330     LDA RANDOM   ;see if ape
2340     CMP #150     ;gets you
2350     BCS SA3      ;yes
2360      PRINT  M37 ;'Charging'
2370     RTS
2380 SA3 LDA #1       ;'Crushed!'
2390     JMP DEAD
2400 ;
2410 ;Check: correct Direct Object,
2420 ;Prepositional Object, use of
2430 ;'WITH' and ownership of PO
2440 ;Enter with X=test value of DO
2450 ;Y= test value of PO
2460 ;
2470 CKWITH CPX DO    ;X = DO?
2480     BNE WP1      ;nawh
2490     CPY PO       ;Y = PO?
2500     BNE WP2
2510     LDA OBJTBL,Y ;have Y?
2520     CMP #$80
2530     BNE WP3      ;no
2540     LDA WITHF    ;say 'WITH'?
2550     BEQ WP2      ;no
2560     RTS          ;go ahead
2570 WP1 PLA          ;pull return
2580     PLA
2590     JMP WHAFOR   ;'What for?'
2600 ;
2610 WP2 PLA          ;pull return
2620     PLA
2630     LDA FK       ;message flag
2640     BEQ WP5
2650     JMP NOTYT    ;'Not yet'
2660 WP5 JMP NOCAN    ;'Can't do'
2670 ;
2680 WP3 PLA
2690     PLA
2700     JMP HAVNT    ;'Haven't got'
2710 ;
2720 ;Lift the plant
2730 ;
2740 DLIF LDA WHERE   ;in stairs?
2750     CMP #$0D
2760     BNE WHAFOR   ;no!
2770     LDA DO
2780     CMP #$26     ;plant?
2790     BNE WHAFOR   ;so why do it?
2800     LDA F5       ;lift before?
2810     BEQ WHAFOR   ;yes
2820     EOR #1       ;set flag
2830     STA F5
2840     LDA #$0D     ;show key
2850     STA OBJTBL+$11
2860     JMP LK5EE    ;show us
2870 WHAFOR  PRINT  M27 ;'What for?'
2880     RTS
2890 ;
2900 ;Move an object
2910 ;
```

```
2920 DMOV LDA WHERE
2930      CMP #$0F     ;office?
2940      BNE MVC      ;no
2950      LDA DO
2960      CMP #$29     ;picture?
2970      BEQ MV2      ;yup
2980 MV1 JMP NOCAN     ;no can do
2990 MV2 LDA FA        ;safe already
3000      BEQ PDON     ;visible?
3010      EOR #1       ;no. set flag
3020      STA FA
3030      LDA # <D5AF  ;say it
3040      STA SAFROM
3050      LDA # >D5AF
3060      STA SAFROM+1
3070      JMP LK5EE
3080 ;
3090 PDON  PRINT  M2 ;'Done'
3100      RTS
3110 ;
3120 MVC CMP #$1B      ;storage?
3130      BNE MV1      ;no
3140      LDA DO
3150      CMP #$1B     ;cabinet?
3160      BNE MV1      ;no
3170      LDA FC       ;moved before?
3180      BEQ PDON     ;yes
3190      EOR #1
3200      STA FC       ;set flag
3210      LDA #$1D     ;show exit down
3220      STA EXITAB+167
3230      JMP LK5EE
3240 ;
3250 ;Push (or Press) buttons
3260 ;
3270 DPU5 LDA WHERE    ;storage?
3280      CMP #7
3290      BNE P52      ;no
3300      LDA DO
3310      CMP #$27     ;red button?
3320      BNE P53
3330      LDA #0       ;turn off lasers
3340 P54 STA F2
3350      JMP PCLIK    ;'Click'
3360 P53 CMP #$28      ;blue?
3370      BEQ P51      ;yes
3380 P56 JMP NOCAN     ;'Can't '
3390 P51 LDA #1        ;turn lasers on
3400      BNE P54
3410 P52 CMP #$1D      ;small room
3420      BNE P56      ;(airlock)
3430      LDA DO
3440      CMP #$28     ;blue button
3450      BNE P56      ;no
3460      LDA #$1E     ;open east
3470      STA EXITAB+176
3480      LDA #$FF     ;close north
3490      STA EXITAB+174
3500      J5R PCLIK    ;'Click'
3510      JMP LK5EE
3520 ;
3530 ;Player's Inventory
3540 ;
3550 INVENT LDX #0     ;'You have:'
3560 ILL LDA YOUH,X
3570      STA IBUF,X
3580      INX
3590      CPX #10
3600      BNE ILL
3610      LDY #0       ;see which ones
3620 ILQ LDA OBJTBL,Y ;you own
3630      CMP #$80
3640      BNE ILN      ;no this one
3650      TYA
3660      PHA          ;save Y
3670      A5L A        ;x2 for an
3680      TAY          ;offset
3690      LDA OBJNAM,Y ;into names
3700      STA 5L
3710      LDA OBJNAM+1,Y
3720      STA 5L+1
3730      LDY #0       ;move name
3740 ILM LDA (5L),Y    ;to buffer
3750      BEQ ILP      ;done at 0
3760      STA IBUF,X
3770      INX
3780      INY
3790      BNE ILM
3800 ILP LDA #',       ;put in ', '
3810      STA IBUF,X
3820      INX
3830      LDA #$20
3840      STA IBUF,X
3850      INX
3860      PLA          ;get Y
3870      TAY
3880 ILN INY           ;next object
3890      CPY #NUM5HO  ;at end?
3900      BNE ILQ      ;no
3910      CPX #10      ;X still 10?
3920      BNE ILW      ;no
3930      LDY #0       ;must have
3940 NOH LDA NOTH,Y    ;'Nothing!'
3950      STA IBUF,X
3960      INX
3970      INY
3980      CPY #10
3990      BNE NOH
4000      JMP ILU      ;print it
4010 ILW LDA #0        ;put in 'EOL'
4020      STA IBUF-1,X
4030      LDA #$20     ;erase last ,
4040      STA IBUF-2,X
4050 ILU  PRINT  IBUF ;print it
4060      RTS
4070 ;
4080 ;You have failed the mission
4090 ;Enter with A = type of death
4100 ;
4110 DEAD A5L A        ;x2
4120      TAX          ;offset
4130      LDA DEADTAB+1,X ;get message
4140      TAY          ;high byte
4150      LDA DEADTAB,X
4160      TAX          ;low of address
4170      J5R PRINTE   ;print it
4180       PRINT  M41 ;'Failed'
4190      JMP WHANOW
4200 ;
4210 ;Before allowing a turn see if
4220 ;the player is in the ape or
4230 ;robot room. Don't allow them to
4240 ;move out.
4250 ;
4260 APEROB LDA WHERE
4270      CMP #$1A     ;with robot?
4280      BNE DOAPE    ;no
4290      LDA FJ       ;robot alive?
4300      BNE EYA      ;yes
4310      RTS          ;whew!
4320 EYA J5R INTPRET ;get command
4330      LDA ERFLG    ;test for error
4340      BNE EYB      ;error
4350      LDA VP       ;get verb #
4360      CMP #$0A     ;shoot?
4370      BNE EYB      ;nope
4380      JMP D5HO     ;goto shoot
4390 EYB LDA RANDOM   ;robot shoots
4400      CMP #200
4410      BCC EYC
4420      LDA #0       ;hit you
4430      JMP DEAD     ;done for
4440 EYC  PRINT  M35 ;robot misses
4450      PLA          ;pull return
4460      PLA
4470      JMP DOIN     ;get input
4480 ;
4490 DOAPE CMP #9      ;ape room?
4500      BNE NOEN     ;no
4510      LDA FI       ;in cage?
4520      BNE NOEN     ;yes
4530      LDA F6       ;alive?
4540      BEQ NOEN     ;dead
4550      J5R INTPRET ;get command
4560      LDA ERFLG
4570      BNE EYD      ;error
4580      LDA VP       ;get verb #
4590      CMP #$0A     ;shoot?
4600      BNE EYD      ;no
4610      JMP D5HO     ;do shoot
4620 EYD LDA RANDOM   ;see if ape
4630      CMP #130     ;gets you
4640      BC5 NOEN     ;escape for now
4650      PLA
4660      PLA
4670      LDA #1       ;crushes you
4680      JMP DEAD
4690 NOEN RTS
4700 ;
4710 ;Check for traps in
4720 ;various rooms
4730 ;
4740 CKDEAD LDA WHERE
4750      CMP #5       ;hallway?
4760      BNE DI2
4770      LDA F2       ;lasers on?
4780      BEQ CDOK     ;no
4790      LDA #5       ;blasted!
```

```
4800      JMP DEAD
4810 CDOK RTS          ;ok so far
4820 DI2 CMP #$17      ;security?
4830      BNE DI3
4840      LDA FD       ;electric floor
4850      BEQ CDOK     ;ok
4860      LDA #2       ;Zzzzzit
4870      JMP DEAD
4880 DI3 CMP #$11      ;X-ray room?
4890      BNE DI4
4900      LDA FG       ;switch
4910      BNE CDOK     ;off!
4920      LDA #4       ;radiation
4930      JMP DEAD
4940 DI4 CMP #$12      ;security?
4950      BNE DI5
4960      LDA FH       ;gas mask on?
4970      BEQ CDOK     ;yes
4980      LDA #3       ;poison gas
4990      JMP DEAD
5000 DI5 CMP #$1E      ;tunnel?
5010      BNE CDOK
5020      LDA FE       ;wearing scuba?
5030      BEQ CDOK
5040      LDA #6       ;drowned
5050      JMP DEAD
5060 ;
5070 ;Check the status of objects
5080 ;and show the score
5090 ;
5100 DOSCOR LDA #0     ;zero
5110      STA SCORE
5120      LDY #4       ;check for the
5130 DYA LDA OBJTBL+$0A,Y
5140      CMP #$80     ;4 objects
5150      BEQ UPO
5160 DYB DEY
5170      BPL DYA
5180      LDA OBJTBL+$0C ;don't
5190      CMP #$80     ;count memo
5200      BNE DYC      ;next time
5210      DEC SCORE    ;(-1 for memo)
5220 DYC LDA SCORE     ;I'll put them
5230      ORA #$10     ;next to one
5240      STA PLACE+37 ;another
5250      RTS          ;show score
5260 UPO INC SCORE     ;plus 1
5270      JMP DYB
5280 ;
5290 ;See if you've won after an
5300 ;escape from the lab
5310 ;
5320 ENDGAM  PRINT  M42 ;'Escaped'
5330      INC ROWCRS
5340      LDA SCORE    ;win?
5350      CMP #4
5360      BNE NOWIN
5370       PRINT  M44 ;You won!
5380 ENL JMP ENL
5390 NOWIN  PRINT  M41 ;not quite
5400      JMP WHANOW
5410 ;
5420 ;Save initial condition in
5430 ;the event of a restart
5440 ;
5450 KEEP LDA # <SAVST ;start of
5460      STA J        ;data
5470      LDA # >SAVST
5480      STA J+1
5490      LDA # <HOLD  ;backup area
5500      STA ML
5510      LDA # >HOLD
5520      STA ML+1
5530 KK3 LDY #0        ;one byte at
5540      LDA (J),Y    ;a time
5550      STA (ML),Y
5560      INC ML       ;bump pointers
5570      BNE KK1
5580      INC ML+1
5590 KK1 INC J
5600      BNE KK2
5610      INC J+1
5620 KK2 LDA J+1       ;at the
5630      CMP # >SAVEN ;end yet?
5640      BNE KK3
5650      LDA J
5660      CMP # <SAVEN
5670      BNE KK3
5680      RTS          ;done
5690 ;
5700 ;Ok. Something happened to
5710 ;bring us here. Either press
5720 ;START or SYSTEM RESET
5730 ;
5740 WHANOW  PRINT  M49 ;Press START
5750 NNW LDA CONSOL    ;check consol
5760      CMP #6       ;START?
5770      BEQ REDO     ;try again
5780      BNE NNW
5790 REDO LDA # <SAVST ;move data
5800      STA J        ;back to
5810      LDA # >SAVST ;where the
5820      STA J+1      ;program can
5830      LDA # <HOLD  ;use it
5840      STA ML
5850      LDA # >HOLD
5860      STA ML+1
5870 KR1 LDY #0
5880      LDA (ML),Y
5890      STA (J),Y
5900      INC ML
5910      BNE KR2
5920      INC ML+1
5930 KR2 INC J
5940      BNE KR3
5950      INC J+1
5960 KR3 LDA J+1       ;done yet?
5970      CMP # >SAVEN
5980      BNE KR1
5990      LDA J
6000      CMP # <SAVEN
6010      BNE KR1
6020      JMP BEGIN    ;start over
6030 ;
6040 ;Data
6050 ;
6060 SINGLE .BYTE "NSEWUD"
6070 INTOB .BYTE "INTO "
6080 WITHB .BYTE "WITH "
6090 THE .BYTE "THE "
6100 ;
6110 ;Table of Room Names
6120 ;
6130 ROOMTAB .WORD R0 ;0 outside
6140      .WORD R1     ;1 entrance
6150      .WORD R2     ;2 reception
6160      .WORD R3     ;3 conference
6170      .WORD R4     ;4 kitchn
6180      .WORD R5     ;5 security
6190      .WORD R6     ;6 experiment
6200      .WORD R7     ;7 storage
6210      .WORD R7     ;8 storage
6220      .WORD R8     ;9 animal
6230      .WORD R9     ;a west end
6240      .WORD RA     ;b hallway
6250      .WORD RB     ;c cafeteria
6260      .WORD RC     ;d stairwell
6270      .WORD RC     ;e stairwell
6280      .WORD RD     ;f office
6290      .WORD RE     ;10 chem lab
6300      .WORD RF     ;11 x-ray
6310      .WORD R5     ;12 security
6320      .WORD RC     ;13 stairwell
6330      .WORD RG     ;14 computer
6340      .WORD RH     ;15 photo lab
6350      .WORD R7     ;16 storage
6360      .WORD R5     ;17 security
6370      .WORD RC     ;18 stairwell
6380      .WORD RD     ;19 office
6390      .WORD R5     ;1a security
6400      .WORD R7     ;1b storage
6410      .WORD RI     ;1c electronic
6420      .WORD RJ     ;1d small(air)
6430      .WORD RK     ;1e tunnel
6440      .WORD RL     ;spacer not use
6450      .WORD RL     ;20 cage
6460 ;
6470 ;Direction table for names
6480 ;
6490 DIRECT .WORD XNOR ;north
6500      .WORD XSOU   ;south
6510      .WORD XEAS   ;east
6520      .WORD XWES   ;west
6530      .WORD XUP    ;up
6540      .WORD XDOW   ;down
6550 ;
6560 SAVST ;           start of save
6570 ;
6580 WHERE .DS 1       ;current room
6590 CARRY .DS 1       ;# objs carried
6600 SCORE .DS 1       ;what else?
6610 ;
6620 ;Object Location Table
6630 ;
6640 ;0-31 = In a Room
6650 ;$FF  = Not Visible
6660 ;$80  = Owned by Player
6670 ;
```

```
6680 OBJTBL .BYTE $FF ;badge
6690      .BYTE $FF   ;nailfile
6700      .BYTE $FF   ;card
6710      .BYTE 3     ;pad
6720      .BYTE $20   ;disk    5
6730      .BYTE $FF   ;gun
6740      .BYTE $FF   ;baton
6750      .BYTE 8     ;box
6760      .BYTE $FF   ;gas mask
6770      .BYTE 6     ;folder   10
6780      .BYTE $FF   ;vial
6790      .BYTE $FF   ;formula
6800      .BYTE $0F   ;memo
6810      .BYTE $FF   ;microdot
6820      .BYTE $FF   ;program  15
6830      .BYTE $FF   ;rod
6840      .BYTE $FF   ;coin
6850      .BYTE $FF   ;small key
6860      .BYTE $16   ;scub
6870      .BYTE $1C   ;elec     20
6880      .BYTE $1C   ;flash
6890      .BYTE $FF   ;lar k
6900      .BYTE $FF   ;combo last 1
6910 ;
6920 ;Table of Exits
6930 ;
6940 ;$FF=noexit # = rooms
6950 ;
6960 EXITAB .BYTE $FF,$FF,$FF,$01
6970      .BYTE $FF,$FF ;0 outside
6980      .BYTE $02,$03,$FF,$05
6990      .BYTE $FF,$FF ;1 entrance
7000      .BYTE $FF,$01,$FF,$FF
7010      .BYTE $FF,$FF ;2 reception
7020      .BYTE $01,$07,$FF,$FF
7030      .BYTE $FF,$FF ;3 conference
7040      .BYTE $FF,$05,$FF,$0C
7050      .BYTE $FF,$FF ;4 kitchen
7060      .BYTE $04,$FF,$01,$0B
7070      .BYTE $FF,$FF ;5 security
7080      .BYTE $FF,$08,$FF,$09
7090      .BYTE $FF,$FF ;6 experiment
7100      .BYTE $03,$FF,$FF,$FF
7110      .BYTE $FF,$FF ;7 storage
7120      .BYTE $06,$FF,$FF,$FF
7130      .BYTE $FF,$FF ;8 storage
7140      .BYTE $0A,$FF,$06,$FF
7150      .BYTE $FF,$FF ;9 animal
7160      .BYTE $0D,$09,$0B,$FF
7170      .BYTE $FF,$FF ;a west end
7180      .BYTE $0C,$FF,$05,$0A
7190      .BYTE $FF,$FF ;b hallway
7200      .BYTE $FF,$0B,$04,$FF
7210      .BYTE $FF,$FF ;c cafeteria
7220      .BYTE $FF,$0A,$FF,$FF
7230      .BYTE $0E,$18 ;d stairs
7240      .BYTE $FF,$FF,$17,$FF
7250      .BYTE $FF,$0D ;e stairs
7260      .BYTE $17,$16,$FF,$FF
7270      .BYTE $FF,$FF ;f office
7280      .BYTE $FF,$FF,$11,$17
7290      .BYTE $FF,$FF ;10 chem lab
7300      .BYTE $FF,$12,$FF,$10
7310      .BYTE $FF,$FF ;11 x-ray lab
7320      .BYTE $11,$13,$FF,$FF
7330      .BYTE $FF,$FF ;12 security
7340      .BYTE $12,$FF,$FF,$FF
7350      .BYTE $FF,$07 ;13 stairs
7360      .BYTE $FF,$FF,$13,$FF
7370      .BYTE $FF,$FF ;14 computer
7380      .BYTE $10,$FF,$12,$FF
7390      .BYTE $FF,$FF ;15 photo lab
7400      .BYTE $0F,$FF,$FF,$FF
7410      .BYTE $FF,$FF ;16 storage
7420      .BYTE $FF,$0F,$FF,$0E
7430      .BYTE $FF,$FF ;17 security
7440      .BYTE $FF,$1A,$FF,$FF
7450      .BYTE $0D,$FF ;18 stairs
7460      .BYTE $FF,$1C,$FF,$FF
7470      .BYTE $FF,$FF ;19 office
7480      .BYTE $18,$1B,$1C,$FF
7490      .BYTE $FF,$FF ;1a security
7500      .BYTE $1A,$FF,$FF,$FF
7510      .BYTE $FF,$FF ;1b storage
7520      .BYTE $19,$FF,$FF,$1A
7530      .BYTE $FF,$FF ;1c electronic
7540      .BYTE $1B,$FF,$FF,$FF
7550      .BYTE $FF,$FF ;1d small room
7560      .BYTE $FF,$FF,$21,$FF
7570      .BYTE $FF,$FF ;1e tunnel
7580      .BYTE $FF,$FF,$FF,$FF
7590      .BYTE $FF,$FF ;$21 out!
7600      .BYTE $09,$FF,$FF,$FF
7610      .BYTE $FF,$FF ;$20 cage
7620 ;
7630 ;Flag Table
7640 ;
7650 F1   .BYTE 1     ;desk R2 lock
7660 F2   .BYTE 1     ;lasers R5 on
7670 F3   .BYTE 1     ;vend mach R4
7680 F5   .BYTE 1     ;plant Rd key
7690 F6   .BYTE 1     ;ape alive R9
7700 F7   .BYTE 1     ;desk R1
7710 F8   .BYTE 1     ;cabinet R19
7720 F9   .BYTE 1     ;box - gas
7730 FA   .BYTE 1     ;picture moved
7740 FB   .BYTE 1     ;safe Rf
7750 FC   .BYTE 1     ;cabinet moved
7760 FD   .BYTE 1     ;elect. floor
7770 FE   .BYTE 1     ;wear scuba
7780 FF   .BYTE 1     ;door
7790 FG   .BYTE 1     ;x-rays
7800 FH   .BYTE 1     ;wear gas mask
7810 FI   .BYTE 1     ;ape in cage
7820 FJ   .BYTE 1     ;robot alive
7830 FK   .BYTE 1     ;shoot msg
7840 FM   .BYTE 1     ;used combinat
7850 FN   .BYTE 1     ;used large key
7860 ;
7870 SAVEN ;          end of save
7880 ;
7890 ;Tables of offsets into
7900 ;the table of Exits
7910 ;Multiples of 6
7920 ;
7930 EXOF .BYTE 0,6,12,18,24
7940      .BYTE 30,36,42,48,54
7950      .BYTE 60,66,72,78,84
7960      .BYTE 90,96,102,108,114
7970      .BYTE 120,126,132,138,144
7980      .BYTE 150,156,162,168,174
7990      .BYTE 180,186,192,198,204
8000 ;
8010 ;Verb Table - only the
8020 ;first 3 letters are used
8030 ;
8040 VERBT .BYTE "GETDROOPEREAINS"
8050      .BYTE "WEATURFLIEXAWAVSHO"
8060      .BYTE "TAKLIFMOVPUSUNLPRE"
8070      .BYTE "PUT"
8080 ;
8090 ;Noun Table - first 3 letters
8100 ;
8110 NOUNT .BYTE "BADNAICARPADDIS"
8120      .BYTE "GUNBATBOXGASFOLVIA"
8130      .BYTE "FORMEMMICPRORODCOI"
8140      .BYTE "SMASCUELEFLALARCOM"
8150      .BYTE "KNOHOOSWIROBCABDES"
8160      .BYTE "SAFSLOCAGLEVLOCVEN"
8170      .BYTE "MACAPEDOOPLAREDBLU"
8180      .BYTE "PICENLDRIHOLALL"
8190 ;
8200 ;Objects
8210 ;
8220 OB0 .BYTE "badge",0
8230 OB1 .BYTE "nailfile",0
8240 OB2 .BYTE "card",0
8250 OB3 .BYTE "pad",0
8260 OB4 .BYTE "disk",0
8270 OB5 .BYTE "gun",0
8280 OB6 .BYTE "baton",0
8290 OB7 .BYTE "box",0
8300 OB8 .BYTE "gas mask",0
8310 OB9 .BYTE "folder",0
8320 OBA .BYTE "vial",0
8330 OBB .BYTE "formula",0
8340 OBC .BYTE "memo",0
8350 OBD .BYTE "microdot",0
8360 OBE .BYTE "program",0
8370 OBF .BYTE "rod",0
8380 OBG .BYTE "coin",0
8390 OBH .BYTE "small key",0
8400 OBI .BYTE "scuba gear",0
8410 OBJ .BYTE "electronic device"
8420      .BYTE 0
8430 OBK .BYTE "flashing ball",0
8440 OBL .BYTE "large key",0
8450 OBM .BYTE "combination",0
8460 ;
8470 ;Table for the object names
8480 ;
8490 OBJNAM .WORD OB0
8500      .WORD OB1
8510      .WORD OB2
8520      .WORD OB3
8530      .WORD OB4
```

```
8540      .WORD OB5
8550      .WORD OB6
8560      .WORD OB7
8570      .WORD OB8
8580      .WORD OB9
8590      .WORD OBA
8600      .WORD OBB
8610      .WORD OBC
8620      .WORD OBD
8630      .WORD OBE
8640      .WORD OBF
8650      .WORD OBG
8660      .WORD OBH
8670      .WORD OBI
8680      .WORD OBJ
8690      .WORD OBK
8700      .WORD OBL
8710      .WORD OBM
```

LISTING 4: ASSEMBLY

```
0100 ;SAVE#D:SPYPT3.M65
0110 ;
0120 ;-----------------------;
0130 ; Cursive Character Set ;
0140 ; for Secret Agent      ;
0150 ;-----------------------;
0160 ;
0170 ;
0180 ;This is the first half
0190 ;of the Character Set
0200 ;
0210      *=  CHSET
0220      .BYTE $00,$00,$00,$00
0230      .BYTE $00,$00,$00,$00
0240      .BYTE $00,$18,$18,$18
0250      .BYTE $18,$00,$18,$00
0260      .BYTE $00,$66,$66,$66
0270      .BYTE $00,$00,$00,$00
0280      .BYTE $00,$66,$FF,$66
0290      .BYTE $66,$FF,$66,$00
0300      .BYTE $18,$3E,$60,$3C
0310      .BYTE $06,$7C,$18,$00
0320      .BYTE $00,$66,$66,$18
0330      .BYTE $30,$66,$46,$00
0340      .BYTE $1C,$36,$1C,$38
0350      .BYTE $6F,$66,$3B,$00
0360      .BYTE $00,$18,$18,$18
0370      .BYTE $00,$00,$00,$00
0380      .BYTE $00,$0E,$1C,$18
0390      .BYTE $18,$1C,$0E,$00
0400      .BYTE $00,$70,$38,$18
0410      .BYTE $18,$38,$70,$00
0420      .BYTE $00,$66,$3C,$FF
0430      .BYTE $3C,$66,$00,$00
0440      .BYTE $00,$18,$18,$7E
0450      .BYTE $18,$18,$00,$00
0460      .BYTE $00,$00,$00,$00
0470      .BYTE $00,$18,$18,$30
0480      .BYTE $00,$00,$00,$7E
0490      .BYTE $00,$00,$00,$00
0500      .BYTE $00,$00,$00,$00
0510      .BYTE $00,$18,$18,$00
0520      .BYTE $00,$06,$0C,$18
0530      .BYTE $30,$60,$40,$00
0540      .BYTE $00,$3C,$66,$6E
0550      .BYTE $76,$66,$3C,$00
0560      .BYTE $00,$18,$38,$18
0570      .BYTE $18,$18,$7E,$00
0580      .BYTE $00,$3C,$66,$0C
0590      .BYTE $18,$30,$7E,$00
0600      .BYTE $00,$7E,$0C,$18
0610      .BYTE $0C,$66,$3C,$00
0620      .BYTE $00,$0C,$1C,$3C
0630      .BYTE $6C,$7E,$0C,$00
0640      .BYTE $00,$7E,$60,$7C
0650      .BYTE $06,$66,$3C,$00
0660      .BYTE $00,$3C,$60,$7C
0670      .BYTE $66,$66,$3C,$00
0680      .BYTE $00,$7E,$06,$0C
0690      .BYTE $18,$30,$30,$00
0700      .BYTE $00,$3C,$66,$3C
0710      .BYTE $66,$66,$3C,$00
0720      .BYTE $00,$3C,$66,$3E
0730      .BYTE $06,$0C,$38,$00
0740      .BYTE $00,$00,$18,$18
0750      .BYTE $00,$18,$18,$00
0760      .BYTE $00,$00,$18,$18
0770      .BYTE $00,$18,$18,$30
0780      .BYTE $06,$0C,$18,$30
0790      .BYTE $18,$0C,$06,$00
0800      .BYTE $00,$00,$7E,$00
0810      .BYTE $00,$7E,$00,$00
0820      .BYTE $60,$30,$18,$0C
0830      .BYTE $18,$30,$60,$00
0840      .BYTE $00,$3C,$66,$0C
0850      .BYTE $18,$00,$18,$00
0860      .BYTE $00,$3C,$66,$6E
0870      .BYTE $6E,$60,$3E,$00
0880      .BYTE $00,$1E,$37,$67
0890      .BYTE $67,$6F,$3B,$00
0900      .BYTE $00,$1E,$33,$73
0910      .BYTE $7E,$73,$7F,$00
0920      .BYTE $00,$1E,$33,$60
0930      .BYTE $60,$70,$3F,$00
0940      .BYTE $00,$3C,$66,$63
0950      .BYTE $63,$63,$7E,$00
0960      .BYTE $00,$1E,$33,$60
0970      .BYTE $7C,$60,$7F,$00
0980      .BYTE $00,$1E,$33,$60
0990      .BYTE $7C,$60,$60,$00
1000      .BYTE $00,$1E,$33,$60
1010      .BYTE $6E,$63,$3E,$00
1020      .BYTE $00,$63,$63,$63
1030      .BYTE $7F,$63,$63,$00
1040      .BYTE $00,$7F,$18,$18
1050      .BYTE $18,$18,$7F,$00
1060      .BYTE $00,$03,$03,$03
1070      .BYTE $73,$36,$3C,$00
1080      .BYTE $00,$67,$6E,$7C
1090      .BYTE $7C,$6C,$6F,$00
1100      .BYTE $00,$70,$70,$60
1110      .BYTE $60,$63,$7F,$00
1120      .BYTE $00,$63,$63,$77
1130      .BYTE $7F,$6B,$63,$00
1140      .BYTE $00,$7C,$76,$76
1150      .BYTE $76,$76,$77,$00
1160      .BYTE $00,$1C,$36,$63
1170      .BYTE $63,$36,$1C,$00
1180      .BYTE $00,$1E,$33,$33
1190      .BYTE $3E,$30,$30,$00
1200      .BYTE $00,$1C,$36,$63
1210      .BYTE $63,$6F,$3E,$03
1220      .BYTE $00,$3C,$36,$36
1230      .BYTE $3E,$33,$33,$00
1240      .BYTE $00,$1E,$33,$60
1250      .BYTE $3E,$03,$7F,$00
1260      .BYTE $00,$3F,$6C,$6C
1270      .BYTE $0C,$0C,$1B,$00
1280      .BYTE $00,$33,$33,$33
1290      .BYTE $33,$63,$3E,$00
1300      .BYTE $00,$63,$63,$63
1310      .BYTE $36,$3C,$18,$00
1320      .BYTE $00,$63,$63,$6B
1330      .BYTE $7F,$77,$63,$00
1340      .BYTE $00,$63,$66,$3C
1350      .BYTE $1C,$36,$63,$00
1360      .BYTE $00,$63,$63,$36
1370      .BYTE $1E,$0C,$18,$00
1380      .BYTE $00,$3F,$66,$0C
1390      .BYTE $18,$33,$7E,$00
1400      .BYTE $00,$1E,$18,$18
1410      .BYTE $18,$18,$1E,$00
1420      .BYTE $00,$40,$60,$30
1430      .BYTE $18,$0C,$06,$00
1440      .BYTE $00,$78,$18,$18
1450      .BYTE $18,$18,$78,$00
1460      .BYTE $00,$08,$1C,$36
1470      .BYTE $63,$00,$00,$00
1480      .BYTE $00,$00,$00,$00
1490      .BYTE $00,$00,$FF,$00
1500      .BYTE $00,$36,$7F,$7F
1510      .BYTE $3E,$1C,$08,$00
1520      .BYTE $18,$18,$18,$1F
1530      .BYTE $1F,$18,$18,$18
1540      .BYTE $03,$03,$03,$03
1550      .BYTE $03,$03,$03,$03
1560      .BYTE $18,$18,$18,$F8
1570      .BYTE $F8,$00,$00,$00
1580      .BYTE $18,$18,$18,$F8
1590      .BYTE $F8,$18,$18,$18
1600      .BYTE $00,$00,$00,$F8
1610      .BYTE $F8,$18,$18,$18
1620      .BYTE $03,$07,$0E,$1C
1630      .BYTE $38,$70,$E0,$C0
1640      .BYTE $C0,$E0,$70,$38
1650      .BYTE $1C,$0E,$07,$03
1660      .BYTE $01,$03,$07,$0F
1670      .BYTE $1F,$3F,$7F,$FF
1680      .BYTE $00,$00,$00,$00
1690      .BYTE $0F,$0F,$0F,$0F
1700      .BYTE $80,$C0,$E0,$F0
```

continued on page 56

```
1710      .BYTE $F8,$FC,$FE,$FF
1720      .BYTE $0F,$0F,$0F,$0F
1730      .BYTE $00,$00,$00,$00
1740      .BYTE $F0,$F0,$F0,$F0
1750      .BYTE $00,$00,$00,$00
1760      .BYTE $FF,$FF,$00,$00
1770      .BYTE $00,$00,$00,$00
1780      .BYTE $00,$00,$00,$00
1790      .BYTE $00,$00,$FF,$FF
1800 ;
1810 ;This is the fourth part
1820 ;of the Character Set
1830 ;I skipped the control
1840 ;characters
1850 ;
1860      *=  CHSET+$0300
1870 ;
1880      .BYTE $00,$18,$3C,$7E
1890      .BYTE $7E,$3C,$18,$00
1900      .BYTE $00,$00,$1E,$36
1910      .BYTE $76,$76,$9F,$00
1920      .BYTE $00,$60,$60,$7C
1930      .BYTE $66,$66,$FD,$00
1940      .BYTE $00,$00,$3C,$70
1950      .BYTE $60,$70,$DF,$00
1960      .BYTE $00,$06,$06,$3E
1970      .BYTE $66,$66,$FF,$00
1980      .BYTE $00,$00,$1C,$36
1990      .BYTE $36,$1C,$F7,$00
2000      .BYTE $00,$1C,$38,$30
2010      .BYTE $3E,$7B,$D9,$1C
2020      .BYTE $00,$00,$1E,$33
2030      .BYTE $73,$DE,$87,$3C
2040      .BYTE $00,$60,$60,$60
2050      .BYTE $7C,$66,$E7,$00
2060      .BYTE $00,$18,$00,$18
2070      .BYTE $18,$7E,$C3,$00
2080      .BYTE $00,$1C,$00,$1C
2090      .BYTE $3C,$6F,$CD,$3C
2100      .BYTE $00,$30,$30,$3E
2110      .BYTE $36,$3C,$F7,$00
2120      .BYTE $00,$18,$18,$18
2130      .BYTE $18,$18,$E7,$00
2140      .BYTE $00,$00,$33,$7F
2150      .BYTE $7F,$DB,$DB,$00
2160      .BYTE $00,$00,$7C,$66
2170      .BYTE $66,$66,$E7,$00
2180      .BYTE $00,$00,$3C,$66
2190      .BYTE $67,$E7,$3C,$00
2200      .BYTE $00,$00,$7C,$66
2210      .BYTE $66,$7C,$E7,$60
2220      .BYTE $00,$00,$3E,$66
2230      .BYTE $66,$7E,$8F,$0E
2240      .BYTE $00,$00,$60,$7E
2250      .BYTE $66,$66,$C3,$00
2260      .BYTE $00,$00,$0E,$1B
2270      .BYTE $33,$63,$DF,$00
2280      .BYTE $00,$0C,$3F,$0C
2290      .BYTE $0C,$1C,$F7,$00
2300      .BYTE $00,$00,$73,$33
2310      .BYTE $33,$33,$DF,$00
2320      .BYTE $00,$00,$36,$36
2330      .BYTE $36,$7F,$CD,$00
2340      .BYTE $00,$00,$63,$63
2350      .BYTE $6B,$7F,$DD,$00
2360      .BYTE $00,$00,$33,$7E
2370      .BYTE $6E,$DB,$B1,$00
2380      .BYTE $00,$00,$1B,$1B
2390      .BYTE $1F,$36,$CF,$18
2400      .BYTE $00,$00,$00,$66
2410      .BYTE $5B,$DB,$83,$3E
2420      .BYTE $00,$18,$3C,$7E
2430      .BYTE $7E,$18,$3C,$00
2440      .BYTE $18,$18,$18,$18
2450      .BYTE $18,$18,$18,$18
2460      .BYTE $00,$7E,$70,$7C
2470      .BYTE $6E,$66,$06,$00
2480      .BYTE $08,$18,$38,$78
2490      .BYTE $38,$18,$08,$00
2500      .BYTE $10,$18,$1C,$1E
2510      .BYTE $1C,$18,$10,$00
```

LISTING 5: ASSEMBLY

```
0100 ;SAVE#D:SPYPT4.M65
0110 ;
0120 ;--------------------;
0130 ;    Secret Agent    ;
0140 ; By: Barry Kolbe    ;
0150 ; Disk I/O routines  ;
0160 ; (c) 1988           ;
0170 ;--------------------;
0180 ;
0190 ;CIO equates
0200 ;
0210 ICCOM = $0342   ;command
0220 ICBAL = $0344   ;buffer address
0230 ICBAH = $0345
0240 ICBLL = $0348   ;buffer length
0250 ICBLH = $0349
0260 AUX1 =  $034A   ;type access
0270 AUX2 =  $034B
0280 CIO =   $E456
0290 ;
0300 SVEGAM JSR CLOSE1 ;close IOCB 1
0310      LDA #3       ;open
0320      LDX #$10     ;channel 1
0330      STA ICCOM,X
0340      LDA #8       ;to write
0350      STA AUX1,X
0360      LDA # <FNAM ;file name
0370      STA ICBAL,X
0380      LDA # >FNAM
0390      STA ICBAH,X
0400      LDA #0
0410      STA AUX2,X
0420      JSR CIO      ;open it!
0430      BMI IOERR    ;show error
0440      LDA #$0B     ;put bytes
0450      LDX #$10     ;channel 1
0460      STA ICCOM,X
0470      LDA # <SAVST ;starting
0480      STA ICBAL,X ;address
0490      LDA # >SAVST
0500      STA ICBAH,X
0510      LDA # <[SAVEN-SAVST+1]
0520      STA ICBLL,X ;number of
0530      LDA # >[SAVEN-SAVST+1]
0540      STA ICBLH,X ;bytes
0550      JSR CIO      ;save data
0560      BMI IOERR    ;woops!
0570 IODON  PRINT  M2 ;'done'
0580      JSR CLOSE1  ;close file
0590      RTS
0600 IOERR JSR CLOSE1 ;close IOCB 1
0610       PRINT  IOMSG ;say 'Error'
0620      RTS
0630 ;
0640 CLOSE1 LDX #$10 ;close IOCB 1
0650      LDA #$0C     ;close commnd
0660      STA ICCOM,X
0670      JSR CIO
0680      RTS
0690 ;
0700 LOADGAM JSR CLOSE1 ;close IOCB 1
0710      LDA #3       ;open
0720      LDX #$10
0730      STA ICCOM,X
0740      LDA # <FNAM
0750      STA ICBAL,X
0760      LDA # >FNAM
0770      STA ICBAH,X
0780      LDA #0
0790      STA AUX2,X
0800      LDA #4       ;read
0810      STA AUX1,X
0820      JSR CIO
0830 IOJ  BMI IOERR    ;'Error'
0840      LDA #7       ;get bytes
0850      LDX #$10
0860      STA ICCOM,X
0870      LDA # <SAVST ;starting
0880      STA ICBAL,X ;address
0890      LDA # >SAVST
0900      STA ICBAH,X ;get number
0910      LDA # <[SAVEN-SAVST+1]
0920      STA ICBLL,X ;of bytes
0930      LDA # >[SAVEN-SAVST+1]
0940      STA ICBLH,X
0950      JSR CIO
0960      BMI IOJ      ;'Error'
0970 ;
0980 ;Reset proper Message
0990 ;
1000      LDA FJ       ;robot dead?
1010      BEQ RBK      ;yes
1020      LDA # <D1A   ;say 'guarding'
1030      STA ROBROM
1040      LDA # >D1A
1050      STA ROBROM+1
1060      JMP IODON
1070 RBK  LDA # <D1F   ;'mangled
1080      STA ROBROM   ;robot'
1090      LDA # >D1F
```

```
1100      STA ROBROM+1
1110      JMP IODON
1120 ;
1130 FNAM .BYTE "D:SPY.DAT",EOL
1140 IOMSG .BYTE "File error. ",0
```

LISTING 6: ASSEMBLY

```
0100 ;SAVE#D:SPYPT4.M65
0120 ;
0130 ;--------------------;
0140 ;   Secret Agent     ;
0150 ;     (c) 1988       ;
0160 ;  by Barry Kolbe    ;
0170 ;--------------------;
0180 ;
0190 ;Table of Verbs (Commands)
0200 ;
0210 CMDTBL .WORD DGET ;get
0220      .WORD DDRO  ;drop
0230      .WORD DOPE  ;open
0240      .WORD DREA  ;read
0250      .WORD DINS  ;insert
0260      .WORD DWEA  ;wear
0270      .WORD DTUR  ;turn
0280      .WORD DFLI  ;flip
0290      .WORD DEXA  ;examine
0300      .WORD DWAV  ;wave
0310      .WORD DSHO  ;shoot
0320      .WORD DGET  ;take
0330      .WORD DLIF  ;lift
0340      .WORD DMOV  ;move
0350      .WORD DPUS  ;push
0360      .WORD DOPE  ;unlock
0370      .WORD DPUS  ;press
0380      .WORD DINS  ;put ($12)
0390 ;
0400 ;Table of Room Descriptions
0410 ;
0420 DESTAB .WORD D0
0430      .WORD D1
0440      .WORD D1
0450      .WORD D3
0460      .WORD D4
0470      .WORD D5
0480      .WORD D1
0490      .WORD D7
0500      .WORD D8
0510 APEROM .WORD DE
0520      .WORD DE
0530      .WORD D5
0540      .WORD DC
0550      .WORD DD
0560      .WORD DE
0570 SAFROM .WORD DF
0580      .WORD D10
0590      .WORD D11
0600      .WORD D12
0610      .WORD D13
0620      .WORD D14
0630      .WORD D15
0640      .WORD DFAKE
0650      .WORD D17
0660      .WORD DE
0670      .WORD D19
0680 ROBROM .WORD D1A
0690      .WORD D1B
0700      .WORD D1C
0710      .WORD D1D
0720      .WORD DFAKE
0730      .WORD DFAKE
0740      .WORD DFAKE
0750 ;
0760 ;Room Names
0770 ;
0780 R0   .BYTE "Outside",0
0790 R1   .BYTE "Entrance",0
0800 R2   .BYTE "Reception area",0
0810 R3   .BYTE "Conference room",0
0820 R4   .BYTE "Kitchen",0
0830 R5   .BYTE "Security area",0
0840 R6   .BYTE "Experiment room",0
0850 R7   .BYTE "Storage room",0
0860 R8   .BYTE "Animal room",0
0870 R9   .BYTE "West end of hallway"
0880      .BYTE 0
0890 RA   .BYTE "Hallway",0
0900 RB   .BYTE "Cafeteria",0
0910 RC   .BYTE "Stairwell",0
0920 RD   .BYTE "Office",0
0930 RE   .BYTE "Chemical Lab",0
0940 RF   .BYTE "X-ray Lab",0
0950 RG   .BYTE "Computer Lab",0
0960 RH   .BYTE "Photo Lab",0
0970 RI   .BYTE "Electronics Lab",0
0980 RJ   .BYTE "Small room",0
0990 RK   .BYTE "Tunnel",0
1000 RL   .BYTE "Cage",0
1010 ;
1020 ;Messages
1030 ;
1040 XNOR .BYTE "North, ",0
1050 XSOU .BYTE "South, ",0
1060 XEAS .BYTE "East, ",0
1070 XWES .BYTE "West, ",0
1080 XUP  .BYTE "Up, ",0
1090 XDOW .BYTE "Down, ",0
1100 ;
1110 YOUH .BYTE "You have: "
1120 NOTH .BYTE "Nothing ",0
1130 YOUS .BYTE "You see: "
1140 M1   .BYTE "What?",0
1150 M2   .BYTE "Done.",0
1160 M3   .BYTE "It's not here. ",0
1170 M4   .BYTE "Taken. ",0
1180 M5   .BYTE "You don't have that"
1190      .BYTE ". ",0
1200 M6   .BYTE "Dropped. ",0
1210 M7   .BYTE "You can't carry "
1220      .BYTE "anymore. ",0
1230 M8   .BYTE "You can't go that "
1240      .BYTE "way. ",0
1250 M9   .BYTE "The exits are: ",0
1260 M10  .BYTE "You see: ",0
1270 M11  .BYTE "You already have "
1280      .BYTE "that. ",0
1290 M12  .BYTE "You see nothing "
1300      .BYTE "special. ",0
1310 M13  .BYTE "It's open. ",0
1320 M14  .BYTE "Not yet. ",0
1330 M16  .BYTE "It won't fit. ",0
1340 M17  .BYTE "You are wearing the"
1350      .BYTE " scuba gear. ",0
1360 M18  .BYTE "Nothing happens. "
1370      .BYTE 0
1380 M19  .BYTE "You can't do tha"
1390      .BYTE "t. ",0
1400 M20  .BYTE "'The blue button tu"
1410      .BYTE "rns off the lasers"
1420      .BYTE ".' ",0
1430 M21  .BYTE "The title is: The "
1440      .BYTE "Effect of Bio-Agents"
1450      .BYTE " on Mammals. ",0
1460 M22  .BYTE "It says: 'Tighten "
1470      .BYTE "security on all fl"
1480      .BYTE "oors.' ",0
1490 M23  .BYTE "You can't read tha"
1500      .BYTE "t. ",0
1510 M24  .BYTE "You can't get tha"
1520      .BYTE "t! ",0
1530 M25  .BYTE "You can't open t"
1540      .BYTE "hat. ",0
1550 M27  .BYTE "What for? ",0
1560 M28  .BYTE ".. Click.. ",0
1570 M29  .BYTE "The safe opens. ",0
1580 M30  .BYTE "There's an ON/OFF s"
1590      .BYTE "witch on it. ",0
1600 M31  .BYTE "You are wearing the"
1610      .BYTE " gas mask. ",0
1620 M33  .BYTE "You missed. ",0
1630 M34  .BYTE "A laser blast from "
1640      .BYTE "the robot atomizes "
1650      .BYTE "you! ",0
1660 M35  .BYTE "The robot fires, but"
1670      .BYTE " misses. ",0
1680 M36  .BYTE "The ape crushes your"
1690      .BYTE "bones.. Crack.. ",0
1700 M37  .BYTE "The ape is charging "
1710      .BYTE "at you! ",0
1720 M38  .BYTE "The floor is electr"
1730      .BYTE "ified. Zzzit.. ",0
1740 M39  .BYTE "You choke as poison"
1750      .BYTE "ed gas seeps into y"
1760      .BYTE "our lungs! ",0
1770 M40  .BYTE "Deadly radiation mel"
1780      .BYTE "ts your flesh. ",0
1790 M41  .BYTE "You failed your miss"
1800      .BYTE "ion. ",0
1810 M42  .BYTE "You escaped from the"
1820      .BYTE " lab. ",0
1830 M43  .BYTE "Suddenly laser burst"
1840      .BYTE "s cut through you. "
```

continued on page 82

DISK DIRECTORY ALPHABETIZER

by Craig J. Stadler

Just about every computer user has a program in his jumble of files that alphabetizes a disk directory for printing or displaying on the screen, right? Yes, but do they *really* alphabetize the directory? Probably not! The *Disk Directory Alphabetizer (DDA)* is the real thing. It actually alphabetizes the on-disk directory, and then writes the new directory and patches to the disk—not just a temporary display.

It actually alphabetizes the on-disk directory, and then writes the new directory and patches to the disk—not just a temporary display.

What?

If you don't fully understand, here is an explanation. Most other directory alphabetizing programs operate in this manner: 1) read the directory into memory, 2) alphabetize the directory in memory and 3) display and or print the directory. And that's that!

However, the *DDA* operates in this manner: 1) read the directory into memory, 2) alphabetize the directory in memory, 3) write the alphabetized directory to disk and 4) read, patch and write all file and sector data necessary for proper operation.

First there is something we must clear up. Although *DDA* rewrites file and sector data, it does not write over information data, only sector data such as file number and next sector. Most people feel that it is not possible to achieve what *DDA* performs because of No. 4. We will go into this later when we discuss how *DDA* works.

How to Use DDA

After typing Listing 1 (and *please* check your typing with BASIC Editor II on page 54; typing mistakes in this program could result in serious damage to your disk files), run the program. You will be given three choices:

C)hange Drive or V)erify: Use this option to change your drive number to any other than Drive 1 (default). Verify is a feature that checks for errors when writing to a disk.

P)rocess Directory: Press "P" to begin the program.

D)isk Operating System: Press "D" to go to DOS. Default settings are Drive-1 and Verify On.

Change defaults if necessary and press "P" to begin. The first prompt is self-explanatory. Remember, make sure there is a write protect tab on the disk. At this point *DDA* reads and alphabetizes the directory. The screen shuts off so the computer can operate faster. Wait for the second prompt.

The second prompt is the data-write

prompt, so be sure to follow the instructions, and don't forget to take the write protect tab off the disk; if you don't, *DDA* won't be able to write the new directory to the disk. First, *DDA* writes the directory back onto the disk and then modifies the necessary file and sector patches.

How Can This Be So Complex?

Reading and writing a disk directory should not be a problem for the experienced programmer. In the case of *DDA*, one would logically think that it would be possible to just read, alphabetize and write the directory. But it's not that easy. You see, Atari DOS and the Volume Table of Contents operate rather differently from other disk operating systems.

When you write a file to your disk, this is what happens:

1) The VTOC finds space on the disk.

2) The filename is written onto the end of the directory listing and the VTOC remembers which file number was assigned to that file. The file is then written onto the disk where ever space is available. When the file is written, there are a few bytes of data at the end of each 128/125 byte sector. These bytes tell what file number the file sector belongs to, next sector and number of bytes of data contained in the sector. The few bytes of informational data for use by the VTOC are written on every sector of data for that particular file.

According to several manuals and magazines, a normal data sector contains 125 actual data bytes. Byte #125 signifies the number of bytes used in the sector. Byte #126 signifies the file number and the next sector number (high two bits). Byte #127 signifies the next sector number (low eight bits). After reading and hopefully understanding these facts, you can understand why a simple read/alphabetize/write process is ridiculous.

When you alphabetize the directory, the file numbers do not match up any longer! When VTOC and DOS try to access any of these files, you will get ERROR 164. This is why *DDA* performs all the patches.

How Does DDA Patch?

There are even more difficulties. One cannot just go into a data sector and change Byte #126 (file number and next sector number) because these two work together along with Byte #127. Now, *DDA* must read the start sector of the first file, calculate the next sector, change Bytes 126 and 127, read the next sector and so on until that particular file is finished being modified. After this, *DDA* goes to the next file and so on. By using algebra and some friendly help, as some programmers do *(Ahem!)*, one can calculate a formula for changing Bytes 127 and 128 to get the desired result.

Reading and writing a disk directory should not be a problem for the experienced programmer. But it's not that easy.

One Last Thing

If any of you are considering using *DDA* on your RAMdisk, there are doubts about it working with memory rather than disk sectors. Be careful!

The author would like to thank Alicia Crowell and Christopher Gibson.

LISTING 1: BASIC

```
AE 10 REM ******************************
VM 20 REM *       DISK DIRECTORY       *
HN 30 REM *         ALPHABETIZER       *
CK 40 REM *     by Craig J. Stadler    *
FW 50 REM *                            *
BS 60 REM *       COPYRIGHT 1989       *
BB 70 REM *     BY ANALOG COMPUTING    *
AL 80 REM ******************************
BG 90 REM
YB 100 DIM D$(1024),SD$(128),N$(16):DRV=1
   :VRF=1
ON 110 DO=ADR(SD$):SD$="♥":SD$(128)="♥":S
   D$(2)=SD$
NB 120 GRAPHICS 0:POKE 752,1:POKE 82,0
WP 130 POSITION 0,4:? "
                                "
BF 140 POSITION 0,6:? "__The_Atari_Disk_D
   irectory_Alphabetizer_"
EG 150 POSITION 0,8:? "  Drive:
            Verify:     "
AG 160 POSITION 8,8:? CHR$(DRV+176)
WS 170 IF VRF=1 THEN POSITION 36,8:? "On
   "
JG 180 IF VRF=0 THEN POSITION 36,8:? "Off
   "
VA 190 POSITION 0,12:? "▸   Change Drive
   / Verify":? :? "▸   Process Directory"
FI 200 ? :? "▸   Disk Operating System":G
   OSUB 650:IF A=68 THEN DOS
LY 210 IF A=67 THEN 240
MA 220 IF A=80 THEN 270
MI 230 GOTO 210
MY 240 POKE 752,1:? :? "▸       Drive (1-
   8):";:GOSUB 650:A=A-48:DRV=A:? DRV:IF
   DRV<1 OR DRV>9 THEN 240
QV 250 ? "▸  Verify (1-on) (0-off):";:GOS
   UB 650:VRF=A:VRF=VRF-48:? VRF:IF VRF<0
```

```
   OR VRF>1 THEN 250
NJ 260 POKE 752,1:GOTO 120
VY 270 GOSUB 660
ZF 280 ? "▶    ___Directory Read___":? :?
   "Please insert disk to be read and MAK
   E"
SB 290 ? "SURE there is a WRITE PROTECT T
   AB on it."
XP 300 ? "▶      Press Any Key ":GOSUB 6
   50
FK 310 IF VRF=1 THEN POKE 1913,87
WY 320 IF VRF=0 THEN POKE 1913,80
VR 330 GOSUB 660
SM 340 OP2=1:OP=82:S1=361:S2=368:GOSUB 67
   0
MZ 350 GOSUB 660:POKE 559,22
ZW 360 TRAP 380:FOR A=1 TO 993 STEP 16:IF
    D$(A+22,A+32)="♥♥♥♥♥♥♥♥♥♥♥♥" THEN 380
DF 370 NEXT A
OI 380 XND=A:FOR Z=1 TO INT(XND/16)
VM 390 FOR X=0 TO XND-16 STEP 16
OK 400 IF D$(X+6,X+16)>D$(X+22,X+32) THEN
    N$=D$(X+1,X+16):D$(X+1,X+16)=D$(X+17,
   X+32):D$(X+17,X+32)=N$
SB 410 NEXT X:NEXT Z:POKE 559,42
XF 420 ? "↑▶      ___Data Write___":? :? "
   Please REMOVE the WRITE PROTECT TAB an
   d re-insert ";
SP 430 ? "the disk for directory write."
OT 440 ? :? "  After this section, the sc
   reen will":? "  blank and ADDA will re
   -write file &"
DX 450 ? "  sector data.  PLEASE WAIT!!"
PQ 460 ? :? "▶      Press Any Key "
DT 470 POKE 752,1:GOSUB 650
WC 480 GOSUB 660
HJ 490 S1=361:S2=368:OP=87:OP2=2
CY 500 POP :GOSUB 670:FN=0:POKE 559,22
AI 510 FOR X=4 TO 16*(INT((XND+17)/16)) S
   TEP 16
SO 520 S1=ASC(D$(X+1,X+1))*256+ASC(D$(X,X
   ))
CI 530 LN=ASC(D$(X-2,X-2)):IF D$(X-3,X-3)
   =CHR$(128) OR LN=0 THEN 630
YY 540 S2=S1
TK 550 OP=82:OP2=0
WL 560 GOSUB 670
MQ 570 Z=ASC(SD$(126,126)):Y=ASC(SD$(127,
   127)):NXTS=256*(Z-4*(INT(Z/4)))+Y
GT 580 Z=4*FN+((NXTS-Y)/256):SD$(126,126)
   =CHR$(Z)
PJ 590 OP=87:OP2=0:GOSUB 670
NF 600 S1=NXTS:S2=S1:OP2=0
NA 610 IF NXTS=0 THEN 630
PG 620 GOTO 550
TK 630 FN=FN+1:NEXT X:POKE 559,42
SZ 640 ? :? "DIRECTORY NOW ALPHABETIZED!!
   !!!":FOR A=1 TO 800:NEXT A:GRAPHICS 0:
   END
KQ 650 CLOSE #1:OPEN #1,4,0,"K:":GET #1,A
   :CLOSE #1:RETURN
SR 660 FOR A=12 TO 21:COLOR 32:PLOT 0,A:D
   RAWTO 39,A:NEXT A:POSITION 0,12:RETURN
DT 670 REM ----- READ + WRITE SUB -----
HU 680 POKE 769,DRV:POKE 770,OP:Z=0
PD 690 FOR SN=S1 TO S2
MZ 700 POKE 779,0
AX 710 IF OP2=2 THEN GOSUB 800
ZM 720 SO=DO-256*INT(DO/256)
VV 730 LO=INT(DO/256)
RN 740 POKE 772,SO:POKE 773,LO
CO 750 POKE 778,SN-256*INT(SN/256)
DF 760 POKE 779,INT(SN/256)
VL 770 HO=USR(ADR("h 5d♦"))
HZ 780 IF OP2=1 THEN D$(LEN(D$)+1)=SD$
ST 790 NEXT SN:RETURN
OK 800 SD$(1,128)=D$(Z+1,Z+128)
PD 810 Z=Z+128
ZI 820 RETURN
```

BARGAINS!

1020 COLOR PRINTER PLOTTER

Complete with:
- 2 Pen Sets
- 1 Roll Paper
- Power Supply & Cable

$14.98 Brand New

EXTRA PEN SETS ..... $ .89 (Black) $3.98 (Color)

ATTN. DEALERS

PAC-MAN Case of 120 **$69.**

800 48K COMPUTER w/PAC-MAN **$69.95** Reconditioned

1025 PRINTER
- 80 Column
- Dot Matrix
- Friction/Tractor
- Direct Connect

$79.98 Reconditioned

1200XL 64K COMPUTER

w/PAC-MAN **$49.** Reconditioned

1027 PRINTER

- 80 Column
- Direct Connect
- Letter Quality

$79. Brand New

1050 DISK DRIVE $169.

Reconditioned

800 BOARDS

- Mother
- CPU
- Power
- ROM

YOUR CHOICE **$8.98**

SYNTREND $4.98

850 Interface Serial, parallel **$79.98** Reconditioned

825 80 Column Printer
- Friction, Trac
- Dot Matrix

$69. Reconditioned

BASIC TUTOR 4 BOOKS **$4.98**

ARCADE CHAMP
- 2 JOYSTICKS
- QIX
- PAC-MAN

$14.98

NEW **DOS XE** .. **$9.95**

1010 Tape drive .... $29.
10' Joystick ext. .... $ .99
Trackball $9.95

CARTRIDGES FOR 800, XL, XE

BASIC CARTRIDGE	$4.95	DELTA DRAWING	$8.98	BATTLEZONE	$19.98
BASIC TUTOR (4 BOOKS)	$4.95	HEY DIDDLE DIDDLE	$8.98	FOOD FIGHT	$19.98
TURMOIL	$4.95	SLIME (400/800)	$8.98	HARDBALL	$19.98
PAC-MAN (no box)	$4.95	ALPHABET ZOO	$8.98	FIGHT NIGHT	$19.98
DONKEY KONG (no box)	$4.95	ALF	$8.98	ONE ON ONE BASKETBALL	$19.98
GORF (400,800)	$4.95	ADVENTURE CREATOR	$8.98	DESERT FALCON	$19.98
DEMON ATTACK (400,800)	$4.95	SKY WRITER	$14.95	NECROMANCER	$19.98
DELUXE INVADERS	$4.95	FOOTBALL	$14.95	RESCUE ON FRACTALUS	$19.98
JOURNEY TO THE PLANETS	$4.95	DEFENDER	$14.95	BALLBLAZER	$19.98
MATH ENCOUNTER	$7.98	ROBOTRON	$19.98	BLUE MAX	$19.98
DANCE FANTASY	$8.98	TENNIS	$19.98	STAR RAIDERS II	$19.98
LOGIC LEVELS	$8.98	FINAL LEGACY	$19.98	DAVID'S MIDNIGHT MAGIC	$19.98
MEMORY MANOR	$8.98	MARIO BROS.	$19.98	ARCHON	$19.98
LINKING LOGIC	$8.98	DONKEY KONG JR.	$19.98	GATO	$24.98
CHICKEN	$8.98	JUNGLE HUNT	$19.98	ACE OF ACES	$24.98
CLAIM JUMPER	$8.98	MOON PATROL	$19.98	LODE RUNNER	$24.98
				BARNYARD BLASTER (XE gun)	$24.98

DISK SOFTWARE FOR 800, XL, XE

DAVID'S MIDNIGHT MAGIC	$4.98	CROSSCHECK	$4.98	SPIDERMAN	$4.98
ZORRO	$4.98	MOLECULE MAN	$4.98	HULK	$4.98
BANDITS (48K 400,800)	$4.98	CRYSTAL RAIDER	$4.98	VISICALC	$24.98
PROTECTOR II	$4.98	DESPATCH RIDER	$4.98	BOOKKEEPER W/ numeric keypad	$29.98
CLAIM JUMPER	$4.98	MISSION ASTEROID	$4.98	GET RICH	$39.98
SYNTREND	$4.98				

810 DISK DRIVES $129.

8-BIT I/O CABLE $4.98

SAN JOSE COMPUTER

THE ATARI STORE

Sunrise Plaza 640 Blossom Hill Rd. San Jose, CA 95123
(408) 224-8575 • BBS (408) 224-9052

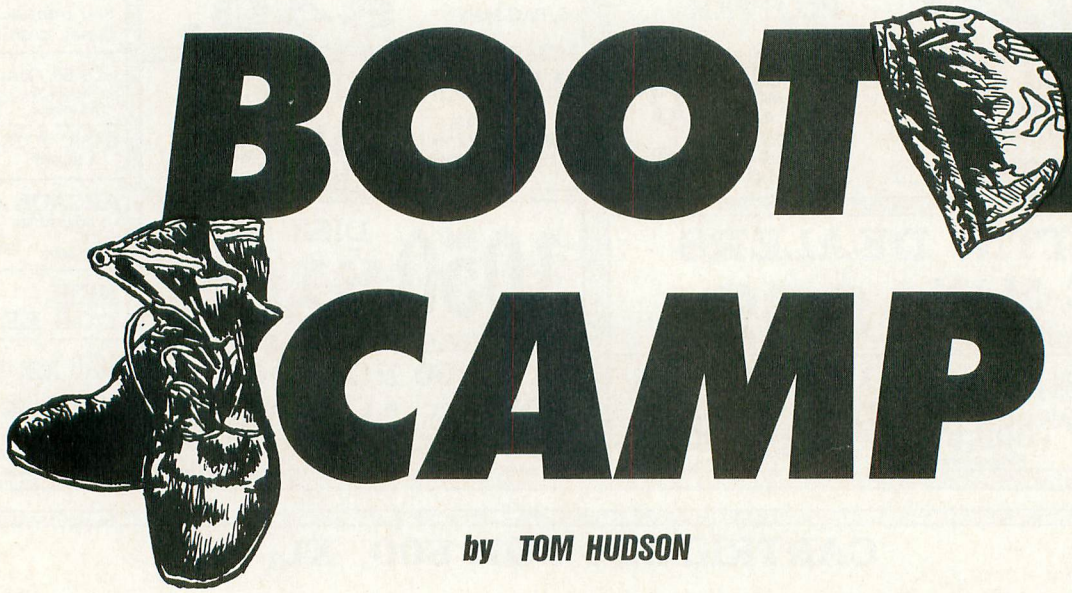
5.25" DISKS **20 CENTS EA.***

QTY.	PRICE
10	$4.00
100	$29.95
*1000	$200

MAJORITY ARE UNNOTCHED CONTAINING OLD SOFTWARE

SHIPPING: ADD $5.00 TO ALL ORDERS. AIR AND INTERNATIONAL SHIPPING EXTRA. THAT'S IT.
WARRANTY: 90 DAY WARRANTY ON ALL ITEMS. **TAX:** CALIFORNIA RESIDENTS ADD 7% SALES TAX.
PREPAYMENT: USE VISA, MASTERCARD, MONEY ORDER, CASHIER'S CHECK OR PERSONAL CHECK.
PERSONAL CHECK MUST CLEAR PRIOR TO SHIPMENT. **C.O.D.:** CASH, CASHIER'S CHECK OR M.O. ONLY.
Prices subject to change without notice. Brand and/or product names are trademarks or registered trademarks of their respective holders.
Ad produced on an ATARI ST.

CIRCLE #102 ON READER SERVICE CARD.



BOOT CAMP

by TOM HUDSON

Last issue, you were introduced to the concept of various numbering systems, including Base 2, Base 10 and Base 16. We also covered the basics of assembly language and the registers of the 6502 microprocessor.

In this issue, we'll talk about the ways the 6502 can address memory and begin looking at the 6502 instruction set.

Address Unknown?

In order to perform useful work for us, the 6502 microprocessor chip must be able to get numbers from memory, manipulate them and place the results back in memory. Each memory location has its own number, or address. The 6502 can reference up to 65,536 bytes of memory ($0000-$FFFF).

If you've used the BASIC PEEK and POKE functions, you've used the 6502's addressing ability already. Consider, for example, the BASIC command:

```
POKE 559,0
```

This command places a 0 in address 559 ($22F), which turns off the computer's screen display.

Luckily for us programmers, the designers of the 6502 gave us quite a bit of flexibility in how we reference memory locations. These ways are listed below.

Immediate addressing allows us to place one number we are working with (or OPERAND) right after the operation code. The operand must be preceded with then"#" symbol. For example, the assembly instruction:

```
LDA #23
```

places the number 23 in the accumulator. In this example we specified the number in decimal. If we wanted, we could have given the number in hexadecimal (Base 16):

```
LDA #$17
```

Note that decimal numbers require no special marking, but hex numbers are always preceded by a "$" symbol.

Absolute addressing tells the computer we want to get the operand from a certain address somewhere in memory. For example, let's say we want to turn off the screen as we did before in the above BASIC example. Instead of a POKE 559,0 command, we could use the following two assembly instructions:

```
LDA #0
STA 559
```

The first instruction, as we learned above, will load the accumulator with a 0. The second instruction uses the absolute addressing mode to store the contents of the accumulator into memory address 559. What could be easier?

Implied addressing means that no addresses are used in the instruction. The CLC (clear carry) and RTS (return from

subroutine) instructions are good examples of implied addressing instructions.

Accumulator addressing is used for those instructions that use only the accumulator, such as ASLA (arithmetic shift left).

Index addressing is a useful type of addressing which makes table operations very simple. In this mode, the X or Y register is used as an index. For example, in the following instruction:

LDA TABLE, X

If the X register contains a 7, the accumulator will be loaded with whatever is in the seventh byte after TABLE. It's a very simple concept, and works the same with the Y register.

Indirect addressing is only used with the JMP (jump to location) instruction. In the following example:

JMP ($3000)

The JMP will not go to address $3000, but it will take the contents of $3000 and $3001 and jump to the address indicated by their contents. If, for example, $3000 contains $3F and $3001 contains $50, the program will jump to $503F. This instruction is rarely used, but it can be irreplaceable under certain circumstances.

Pre-indexed indirect addressing uses the X register and an operand byte to address a byte in the first 256 bytes of memory. In the following example:

LDA ($AF,X)

If the X register contains $12, the computer adds $AF and $12, giving a result of $C1. The computer then takes the contents of $C1 and and $C2 and loads the accumulator from the address contained in these bytes. For example, if Location $C1 contains $50 and a Location $C2 contains $3F, the accumulator will be loaded from Location $3F50.

Point-indexed indirect addressing uses the Y register and an address in the first 256 bytes of memory to point to another address. In the following example:

LDA ($AF),Y

The computer takes the contents of bytes $AF and $B0 and adds the Y register to this address for a final address. If $AF contains $00 and $B0 contains $40 the computer first points to $4000, then uses the Y register as an offset. If the Y register contains $50, the accumulator would be loaded from $4050. This addressing mode is used fairly often.

Relative addressing is used in all branch-on-condition instructions in the 6502. Usually after a comparison the programmer will branch on a condition. This is the same as an IF/THEN statement in BASIC. In the following example:

BNE START

The computer will calculate the number of bytes between the branch instruction and the location referenced by START at assembly time. During execution, the image in memory may look like:

BNE $30

This indicates that START was 48 bytes from the branch instruction. If the branch is executed, the computer will skip 48 bytes and continue executing as the part of the program labeled START. There is only one drawback to this addressing mode: The branch cannot be farther than −126 or +129 bytes. Longer branches require the use of the JMP instruction.

Assembler Syntax

Every computer language has a set of rules known as syntax. These rules are established so that the programmer will enter program code in a way that the computer will understand. Assembly language has a very simple syntax, shown in Figure 1.

LABEL
OP CODE
OPERAND
COMMENTS

Figure 1.

If you have ever looked at assembly language source listings in ANALOG, you have noticed the neat columns of "gibberish." This is the way assembly language is structured.

Each column of information in the assembly source listing is known as a field. Each field is separated by one or more spaces.

The first field, or label field, is optional. If the code you are writing will be referenced elsewhere in the program, you should place an appropriate label in the label field.

A label should give some ideal of what the section of code does. For example, L0001 tells nothing about the code, whereas VBLANK tells us that the code is part of the vertical blank cycle. Meaningful labels should be included wherever possible.

Labels should start with a letter, but can contain numbers in them.

Many assemblers use only the first five or six characters of a label, so the labels will be limited to six characters. This will enable the readers with assemblers other than the ATARI cartridge to use the pro-

In BASIC, the programmer doesn't really care where the program is placed in memory. This is one of the benefits of a high-level language.

gram listings with as little modification as possible.

The second field in an assembler statement is the operation code. This is usually a three-character standard 6502 instruction, such as LDA, STA or JMP.

Each assembler also has a set of directives, or pseudo-operations. These operations are not commands to the 6502, but are processed by the assembler program at assembly time. The most common directives are ".BYTE," ".WORD," "EQU" or "=" and "ORG" or "*=". These will be discussed in detail later.

The third field in an assembler statement is the operand. This field contains data or addresses required by the operation code. Operands are not needed by all operation codes.

Operands are usually given in decimal or hexadecimal. Decimal numbers require no special prefix, but hex numbers must be preceded by the "$" character.

Operands can also be labels defined elsewhere in the program. For example, instead of:

```
JMP $4000
```

We could have used the EQUATE directive to define a label called START and set it to the value of $4000 as follows:

```
START = $4000
JMP START
```

By using labels in operands instead of absolute numbers, programs are easier to change if the need arises. Imagine having to change 50 "JMP $4000" instructions to "JMP $5000." If we used "JMP START" instead, we'd only have to change the "START = $4000" to "START = $5000." This would automatically change the 50 JMP instructions!

The last field in an assembler statement is the comment. Comments are optional, but encouraged. Comments are like REMarks in BASIC—they help document what the programmer is doing. This is especially important in assembler programs, which are somewhat difficult to decipher.

Comments are preceded by a semicolon (;). Everything after the semicolon is ignored by the assembler. Comments should be used as often as possible, especially when a section of code is fairly complex. This will not only help others who use the program, but will help you if you need to make changes to the program at a later date.

Where to Put the Program?

In BASIC, the programmer doesn't really care where the program is placed in memory. BASIC handles all these messy details for the programmer, who simply writes program code. This is one of the benefits of a high-level language like BASIC.

As mentioned last issue, the assembly language programmer must know at all times what locations a program is using. Without total knowledge of a program's location, it is possible to overlap memory used by the system and cause an irrecoverable "lock-up."

Let's look at what memory locations are available to us in the Atari computer system. This discussion will apply to users of the Atari assembler-editor cartridge only.

Plug your cartridge into the computer and turn on the power. When the EDIT prompt appears, type SIZE and press Return. The cartridge will show three nubers, such as:

```
0700    0800    3CIF
```

The first number is the bottom of RAM, the second is the end of the text-editor buffer, and the third is the highest available RAM address.

Since readers have different amounts of memory and since cassette and disk systems use different amounts of memory, each reader must decide where to place the object program in memory. To do this, subtract about $600 (1536 bytes) from the last number above. In this case, the number is $3C1F-$0600=$361F. Round this down to the nearest 256 bytes and you have $3600. This will be the starting address of your object program. Use this address in the "*=" directive of the program in this column.

There are also 256 bytes available for use at $0600=$06FF, or page 6. We will be using this area later for subroutines called by BASIC. The term "page" is used to refer to a 256-byte section of memory. The page number comes from the first two digits of the hex address. $0200-$02FF is page 2, $0800-08FF is page 8, etc.

The last memory available to us has special significance. This memory lies on page 0, $0000-$00FF. When the 6502 knows a byte is on page 0, it only needs the last two hex digits to address it. This allows the 6502 to access the information faster, with a smaller program, since only one byte is needed in the operand instead of the usual two needed for an address.

Since page 0 addresses can be accessed faster with less program memory, it is obviously good to use page 0 whenever possible. The problem is, the system uses some of page 0 for its own needs. The entire first half of page 0 ($0000-$007F) is always used by the system. The second half ($0080-$00FF) is available to assembly language programs if no cartridges are in use.

A Few Instructions

Now we're ready to look at a few 6502 operation codes and see how they work. We'll start with the most frequently used instructions and work our way up to the rarely used instructions.

Without a doubt, the most frequently used 6502 operation code is LDA, or LoaD Accumulator. This instruction places a desired number in the A register, or accumulator.

The accumulator is used in all addition and subtraction operations, as well as most other arithmetic that can be performed on the 6502. You must move numbers in and out of the accumulator constantly, keeping track of the results. At times, you'll feel like a traffic cop trying to direct hundreds of cars through an ordinary doorway. After just a few hours of assembly programming, you'll see how important the accumulator is.

The LDA instruction has eight different formats, each with its own addressing method:

LDA	#n	(Immediate)
LDA	nn	(Absolute)
LDA	n	(Page zero)
LDA	(n,X)	(Pre-indexed indirect)
LDA	(n),Y	(Post-indexed indirect)
LDA	n,X	(Zero page indexed X)
LDA	nn,X	(Indexed X)
LDA	nn,Y	(Indexed Y)

Each of these instructions work differently in order to load the accumulator. They find the address from which they are to get the number and place it in the accumulator, destroying whatever was there before. Once the number is placed in the accumulator, however, the instructions act alike.

Let's assume the number loaded into the accumulator was $94, shown below in its binary form (note the "%" sign preceding the binary number).

$94 = %10010100

All LDA instructions take special information from the number loaded and set microprocessor status flags accordingly. The two flags changed are the sign flag and the zero flag.

The zero flag is set to 1 if the number loaded was 0, and is set to 0 if the number was not 0. This flag is mainly used for branching, which we will cover later.

The sign flag is set to the value of the high-order (or leftmost) bit of the number loaded. You should remember that an 8-bit byte can contain numbers from 0-255. This is true when we are considering the numbers to be unsigned. The 6502 uses a signed numbering system that can be somewhat confusing.

Whenever a number's high-order bit is a 1, the number is considered to be negative. Using this method, a byte can contain numbers from −128 to 127. How does this work? Let's start with the positive numbers.

Positive numbers in the 6502 signed number scheme range from 0 (which is always considered positive) to 127. The upper limit of 127 is set because if the number is 128, the high-order bit will be set to 1 and the number is negative.

Negative numbers range from −1 to −128 in the 6502 system. If we subtract 1 from 0 in the 8-bit byte format, the byte's contents will "wrap around" to the bit pattern 11111111, which is 255. 255 corresponds to −1 in this scheme. An easy way to remember the relationship here is the following calculation:

UNSIGNED NUMBER −256 = SIGNED NUMBER

Using this formula with the unsigned number 255, we can see that 255−256= −1, which is correct. We can easily find the signed counterpart to 128, or 128−156 = −128.

Now you can see exactly how the sign flag works. This flag will be very important later when we perform comparisons.

The next instruction, which is used almost as much as the LDA instruction is STA or STore Accumulator. this instruction does almost the same thing as the LDA, but in reverse.

The STA instruction has the following formats:

STA	nn	(Absolute)
STA	n	(Page zero)
STA	(n,X)	(Pre-indexed X)
STA	(n),Y	(Post-indexed Y)
STA	n,X	(Page zero indexed X)
STA	nn,X	(Absolute X)
STA	nn,Y	(Absolute Y)

You will notice that the STA instruction has the same formats as the LDA instructor except for the immediate format. Think about it for a minute and the reason should be obvious.

At times, you'll feel like a traffic cop trying to direct hundreds of cars through an ordinary doorway. After just a few hours, you'll see how important the accumulator is.

Unfortunately, the Atari assembler-editor cartridge only allows you to use Locations $B0-$CF. These locations are probably sufficient for most testing purposes.

When writing assembly language programs to be called as subroutines by Atari BASIC, only Locations $CB-$D1 and $D4-$D5 can be used without conflict with BASIC's work areas. If an assembly subroutine needs temporary work areas, Locations $D6-$F1 can be used. These areas will probably be changed by BASIC after the assembly subroutine ends, but they will work fine as temporary storage locations.

The STA instruction simply places whatever number is in the accumulator into the address specified in the operand. The number in the accumulator will be unaffected, and will still be available for your use.

The STA instruction does not affect any status.

A third instruction that is widely used is the JMP instruction. This instruction is just like BASIC's GOTO statement. Whenever this instruction is executed, the program will jump to the address specified and continue processing. The address jumped to must contain executable program statements, so take care.

The JMP instruction has two formats:

JMP nn (Absolute)
JMP (nn) (Indirect)

As noted above in the discussion of the indirect format, the indirect jump is rarely used, but can be very helpful in special situations.

The absolute jump instruction is the most-used form of the JMP operation code. The address specified can either be a hex or a decimal number or a label that is defined elsewhere in the program.

The JMP instruction does not affect any status flags.

Applying the Instructions

Now that we've described the LDA, STA and JMP instructions, let's apply them in a short program.

The program in Figure 2 is essentially a "do-nothing." It will simply move numbers around in memory until we stop it. Type the program into your computer, remembering to set your origin value (*= in Line 140) as described above.

When you have entered the program and set the origin at Line 140, type ASM and press Return. The program will be assembled into memory and ready to execute.

Before executing the program, let's look at Figure 2. The first thing you'll notice in the listing is the presence of comments. I can't overemphasize the importance of comments in an assembly language program. They're simply a must whenever you're writing programs, even for yourself. You'll notice that some comment lines are simply semicolons with no comment. These are used as separators to break up sections of code. For example, each label group (i.e., START, PART1, PART2, etc.) is a distinct group in the listing.

Remember, comments don't take up any program space in assembly language, so use them as often as possible!

```
0100 ;** DO-NOTHING DEMO PROGRAM **
0110 ;
0120 ;BY TOM HUDSON
0130 ;
0140         *=  $????    ;YOUR ORIGIN!
0150 ;
0160 START   LDA BYTE1    ;COPY BYTE1..
0170         STA BYTE2    ;TO BYTE2
0180         LDA #7       ;PUT A 7...
0190         STA BYTE3    ;IN BYTE3
0200         JMP PART2    ;JUMP!
0210 ;
0220 PART1   LDA BYTE2    ;MOVE BYTE2...
0230         STA BYTE4    ;TO BYTE4
0240         JMP PART3    ;AND JUMP
0250 ;
0260 PART2   LDA RANDOM   ;MOVE RANDOM..
0270         STA BYTE1    ;TO BYTE1
0280         JMP PART1    ;AND JUMP!
0290 ;
0300 PART3   LDA BYTE4    ;MOVE BYTE4...
0310         STA BYTE5    ;TO BYTE5
0320         JMP START    ;AND JUMP!
0330 ;
0340 ;DATA BYTE5 START HERE!
0350 ;
0360 BYTE1   .BYTE 1      ;NUMBER 1
0370 BYTE2   .BYTE 2      ;NUMBER 2
0380 BYTE3   .BYTE 3      ;NUMBER 3
0390 BYTE4   .BYTE 4      ;NUMBER 4
0400 BYTE5   .BYTE 5      ;NUMBER 5
0410 RANDOM =   $D20A     ;RANDOM NUMBER
0420 ;
0430         .END
```

Figure 2.

Line 160—loads the accumulator with the number 7, wiping out whatever was previously in the accumulator. Remember that whenever the accumulator is loaded, the contents of the accumulator before the load will be lost.

Line 190—stores the 7 just loaded into the accumulator at the location labeled BYTE3. This is also a very common operation.

Line 200—jumps to PART2, and execution continues there.

Line 220—labeled PART1, loads the accumulator from the location marked BYTE2.

Line 230—stores the value just loaded from BYTE2 into the location labeled BYTE4.

Line 240—jumps to PART3.

Line 260—labeled PART2, loads a byte from the computer's random number generator at $D20A. This location gives a random number from 0-255.

Line 270—stores the random number at the location labeled BYTE1.

Line 280—jumps to PART1.

Line 300—labeled PART3, loads the accumulator from the location labeled BYTE4.

Line 310—stores the number just loaded at Location BYTE5.

Line 320—jumps to START. This causes the program to loop forever until you press the Break key.

Lines 360-400—define the bytes labeled BYTE1-BYTE5. The .BYTE directive is used to assign initial values to the locations. BYTE1 will contain 1, BYTE2 will contain 2, etc.

Line 410—uses the EQUATE directive to define the address of the label RANDOM. This location is $D20A (53770 decimal). Whenever the label RANDOM is referenced, the computer will use the value $D20A.

Line 430—uses the .END directive to tell the assembler the end of the code has been reached. This directive is optional, but recommended.

Tracing the Action

Now you can execute the above program and see what it does. Note the address you used in Line 140. With the EDIT prompt on the screen, type BUG and press Return. The DEBUG prompt will appear.

Type L followed by the address you used in Line 140 and press Return. For example, if your Line 140 reads:

```
* = $4300
```

You should type L4300 and press Return. The computer will show you how your program appears in memory, and it should look something like Figure 3.

```
6000   AD 29 60   LDA $6029
6003   8D 2A 60   STA $602A
6006   A9 07      LDA #$07
6008   8D 2B 60   STA $602B
600B   4C 17 60   JMP $6017
600E   AD 2A 60   LDA $602A
6011   8D 2C 60   STA $602C
6014   4C 20 60   JMP $6020
6017   AD 0A D2   LDA $D20A
601A   8D 29 60   STA $6029
601D   4C 0E 60   JMP $600E
6020   AD 2C 60   LDA $602C
6023   8D 2D 60   STA $602D
6026   4C 00 60   JMP $6000
6029   01 02      ORA ($02,X)
602B   03         ???
602C   04         ???
602D   05 00      ORA #$00
602F   00         BRK
6030   00         BRK
```

Figure 3.

Your listing will probably vary from this illustration, which was assembled to Location $6000. Note that the BYTE1-BYTE5 values appear in memory from $6029-$602D, and the computer tries to show the bytes as instructions (like ORA #$00). Simply ignore such instruction whenever you know they are misinterpreted data.

If your program is at the proper location, you are ready to watch its execution. Type T followed by the address in Line 140 and press Return.

The computer will begin tracing the execution of your program one line at time. Each instruction will be shown along with its address and the contents of the 6502 registers after the instruction executes. Page 40 of the Atari assembler-editor manual describes the trace operation in detail.

At any time in the execution you may stop the program with the Break key and examine the BYTE1-BYTE5 locations (note their addresses at assembly time) by using the Dnnnn command, described on page 36 of the assembler-editor manual.

We are interested in seeing how the instructions we used are executed and how they affect memory. Figure 4 shows the lines of the program as they are executed and the status of the variables BYTE1-BYTE5 after each statement is executed. Note that the value present in RANDOM cannot be predicted and is indicated by "R#."

			STATEMENT	A RG.	BT. 1	BT. 2	BT. 3	BT. 4	BT. 5
160	START	LDA	BYTE1	01	01	02	03	04	05
170		STA	BYTE2	01	01	01	03	04	05
180		LDA	#7	07	01	01	07	04	05
190		STA	BYTE3	07	01	01	07	04	05
200		JMP	PART2	07	01	01	07	04	05
260	PART2	LDA	RANDOM	R#	01	01	07	04	05
270		STA	BYTE1	R#	R#	01	07	04	05
280		JMP	PART1	R#	R#	01	07	04	05
220	PART1	LDA	BYTE2	01	R#	01	07	04	05
230		STA	BYTE4	01	R#	01	07	01	05
240		JMP	PART3	01	R#	01	07	01	05
300	PART3	LDA	BYTE4	01	R#	01	07	01	05
310		STA	BYTE5	01	R#	01	07	01	01
320		JMP	START	01	R#	01	07	01	01
160	START	LDA	BYTE1	R#	R#	01	07	01	01
170		STA	BYTE2	R#	R#	R#	07	01	01

Figure 4.

(Try the next 10 steps yourself!)

As stated earlier, this is a "do-nothing" program, and will continue to execute forever unless it is stopped by the user. If you'd like a demonstration of this infinite execution, type G followed by the address in Line 140 and press Return. The computer will begin executing the do-nothing at an unbelievable speed, and won't stop until you press Break. You won't see anything happen during the program's execution, but you can rest assured that the computer is following your instructions to the letter.

Stay Tuned

Next issue, we'll start digging into more 6502 operation codes, learn to add and subtract, and work with index registers. Until then, make your own short programs using the instruction we've covered. I realize these three aren't enough to create complex programs, but knowledge of their use is essential to future lessons.

Super Command Processor, Part II

by Bryan Schappel

Welcome back! This month we will discuss external commands, how external commands interface with BBKCP, and how to write your own external commands. There are also four external commands supplied with this installment. Let's get to it.

More on External Commands

Last month we said an external command is something that the internal or memory resident portion of BBKCP does not understand. Such a command is loaded from disk each time it is called. Almost all external commands are destructive to memory, meaning that when an external command is executed *any program in memory will be lost*!

External commands interface directly by calling routines built into BBKCP. By doing this, filenames, addresses and numbers may be sent to an external command. Table 1 gives the name, address and description of special routines in BBKCP. These routines are called by a 6502 JSR instruction. Any parameters or initial conditions for the routines will be given also.

By using the routines in the table, we may write any number of external commands, performing almost any operation. Here are some samples.

The External Goodies

Listings 1 through 4 are the data needed to create your copies of the four external commands described below. You should type in this data using the M/L Editor on page 61. Listings 5 through 8 are the assembly language source codes for the four external commands. You need not type these listings; they are provided only for those who are interested in assembly language programming.

COMMAND #1: DISBAT**

This command will toggle the batch-file processing feature of BBKCP on or off. After the file executes, a message will be displayed telling you about the status of batch processing.

This command is *not* destructive to memory. It was included because certain programs and games published in ANALOG will fail to run if batch processing is left on.

COMMAND #2: DUMP *[addr]*

This is a memory dump utility that loads and runs at $2480. It will begin dumping the contents of memory on the screen from the address you specify. Press Start or Break to return to BBKCP.

COMMAND #3: SAVE fname start end *[init run]*

This allows you to perform a binary save of memory. You must provide a legal filename, a start address, an end address and optional initialize and run addresses.

COMMAND PROCESSOR

NAME	HEX ADDRESS	DESCRIPTION
ADDRESS	23B7	Contains the last RUN address of an external command. Contains address for the RUN command.
CLOSE1	1EE4	Closes IOCB #1 Returns with X =$10
CLOSE2	1EE8	Closes IOCB #2. Returns with X=$20.

CLOSEIT	**1EEA**	Closes ANY IOCB. enter with X=IOCB offset. ($00, $10, $20,...)
DEFDEV	**2354**	This is the default device and device number in the format D1.
DEVICE	**23B9**	Contains a correct DOS filename in the format of D1:filename.ext.
EPRINT	**1E64**	Will print a NULL terminated string to the screen. Enter with A,Y being the LO,HI of your text. (NULL=0).
FINDARG	**21F4**	Will find the start of the next argument on the command line. Enter with Y=search offset. (IE Y=3 says start search on Character 4.) Exits with Y and LNPOS = start of argument.
FINDFILE	**1FDB**	Enter with Y=start of name and DEVICE will return the filename in the D1:filename.ext format.
FNAME	**23BC**	Start of formatted filename. Same as DEVICE+3.
FRO	**00D4**	Contains the result of a GRAB_HEX. $D4 is LO byte and $D5 is HI byte.
GOCART	**1F11**	Tries to RUN a cartridge. Calling this is the same as issuing a CAR command.
GRAB_HEX	**1E9D**	Pulls a hex address from the command line. Enter with Y=offset. Exit: Carry Clear means good number Carry Set means bad number.
HEXDIG	**236B**	Contains the ATASCII hexadecimal digits. 0-9 and A-F.
INPUT	**1EF7**	Prints the current prompt and accepts one 64-byte line from the screen editor.
IOERROR	**1FAC**	Closes IOCB #1 and prints the I/O error number on the screen in decimal. Enter with Y=error number.
LNPOS	**00DF**	Contains the current offset to the argument on the command line.
MYBUF	**0500**	64-byte command line buffer.
WARMST	**0008**	Warm start flag. If the value here is $FF, memory contents are OK. If value is $00, memory will be wiped out.

COMMAND #4: FORMAT**

This is a diskette initialization program. The menu contains five options: Format, Write DOS, Format and Write DOS, Change Drive and Exit. Just press the number next to the description to select your option. If you choose an option between 1 and 3, you are asked to insert a disk and press Return. Pressing Return begins the process; pressing any other key aborts the option, returning you to the menu.

When you choose to write DOS, FORMAT not only writes the DOS.SYS file, but it also writes the AUTORUN.SYS version of BBKCP to the disk.

Option 4 allows you to alter the drive number that the commands are being executed on. Just type the new drive number, and you are returned to the menu.

Option 5 returns you to BBKCP.

Writing External Commands

All external commands must be written in machine language, therefore you must be familiar with 6502 assembly language. Here are a couple of rules to follow.

Rule #1: Assemble all your commands with an origin not less than $2410, otherwise part of BBKCP would be overwritten, causing a system crash.

Rule #2: While a batch file is being processed *never* use IOCB #5. Doing so will most likely cause a system crash. This is the IOCB that the input is being redirected from—take care.

Some Tips

If you write a small external command that is less than 448 bytes long, you may load it at $0540 and set WARMST back to a $FF, thus preserving the contents of memory.

Study the source code for the external commands given in this issue. They will guide you in the method of writing your own commands. Some extra commands you may consider writing would be a multifile copy program, a disk copier or a simple assembler/disassembler. The possibilities are quite endless.

By the time you read this, Bryan and Carol will have been happily married for a while. Their new apartment contains a new Mega ST2, which shares the computer room with the 800XL. The compu-kids get along very well—if you overlook the constant battle for the printer.

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,5,84,5,104,104,17
3,8,34,201,160,240,32,169,6842
1010 DATA 160,141,8,34,169,0,141,9,34
,169,169,141,10,34,169,185,5164
1020 DATA 141,247,32,169,26,141,248,3
2,169,3,141,249,32,208,41,169,8132
1030 DATA 76,141,247,32,141,8,34,169,
51,141,248,32,169,33,141,249,8339
1040 DATA 32,169,125,141,9,34,169,33,
141,10,34,172,216,35,169,0,3685
1050 DATA 153,26,3,169,228,153,27,3,7
6,116,228,0,0,0,0,0,8563
```

LISTING 2: M/L EDITOR DATA

```
1000 DATA 255,255,128,36,78,37,164,22
3,185,0,5,201,155,208,13,173,7859
1010 DATA 181,35,133,212,173,182,35,1
33,213,76,156,36,32,154,30,144,5925
1020 DATA 1,96,165,212,133,203,165,21
3,133,204,165,204,160,0,32,8,7486
1030 DATA 37,165,203,160,2,32,8,37,16
0,0,132,224,177,203,72,201,8219
1040 DATA 155,208,2,169,27,153,62,37,
166,224,189,71,37,168,104,32,5646
1050 DATA 8,37,164,224,200,192,8,208,
225,152,24,101,203,133,203,144,1263
1060 DATA 2,230,204,162,0,169,9,141,6
6,3,169,33,141,68,3,169,3400
1070 DATA 37,141,69,3,169,1,141,73,3,
141,254,2,32,86,228,48,4093
1080 DATA 7,173,31,208,201,7,240,162,
169,0,141,254,2,96,72,41,5607
1090 DATA 15,170,189,105,35,153,34,37
,104,41,240,74,74,74,74,170,4761
1100 DATA 189,105,35,153,33,37,96,48,
48,48,48,32,48,48,32,48,8027
1110 DATA 48,32,48,48,32,48,48,32,48,
48,32,48,48,32,48,48,6998
1120 DATA 32,48,48,32,32,32,32,32,32,
32,32,32,155,5,8,11,6077
1130 DATA 14,17,20,23,26,0,0,0,0,0,0,
0,0,0,0,0,1460
```

LISTING 3: M/L EDITOR DATA

```
1000 DATA 255,255,16,36,16,37,164,223
,32,212,31,164,223,32,237,33,7338
1010 DATA 32,154,30,176,47,165,212,13
3,204,165,213,133,205,164,223,32,2160
1020 DATA 237,33,132,203,32,154,30,17
6,27,165,212,133,206,165,213,133,1365
1030 DATA 207,32,221,30,32,195,34,32,
149,31,169,8,157,74,3,32,1148
1040 DATA 86,228,16,3,76,221,30,32,20
5,36,48,248,32,205,36,48,4117
1050 DATA 243,32,252,36,48,238,165,20
4,133,214,165,205,133,215,165,206,483
4
1060 DATA 56,229,214,133,218,165,207,
229,215,133,219,230,218,208,2,230,599
9
1070 DATA 219,32,10,37,48,206,164,203
,32,237,33,132,203,32,154,30,6261
1080 DATA 176,194,32,241,36,169,226,1
33,204,169,228,133,206,169,2,133,1376
1090 DATA 205,133,207,32,252,36,48,17
2,32,227,36,48,167,164,203,32,7052
1100 DATA 237,33,32,154,30,176,157,32
,241,36,169,224,133,204,169,226,2488
1110 DATA 133,206,169,2,133,205,133,2
07,32,252,36,48,135,32,227,36,6616
1120 DATA 76,221,30,169,255,72,162,16
,169,0,157,72,3,157,73,3,2865
1130 DATA 169,11,157,66,3,104,76,86,2
28,173,17,37,32,207,36,48,2950
1140 DATA 16,173,18,37,76,207,36,165,
212,141,17,37,165,213,141,18,6377
1150 DATA 37,96,169,4,133,218,169,0,1
33,219,133,215,169,204,133,214,2960
1160 DATA 169,11,162,16,76,55,30,0,0,
0,0,0,0,0,0,0,2821
```

LISTING 4: M/L EDITOR DATA

```
1000 DATA 255,255,128,36,199,39,173,1
84,35,141,90,38,160,2,185,8,4387
1010 DATA 34,153,36,39,185,30,39,153,
8,34,185,247,32,153,39,39,3394
1020 DATA 185,33,39,153,247,32,136,16
,229,169,65,160,38,32,97,30,3770
1030 DATA 173,90,38,141,36,38,141,55,
38,141,40,38,32,13,38,48,8480
1040 DATA 251,141,20,39,56,233,49,48,
243,201,5,176,239,72,169,20,7528
1050 DATA 160,39,32,97,30,133,203,104
,170,240,11,202,240,14,202,240,1634
1060 DATA 17,202,240,27,208,43,32,29,
37,76,163,36,32,102,37,76,996
1070 DATA 163,36,32,29,37,165,203,208
,3,32,107,37,76,163,36,169,4259
1080 DATA 7,160,39,32,97,30,32,13,38,
41,127,141,90,38,76,163,1936
1090 DATA 36,160,2,185,36,39,153,8,34
,185,39,39,153,247,32,136,4897
1100 DATA 16,241,96,32,22,38,240,3,23
0,203,96,169,211,160,38,32,7305
1110 DATA 97,30,32,221,30,169,35,157,
68,3,169,38,157,69,3,169,3625
1120 DATA 0,157,75,3,157,74,3,169,254
,157,66,3,32,86,228,16,4187
1130 DATA 26,152,72,169,21,160,39,32,
97,30,104,168,32,165,31,230,5150
1140 DATA 203,169,198,160,38,32,97,30
,76,29,38,96,32,22,38,208,1382
1150 DATA 250,169,22?,160,38,32,97,30
,32,252,37,169,54,157,68,3,3571
1160 DATA 169,38,157,69,3,32,86,228,4
8,199,32,252,37,169,39,157,6527
1170 DATA 68,3,169,38,157,69,3,32,86,
228,48,181,162,16,169,24,4382
1180 DATA 157,68,3,169,39,157,69,3,16
9,6,157,72,3,169,0,157,2891
1190 DATA 73,3,169,11,157,66,3,32,86,
228,48,149,162,16,169,124,5497
1200 DATA 157,68,3,169,29,157,69,3,16
9,57,157,72,3,169,6,157,3461
1210 DATA 73,3,169,11,157,66,3,32,86,
228,48,32,162,16,169,11,2305
1220 DATA 157,66,3,169,42,157,68,3,16
9,39,157,69,3,169,158,157,5599
1230 DATA 72,3,169,0,157,73,3,32,86,2
28,16,3,76,75,37,76,390
1240 DATA 221,30,32,221,30,32,195,34,
169,8,157,74,3,169,0,157,3613
1250 DATA 75,3,96,173,37,228,72,173,3
6,228,72,96,169,185,160,38,8095
1260 DATA 32,97,30,32,13,38,201,155,9
6,68,32,58,155,68,32,58,1611
1270 DATA 65,85,84,79,82,85,78,46,83,
89,83,155,68,32,58,68,1607
1280 DATA 79,83,46,83,89,83,155,125,7
0,79,82,77,65,84,32,85,2130
1290 DATA 84,73,76,73,84,89,155,155,6
8,101,118,105,99,101,32,68,3768
1300 DATA 32,58,155,155,160,205,197,2
06,213,160,155,49,46,32,70,111,7272
1310 DATA 114,109,97,116,155,50,46,32
,87,114,105,116,101,32,68,79,2565
1320 DATA 83,155,51,46,32,70,111,114,
109,97,116,32,97,110,100,32,2743
1330 DATA 87,114,105,116,101,32,68,79
,83,155,52,46,32,67,104,97,2116
1340 DATA 110,103,101,32,68,114,105,1
18,101,32,35,155,53,46,32,69,1181
1350 DATA 120,105,116,155,155,79,112,
116,105,111,110,32,62,32,0,73,1680
1360 DATA 110,115,101,114,116,32,68,7
3,83,75,44,32,80,114,101,115,2647
1370 DATA 115,32,82,69,84,85,82,78,0,
155,155,70,111,114,109,97,4520
1380 DATA 116,116,105,110,103,46,46,4
6,0,155,155,87,114,105,116,105,4635
1390 DATA 110,103,32,68,79,83,46,46,4
6,0,155,155,87,114,105,116,3794
1400 DATA 105,110,103,32,66,66,75,67,
80,46,46,46,0,78,101,119,698
1410 DATA 32,68,114,105,118,101,32,35
,32,0,32,155,155,0,255,255,6460
1420 DATA 124,29,198,35,160,0,169,185
,26,3,0,0,0,0,0,0,6063
```

```
1430 DATA 118,30,68,49,58,65,85,84,79
,82,85,78,46,66,65,84,1198
1440 DATA 155,0,10,36,139,36,162,0,14
2,217,35,134,8,232,134,9,4761
1450 DATA 32,199,32,169,83,160,36,32,
97,30,32,225,30,32,195,34,3067
1460 DATA 32,149,31,169,4,157,74,3,32
,86,228,8,32,225,30,40,2471
1470 DATA 16,10,165,6,208,3,108,10,0,
108,250,191,169,81,160,35,6332
1480 DATA 32,97,30,169,186,160,35,32,
97,30,104,104,76,28,34,125,2318
1490 DATA 155,66,66,75,32,67,80,32,45
,32,40,67,41,32,49,57,8250
1500 DATA 56,55,32,65,78,65,76,79,71,
32,67,111,109,112,117,116,3590
1510 DATA 105,110,103,155,98,121,58,3
2,66,114,121,97,110,32,83,99,3578
1520 DATA 104,97,112,112,101,108,155,
0,224,2,2,0,10,36,0,0,7532
```

LISTING 5: ASSEMBLY

```
0100      .OPT NO LIST
0110 ;-----------------------------
0120 ;BATCH File Processing Disable
0130 ;Utility.
0140 ;
0150 ;External Command #1
0160 ;
0170 ;by: Bryan Schappel
0180 ;
0190 ;CP Call:
0200 ;DISBAT
0210 ;-----------------------------
0220 ;
0230 ;Equates
0240 ;
0250 FIND_E = $20F7
0260 BATCH = $2208
0270 MAIN =  $217D
0280 SKIP_BAT = $2133
0290 HATABS = $031A
0300 INDEX = $23D8
0310 RESET = $20C1
0320 ;
0330      *=  $0500
0340 ;
0350 START PLA
0360      PLA
0370      LDA BATCH
0380      CMP #$A0     ;LDY #
0390      BEQ DIS_BAT
0400 ;
0410 ;Re-enable Batch Processing
0420 ;
0430      LDA #$A0     ;LDY #
0440      STA BATCH
0450      LDA #0
0460      STA BATCH+1
0470      LDA #$A9     ;LDA #
0480      STA BATCH+2
0490      LDA #$B9     ;LDA HATABS,Y
0500      STA FIND_E
0510      LDA # <HATABS
0520      STA FIND_E+1
0530      LDA # >HATABS
0540      STA FIND_E+2
0550      BNE LEAVE
0560 ;
0570 ;Disable BATCH Processing
0580 ;
0590 DIS_BAT LDA #$4C ;JMP
0600      STA FIND_E
0610      STA BATCH
0620      LDA # <SKIP_BAT
0630      STA FIND_E+1
0640      LDA # >SKIP_BAT
0650      STA FIND_E+2
0660      LDA # <MAIN
0670      STA BATCH+1
0680      LDA # >MAIN
0690      STA BATCH+2
0700      LDY INDEX
0710      LDA #$00
0720      STA HATABS,Y
0730      LDA #$E4
0740      STA HATABS+1,Y
0750 LEAVE JMP $E474
```

LISTING 6: ASSEMBLY

```
0100      .OPT NO LIST
0110 ;-------------------
0120 ;MEMORY DUMPER
0130 ;External Command #2
0140 ;
0150 ;By: Bryan Schappel
0160 ;
0170 ;CP Call:
0180 ;DUMP [addr]
0190 ;-------------------
0200 ;
0210 ;Equates
0220 ;
0230 HEXDIG = $2369
0240 GRAB_HEX = $1E9A
0250 CONSOL = $D01F
0260 LNPOS = $DF
0270 SAVEY = $E0
0280 EOL =   $9B
0290 FR0 =   $D4
0300 DSPFLG = $02FE
0310 MYBUF = $0500
0320 ADDRESS = $23B5
0330 INDR =  $CB
0340 ICCOM = $0342
0350 ICBAL = $0344
0360 ICBAH = $0345
0370 ICBLL = $0348
0380 ICBLH = $0349
0390 CIOV =  $E456
0400 ;
0410      *=  $2480
0420 ;
0430 START LDY LNPOS
0440      LDA MYBUF,Y
0450      CMP #EOL
0460      BNE GET_HEX
0470      LDA ADDRESS
0480      STA FR0
0490      LDA ADDRESS+1
0500      STA FR0+1
0510      JMP HEX1
0520 ;
0530 GET_HEX JSR GRAB_HEX
0540      BCC HEX1
0550      RTS
0560 ;
0570 ;Memory Dump is Here
0580 ;
0590 HEX1 LDA FR0
0600      STA INDR
0610      LDA FR0+1
0620      STA INDR+1
0630 NEXT LDA INDR+1
0640      LDY #0
0650      JSR MAKE_HEX
0660      LDA INDR
0670      LDY #2
0680      JSR MAKE_HEX
0690      LDY #0
0700 LOOP STY SAVEY
0710      LDA (INDR),Y
0720      PHA
0730      CMP #EOL
0740      BNE NOT_EOL
0750      LDA #27
0760 NOT_EOL STA HEX_BUF+29,Y
0770      LDX SAVEY
0780      LDA POSIT,X
0790      TAY
0800      PLA
0810      JSR MAKE_HEX
0820      LDY SAVEY
0830      INY
0840      CPY #8
0850      BNE LOOP
0860      TYA
0870      CLC
0880      ADC INDR
0890      STA INDR
0900      BCC NO_HI
0910      INC INDR+1
0920 NO_HI LDX #$00
0930      LDA #9
0940      STA ICCOM
0950      LDA # <HEX_BUF
0960      STA ICBAL
0970      LDA # >HEX_BUF
0980      STA ICBAH
0990      LDA #1
1000      STA ICBLH
1010      STA DSPFLG
```

```
1020      JSR CIOV
1030      BMI XIT
1040      LDA CONSOL
1050      CMP #7
1060      BEQ NEXT
1070 XIT  LDA #0
1080      STA DSPFLG
1090      RTS
1100 ;
1110 ;Make a Number Hexadeximal
1120 ;
1130 MAKE_HEX PHA
1140      AND #$0F
1150      TAX
1160      LDA HEXDIG,X
1170      STA HEX_BUF+1,Y
1180      PLA
1190      AND #$F0
1200      LSR A
1210      LSR A
1220      LSR A
1230      LSR A
1240      TAX
1250      LDA HEXDIG,X
1260      STA HEX_BUF,Y
1270      RTS
1280 ;
1290 ;Buffers
1300 ;
1310 HEX_BUF .BYTE "0000 00 00 00 "
1320      .BYTE "00 00 00 00 00 "
1330      .BYTE "          ",EOL
1340 POSIT .BYTE 5,8,11,14,17,20
1350      .BYTE 23,26
```

LISTING 7: ASSEMBLY

```
0100      .OPT NO LIST
0110 ;-------------------------------
0120 ;BINARY SAVE
0130 ;External Command #3
0140 ;
0150 ;by: Bryan Schappel
0160 ;
0170 ;CP Call:
0180 ;SAVE fname start end [init run]
0190 ;-------------------------------
0200 ;
0210 ;Equates
0220 ;
0230 LNPOS = $DF
0240 GRAB_HEX = $1E9A
0250 FINDFILE = $1FD4
0260 FINDARG = $21ED
0270 CALL_CIO = $1E37
0280 CLOSE1 = $1EDD
0290 SET_DEV = $1F95
0300 SET_OPN = $22C3
0310 RUNAD = $02E0
0320 INITAD = $02E2
0330 FR0 =   $D4
0340 ICCOM = $0342
0350 ICBAL = $0344
0360 ICBAH = $0345
0370 ICBLL = $0348
0380 ICBLH = $0349
0390 AUX1 =  $034A
0400 CIOV =  $E456
0410 LPOS =  $CB
0420 S_SAVE = $CC
0430 E_SAVE = $CE
0440 SL   =  $D6
0450 SH   =  $D7
0460 BLL =   $DA
0470 BLH =   $DB
0480 ;
0490      *=  $2410
0500 ;
0510 START LDY LNPOS
0520      JSR FINDFILE ;filename
0530      LDY LNPOS
0540      JSR FINDARG
0550      JSR GRAB_HEX ;start adr
0560      BCS NO_SAVE
0570      LDA FR0
0580      STA S_SAVE
0590      LDA FR0+1
0600      STA S_SAVE+1
0610      LDY LNPOS
0620      JSR FINDARG
0630      STY LPOS
0640      JSR GRAB_HEX ;end adr
0650      BCS NO_SAVE
0660      LDA FR0
0670      STA E_SAVE
0680      LDA FR0+1
0690      STA E_SAVE+1
0700 ;
0710 ;Start Save
0720 ;
0730      JSR CLOSE1  ;open file
0740      JSR SET_OPN
0750      JSR SET_DEV
0760      LDA #8
0770      STA AUX1,X
0780      JSR CIOV
0790      BPL DO_SAVE
0800 ;
0810 NO_SAVE JMP CLOSE1
0820 ;
0830 DO_SAVE JSR PUT_FF ;send $FF
0840      BMI NO_SAVE ;header
0850      JSR PUT_FF
0860      BMI NO_SAVE
0870      JSR PUT_HEADER ;send load
0880      BMI NO_SAVE ;adr's
0890 ;
0900      LDA S_SAVE  ;get len
0910      STA SL      ;to save
0920      LDA S_SAVE+1
0930      STA SH
0940      LDA E_SAVE
0950      SEC
0960      SBC SL
0970      STA BLL
0980      LDA E_SAVE+1
0990      SBC SH
1000      STA BLH
1010      INC BLL
1020      BNE N_HI
1030      INC BLH
1040 N_HI JSR S_SEG  ;send seg
1050      BMI NO_SAVE
1060      LDY LPOS
1070      JSR FINDARG
1080      STY LPOS
1090      JSR GRAB_HEX ;init adr
1100      BCS NO_SAVE
1110      JSR MOVE_2
1120      LDA # <INITAD
1130      STA S_SAVE
1140      LDA # <[INITAD+2]
1150      STA E_SAVE
1160      LDA # >INITAD
1170      STA S_SAVE+1
1180      STA E_SAVE+1
1190      JSR PUT_HEADER ;put addr's
1200      BMI NO_SAVE
1210      JSR PUT_TEMP ;init adr
1220      BMI NO_SAVE
1230      LDY LPOS
1240      JSR FINDARG
1250      JSR GRAB_HEX ;run adr
1260      BCS NO_SAVE
1270      JSR MOVE_2  ;save it
1280      LDA # <RUNAD ;headers
1290      STA S_SAVE
1300      LDA # <[RUNAD+2]
1310      STA E_SAVE
1320      LDA # >RUNAD
1330      STA S_SAVE+1
1340      STA E_SAVE+1
1350      JSR PUT_HEADER ;send 'em
1360      BMI NO_SAVE
1370      JSR PUT_TEMP ;send run adr
1380      JMP CLOSE1  ;all done
1390 ;
1400 ;Send a $FF
1410 ;
1420 PUT_FF LDA #$FF
1430 ;
1440 ;Put 1 byte to file
1450 ;
1460 PUT_BYTE PHA
1470      LDX #$10
1480      LDA #0
1490      STA ICBLL,X
1500      STA ICBLH,X
1510      LDA #11
1520      STA ICCOM,X
```

```
1530      PLA
1540      JMP CIOV
1550 ;
1560 ;Put the TEMP location
1570 ;
1580 PUT_TEMP LDA TEMP
1590      JSR PUT_BYTE
1600      BMI P_RTS
1610      LDA TEMP+1
1620      JMP PUT_BYTE
1630 ;
1640 ;Move FR0 to TEMP
1650 ;
1660 MOVE_2 LDA FR0
1670      STA TEMP
1680      LDA FR0+1
1690      STA TEMP+1
1700 P_RTS RTS
1710 ;
1720 ;Put the Header Out
1730 ;
1740 PUT_HEADER LDA #4
1750      STA BLL
1760      LDA #0
1770      STA BLH
1780      STA SH
1790      LDA #S_SAVE
1800      STA SL
1810 S_SEG LDA #11
1820      LDX #$10
1830      JMP CALL_CIO
1840 ;
1850 TEMP .DS 2
```

LISTING 8: ASSEMBLY

```
0100      .OPT NO LIST
0110 ;--------------------
0120 ;DISK FORMATTER
0130 ;External Command #4
0140 ;
0150 ;by: Bryan Schappel
0160 ;
0170 ;CP Call:
0180 ;FORMAT
0190 ;--------------------
0200 ;
0210 ;Equates
0220 ;
0230 CLOSE1 = $1EDD
0240 CLOSE2 = $1EE1
0250 SET_DEV = $1F95
0260 SET_OPN = $22C3
0270 EPRINT = $1E61
0280 ICCOM = $0342
0290 ICBAL = $0344
0300 ICBAH = $0345
0310 ICBLL = $0348
0320 ICBLH = $0349
0330 AUX1 =  $034A
0340 AUX2 =  $034B
0350 CIOV =  $E456
0360 EOL =   $9B
0370 DEVICE = $23B7
0380 IOERROR = $1FA5
0390 ORIGIN = $1D7C
0400 EPRDN = $1E76
0410 ERROR = $CB
0420 BATCH = $2208
0430 FIND_E = $20F7
0440 ;
0450 ;PRINT macro
0460 ;
0470      .MACRO PRINT
0480      LDA # <%1
0490      LDY # >%1
0500      JSR EPRINT
0510      .ENDM
0520 ;
0530      *=  $2480
0540 ;
0550 START LDA DEVICE+1
0560      STA MY_DEV
0570      LDY #2
0580 SV   LDA BATCH,Y
0590      STA HOLD,Y
0600      LDA NORM,Y
0610      STA BATCH,Y
0620      LDA FIND_E,Y
0630      STA HOLD+3,Y
0640      LDA NORM+3,Y
0650      STA FIND_E,Y
0660      DEY
0670      BPL SV
0680 BEGIN  PRINT  TITLE
0690      LDA MY_DEV
0700      STA FMAT+1
0710      STA DOSNAME+1
0720      STA AUTO+1
0730 ;
0740 M_LOOP JSR GET_KEY
0750      BMI M_LOOP
0760      STA OPTION
0770      SEC
0780      SBC #'1
0790      BMI M_LOOP
0800      CMP #5
0810      BCS M_LOOP
0820      PHA
0830       PRINT  OPTION
0840      STA ERROR
0850      PLA
0860      TAX
0870      BEQ OPT1
0880      DEX
0890      BEQ OPT2
0900      DEX
0910      BEQ OPT3
0920      DEX
0930      BEQ OPT4
0940      BNE OPT5
0950 ;
0960 ;Just Format Disk
0970 ;
0980 OPT1 JSR FORMAT
0990      JMP BEGIN
1000 ;
1010 ;Just Write DOS+BBKCP
1020 ;
1030 OPT2 JSR WRITE_DOS
1040      JMP BEGIN
1050 ;
1060 ;Format and Write DOS
1070 ;
1080 OPT3 JSR FORMAT
1090      LDA ERROR
1100      BNE G_B
1110      JSR WRITE_IT
1120 G_B JMP BEGIN
1130 ;
1140 ;Get new device number
1150 ;
1160 OPT4  PRINT  NEW_DEV
1170      JSR GET_KEY
1180      AND #$7F
1190      STA MY_DEV
1200      JMP BEGIN
1210 ;
1220 ;Return to BBKCP
1230 ;
1240 OPT5 LDY #2
1250 LD   LDA HOLD,Y
1260      STA BATCH,Y
1270      LDA HOLD+3,Y
1280      STA FIND_E,Y
1290      DEY
1300      BPL LD
1310      RTS
1320 ;
1330 ;Format the Disk
1340 ;
1350 FORMAT JSR PROMPT
1360      BEQ GO_F
1370      INC ERROR
1380      RTS
1390 GO_F   PRINT  F_MAT
1400      JSR CLOSE1
1410      LDA # <FMAT
1420      STA ICBAL,X
1430      LDA # >FMAT
1440      STA ICBAH,X
1450      LDA #0
1460      STA AUX2,X
1470      STA AUX1,X
1480      LDA #254
1490      STA ICCOM,X
1500      JSR CIOV
1510      BPL G_BEGIN
1520 ;
```

```
1530 OOPS TYA
1540      PHA
1550       PRINT  EOL2
1560      PLA
1570      TAY
1580      JSR IOERROR
1590      INC ERROR
1600       PRINT  H_KEY
1610      JMP KEY_RET
1620 G_BEGIN RTS
1630 ;
1640 ;Write DOS and BBKCP
1650 ;
1660 WRITE_DOS JSR PROMPT
1670      BNE G_BEGIN
1680 WRITE_IT  PRINT  W_DOS
1690      JSR SET_80
1700      LDA # <DOSNAME
1710      STA ICBAL,X
1720      LDA # >DOSNAME
1730      STA ICBAH,X
1740      JSR CIOV
1750      BMI OOPS
1760 ;
1770      JSR SET_80
1780      LDA # <AUTO
1790      STA ICBAL,X
1800      LDA # >AUTO
1810      STA ICBAH,X
1820      JSR CIOV
1830      BMI OOPS
1840      LDX #$10
1850      LDA # <HEADER
1860      STA ICBAL,X
1870      LDA # >HEADER
1880      STA ICBAH,X
1890      LDA #6
1900      STA ICBLL,X
1910      LDA #0
1920      STA ICBLH,X
1930      LDA #11
1940      STA ICCOM,X
1950      JSR CIOV
1960      BMI OOPS
1970      LDX #$10
1980      LDA # <ORIGIN
1990      STA ICBAL,X
2000      LDA # >ORIGIN
2010      STA ICBAH,X
2020      LDA #$39
2030      STA ICBLL,X
2040      LDA #$06
2050      STA ICBLH,X
2060      LDA #11
2070      STA ICCOM,X
2080      JSR CIOV
2090      BMI J_ERR
2100      LDX #$10
2110      LDA #11
2120      STA ICCOM,X
2130      LDA # <END_DATA
2140      STA ICBAL,X
2150      LDA # >END_DATA
2160      STA ICBAH,X
2170      LDA # <DATLEN
2180      STA ICBLL,X
2190      LDA # >DATLEN
2200      STA ICBLH,X
2210      JSR CIOV
2220      BPL J_CL
2230 J_ERR JMP OOPS
2240 J_CL JMP CLOSE1
2250 ;
2260 ;Set up IOCB #1
2270 ;
2280 SET_80 JSR CLOSE1
2290      JSR SET_OPN
2300      LDA #8
2310      STA AUX1,X
2320      LDA #0
2330      STA AUX2,X
2340      RTS
2350 ;
2360 ;Get a Key
2370 ;
2380 GET_KEY LDA $E425
2390      PHA
2400      LDA $E424
2410      PHA
2420      RTS
2430 ;
2440 ;Prompt for a Disk
2450 ;
2460 PROMPT  PRINT  P_TXT
2470 KEY_RET JSR GET_KEY
2480      CMP #EOL
2490      RTS
2500 ;
2510 ;Program Data
2520 ;
2530 FMAT .BYTE "D :",EOL
2540 AUTO .BYTE "D :AUTORUN.SY
2550 DOSNAME .BYTE "D :DOS.SYS
2560 TITLE .BYTE $7D,"FORMAT "
2570      .BYTE "UTILITY",EOL
2580      .BYTE EOL,"Device D"
2590 MY_DEV .BYTE " :",EOL,EOL
2600      .BYTE "  MENU  ",EOL
2610      .BYTE "1. Format",EOL
2620      .BYTE "2. Write DOS",
2630      .BYTE "3. Format and
2640      .BYTE "Write DOS",EOL
2650      .BYTE "4. Change Drive #"
2660      .BYTE EOL
2670      .BYTE "5. Exit",EOL,EOL
2680      .BYTE "Option > ",0
2690 ;
2700 P_TXT .BYTE "Insert DISK, "
2710 H_KEY .BYTE "Press RETURN",0
2720 F_MAT .BYTE EOL,EOL
2730      .BYTE "Formatting...",0
2740 W_DOS .BYTE EOL,EOL
2750      .BYTE "Writing DOS...",0
2760 W_CP .BYTE EOL,EOL
2770      .BYTE "Writing BBKCP...",0
2780 NEW_DEV .BYTE "New Drive # ",0
2790 OPTION .BYTE " "
2800 EOL2 .BYTE EOL,EOL,0
2810 HEADER .WORD $FFFF,$1D7C,$23C6
2820 NORM .BYTE $A0,$00,$A9
2830      .BYTE $B9,$1A,$03
2840 HOLD .WORD $00,$00,$00
2850 ;
2860 ;CP Run Stuff
2870 ;
2880 END_DATA .WORD EPRDN
2890      .BYTE "D1:AUTORUN.BAT"
2900      .BYTE EOL,0
2910      .WORD $240A
2920      .WORD $248B
2930      LDX #0
2940      STX $23D9
2950      STX $08
2960      INX
2970      STX $09
2980      JSR $20C7
2990      LDA #$53     ;credits
3000      LDY #$24
3010      JSR EPRINT
3020      JSR CLOSE2
3030      JSR SET_OPN
3040      JSR SET_DEV
3050      LDA #4
3060      STA AUX1,X
3070      JSR CIOV
3080      PHP
3090      JSR CLOSE2
3100      PLP
3110      BPL HAV_BAT
3120      LDA $06
3130      BNE G_CART
3140      JMP ($0A)
3150 G_CART JMP ($BFFA)
3160 HAV_BAT LDA #$51 ;prompt
3170      LDY #$23
3180      JSR EPRINT
3190      LDA #$BA     ;fname
3200      LDY #$23
3210      JSR EPRINT
3220      PLA
3230      PLA
3240      JMP $221C
3250 ;
3260      .BYTE $7D,EOL
3270      .BYTE "BBK CP - (C) 1987 "
3280      .BYTE "ANALOG Computing",EOL
3290      .BYTE "by: Bryan Schappel"
3300      .BYTE EOL,0
3310      .WORD $02E0
3320      .WORD $02
3330      .WORD $240A
3340 DATLEN = *-END_DATA
```

B&C ComputerVisions

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

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

800/XL/XE SOFTWARE
ALL TITLES ON DISK

ENTERTAINMENT

12 ADAMS ADVENTURES	14.95
ALIANTS	26.95
ALT. REALITY CITY	26.95
ALT. REALITY DUNGEON	26.95
BEYOND CASTLE WOLF	14.95
BISMARK	26.95
BOP & WRESTLE (64K)	26.95
BORDINO:1812	26.95
BOULDERDASH CONSTR.SET	17.95
BRUCE LEE	17.95
CASTLE WOLFENSTEIN	14.95
DALLAS QUEST	7.95
D-BUG	7.95
F-15 STRIKE EAGLE	31.50
FIGHT NIGHT	17.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS	22.50
GUNSLINGER	26.95
HARD HAT MAC	7.95
JAWBREAKER	9.95
KARATEKA	13.50
KNICKERBOCKERS	13.50
KORONIS RIFT	13.50
LAST V-8	8.95
LEADERBOARD	13.50
MAIL ORDER MONSTERS	13.50
MICROLEAGUE BASEBALL	35.95
MONTEZUMA'S REVENGE	14.95
MOUSEQUEST	17.95
MOON SHUTTLE	7.95
NINJA	8.95
OIL'S WELL	9.95
O'RILEY'S MINE	9.95
PIRATES OF BARB. COAST	22.50
PREPPIE I & II	9.95
RESCUE ON FRACTALAS	13.50
SILENT SERVICE	31.50
SPEEDKING	8.95
SPIDERMAN	7.95
SPITFIRE 40	31.50
STARFLEET I	44.95
SPY VS. SPY III	17.95
STOCKMARKET	22.50
STRIP POKER	26.95
SUMMER GAMES	17.95
TAX DODGE	9.95
THE HULK	7.95
TOMAHAWK (64K)	26.95
TOP GUNNER	17.95
TOUCHDOWN FOOTBALL	13.50
TRAILBLAZER	26.95
UNIVERSE	44.95
ZAXXON	13.50

PROGRAMMING

ACTION!	71.95
ACTION! TOOLKIT	26.95
BASIC XL	53.95
BASIC XL TOOLKIT	26.95
BASIC XE	71.95
DISK I/O	26.95
DRAPER PASCAL	44.95
KYAN PASCAL	62.95
LIGHTSPEED C	35.95
LOGO	29.95
MAC/65	71.95
MAC/65 TOOLKIT	26.95
MACRO ASSEMBLER	22.50
PILOT	19.95
SPARTA DOS X	71.95
TOP DOS 1.5 PLUS	35.95

PRODUCTIVITY

ANIMATION STATION	89.95
ATARIWRITER+	39.95
ATARI BOOKKEEPER	24.95
ATARI MUSIC II	14.95
AWARDWARE (1050)	13.50
BANK STREET WRITER	14.95
BLAZING PADDLES	31.50
CELEBRITY COOKBOOK	26.95
COMPUTE YOUR ROOTS	35.95
DATAMANAGER	17.95
FAMILY FINANCE	6.95
GUITAR WIZARD	26.95
HOME ACCOUNTANT	19.95
HOME FILING MANAGER.	6.95
HOMEPAK	24.95
INVENTORY MASTER	80.95
LETTER WIZARD	29.95
MUSIC CONSTRUCTION SET	13.50
NEWSROOM (1050 - 64K)	13.50
NEWS STATION	26.95
NEWS STA. COMPANION.	26.95
PAGE DESIGNER	26.95
PRINT POWER (1050)	13.50
PRINTKIT (1050)	13.50
PRINTSHOP	34.95
P.S. COMPANION (64K)	24.95
P.S.GRAPHICS LIBRARY 1	17.95
P.S.GRAPHICS LIBRARY 2	17.95
P.S.GRAPHICS LIBRARY 3	17.95
PROOF READER	17.95
PUBLISHING PRO	35.95
RUBBER STAMP	26.95
SYNTREND	14.95
SUPER MAILER	35.95
THE LOTTO PROGRAM	17.95
TIMEWISE	6.95
TURBOWORD/80 COLUMN REQUIRES XEP80	44.95
VIDEO TITLESHOP (64K)	26.95
GRAPHICS COMPANION.	17.95
VIRTUOSO	29.95
VISICALC	24.95

EDUCATION

ATARI LIGHT MODULE (REQ.ATARILAB STARTER)	9.95
BUZZWORD	35.95
GRANDMA'S HOUSE (-10)	9.95
HEY DIDDLE (AGE 3-10)	9.95
MASTER TYPE	14.95
PLANATARIUM	22.50
STATES AND CAPITALS	9.95
TOUCH TYPING	9.95
CBS (AGE 3-6):	
ASTROGROVER	8.95
BIG BIRD SPEC DELIVE	8.95
ERNIE'S MAGIC SHAPE.	8.95
DESIGNWARE:	
MATHMAZE (6-11)	35.95
MISSION ALGEBRA (13+)	35.95
SPELLICOPTER (6-11)	35.95
TINK TONK (AGE 4-6):	
ABC'S	8.95
COUNT AND ADD	8.95
SMART THINKER	8.95
SPELLING	8.95
SUBTRACTION	8.95
THINKING SKILLS	8.95
ALL 6 TINK TONKS	39.95
UNICORN:	
10 LITTLE ROBOTS (PRE-SCHOOL)	26.95
FUN BUNCH (6-ADULT)	26.95
RACECAR RITHMETIC (AGE 6+)	26.95
WEEKLY READER (PRE-SCHOOL):	
STICKY BEAR SHAPES	26.95
STICKY BEAR NUMBERS.	26.95
STICKY BEAR ABC'S	26.96
STICKY BEAR OPPOSITE	26.95

800/XL/XE SOFTWARE
ALL TITLES ON CARTRIDGE

ENTERTAINMENT

ALIEN AMBUSH	9.95
ACE OF ACES (XL/XE)	24.95
ARCHON	19.95
ASTEROIDS	15.95
ATARI TENNIS	9.95
BALL BLAZER	19.95
BARNYARD BLASTER	24.95*
BATTLEZONE	19.95
BLUE MAX	19.95
CAVERNS OF MARS	14.95
CENTIPEDE	14.95
CHICKEN	9.95
CHOPLIFTER	14.95
CLAIM JUMPER	9.95
CLOUDBURST	9.95
CRIME BUSTER(XL/XE)	24.95*
CROSSBOW	24.95*
CROSSFIRE	9.95
DAVIDS MIDNIGHT MAGIC	19.95
DEFENDER	14.95
DELUXE INVADERS	7.95
DESERT FALCON	19.95
DIG DUG	19.95
DONKEY KONG	5.00
DONKEY KONG JR.	19.95
EASTERN FRONT (1941)	19.95
E.T. PHONE HOME	9.95
FIGHT NIGHT	19.95
FINAL LEGACY	19.95
FOOD FIGHT (XL/XE)	19.95
FOOTBALL	14.95
FROGGER	14.95
GALAXIAN	19.95
GATO	24.95
GORF (400/800)	5.00
GYRUSS	14.95
HARDBALL	19.95
INTO THE EAGLES NEST	19.95
JOURNEY TO PLANETS	9.95
JOUST	19.95
JUNGLE HUNT	19.95
KABOOM!	14.95
LODE RUNNER	24.95
MARIO BROS.	19.95
MILLIPEDE	19.95
MISSILE COMMAND	5.00
MOON PATROL	19.95
MR. COOL	9.95
MS. PAC MAN	19.95
NECROMANCER	19.95
ONE ON ONE (XL/XE)	19.95
PAC MAN	5.00
PENGO	19.95
POLE POSITION	19.95
POPEYE	14.95
Q-BERT	14.95
QIX	14.95
RESCUE ON FRACTALAS	19.95
RETURN OF THE JEDI	14.95
ROBOTRON:2084	19.95
SKY WRITER	14.95
SLIME (400/800)	9.95
SPACE INVADERS	14.95
STAR RAIDERS	5.00
STAR RAIDERS II	19.95
SUPER BREAKOUT	9.95
TRACK & FIELD	24.95
TURMOIL	9.95
WIZARD OF WOR	5.00

* REQUIRES LIGHT GUN

PRODUCTIVITY

ATARIWRITER	19.95
MICROFILER	22.50

EDUCATION

ATARILAB STARTEER SET	29.95
MATH ENCOUNTERS	9.95
FISHER PRICE (PRE SCHOOL):	
DANCE FANTASY	8.95
LINKING LOGIC	8.95
LOGIC LEVELS	8.95
MEMORY MANOR	8.95
SPINNAKER (AGE 3-10):	
ALF IN COLOR CAVES	9.95
ALPHABET ZOO	9.95
DELTA DRAWING	9.95
FACEMAKER	9.95
KIDS ON KEYS	9.95
KINDERCOMP	9.95
(AGE 7 - ADULT):	
ADV.CREATOR (400/800)	9.95
FRACTION FEVER	9.95

THE BASIC TUTOR
Learn to program in BASIC
Requires a 410 OR 1010 Program Recorder
COMPLETE PACKAGE ONLY **$9.95**

BOOKS ONLY

DE RE ATARI	10.00
LOGO	10.00
ATARIWRITER	10.00
DOS 2.5	12.95
BASIC REFERENCE	5.00
BOOKKEEPER	10.00

SUPER SPECIALS
RECONDITIONED ATARI MERCHANDISE
30 DAY WARRANTY

800 (48K) COMPUTER $79.95	**SPACE AGE JOYSTICK** $5.00	**1030 MODEM WITH EXPRESS!** $24.95
400 (16K) COMPUTER $29.95	**ATARI TRACKBALL** $9.95	**1010 PROGRAM RECORDER** $29.95
1020 COLOR PRINTER/PLOTTER **$19.95** 40 COLUMNS WIDE (NEW IN BOX)	**ATARI BOOKKEEPER** $14.95 - NO BOX **ATARI NUMERIC KEYPAD $7.95**	**DISKETTES AS LOW AS 20 CENTS** 10 FOR $4.00 100 FOR $29.95 1000 FOR $200 MOST ARE UNNOTCHED WITH OLD SOFTWARE

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

Phone orders accepted TUESDAY THROUGH FRIDAY from 10:00 am to 6:00 pm PST.

We carry a complete line of ATARI products and have a large public domain library.
Write or call for free catalogue. (408) 749-1003 TUE - FRI 10AM - 6 PM

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

CIRCLE #103 ON READER SERVICE CARD.

BASIC Editor II

by Clayton Walnum

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

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

Typing in the Editor

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

Disk version:

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

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

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

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

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

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

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

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

Cassette version:

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

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

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

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

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

(6) Rewind the cassette.

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

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

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

Using the Editor

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

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

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

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

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

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

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

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

Leaving the Editor

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

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

Large listings

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

The post-processor

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

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

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

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

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

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

Murphy's Law

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

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

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

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

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

Listing 1.
BASIC listing.

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

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

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

Listing 2.
BASIC listing.

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

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

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

continued from page 35

```
1850      .BYTE 0
1860 M44  .BYTE "Your Mission was suc"
1870      .BYTE "cessful. ",0
1880 M45  .BYTE "Water floods the com"
1890      .BYTE "partment. ",0
1900 M46  .BYTE "You drown in the mur"
1910      .BYTE "ky waters..blub.. ",0
1920 M47  .BYTE "Secret Agent: Missi"
1930      .BYTE "on 1 ",0
1940 M48  .BYTE "  (c) 1988 by Barry"
1950      .BYTE " Kolbe ",0
1960 M49  .BYTE "Press START to try "
1970      .BYTE "again. ",0
1980 M50  .BYTE "Shoot at what? ",0
1990 M51  .BYTE "There's no room. ",0
2000 M52  .BYTE "The escaping viurs a"
2010      .BYTE "ttacks your body. ",0
2020 M53  .BYTE "A steel door blocks "
2030      .BYTE "the east door. ",0
2040 M54  .BYTE "It has a gravity con"
2050      .BYTE "trolled switch. ",0
2060 ;
2070 ;They Gotcha!
2080 ;
2090 DEADTAB .WORD M34 ;robot
2100      .WORD M36   ;ape
2110      .WORD M38   ;floor
2120      .WORD M39   ;gas
2130      .WORD M40   ;x-ray
2140      .WORD M43   ;lasers
2150      .WORD M46   ;flood
2160      .WORD M52   ;drop vial
2170 ;
2180 ;Descriptions of Rooms
2190 ;
2200 D0   .BYTE "The entrance to the "
2210      .BYTE "Hanover-Tyler resea"
2220      .BYTE "rch facility lies to"
2230      .BYTE " the west. ",0
2240 D1   .BYTE "A desk is in the co"
2250      .BYTE "rner of the room. "
2260      .BYTE 0
2270 D3   .BYTE "There is an oval ta"
2280      .BYTE "ble with chairs. ",0
2290 D4   .BYTE "A food cabinet is "
2300      .BYTE "on the east wall. ",0
2310 D5   .BYTE "Hanging on the wall"
2320      .BYTE " is a metal cabinet "
2330      .BYTE "with a slot. ",0
2340 D7   .BYTE "Red and blue button"
2350      .BYTE "s are near the door"
2360      .BYTE ". ",0
2370 D8   .BYTE "Shelves make up the"
2380      .BYTE " south wall. ",0
2390 D9   .BYTE "An albino ape is in "
2400      .BYTE "a cage. ",0
2410 DC   .BYTE "There are tables, "
2420      .BYTE "chairs, and a vend"
2430      .BYTE "ing machine. ",0
2440 DD   .BYTE "A decorative plant"
2450      .BYTE " sits in the corne"
2460      .BYTE "r. ",0
2470 DE   .BYTE "There is a knob on"
2480      .BYTE " the wall. ",0
2490 DF   .BYTE "A scenic picutre h"
2500      .BYTE "angs on the east wa"
2510      .BYTE "ll and a table lies"
2520      .BYTE " to the west. ",0
2530 D10  .BYTE "Chemicals lie on "
2540      .BYTE "the tables. There's"
2550      .BYTE " a switch by the do"
2560      .BYTE "or. ",0
2570 D11  .BYTE "Looks dangerous i"
2580      .BYTE "n here. ",0
2590 D12  .BYTE "There is a hole "
2600      .BYTE "near the west door"
2610      .BYTE ". ",0
2620 D13  .BYTE "There is a coat ha"
2630      .BYTE "nging on a hook. ",0
2640 D14  .BYTE "There is an ATARI "
2650      .BYTE "130XE computer, driv"
2660      .BYTE "e and printer here."
2670      .BYTE " ",0
2680 D15  .BYTE "A photo-enlarger r"
2690      .BYTE "ests on a lab table"
2700      .BYTE ". ",0
2710 D17  .BYTE "A door in the east"
2720      .BYTE " wall. ",0
2730 D19  .BYTE "The furniture cons"
2740      .BYTE "ists of a desk and a"
2750      .BYTE " cabinet. ",0
2760 D1A  .BYTE "A robot guards the"
2770      .BYTE " exits. ",0
2780 D1B  .BYTE "A 36 inch high cab"
2790      .BYTE "inet is in the cor"
2800      .BYTE "ner. ",0
2810 D1C  .BYTE "Lots of resisters"
2820      .BYTE " and chips clutter "
2830      .BYTE "the tables. ",0
2840 D1D  .BYTE "There is a blue bu"
2850      .BYTE "tton by the door. "
2860      .BYTE 0
2870 ;
2880 ;These replace the regular
2890 ;descriptions when the 'ape'
2900 ;and 'robot' are eliminated
2910 ;
2920 D1F  .BYTE "A mangled robot l"
2930      .BYTE "ies on the floor. ",0
2940 D20  .BYTE "A dead ape lies "
2950      .BYTE "on the floor. ",0
2960 D5AF .BYTE "There is a safe her"
2970      .BYTE "e. ",0
2980 ;
2990 ;No description on some rooms
3000 ;
3010 DFAKE .BYTE $FF
```

LISTING 7: ASSEMBLY

```
0100 ;SAVE#D:Codeit.ASM
0110 ;
0120 ;-------------------------;
0130 ; Text data compressor   ;
0140 ; (c)1988 by Barry Kolbe ;
0150 ;                        ;
0160 ; Original idea by:      ;
0170 ;        Bryan Schappel  ;
0180 ;-------------------------;
0190 ;
0200 ;Takes text and compresses it
0210 ;by first coding it into numbers
0220 ;between 0 and 31. The every 8
0230 ;bytes are compressed into 5.
0240 ;
0250 ;
0260      .OPT NO LIST,OBJ
0270 ;
0280 ;converversions: 'ASCII'=code#
0290 ;
0300 ;     0=0 1-26 =a-z
0310 ;stop =27 as long as the second
0320 ;       set has less than 28
0330 ;       entries
0340 ;codes: .=28 spc=29 caps=30
0350 ;
0360 ;     flag=31 (uses 2nd set)
0370 ;
0380 ;second set of codes
0390 ;
0400 ;flags: '=0  ?=1  !=2  ,=3 -=4
0410 ;     ;=5  /=6  (=7 )=8
0420 ;     0-9 = 9(-)18
0430 ;
0440 ;text must end with $ff
0450 ;
0460      *=  $CB
0470 F    .DS 2         ;from
0480 T    .DS 2         ;to
0490 ;
0500 IBUF =  $7000      ;initial msg
0510 OBUF =  $6000      ;coded msg
0520 DBUF =  $7300      ;decoded msg
0530 CBUF =  $8000      ;cmpressed
0540 UBUF =  $5F00      ;un-comp
0550 OTMP =  $5F00      ;cmpress
0560 CTMP =  $5F20      ;buffers
0570 ;
0580      *=  $4800
0590 ;
0600 BEGIN LDA # <IBUF ;set pointers
0610      STA F
0620      LDA # >IBUF
0630      STA F+1
0640      LDA # <OBUF
0650      STA T
0660      LDA # >OBUF
0670      STA T+1
0680 CODE LDY #0       ;get a byte
0690      LDA (F),Y
0700      BEQ CODY     ;handle 0
0710      CMP #$FF     ;end?
0720      BNE CODD     ;no
0730      LDA #27      ;set end flag
0740      STA (T),Y
```

```
0750      JSR COMPRS  ;compress it
0760      BRK         ;done phase1
0770 ;
0780      JSR UNCOMP  ;uncompress
0790      JSR DECODE  ;decode msg
0800      BRK
0810 ;
0820 CODD CMP #$61    ;lower case?
0830      BCS LOWCASE
0840      CMP #$41    ;upper case?
0850      BCS UPCASE
0860      LDX #1      ;get char
0870 EE1  CMP ENTB1,X
0880      BEQ DO1
0890      DEX
0900      BPL EE1
0910      LDX #18     ;try  2nd set
0920 EE2  CMP ENTB2,X
0930      BEQ DO2
0940      DEX
0950      BPL EE2
0960      SEC         ;error!
0970      BRK
0980 ;
0990 DO2  LDA #31     ;set 2nd flag
1000      STA (T),Y
1010      JSR BUMPT   ;next position
1020      TXA         ;X to A
1030      JMP CODY    ;put in code
1040 ;
1050 DO1  TXA         ;in 1st set
1060      CLC         ;add 28 to 'X'
1070      ADC #28
1080      BNE CODY
1090 ;
1100 ;tables of characters
1110 ;X used as index
1120 ;
1130 ENTB1 .BYTE ", "
1140 ENTB2 .BYTE "'?!,-:/()"
1150      .BYTE "0123456789"
1160 ;
1170 LOWCASE SEC      ;make 1-26
1180      SBC #$60
1190 CODY STA (T),Y   ;save
1200      JSR BUMBOTH ;increase
1210      JMP CODE    ;both indirects
1220 ;
1230 UPCASE PHA       ;save it
1240      LDA #30     ;set caps flag
1250      STA (T),Y
1260      JSR BUMPT   ;up pointer
1270      PLA         ;get char
1280      SEC
1290      SBC #$40    ;1-26
1300      STA (T),Y
1310      JSR BUMBOTH ;bump ptrs
1320      JMP CODE    ;again
1330 ;
1340 BUMPF INC F      ;bump ptr f
1350      BNE NOF
1360      INC F+1
1370 NOF RTS
1380 ;
1390 BUMPT INC T      ;bump ptr t
1400      BNE NOT
1410      INC T+1
1420 NOT RTS
1430 ;
1440 BUMBOTH JSR BUMPF
1450      JSR BUMPT
1460      RTS
1470 ;
1480 ;the decode section
1490 ;
1500 DECODE LDA # <UBUF ;set pntrs
1510      STA F
1520      LDA # >UBUF
1530      STA F+1
1540      LDA # <DBUF
1550      STA T
1560      LDA # >DBUF
1570      STA T+1
1580 DCODE LDY #0     ;get a byte
1590      LDA (F),Y
1600      BEQ DCC     ;zero?
1610      CMP #31     ;second set?
1620      BEQ BIG     ;yes
1630 ;less than 31
1640      CMP #30     ;capitals?
1650      BEQ CAPS
1660      CMP #27     ;end of text?
1670      BNE UNO
1680      LDA #$FF    ;set flag
1690      STA (T),Y
1700      RTS
1710 ;
1720 UNO BCC LOOW     ;less than 27
1730 SPEC SEC         ;over so -28
1740      SBC #28
1750      TAX
1760      LDA ENTB1,X ;get ASCII
1770      JMP DCC
1780 ;lower case
1790 ;
1800 LOOW CLC         ;add proper
1810      ADC #$60    ;amount
1820 DCC STA (T),Y
1830      JSR DBUMB   ;double bump
1840      JMP DCODE   ;more
1850 ;
1860 CAPS JSR BMF     ;bump from
1870      LDA (F),Y   ;get code
1880      CLC         ;add amount
1890      ADC #$40
1900      BNE DCC     ;decode
1910 ;
1920 BIG JSR BMF      ;bump from
1930      LDA (F),Y
1940      TAX         ;get second
1950      LDA ENTB2,X ;decode byte
1960      JMP DCC
1970 ;
1980 STOP CLC         ;all done
1990      LDA #$FF    ;show end of
2000      STA (T),Y   ;text
2010      RTS
2020 ;
2030 BMF INC F        ;bump f by 1
2040      BNE BM1
2050      INC F+1
2060 BM1 RTS
2070 ;
2080 BMD INC T        ;bump t by 1
2090      BNE BM2
2100      INC T+1
2110 BM2 RTS
2120 ;
2130 DBUMB JSR BMF    ;bump both f
2140      JSR BMD     ;and t
2150      RTS
2160 ;
2170 ;Compress Text:8 bytes to 5
2180 ;
2190 COMPRS LDA # <OBUF ;set pointrs
2200      STA F
2210      LDA # >OBUF
2220      STA F+1
2230      LDA # <CBUF
2240      STA T
2250      LDA # >CBUF
2260      STA T+1
2270 ;
2280 ;get 8 bytes
2290 ;
2300 GET8 LDA #0      ;set end of tex
2310      STA EFLAG   ;flag
2320      LDY #0
2330 C3   LDA (F),Y
2340      CMP #27     ;end?
2350      BNE C9
2360      INC EFLAG   ;yes
2370 C9   STA CTMP,Y  ;store bytes
2380      INY
2390      CPY #8
2400      BNE C3
2410 ;
2420 ;compress it:all bytes are in
2430 ;the lower 5 bits-roll them
2440 ;to the top 5 bits and then
2450 ;roll them into a 5 byte
2460 ;output buffer.
2470 ;
2480      LDY #0
2490 C1   LDA CTMP,Y  ;get a byte
2500      ASL A       ;shift up
2510      ASL A
2520      ASL A       ;to hi 5 bits
2530      LDX #6      ;shift 5 bits
2540 C2   ASL A       ;into output
2550      ROL OTMP+4  ;buffer
2560      ROL OTMP+3
2570      ROL OTMP+2
2580      ROL OTMP+1
2590      ROL OTMP
2600      DEX
2610      CPX #1      ;done 5 yet?
2620      CLC
2630      BNE C2
2640      INY
2650      CPY #8      ;done 8 bytes?
2660      BNE C1
2670 ;
2680      LDY #0      ;now move the
2690 C5   LDA OTMP,Y  ;5 bytes out
2700      STA (T),Y   ;to contiguous
2710      INY         ;output memory
2720      CPY #5      ;where it
2730      BNE C5      ;can be SAVEd
2740      LDA T       ;add 5 to
2750      CLC         ;'TO' indirect
2760      ADC #5
2770      STA T
2780      LDA T+1
2790      ADC #0
2800      STA T+1
2810      LDA F       ;add 8 to 'FROM'
2820      CLC         ;indirect
2830      ADC #8
2840      STA F
2850      LDA F+1
2860      ADC #0
2870      STA F+1
2880      LDA EFLAG   ;end yet?
2890      BEQ GET8    ;no
2900      RTS         ;quit
2910 EFLAG .DS 1
2920 ;
2930 ;Uncompress data:get 5 bytes
2940 ;and expand them to 8 bytes
2950 ;
2960 UNCOMP LDA # <CBUF ;set pointrs
2970      STA F
2980      LDA # >CBUF
2990      STA F+1
3000      LDA # <UBUF
3010      STA T
3020      LDA # >UBUF
3030      STA T+1
3040      LDY #7      ;clear out
3050      LDA #0      ;output buffer
3060 U1   STA OTMP,Y
3070      DEY
3080      BPL U1
3090 ;
3100 GET5 LDY #4      ;get 5 bytes
3110 GG   LDA (F),Y
3120      STA CTMP,Y
3130      DEY
3140      BPL GG
3150 ;
3160      LDX #7      ;expand 5 to 8
3170      LDA #0
3180 U4   LDY #6
3190 U2   ROR CTMP    ;slide them
3200      ROR CTMP+1  ;to the right
3210      ROR CTMP+2
3220      ROR CTMP+3
3230      ROR CTMP+4
3240      ROR A       ;A has top
3250      DEY         ;5 bits
3260      CPY #1
3270      CLC
3280      BNE U2
3290      ROR A       ;slide 'em to
3300      ROR A       ;lower 5 bits
3310      ROR A
3320      STA OTMP,X  ;save in the
3330      DEX         ;buffer
3340      BPL U4
3350 ;
3360 ;check if done
3370 ;
3380      LDY #7      ;move to
3390 MM2  LDA OTMP,Y  ;contiguous
3400      STA (T),Y   ;memory
3410      DEY
3420      BPL MM2
3430      LDY #7
3440 CK   LDA OTMP,Y
3450      CMP #27     ;at end?
3460      BEQ DONE    ;yup
3470      DEY
3480      BPL CK
3490 ;
3500      LDA F       ;add 5 to 'FROM'
3510      CLC         ;pointer
3520      ADC #5
3530      STA F
3540      LDA F+1
3550      ADC #0
3560      STA F+1
3570      LDA T       ;add 8 to 'TO'
3580      CLC         ;pointer
3590      ADC #8
3600      STA T
3610      LDA T+1
3620      ADC #0
3630      STA T+1
3640      JMP GET5    ;get some more
3650 ;
3660 DONE RTS         ;done
```

End User

by Arthur Leyenberger

I have just returned from my monthly Atari Users' Group Meeting. Every month I am amazed that more Atari 8-bit owners don't take advantage of these groups, which are a valuable source of information about Atari computers and how to use them. Where else can you find people who can answer both the common and esoteric questions that constantly seem to arise?

I met a couple of new 8-bit users this month, and it brought back memories of when I first started out. One guy, a man in his late twenties or early thirties, asked me what a modem was. When I explained that it was simply a device that allowed two computers to "talk" to each other via a telephone line, I could see his eyes light up as he understood the concept and started thinking about what he might use one for.

Like the majority of other group members, I am always happy to share whatever information I have if it might help someone else get started with their computer. I remember when I first began using an Atari 800. I had what seemed like a million questions and, thanks to the other club members, was able to learn about the computer and how to use it. That's what users' groups are all about.

After this fellow asked me about modems, he then wanted to know how to actually connect one to his computer. I told him about serial ports (for modems) and parallel ports (for printers) and the various options he had. At first the discussion of the Atari 850 Interface, ICD's P:R: Connection and other products seemed to overwhelm him. But after asking a few more questions he understood what the options were, what might be best for him and where he could buy what he wanted. He was glad to have learned something new, and I was glad to have helped him.

Not only do users' groups provide a source of valuable information, they also help in other ways. Most users' groups have a disk library of public domain files that can be had usually for the price of the disk. There are hundreds of useful public domain programs available that perform just about every function. There are simple word processors, spreadsheets, file utilities, file format-conversion programs and much, much more. In addition, most clubs have hundreds of picture files, font files and other types of files that can be used with commercial programs.

Many users' groups also publish a newsletter. As a member of the club, you are entitled to receive the newsletter which usually contains software and hardware reviews, tutorial articles, programs and advertisements by local Atari retailers. It too is a wealth of information.

Users' group meetings also typically have demonstrations of new software and equipment. Some groups even have guest speakers from Atari or various software companies who will demonstrate their products and answer questions. Finally, a users' group can have social benefits as well. I have made a number of friends over the years at our club and you will too, once you join and get involved.

If you just bought an Atari 8-bit computer, or even if you are an old-timer, I urge you to get involved with a local users' group. Not only will you be able to take advantage of the best source of information there is, but you could also provide the help necessary for someone else to overcome a particular hurdle. For more information on Atari Users' Groups, contact Atari Corp., 1196 Borregas Ave., Sunnyvale, CA 94088 (408-745-2000). Send your correspondence to the attention of "Users' Groups."

Get involved with an Atari Users' Group. You'll be glad you did.

SpartaDOS X

ICD has recently released the latest version of their 8-bit disk operating system: *SpartaDOS X*. *SpartaDOS X* comes in a special 64K cartridge that consumes none of the computer's RAM and also has the provision to accept another cartridge. This piggyback feature is useful in that you don't have to remove *SpartaDOS* to use another cartridge. Just plug the other cartridge into the top of the *SpartaDOS* cartridge and you can use both.

SpartaDOS X gives the Atari computer a full complement of DOS commands. In addition, it allows you to create subdirectories that can significantly improve your file management capabilities. Subdirectories are used to help organize the contents of the disk and although they are useful for floppy disks, their use is essential with a hard disk.

If you are familiar with MS-DOS, the disk operating system used with IBM PCs and compatibles, you can appreciate the extent of the *SpartaDOS* commands and how they are used. Just as with MS-DOS, *SpartaDOS X* categorizes the commands into internal and external commands. Internal commands are internal to the command processor itself, and they require no other program to perform the command.

A number of internal file commands are available. The ATTR (Attributes) command lets you change the status any of a number of file attributes directly. These include archive (set when the file is backed up), hidden (allows you to hide or unhide a file or subdirectory and protect (when set, prevents the file from being erased.

Other internal file commands include TYPE (allows you to display the contents of the file), COPY, ERASE, RENAME and DIR (displays a list of files on the disk or in the current subdirectory).

Another two commands, TIME and DATE, let you set the time and date of the R-Time 8 cartridge (if you have it) or the system clock.

Four internal commands let you manipulate subdirectories: MKDIR, RMDIR, CHDIR and PATH. MKDIR creates a subdirectory under a specified drive or another directory. RMDIR removes a directory, but it must not contain any files before it can be removed. CHDIR allows you to change the current working directory or, if no subdirectory name is given, will display the current subdirectory name.

Also called folders or directories, subdirectories can contain other subdirectories in a hierarchical order. The PATH command is used to specify which directories should be searched for commands or programs before the current directory is searched. If, for example, you are two directory levels deep, and you want to run a program in another directory, the PATH commands must have previously been given telling the command processor where to look for the program. Other internal commands allow you to cold boot the computer, PEEK or POKE at memory locations and FORMAT a disk.

There are a number of useful external commands. These commands are found in the CAR: (cartridge) directory and total 48K of cartridge memory. These commands allow you to enter the internal BASIC language in your XL or XE computer (BASIC), enter a cartridge plugged into the top of the *SpartaDOS X* cartridge (CAR), locate a particular file on the disk (FIND) and perform any number of commands on a selected group of files (MENU).

SpartaDOS X also contains a full-featured ARC (archive) utility command (ARC). Compatible with the ARC utilities for the Atari ST and IBM PC, it will take a group of specified files and combine and compress them into a single archive file, requiring a minimum amount of disk space. Existing archive files can also be "unARCed" or broken down into the separate files and uncompressed. Archive files are particularly useful for saving time when uploading and downloading files with a modem.

Similar to MS-DOS, *SpartaDOS X* also contains the capability for batch files and I/O redirection. Batch files are simply a list of valid *SpartaDOS X* commands that are kept in a file and executed all at one time. All batch files have a .BAT filename extension, and you can pass parameters to the batch file by including them in the command line.

For example, suppose you had a batch file called "TEST.BAT" which looked like this:

```
COPY %1 %3

COPY %2 %3/A
```

When you gave the command:

```
COPY FILE1 FILE2 OUTPUT
```

SpartaDOS X would assign the name FILE1 to parameter %1, the name OUTPUT to parameter %3, and combine the two files under the name OUTPUT. Then it would assign the name FILE2 to parameter %2 and append FILE2 to OUTPUT (since the name OUTPUT was already assigned to parameter %3). Up to nine parameters can be passed to a batch file.

Batch files are especially useful for automating a routine chore. For example, you may periodically create a new file, want to append it to an existing file and then want to erase the original file. All of these commands could be placed into a batch file and executed just by typing the name of the batch file.

There is a special batch file called AUTOEXEC.BAT. It can be created like any other batch file, but since it is the default batch file, it will be executed when the command processor is first entered. Any system setup commands such as defining a PATH, changing to a particular drive or directory or sending some printer setup codes to the printer could be put into this file. In this way, these setup commands will automatically be performed when the computer is turned on with the *SpartaDOS X* cartridge installed.

The other MS-DOS-like function is in-

put/output redirection. With *SpartaDOS X* you can divert the output of a single command by including "> >" after the command and specifying a device or filename. For example, let's say you wanted a listing of all of the files on Drive A contained in a separate file. Assuming that Drive A was the default, the DIR command would list the contents of the disk on the screen. By using redirection, you can divert the output of the command that would normally go to the screen, to a file, like this:

DIR > > DIRFILE.

After executing this command, the file called DIRFILE will contain a list of the files on the disk in Drive A.

I have not listed every command available with *SpartaDOS X*. To do so would require more space than I have here. However, I have tried to give you a sample of the power that is available with ICD's *SpartaDOS X*. As you can see, it is the most powerful DOS currently available for the 8-bit Atari computer. It gives the 8-bit Atari user the power that has previously only been available to IBM PC owners.

The version of *SpartaDOS X* I have contains only a preliminary manual. The preliminary manual documents only the user interface and provides general information on the commands. There is no index or table of contents, but it is enough to get you started using the DOS. However, by the time you read this, the final version of the manual will be complete and packaged with the product. Should you buy a copy of the program with the preliminary manual, just send in your warranty card, and ICD will send you the final manual, plus you will be entitled to one ROM upgrade for $8 (normal upgrade price is $20).

ICD has long been supporting the Atari 8-bit computers with such products as the US Doubler (an add-in chip for the Atari 1050 disk drive which makes it a true double-density drive), the P:R: Connection (a hardware interface which provides one parallel and two serial ports), Rambo XL (a 256K memory upgrade for the 800XL and 1200XL computers), and R-Time 8 (a battery-backed clock cartridge that provides time/date stamping for files).

The *SpartaDOS X* cartridge represents the most complete 8-bit DOS now available, is supported by a fine company and retails for $80. For information on *SpartaDOS X* or any of the other fine ICD products, contact ICD, Inc., 1220 Rock St., Rockford, IL 61101-1437. You can also call them at (815) 968-2228.

Arthur Leyenberger is a human factors psychologist and freelance writer living in beautiful New Jersey.

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

```
THE ANALOG #73 DISKETTE CONTAINS 24
MAGAZINE FILES.  THEY ARE LISTED BELOW:

SIDE 1:

FILENAME.EXT  LANG.   LOAD  ARTICLE NAME
------------  -----   ----  ------------

ACCESS1 .BAS  BASIC   LOAD  ACCESS HIDDEN MEM., L1
ACCESS2 .BAS  BASIC   LOAD  ACCESS HIDDEN MEM., L2
AGENT   .OBJ  ML      (#3)  SECRET AGENT
ALPHABET.BAS  BASIC   LOAD  DIRECTORY ALPHABETIZER
FORMAT  .COM  ML      (#5)  COMMAND PROCESSOR II
DUMP    .COM  ML      (#5)  COMMAND PROCESSOR II
DISBAT  .COM  ML      (#5)  COMMAND PROCESSOR II
SAVE    .COM  ML      (#5)  COMMAND PROCESSOR II
SEC2PRNT.BAS  BASIC   LOAD  SECTOR TO PRINTER
MARBLEMG.BAS  BASIC   LOAD  MARBLE MAGIC
MLEDITOR.BAS  BASIC   LOAD  M/L EDITOR
EDITORII.LST  BASIC   ENTER BASIC EDITOR II


SIDE 2:

FILENAME.EXT  LANG.   LOAD  ARTICLE NAME
------------  -----   ----  ------------

SPYPT1  .M65  MAC/65  LOAD  SECRET AGENT
SPYPT2  .M65  MAC/65  LOAD  SECRET AGENT
SPYPT3  .M65  MAC/65  LOAD  SECRET AGENT
SPYPT4  .M65  MAC/65  LOAD  SECRET AGENT
SPYPT5  .M65  MAC/65  LOAD  SECRET AGENT
SPYPT6  .M65  MAC/65  LOAD  SECRET AGENT
CODEIT  .M65  MAC/65  LOAD  SECRET AGENT
FORMAT  .M65  MAC/65  LOAD  COMMAND PROCESSOR II
DISBAT  .M65  MAC/65  LOAD  COMMAND PROCESSOR II
DUMP    .M65  MAC/65  LOAD  COMMAND PROCESSOR II
SAVE    .M65  MAC/65  LOAD  COMMAND PROCESSOR II
BOOTCAMP.ASM  ASM/ED  ENTER BOOT CAMP
```

```
TO LOAD YOUR ANALOG DISK
------------------------

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

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

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

 EXT  DESCRIPTION
 ---  -----------

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

LOADING NOTES
-------------

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

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

U T I L I T Y

M/L EDITOR

For use in machine-language entry.

by Clayton Walnum

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LISTING 1: BASIC LISTING

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

BASIC TRAINING

by Clayton Walnum

Last month, we learned what a program actually is, and along the way we discovered constants, variables and line numbers. We also started to study two BASIC commands: INPUT and PRINT. However, though we discussed these instructions in a general way, we didn't truly explore their capabilities. Both INPUT and PRINT are very flexible; this month we'll see just *how* flexible.

More About INPUT

More than any other command, INPUT is responsible for making BASIC an interactive language, one that allows for the creation of programs with the ability to easily get data from a user. Previously, we used the INPUT command to get a single value from the keyboard, but its abilities go far beyond that. We can also use INPUT to get multiple values, with a single instruction—and those values need not be of the same data type. (In BASIC, the data types we generally refer to are numerical and string; we'll talk about strings next month.)

To retrieve more than one value with INPUT, we simply add to the command a list of the variables, separated by commas, in which we want the data stored. For example, the command INPUT A, B, C will accept three numerical values from the keyboard and store those values in A, B and C.

There are two ways the user can enter these values. The first method is to type all three values, separated by commas, on the same line, and then press Return. For example, if we respond:

```
? 1,2,3
```

to our example INPUT command, A will get the value 1, B will get the value 2, and C will get the value 3. (Remember, the question mark is the computer's signal to enter something; you don't actually type it.)

The second method is to type each value singly followed by Return. The computer will keep prompting the user with question marks until it gets the number of values it needs:

```
?1
?2
?3
```

These responses give us the same results as before: A gets 1, B gets 2, and C gets 3.

The INPUT command can also be used to get data from other devices, such as a disk or cassette drive. But we're not going to talk about that just yet. We don't want to get ahead of ourselves.

More About PRINT

Just as with INPUT, there are many ways to use PRINT. Some of them are too advanced for this month's discussion, such as PRINTing to a disk file. We'll cover that usage later in this series. One topic we can cover, though, is using PRINT to format output.

We just discussed how an INPUT statement can incorporate a variable list. If you think for a minute, you'll realize that PRINT (which is a form of output) is really just the opposite of INPUT (which is a form of, well, *input*), so it follows that a PRINT command can also have a data list. The difference is that the values in the list are output rather than input, and it's possible to use not only variables but constants, as well. In fact, we can even use mathematical expressions. Further, by using commas and semicolons, we have some control over the format in which the data is output. As you can see, PRINT is one of the most powerful commands in BASIC.

Here's an example of a PRINT command with a data list:

```
PRINT "A+B+C=";A+B+C
```

Do you know what will happen here? Let's assume that right before this PRINT we had the INPUT statement we discussed previously, and let's further assume that we had entered the same values: 1, 2 and 3. The output of this PRINT would look like this:

```
A+B+C=6
```

Let's take a closer look at how we got this result.

The "A+B+C=" in the output is the string constant immediately following the PRINT command. You'll recall that anything within quotes is a string constant and that PRINT will output the string exactly as it appears, minus the quotes.

Following the string constant, we have a semicolon, which tells PRINT to stay on the same line and place the next piece of output immediately after the previous one. In our case the next item in the list is an arithmetic expression, which must be solved in order

MORE About INPUT and PRINT

to get the data we want to display. The expression's solution is, of course, 6, so a "6" is printed immediately following the string constant.

Here is a sample program that illustrates all of the above concepts, along with a typical "run" of the program (what we might see on the screen were we to run the program and respond to the prompts).

```
10 PRINT "Type three numbers:"
20 INPUT NUM1,NUM2,NUM3
30 PRINT
40 PRINT "The average of your numbers"
50 PRINT "(";NUM1;", ";NUM2;" and ";NU
M3;") is ";(NUM1+NUM2+NUM3)/3;"."
```

Sample run:

```
Type three numbers:
?5,10,15

The average of your numbers
(5, 10 and 15) is 10.
```

The INPUT command in the above example should be easy to understand, but can you follow what's happening with the PRINT statements in Lines 30 through 50? In Line 30, because there is no "argument" (data to be processed by a command) following the PRINT statement, it simply prints a blank line. On our screen, that gives us some space between our input and the results of that input, making the display easier to read.

Line 40 simply prints this string constant:

```
The average of your numbers
```

No problem understanding that, right? Anything between the quotes is printed verbatim.

After the string is printed, we go to a new line, because the PRINT statement didn't end with a semicolon.

Line 50 is a bit tricky. When analyzing this line of code, you have to match the open and close quotes carefully, so you'll be able to accurately separate the constants from the variables. There are actually nine arguments after this PRINT statement. They are:

```
"("
NUM1
", "
NUM2
" and "
NUM3
") is "
(NUM1+NUM2=NUM3)/3
"."
```

Each of these arguments is separated by a semicolon, which means that they will be printed one right after the other, without any extra spaces and without going to a new line. The first piece of data, a string constant, is printed on the line immediately following the string that was printed in Line 40, giving us this on the screen:

```
The average of your numbers
(
```

Next, the value of the variable NUM1 is printed:

```
The average of your numbers
(5
```

The third argument is a string constant made up of a comma and a space. This string is printed, giving us:

```
The average of your numbers
(5,
```

The value of NUM2 is then printed:

```
The average of your numbers
(5, 10
```

Then the next string constant:

```
The average of your numbers
(5, 10 and
```

Now the value of NUM3 is added:

```
The average of your numbers
(5, 10 and 15
```

Then another string constant:

```
The average of your numbers
(5, 10 and 15) is
```

And a mathematical expression, the result of which will be printed:

```
The average of your numbers
(5, 10 and 15) is 10
```

And finally, we conclude with the last string constant, a period:

```
The average of your number
(5, 10 and 15) is 10.
```

In all our examples, we've been separating the data in the list with semicolons, but you can also use commas, which will cause each piece of data to be printed at the next tab stop. The default tabs for the Atari are eight spaces apart. For example the statement:

```
PRINT "ONE","TWO"
```

will give you:

```
ONE     TWO
```

You should note that you may place a semicolon or a comma at the end of a PRINT statement. For example, look at these two statements:

```
PRINT "THIS IS A TEST";
PRINT "THIS IS NOT"
```

They will give you this result:

```
THIS IS A TESTTHIS IS NOT
```

The semicolon at the end of the first PRINT statement prevents the cursor from advancing to the next line. (The cursor marks where your next output will appear.)

Till Next Time

I'm sure that our discussion of Atari BASIC's flexible INPUT and PRINT command has left your head spinning, so we'll take a break here. Till next time, keep studying.

continued from page 19

cation. This section uses a much simpler filtering system composed of two strings of 15 characters each; one contains the "before" data and the other contains the "after" data. For each of the 128 bytes in a sector, the current character is searched for in the first string and, when found, is changed to the corresponding character in the second string. If it is not found, the character is changed to a "?" for further editing by the programmer.

The dungeon printing portion is similar to that of the city, but it prints a border of asterisks around the map and only prints 16 character lines. If the continue option is chosen after the completion of a level, the program adds 2 to the sector number and goes back to the reading part of the dungeon section to read the next level. Lines 4000 through 4990 contain a simple LPRINT set that prints a map legend for quick identification of map features, and at Line 10000 the subroutines begin.

The 10000 block loads in one sector and stores it in page 6 memory by passing the address of the ML routine, an "82" (the ATASCII code for read) and the sector number to be read to the ML routine via a USR call. In the same manner, the 11000 block filters the 128 bytes in page 6 using the data contained in FILTER$ and stores the new data at the location of TCITY$ to be loaded by the BASIC program into CITY$ and later printed. The 30000 block merely contains the massive amounts of data for the routines and the sections that load the data into the proper strings.

Possible Modifications

Since this program grew by bits and pieces, added here and there, I had to entirely rewrite it to produce the code shown here. But it will be worth the effort if it enables people to adapt this code to their own purposes more easily. Possible changes include: a screen viewer, a user-modifiable filter and a variable size of map to be loaded. To develop a program to map a different game, the read routines, the filter and the arrays must be altered to fit the size and characters of the new data. It sounds like a lot of work—and it is—but the experience can be educational and the results helpful.

David Hill is an electrical engineering major at the University of Texas in Austin, where he is a member of the Longhorn Marching Band. He began his programming on a TRS-80 Model 1, then progressed to an Atari 800XL, 130XE and finally a 1040ST.

LISTING 1: BASIC

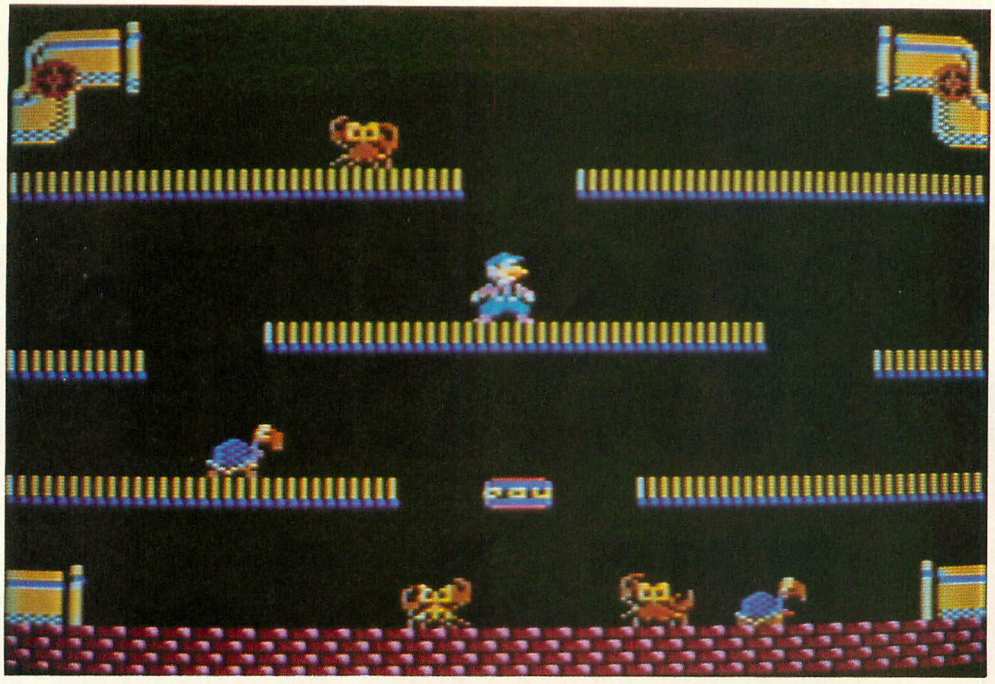
```
EH 0 REM ********************************
DC 2 REM *  SECTOR TO PRINTER MAPPING  *
VM 3 REM *            PROGRAM          *
HU 4 REM *         by David Hill       *
TC 5 REM *                             *
OY 6 REM *        COPYRIGHT 1989       *
LW 7 REM *     BY ANALOG COMPUTING     *
EP 8 REM ********************************
KV 10 REM INITIALIZE AND SETUP ARRAYS
NJ 15 DIM FILTER$(256),CITY$(4096),T$(1)
LF 20 DIM SECTRW$(68),TCITY$(128)
UG 25 DIM MACHFILTER$(38),DUN$(256)
IX 30 DIM DFILT1$(15),DFILT2$(15)
SN 35 DIM CHAR$(1)
QI 40 TCITY$(1)=".":TCITY$(128)="."
FO 45 TCITY$(2)=TCITY$(1)
XW 50 ? "{CLEAR}":? :? :? :? "INITIALIZING..."
CA 55 ? "PLEASE WAIT"
FH 60 GOSUB 30000:REM READ SECTOR INIT
WC 65 GOSUB 30200:REM FILTER INIT
VA 70 GOSUB 30400:REM MACH FILTER INIT
XK 75 GOSUB 30600:REM DUN FILTER INIT
JT 80 ? "PLEASE INSERT DISK AND HIT RETUR
   N";:INPUT T$
XI 85 SEC=1:GOSUB 10000
DK 1000 REM MAIN ROUTINE STARTS HERE
KT 1010 ? "{CLEAR}"
KA 1015 ? "          MAIN MENU:"
YZ 1020 ? "============================"
EA 1025 ? "     (1)  PRINT A CITY"
EO 1030 ? "     (2)  PRINT A DUNGEON"
RI 1035 ? "     (3)  PRINT A MAP LEGEND"
ZR 1040 ? "     (4)  QUIT"
OB 1045 ? "ENTER CHOICE:(1-4)";:INPUT CH
CZ 1050 IF CH<1 OR CH>4 THEN 1000
UL 1055 ON CH GOTO 2000,3000,4000
WD 1060 ? "DO YOU WANT TO QUIT?(Y/N)";
GN 1065 INPUT T$:IF T$<>"Y" THEN 1000
FI 1070 END
OA 2000 REM PRINT CITY SECTION
MK 2010 ? "{CLEAR}":?
XY 2015 ? "          CITY PRINT MENU:"
ZA 2020 ? "============================"
IR 2025 ? "     (1)  LCB'S CASTLE"
UV 2030 ? "     (2)  EXODUS' CASTLE"
FJ 2035 ? "     (3)  LCB'S TOWN"
SI 2040 ? "     (4)  MOON"
OM 2045 ? "     (5)  YEW"
PR 2050 ? "     (6)  MONTOR EAST"
MH 2055 ? "     (7)  MONTOR WEST"
TY 2060 ? "     (8)  GREY"
LS 2065 ? "     (9)  DAWN"
SW 2070 ? "     (10) DEVIL GUARD"
TB 2075 ? "     (11) FAWN"
JF 2080 ? "     (12) DEATH GULCH"
CC 2085 ? "     (13) AMBROSIA"
PU 2090 ? "     (14) SOSARIA"
YS 2095 ? "ENTER # OR 15 TO GO BACK:";
FC 2100 INPUT S:IF S=15 THEN 1000
KI 2105 IF S<1 OR S>14 THEN 2000
SW 2110 S=(S-1)*32+1:? "{CLEAR}":? "READING.."
VD 2115 FOR OFFSET=0 TO 31:SEC=S+OFFSET
FN 2120 GOSUB 10000:REM READ SECTOR
WK 2125 POKE ADR(MACHFILTER$),104
JJ 2130 GOSUB 11000:REM FILTER SECTOR
RT 2135 CITY$(OFFSET*128+1,OFFSET*128+128
   )=TCITY$(1,128)
DF 2140 NEXT OFFSET
AP 2145 ? :? "PRINTING..."
HB 2150 FOR LINE=1 TO 64
ED 2155 START=(LINE-1)*64+1
WV 2160 LPRINT CITY$(START,START+63)
IJ 2165 NEXT LINE:GOTO 2000
WB 3000 REM DUNGEON PRINT SECTION
```

```
WD 3010 REM PUT YOUR PRINTER CODE HERE
AJ 3015 ? "K":? :? "      DUNGEON MENU:"
ZB 3020 ? "============================"
OT 3025 ? "     (1)  SOUTH COAST"
FM 3030 ? "     (2)  DUNGEON OF FIRE"
CH 3035 ? "     (3)  TIME AWAITS"
EN 3040 ? "     (4)  CLUES TO FOLLOW"
LT 3045 ? "     (5)  PERINIAN DEPTHS"
PR 3050 ? "     (6)  MINES OF MORINIA"
TP 3055 ? "  ENTER # OR 7 TO GO BACK:";
FK 3060 INPUT S:IF S<1 OR S>7 THEN 3000
CE 3065 IF S=7 THEN 1000
HO 3070 S=S*16+433:? :? "READING..."
TK 3075 FOR OFFSET=0 TO 1:SEC=S+OFFSET
PY 3080 GOSUB 10000:REM READ NEXT SECTOR
DT 3085 FOR BYTE=1 TO 128
RU 3090 CHAR$=CHR$(PEEK(1663+BYTE))
BV 3095 FOR CHAR=1 TO 15
HA 3100 IF DFILT1$(CHAR,CHAR)=CHAR$ THEN
   CHAR$=DFILT2$(CHAR,CHAR):GOTO 3110
MK 3105 NEXT CHAR:CHAR$="?"
GJ 3110 DUN$(OFFSET*128+BYTE)=CHAR$
CI 3115 NEXT BYTE:NEXT OFFSET
ZQ 3120 ? :? "PRINTING..."
YU 3125 LPRINT "********************"
EQ 3130 FOR LINE=1 TO 16
YP 3135 LPRINT "*";DUN$((LINE-1)*16+1,LIN
   E*16);"*"
MI 3140 NEXT LINE
ZA 3145 LPRINT "********************"
NS 3150 ? "TYPE (1) TO READ NEXT LEVEL,"
HP 3155 ? "OR TYPE (2) TO GO BACK:";
WI 3160 INPUT CH:IF CH<>1 THEN 3000
IJ 3165 S=S+2:GOTO 3075
RW 4000 REM PRINT A MAP LEGEND
GR 4010 ? "K":? "  MAP LEGEND PRINTER"
ZT 4015 ? "============================"
YN 4020 ? "TYPE 1 TO PRINT OR 2 TO GO BAC
   K:";
UJ 4025 INPUT CH:IF CH<>1 THEN 1000
CI 4030 LPRINT :LPRINT :LPRINT
ZH 4035 LPRINT "                MAP LEGEND:
   "
OR 4040 LPRINT "========================
   ============"
RV 4041 LPRINT "     [ ] ------WALKWAY"
XM 4042 LPRINT "     [%] ------MOONGATE"
XY 4045 LPRINT "     [#] ------WALL"
LG 4050 LPRINT "     [$] ------CHEST"
OH 4055 LPRINT "     [(] ------LAVA"
NO 4060 LPRINT "     [*] ------SHRINE"
AU 4065 LPRINT "     [+] ------DARK FOREST
   "
WT 4070 LPRINT "     [-] ------LIGHT FORES
   T"
DE 4075 LPRINT "     [.] ------GRASS"
TF 4080 LPRINT "     [:] ------EXODUS"
LM 4085 LPRINT "     [=] ------'BLANK' WAL
   L"
KR 4090 LPRINT "     [>] ------MOUNTAIN"
BS 4095 LPRINT "     [@] ------WHIRLPOOL"
WH 4100 LPRINT "     [B] ------BALRON/DEVI
   L"
NK 4105 LPRINT "     [C] ------CASTLE/CLER
   IC"
RQ 4110 LPRINT "     [D] ------DUNGEON/DRA
   GON/DAEMON"
TL 4115 LPRINT "     [F] ------FIGHTER"
DW 4120 LPRINT "     [G] ------GUARD"
VU 4125 LPRINT "     [H] ------HORSE"
ER 4130 LPRINT "     [I] ------DOOR"
SC 4135 LPRINT "     [J] ------JESTER"
TS 4140 LPRINT "     [M] ------MERCHANT/MA
   N 'O WARS"
ZF 4145 LPRINT "     [P] ------PINCHER"
GH 4150 LPRINT "     [R] ------RANGER"
HI 4155 LPRINT "     [S] ------SEA MONSTER
   /SKELETONS"
JN 4160 LPRINT "     [T] ------THIEF"
EB 4165 LPRINT "     [W] ------WIZARD"
MV 4170 LPRINT "     [|] ------GREAT SERPE
   NT"
ES 4175 LPRINT :LPRINT "    DUNGEON LEGEN
   D"
YC 4180 LPRINT "========================"
EA 4185 LPRINT "     [ ] ------HALLWAY"
XV 4190 LPRINT "     [#] ------WALL"
HZ 4195 LPRINT "     [X] ------SECRET DOOR
   "
XI 4200 LPRINT "     [I] ------NORMAL DOOR
   "
LP 4205 LPRINT "     [$] ------CHEST"
OP 4210 LPRINT "     [T] ------TRAP"
IZ 4215 LPRINT "     [F] ------FOUNTAIN"
KG 4220 LPRINT "     [W] ------WINDS"
IE 4225 LPRINT "     [G] ------GREMLINS"
MG 4230 LPRINT "     [+] ------2 WAY LADDE
   R"
FC 4235 LPRINT "     [^] ------UP LADDER"
ZY 4240 LPRINT "     [v] ------DOWN LADDER
   "
QU 4245 LPRINT "     [R] ------RUNES"
VN 4250 LPRINT "     [L] ------LORD OF TIM
   E"
PR 4990 GOTO 4000
HN 10000 REM READ SECTOR ROUTINE
KP 10010 X=USR(ADR(SECTRW$),82,SEC)
GM 10015 IF X=1 THEN RETURN
BM 10020 ? "ERROR AT SECTOR:";SEC
YN 10025 END
VR 11000 REM MACHINE LANGUAGE FILTER
VC 11010 A=USR(ADR(MACHFILTER$),ADR(TCITY
   $),ADR(FILTER$)):RETURN
RD 30000 REM INITIALIZE READING ROUTINE
MK 30010 FOR K=1 TO 68:READ Q
OI 30015 SECTRW$(K,K)=CHR$(Q):NEXT K
DH 30020 RETURN
RE 30025 DATA 104,104,104,201,083,169,082
   ,144
FH 30030 DATA 002,169,087,072,169,000,072
   ,169
XJ 30035 DATA 001,072,169,000,072,169,128
   ,072
ZF 30040 DATA 169,006,072,072,104,104,141
   ,005
JI 30045 DATA 003,104,141,004,003,104,104
   ,141
BL 30050 DATA 001,003,104,104,141,002,003
   ,104
BD 30055 DATA 141,011,003,104,141,010,003
   ,032
MY 30060 DATA 083,228,173,003,003,133,212
   ,169
IP 30065 DATA 000,133,213,096
JS 30200 REM INITIALIZE FILTER DATA
JL 30210 FOR FIL=1 TO 256:READ A
NF 30215 FILTER$(FIL,FIL)=CHR$(A)
PK 30220 NEXT FIL:RETURN
QW 30225 DATA 126,001,002,003,046,005,006
   ,007
LS 30230 DATA 045,009,010,011,043,013,014
   ,015
DP 30235 DATA 062,017,018,019,068,021,022
   ,023
IJ 30240 DATA 084,025,026,027,067,029,030
   ,031
MU 30245 DATA 032,033,034,035,036,036,036
   ,036
SF 30250 DATA 072,041,042,043,038,045,046
   ,047
SR 30255 DATA 064,049,050,051,083,053,054
   ,055
DS 30260 DATA 077,057,058,059,038,061,062
   ,063
```

SECTOR TO PRINTER MAPPING

continued from page 19

```
JE 30265 DATA 077,065,066,067,074,069,070
   ,071
AY 30270 DATA 071,073,074,075,066,077,078
   ,079
VJ 30275 DATA 070,081,082,083,067,085,086
   ,087
WS 30280 DATA 087,089,090,091,084,093,094
   ,095
TM 30285 DATA 096,097,098,099,083,101,102
   ,103
ZV 30290 DATA 104,105,106,107,068,109,110
   ,111
WU 30295 DATA 080,113,114,115,068,117,118
   ,119
HQ 30300 DATA 066,121,122,123,124,125,126
   ,127
BQ 30305 DATA 043,129,130,131,040,133,134
   ,135
QR 30310 DATA 037,137,138,139,035,141,142
   ,143
UJ 30315 DATA 144,145,146,147,061,149,150
   ,151
VA 30320 DATA 065,153,154,155,066,157,158
   ,159
SF 30325 DATA 067,161,162,163,068,165,166
   ,167
JM 30330 DATA 069,169,170,171,070,173,174
   ,175
SM 30335 DATA 071,177,178,179,072,181,182
   ,183
RX 30340 DATA 073,185,082,082,085,189,190
   ,191
BI 30345 DATA 089,193,194,195,076,197,198
   ,199
LQ 30350 DATA 077,201,202,203,078,205,206
   ,207
ER 30355 DATA 079,209,210,211,080,213,214
   ,215
PD 30360 DATA 087,217,218,219,082,221,222
   ,223
UX 30365 DATA 083,225,226,227,084,229,230
   ,231
HC 30370 DATA 124,233,234,235,124,237,238
   ,239
DN 30375 DATA 240,241,242,243,244,245,246
   ,247
BZ 30380 DATA 042,249,250,251,082,253,254
   ,255
WD 30400 REM MACHINE LANGUAGE FILTER
XB 30410 FOR MAC=1 TO 38:READ A
IH 30415 MACHFILTER$(MAC,MAC)=CHR$(A)
KA 30420 NEXT MAC:RETURN
QK 30425 DATA 104,104,133,204,104,133,203
   ,104
RS 30430 DATA 133,206,104,133,205,169,128
   ,133
GP 30435 DATA 207,166,207,188,127,006,177
   ,205
CB 30440 DATA 164,207,136,145,203,198,207
   ,162
PE 30445 DATA 000,228,207,208,236,096
OO 30600 REM DUNGEON FILTER INIT
GT 30610 FOR Q=1 TO 15:READ A,B
LG 30615 DFILT1$(Q)=CHR$(A)
LU 30620 DFILT2$(Q)=CHR$(B)
HC 30625 NEXT Q:RETURN
FZ 30630 DATA 000,032,128,035,160,088
HZ 30635 DATA 192,073,064,036,004,084
XJ 30640 DATA 002,070,003,087,006,071
HK 30645 DATA 016,094,032,118,048,043
DZ 30650 DATA 005,077,008,082,001,076
```



MARIO BROTHERS

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
XL/XE Cartridge $19.95

Reviewed by Matthew J.W. Ratcliff

Mario Brothers is a faithful conversion of the popular 1983 Nintendo coin-op game of the same name. This five-year-old game still has a lot of charm, despite the fact that Nintendo has released *Super Mario Brothers* and *Super Mario Brothers 2* for their game system. Why is it that Atari is a full five years behind the rest of the video game industry? It is difficult to get excited about a "new" game from Atari that is ancient by current standards (seldom does a coin-op video game last more than a year), but *Mario Brothers* is such a delightful game that one can get fired up about it.

In the two-player mode, Mario the carpenter is joined by his brother Luigi in a combined effort to rid their home's water pipes of some pesky critters. *Mario Brothers* is a running-and-jumping graphics extravaganza. It is not an adventure game where one must explore room after room, searching for villains and treasures.

Its structure is similar to *Donkey Kong*, where Mario first appeared. The screens are all basically the same. The challenges are presented by the creatures, and, of course, the house is never completely rid of the critters. It goes on forever, in the quest for more points. The two-player mode, with each player assisting the other, is a real hoot.

The screen is filled with platforms, with openings that allow Mario to jump from one level to the next. Creatures walk out of water pipes at the top of the screen, along the platforms, fall through the openings and eventually back into the pipes again at the bottom of the display. Each trek through the pipes seems to pep up the little buggers as they go ever faster. Mario must run along underneath the platforms and jump up at the correct moment using his punch to immobilize the critter. He may then jump up to that platform and kick the varmint off.

The simplest pests to deal with are the Shellcreepers. They can be punched from below and then kicked off, if Mario doesn't waste any time. Eventually these little turtle look-alikes will right themselves and begin infesting the pipes once again, if Mario dawdles too much.

Sidesteppers, crab-like creatures, are a bit tougher, since they take two punches before Mario can give them the boot. Fighterflies do not walk, they hop. This makes life even more difficult for Mario, since they can only be punched while touching a platform. When a Fighterfly has been punched once, it can be kicked off. The Slip Ice, a blue pumpkin head, likes to freeze the floors. This can make for a slippery mess. Fortunately, they need be punched only once to be eliminated. If Mario punches the icicles as they are formed by a Slip Ice, they can be prevented.

Fireballs are also a hazard. They don't burn down the house, but Mario and Luigi had best avoid them. These flying, spinning, bouncing sparklers can be punched to rack up some really big points. The comical look on Mario's face when he gets fried is a smashing effect.

Coins come bouncing out of the water pipes occasionally, to be punched or collided with for bonus points. Between some phases a bonus screen, filled with coins, will appear, permitting Mario a few precious seconds to collect them.

When things get particularly hurried and harried, Mario can hit the POW switch, a large button at the bottom center level of each screen. When hit, it is equivalent to punching every creature on the screen once. It can only be used three times per phase, so it must be employed judiciously.

Playing *Mario Brothers* solo is just another video game, an exercise for the wrist. With a friend, either in cooperative or "combat" mode (try to knock your brother off into a creature), this game is an absolute blast to play. Mario and Luigi can jump and bounce off each other to perform some rather unique gymnastic feats in obliterating the little pests.

At the start of each phase, when a new creature is introduced, a brief show is presented, which graphically illustrates the proper method of elimination. This is a nice touch, allowing a novice to sit down and play without any documentation.

Mario begins with four lives and gets a single bonus life at 20,000 points. Beyond that, it is simply a quest for more points. Graphics and sound effects are stunning. There is an occasional flicker of players, when a lot of creatures are on the same platform—limitations of the Atari's player missile graphics.

Mario Brothers is a good game for one, but a fantastic game for two. When two people play this game, it becomes a social event, and a lot of good-natured fun is had by all who watch and play. As a one-player game, it will be fun for only a short while. Graphic adventure games provide entertainment for the solo gamer through exploration and discovery, features not to be found in *Mario Brothers*.

Accessing XL Hidden

by Kevin T. Pate

By these techniques user memory is expanded by 60%! This gives greater capability to the already powerful Atari XL.

The Atari XL computer (800XL) contains 64K of random access memory (RAM). The 8K BASIC read only memory (ROM) and the 16K operating system (OS) ROM, however, masks 24K of this RAM, making it inaccessible to the user. This is because the 6502 microprocessor in the Atari can only address 64K bytes at one time and therefore must deselect part of its RAM so it can access the BASIC ROM or the OS ROM. This article shows how nearly all of the Atari XL's hidden memory can be easily accessed.

Two techniques are described here for the Atari XL computer:

1. Accessing the hidden memory on a single-byte basis, similar to single PEEKs and POKEs in normal BASIC.

2. Accessing large sections or banks of the hidden memory. Banks of the hidden memory (the size of the bank is selectable by the user) can be switched into or out of an area of regular RAM (the location of this area can also be selected by the user).

Technique #1 allows simple access to the hidden RAM for use in data storage, for example. Technique #2 can be used for switching in different screen displays (similar to page flipping except instead of changing the display list screen pointer from one part of memory to another, the area of screen memory is fixed but is modified by loading in a different bank of memory). Different large tables of data (for a database program or graphics program) or large character strings (for word processing) could be switched in and out of the normal free RAM area. The action all takes place instantaneously since the complicated work is all done in machine-language routines.

By these techniques user memory is expanded 22K, or by 60%! This gives greater capability to the already powerful Atari XL computer system.

Single-Byte Access to Hidden Memory

Figure 1 shows a simple breakdown of the Atari 64K memory address space. The BASIC ROM (8K) and the OS ROM (14K, not including the I/O port area) can be "turned off" through the PIA (Peripheral Interface Adapter) Port B. (Port A monitors the two joystick inputs.) This is performed by setting particular bits at address 54017 ($D301) as follows:

Atari Memory

Bit	Value	Function
0	1	OS ROM selected
0	0	OS ROM deselected
1	0	BASIC ROM selected
1	1	BASIC ROM deselected

The normal value for $D301 is 253 which has Bit 1 equal to 0 (BASIC ROM on) and Bit 0 equal to 1 (OS ROM on). Switching these bits enables us to turn off and on the ROMs so that we can access the underlying RAM. Turning the BASIC and OS off and on and accessing the underlying RAM, however, must be performed from a machine-language routine, since a BASIC routine could not be understood if the BASIC interpreter is off—a system crash would occur.

Listing 1 is a program which enables single-byte access to the Atari XL hidden memory. For those memory-conscious individuals, the program in its current form uses about 2K of memory, but could be streamlined down to 1K by removal of REM statements.

Line 140 in the program calls the initialization subroutine, which sets up the machine-language routines. Calling the initialization subroutine, GOSUB 30000, does all the setup work—nothing else needs to be done by the user. Lines 150-200 is a loop which shows how the new memory-access routines are called. To use the new POKE routine, called ZPOKE, the following is done:

1. Put the target address in ZAD (in decimal).
2. Put the value to be stored in Z.
3. Call the subroutine—GOSUB ZPOKE.

To use the new PEEK routine, called ZPEEK, the following is done:

1. Put the target address in ZAD (in decimal).
2. Call the subroutine—GOSUB ZPEEK.
3. The value of the target address will be returned in Z.

It's as simple as that! The procedure isn't much more difficult than using the standard "POKE address,value" or "A=PEEK(address)" in regular BASIC. The subroutines and machine-language routines do all the hard work. Line 180 in the program zeroes out the variable Z between the storing and reading steps to prove that the value is indeed read from memory.

Lines 30000 through 31090 are the new POKE subroutine, ZPOKE, and Lines 32000 through 32090 are the new PEEK subroutine, ZPEEK. In ZPOKE the value of Z and the high and low bytes of the target address ZAD are stored directly into the machine-language routine in RAM. ZPEEK returns to BASIC with the read value in location Z$(1), the first byte of the area set aside for the machine-language routines. This value is then transferred to variable Z for the user.

In order to prevent a system crash from an unexpected interrupt while the BASIC and OS ROMs are turned off, the interrupt systems (display list, vertical blank, BREAK key, data key, serial input and output, and internal timers) are turned off temporarily. The interrupts are controlled by Locations 16 ($0010), 53774 ($D20E) and 54286 ($D40E) in RAM. A value of 0 disables all interrupts. After the BASIC and OS are back on, the interrupts are enabled by putting the original values back in Locations 16, 53774 and 54286.

The subroutines in 30000 through 32090 (initialization, ZPOKE and ZPEEK) are general routines that can be used in any program (Lines 100-210 only serve as an example of how the subroutines are called). Any part of the 64K memory can now be accessed, except the 2K I/O area (53248-55295). Care should be taken with pages 0-6 of memory that are used by the OS or areas used for the BASIC program, data or screen display. (These areas are still accessible by the normal PEEK and POKE commands, anyway.)

Bank Switching of Hidden Memory

Listing 2 is a program which enables the switching of banks of memory into or out of the Atari XL hidden memory. The program in its current form uses about 4K of memory, but without REM statements uses only 2K.

In the program, the size of the hidden memory banks, ZBSZ, is user selectable and may be from as small as 100 bytes each (allowing 225 banks) to as large as 10240 bytes (allowing two banks). Any size between these two limits is acceptable. (Bank size smaller than 100 bytes is possible, but the dimension of ZB$, the bank address character string, would have to be increased in Line 20030.) The total number of banks, ZBTOT, is automatically calculated from the bank size by the following equation:

ZBTOT = INT(12288/ZBSZ)
+ INT(10240/ZBSZ)

ZBTOT is the total number of banks that can fit into the 12K of hidden memory below the I/O area and the 10K of hidden memory above the I/O area. The banks are identified by the bank pointer variable, ZPTR, which ranges from 1 to ZBTOT.

The user also must select the starting address of the target area, ZAD, in regular RAM where the banks will switch in and out of. Area in regular RAM can be reserved if needed by dimensioning a character string and locating the starting address by "ZAD=ADR(char string variable)." Numeric arrays can be located by dimensioning a one-character string just before the array is dimensioned and using "ZAD=ADR(char string variable)+1" to get the starting address of the array.

Shown in Listing 2 is a "page flipping" type of example where 22 different display banks of 960 bytes each (the GRAPHICS 0 screen size) are created, stored into hidden memory, and then switched one at a time into the screen-memory area in regular RAM. Lines 170 through 220 illustrate what is done by the user for proper setup of the bank switching:

1. Call the initialization subroutine by GOSUB 20000. This dimensions variables and loads in the machine-language routine.

2. Set ZBZ, the bank size in bytes. In this example, the bank size is 960 bytes for the GRAPHICS 0 screen.

3. Call the bank address-setup subroutine by GOSUB 21000. This calculates and assigns bank addresses to string variable ZB$ on a high byte/low byte basis. The addresses are stored this way, two string bytes per bank address, to conserve memory compared with using a numeric array which would have used six bytes per address.

4. Set ZAD, the starting address of the target area in regular RAM. In this example, the address is the start of screen memory as determined by the fourth (low byte of address) and the fifth (high byte of the address) bytes of the display list.

5. The bank-switching is now ready to use.

The bank switching routines are very flexible in that the target address, ZAD, can be changed at any time by the user if he desires to switch the hidden memory bank into another area of regular RAM. Also, the bank address-setup subroutine may be recalled during a program with a different bank size if the user desires a different hidden memory configuration.

The machine-language routine consists of a block move of the memory banks into and out of regular RAM. The BASIC and OS ROMs are turned off at the start of the routine so that the hidden memory can be accessed and then back on at the end of the routine. The machine-language routine is fully relocatable except that it uses Locations 214 through 218 ($D6-$DA) in page 0 for intermediate data storage. To use the bank switching, the user does the following. To switch a section of regular RAM out to a bank of hidden memory, i.e., store it:

1. Set the bank pointer, ZPTR, to the desired bank (range 1 to ZBTOT).

2. Call the bank-switch-out subroutine, GOSUB ZBNKOUT.

To switch a hidden memory bank back into regular RAM:

1. Set the bank pointer, ZPTR, to the desired bank.

2. Call the bank-switch-in subroutines, GOSUB ZBNKIN.

Pretty simple! All the hard work is done by the subroutines and the machine-language routine and occurs instantaneously.

The bank-switch subroutines, ZBNKIN (Lines 22000-22070) and ZBNKOUT (Lines 23000-23070), are identical except for the machine-language calls. First the display is turned off temporarily to speed up the move operation and prevent flickering of the display while the OS ROM is off. The system interrupts are then turned off to prevent accidental crashing. The machine-language call is of the form USR(ADR(MV$),FROM,TO, LENGTH) where FROM and TO are addresses of origin and destination and LENGTH is the number of bytes to move. In the case of ZBNKIN the call is ZZ=USR(ADR(MV$),ZB,ZAD,ZBSZ). The call for ZBNKOUT subroutine is ZZ=USR(ADR(MV$), ZAD,ZB,BSZ). After this the interrupts are enabled, the display is turned back on, and return made back to the main program.

Summary

This article has shown how to access the 22K of Atari XL hidden memory two different ways. Single-byte access or bank switching can be accomplished very easily and fast through the BASIC subroutines and machine-language routines. The subroutines are general and flexible for use in any user routines. The subroutines of both techniques can be combined by using LIST and ENTER. An example of using both would be a program which has four 4K (GRAPHICS 7) screen displays switched in and out by ZBNKIN and

ZBNKOUT, and the rest (6K) of the hidden memory for general data storage accessed by ZPEEK and ZPOKE.

A further interesting advantage of now being able to access the masked RAM is that this memory is not accessible to normal BASIC or the OS and therefore is fully protected, i.e., it cannot be accidentally written into.

FIGURE 1: MEMORY MAP OF THE ATARI XL COMPUTERS

DEC	HEX	FUNCTION
00000-40959	0000-9FFF	Free RAM for program, data, and screen display (pages 0-6 also used by OS) —40K
40960-49151	A000-B999	BASIC ROM (or free RAM if BASIC is deselected)—8K
49152-53247	C000-C999	OS ROM (or free RAM if OS is deselected) — 4K
53248-55295	D000-D799	Input/Output ports (not available as free RAM) — 2K
55296-65535	D800-FFFF	OS ROM (or free RAM if OS deselected)— 10K

LISTING 1: BASIC

```
AE 10 REM ********************************
JN 20 REM *   SINGLE BYTE ACCESS TO   *
HA 30 REM *   ATARI XL HIDDEN MEMORY  *
WP 40 REM *      by Kevin T. Pate     *
FW 50 REM *                           *
BS 60 REM *       COPYRIGHT 1989      *
BB 70 REM *     BY ANALOG COMPUTING   *
AL 80 REM ********************************
BG 90 REM
IV 120 REM TARGET ADDRESS IS ZAD
DC 130 REM DATA VARIABLE IS Z
GP 140 GOSUB 30000:REM RAM ACCESS INITIAL
   IZATION SUBROUTINE
HO 145 PRINT "{CLEAR}SINGLE BYTE ACCESS TO ATAR
   I XL HIDDEN MEMORY DEMONSTRATION"
XH 146 PRINT :PRINT "HIDDEN MEMORY ADDRES
   SES ARE           40960-53247 AND 5529
   6-65535"
JI 150 PRINT :PRINT "TARGET ADDRESS";:INP
   UT ZAD
KR 160 PRINT "VALUE";:INPUT Z
LL 170 GOSUB ZPOKE:PRINT "NOW POKING VALU
   E INTO HIDDEN MEMORY"
SY 180 Z=0:PRINT "NOW ZEROING OUT VALUE"
XZ 190 GOSUB ZPEEK:PRINT "NOW PEEKING HID
   DEN MEMORY ADDRESS"
YA 200 PRINT "ADDRESS=";ZAD,"VALUE=";Z
NM 210 GOTO 150
QR 220 REM
EE 30000 REM RAM ACCESS INITIALIZATION SU
   BROUTINE
LL 30010 REM ML SUBROUTINES ARE RELOCATAB
   LE
MY 30020 DIM Z$(50):REM STORAGE AREA FOR
   ML ROUTINES
DR 30030 REM POKE ML ROUTINE RETURNS VALU
   E TO Z$(1)
YF 30040 POK=ADR(Z$)+1:REM POKE ML ROUTIN
   E STARTING ADDRESS
LG 30050 ZPOKE=31000:REM POKE BASIC SUBRO
   UTINE
JE 30060 PEK=POK+17:REM PEEK ML ROUTINE S
   TARTING ADDRESS
ZW 30070 ZPEEK=32000:REM PEEK BASIC SUBRO
   UTINE
NA 30080 REM ML LOADER
NM 30090 RESTORE 30190
BY 30100 FOR Z=0 TO 34
VK 30110 READ ZZ:POKE POK+Z,ZZ
NS 30120 NEXT Z
RU 30130 ZHA=INT(ADR(Z$)/256):REM HIGH BY
   TE OF Z$(1) ADDRESS
MQ 30140 ZLA=ADR(Z$)-256*ZHA:REM LOW BYTE
    OF Z$(1) ADDRESS
HI 30150 POKE PEK+10,ZLA
FB 30160 POKE PEK+11,ZHA
EE 30170 RETURN
AA 30180 REM POKE ML DATA
AN 30190 DATA 104,162,2,142,1,211
EE 30200 DATA 162,00,142,000,0000,162
GJ 30210 DATA 253,142,1,211,96
UV 30220 REM PEEK ML DATA
ZS 30230 DATA 104,162,2,142,1,211
WR 30240 DATA 174,000,0000,142,000,0000
WH 30250 DATA 162,253,142,1,211,96
CI 30260 REM
MR 31000 REM POKE SUBROUTINE
SX 31010 ZHA=INT(ZAD/256):REM HIGH BYTE O
   F TARGET ADDRESS
BJ 31020 ZLA=ZAD-256*ZHA:REM LOW BYTE OF
   TARGET ADDRESS
YE 31030 POKE POK+7,Z
BC 31040 POKE POK+9,ZLA
JF 31050 POKE POK+10,ZHA
YA 31060 ZINT=PEEK(16):POKE 16,0:POKE 537
   74,0:POKE 54286,0:REM DISABLE INTERRUP
   TS
UH 31070 ZZ=USR(POK):REM CALL POKE ML ROU
   TINE
SD 31080 POKE 16,ZINT:POKE 53774,ZINT:POK
   E 54286,255:REM ENABLE INTERRUPTS
EL 31090 RETURN
BJ 31100 REM
IJ 32000 REM PEEK SUBROUTINE
SZ 32010 ZHA=INT(ZAD/256):REM HIGH BYTE O
   F TARGET ADDRESS
BL 32020 ZLA=ZAD-256*ZHA:REM LOW BYTE OF
   TARGET ADDRESS
UU 32030 POKE PEK+7,ZLA
SQ 32040 POKE PEK+8,ZHA
XY 32050 ZINT=PEEK(16):POKE 16,0:POKE 537
   74,0:POKE 54286,0:REM DISABLE INTERRUP
   TS
DN 32060 ZZ=USR(PEK):REM CALL PEEK ML ROU
   TINE
BP 32070 Z=PEEK(ADR(Z$))
SF 32080 POKE 16,ZINT:POKE 53774,ZINT:POK
   E 54286,255:REM ENABLE INTERRUPTS
EN 32090 RETURN
```

LISTING 2: BASIC

```
AE 10 REM ********************************
AH 20 REM *     BANK SWITCHING OF        *
HA 30 REM *   ATARI XL HIDDEN MEMORY     *
WP 40 REM *      by Kevin T. Pate        *
FW 50 REM *                              *
BS 60 REM *       COPYRIGHT 1989         *
BB 70 REM *     BY ANALOG COMPUTING      *
AL 80 REM ********************************
BG 90 REM
PB 120 REM USER SELECTED BANK SIZE IS VAR
   IABLE ZB5Z
YL 130 REM BANK SIZE IS SELECTABLE FROM 2
   25 X 100 BYTE BANKS MAX., 2 X 10K (102
   40) BYTE BANKS MIN.
QN 140 REM BANK RAM STARTING ADDRESSES (I
   N HIDDEN MEMORY) ARE IN ZB$
RA 150 REM BANK POINTER NUMBER (1 TO ZBTO
   T); ZBTOT IS TOTAL NUMBER OF BANKS
NU 160 REM USER SELECTED TARGET RAM START
   ING ADDRESS (IN REGULAR RAM AREA) IS V
   ARIABLE ZAD
GK 170 GOSUB 20000:REM RAM ACCESS INITIAL
   IZATION SUBROUTINE
VT 180 REM THIS EXAMPLE WILL SWITCH INTO
   SCREEN MEMORY 22 DIFFERENT DISPLAY BAN
   KS
NY 190 ZB5Z=960:REM BANK SIZE = 960 BYTES
    FOR GR.0 SCREEN
TS 200 DLIST=PEEK(560)+256*PEEK(561):REM
   DISPLAY LIST ADDRESS
BR 210 ZAD=PEEK(DLIST+4)+256*PEEK(DLIST+5
   ):REM SCREEN MEMORY STARTING ADDRESS
LG 220 GOSUB 21000:REM SETUP BANK ADDRESS
   ES BASED ON BANK SIZE
CH 230 REM CREATE DISPLAYS AND STORE BANK
   S IN HIDDEN MEMORY
XV 240 POKE 752,1:REM TURN OFF CURSOR
MP 250 PRINT "{CLEAR}{CREATE SCREEN DISPLAYS AND
               STORE IN HIDDEN MEMORY BAN
   KS...}":FOR N=1 TO 1000:NEXT N
AL 260 FOR I=1 TO ZBTOT
KY 270 PRINT "{CLEAR}":POSITION 6,I:PRINT "THIS
    IS DISPLAY BANK #";I:REM CLEAR SCREEN
    AND PRINT MESSAGE AT VARYING ROW NUMB
KH 280 ZPTR=I:GOSUB ZBNKOUT:REM SET BANK
   POINTER AND STORE SCREEN MEMORY TO BAN
   K
GK 290 NEXT I
VB 300 REM NOW SWITCH IN BANKS ONE BY ONE
   , PAUSING BETWEEN EACH DISPLAY
ZJ 310 PRINT "{CLEAR}{NOW RECALL SCREEN DISPLAYS
               FROM HIDDEN MEMORY BANKS.
   .}":FOR N=1 TO 1000:NEXT N
AE 320 FOR I=1 TO ZBTOT
FV 330 ZPTR=I:GOSUB ZBNKIN:REM SET BANK P
   OINTER AND READ IN BANK TO SCREEN MEMO
   RY
OE 340 FOR N=1 TO 100:NEXT N:REM PAUSE
GD 350 NEXT I
YW 360 GOTO 320:REM REPEAT DISPLAY LOOP
RC 370 REM
ED 20000 REM RAM ACCESS INITIALIZATION SU
   BROUTINE
GL 20010 REM ML SUBROUTINE IS RELOCATABLE
LZ 20020 DIM MOV$(50):REM STORAGE AREA FO
   R ML ROUTINE
JT 20030 DIM ZB$(450):REM HIDDEN MEMORY S
   TARTING ADDRESSES, STORED AS HIGH BYTE
   /LOW BYTE
TI 20040 ZBNKIN=22000:REM BANK SWITCH IN
   BASIC SUBROUTINE
CE 20050 ZBNKOUT=23000:REM BANK SWITCH OU
   T BASIC SUBROUTINE
MR 20060 REM ML LOADER
IK 20070 RESTORE 20130
NG 20080 FOR ZZ=0 TO 48
DS 20090 READ ZZZ:POKE ADR(MOV$)+ZZ,ZZZ
GJ 20100 NEXT ZZ
DF 20110 RETURN
FV 20120 REM ML DATA
ZO 20130 DATA 104,162,2,142,1,211
DT 20140 DATA 104,133,215,104,133,214,104
HR 20150 DATA 133,217,104,133,216,104,133
   ,218
LF 20160 DATA 104,170,160,0,177,214,145,2
   16
VV 20170 DATA 200,208,4,230,215,230,217,2
   02
GV 20180 DATA 208,242,198,218,16,238
WT 20190 DATA 162,253,142,1,211,96
BJ 20200 REM
OY 21000 REM SETUP BANK ADDRESSES BASED O
   N BANK SIZE
BU 21010 ZLTOT=INT(12288/ZB5Z):REM TOTAL
   NUMBER OF BANKS IN LOWER PART OF HIDDE
   N MEMORY
ZZ 21020 ZUTOT=INT(10240/ZB5Z):REM TOTAL
   NUMBER OF BANKS IN UPPER PART OF HIDDE
   N MEMORY
ZW 21030 ZBTOT=ZLTOT+ZUTOT:REM TOTAL NUMB
   ER OF BANKS
PN 21040 FOR ZZ=0 TO ZLTOT-1
TV 21050 ZA=40960+INT(ZZ*ZB5Z+0.5):REM BA
   NK STARTING ADDRESS
LR 21060 ZHA=INT(ZA/256):REM HIGH BYTE
NJ 21070 ZLA=ZA-256*ZHA:REM LOW BYTE
EA 21080 POKE ADR(ZB$)+2*ZZ,ZHA:REM STORE
    HIGH BYTE
BJ 21090 POKE ADR(ZB$)+2*ZZ+1,ZLA:REM STO
   RE LOW BYTE
GL 21100 NEXT ZZ
WC 21110 FOR ZZ=0 TO ZUTOT-1
XM 21120 ZA=55296+INT(ZZ*ZB5Z+0.5):REM BA
   NK STARTING ADDRESS
LI 21130 ZHA=INT(ZA/256):REM HIGH BYTE
NA 21140 ZLA=ZA-256*ZHA:REM LOW BYTE
UV 21150 POKE ADR(ZB$)+2*ZLTOT+2*ZZ,ZHA:R
   EM STORE HIGH BYTE
HO 21160 POKE ADR(ZB$)+2*ZLTOT+2*ZZ+1,ZLA
   :REM STORE LOW BYTE
HN 21170 NEXT ZZ
EJ 21180 RETURN
CS 21190 REM
YP 22000 REM BANK SWITCH IN SUBROUTINE
VJ 22010 POKE 559,0:REM TURN OFF DISPLAY
RE 22020 ZB=PEEK(ADR(ZB$)+2*ZPTR-2)*256+P
   EEK(ADR(ZB$)+2*ZPTR-1):REM BANK STARTI
   NG ADDRESS
XP 22030 ZINT=PEEK(16):POKE 16,0:POKE 537
   74,0:POKE 54286,0:REM DISABLE INTERRUP
   TS
RJ 22040 ZZ=USR(ADR(MOV$),ZB,ZAD,ZB5Z):RE
   M SWITCH BANK RAM INTO TARGET AREA IN
   REGULAR RAM
RS 22050 POKE 16,ZINT:POKE 53774,ZINT:POK
   E 54286,255:REM ENABLE INTERRUPTS
WP 22060 POKE 559,46:REM TURN ON DISPLAY
EE 22070 RETURN
CN 22080 REM
SP 23000 REM BANK SWITCH OUT SUBROUTINE
VL 23010 POKE 559,0:REM TURN OFF DISPLAY
RG 23020 ZB=PEEK(ADR(ZB$)+2*ZPTR-2)*256+P
   EEK(ADR(ZB$)+2*ZPTR-1):REM BANK STARTI
   NG ADDRESS
XR 23030 ZINT=PEEK(16):POKE 16,0:POKE 537
   74,0:POKE 54286,0:REM DISABLE INTERRUP
   TS
XL 23040 ZZ=USR(ADR(MOV$),ZAD,ZB,ZB5Z):RE
   M STORE TARGET RAM IN REGULAR AREA TO
   BANK RAM
RU 23050 POKE 16,ZINT:POKE 53774,ZINT:POK
   E 54286,255:REM ENABLE INTERRUPTS
WR 23060 POKE 559,46:REM TURN ON DISPLAY
EG 23070 RETURN
```

Desert Falcon

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
XL/XE cartridge $24.95

Reviewed by Matthew J.W. Ratcliff

Desert Falcon is another variation on the never-ending video game concept of "kill or be killed, in the quest for more points." Only the scenario has changed. Old-time arcaders will get a sudden case of deja vu when playing this game. Has *Desert Falcon* lived another life as *Zaxxon*? It seems that the characters are different, the scenery changed, but the game play is much the same.

Some really bad guys raided Pharaoh's tomb some thirty centuries ago. But Pharaoh was avenged when "desert beasts" dispatched the thieves, leaving no trace of them, not even their camels. However, their loot was scattered over the sands of Egypt. Is it up to the Desert Falcon, with the assistance of a joystick-wrenching professional, to save the jewels and return them to their shrine? No! Nothing so honorable as that. The Desert Falcon is out to steal the jewels once again.

At the opening screen, the user may select one of four difficulty levels. One- or two-player modes may be enabled. In the two-player scenario, each falcon alternates turns, the switch being based on the loss of a life. A cooperative two-player mode would have been more interesting.

The game-play screen is a two-thirds overhead view. The joystick-controlled falcon may hop or fly over the sands, or swim through the water obstacles, avoid pyramids and Washington Monument look-a-like obstacles, collect treasures and hieroglyphs, and zap all the creatures in sight. It seems a bit queer to find so much water in the midst of such a vast desert. The display scrolls smoothly as the bird makes progress along the desert. He may fly in one direction only and cannot backtrack to pick up missed treasures.

The Desert Falcon may hop or hover over treasures and hieroglyphs to pick them up. When three hieroglyphs are accumulated, a superpower is enabled with a rapid double-press of the fire button. Some will zap all enemies on the screen, while others will provide bonus points or even hinder the player. This is the only real depth to the game, beyond zapping the other guys. By painstakingly collecting the right combinations of hieroglyphs, useful powers may be amassed for game advancement.

The object of the game is to collect as many treasures as possible, while killing as many flying, crawling, and burrowing creatures as possible by shooting arrows (an unlimited supply) and avoiding obstacles. To move on to higher levels in the game, there is an encounter with a "howling sphinx" (sounds like something from an old Charlie Chan movie). The sphinx must be shot directly between the eyes, while avoiding the fireballs it blasts.

At higher levels in the game the flying and crawling creatures are augmented by "fire pots" and dart-shooting "mini sphinxes." Naturally, the opponents come in faster and in ever increasing numbers. The only other obvious difference at higher levels is the color change of the sand (green sand?). Immediately after a sphinx is snuffed out, there is a short period of time for the falcon to collect treasures, unmolested by all the annoying flying and crawling creatures. An extra life is earned every 10,000 points, with a maximum of six extras to be held at any one time.

The Atari XEGS has been on the market for over a year now, and *Desert Falcon* is the first joystick game offering from Atari that is not simply a repackaged old disk-based game or a port from another game system. This cut-and-dried theme contributed to the video-game market crash of the mid '80s. It is time that Atari came out with something fresh, new, original, or at least timely; such as a port of a coin-op arcade game with a copyright less than a couple of years old.

Desert Falcon has good graphics, cute sound effects, and a tired, boring theme. If you missed out on *Zaxxon* from the early days of Atari video gaming, and would like a replacement for it, *Desert Falcon* is a good substitute. If you are looking for something new and refreshing, it won't be found here.

Marble M

by Earl Hill

When I was a young boy, in 1935, my grandfather introduced me to a board game using marbles. This game resembled a combination of regular checkers and Chinese checkers. The wooden game board had holes drilled for 32 marbles that were placed in the form of a cross. The object was to jump all the marbles and finish with the last marble in the center of the board. As in checkers, a marble was removed at every jump.

Time marched on, and along with my acquisition of an Atari 8-bit computer came my desire to adapt my grandfather's game to the computer. Some time later I also found that a version of this game was mentioned in the Dell Publishing book *Hackers* by Steven Levy, discussing how Bill Gosper at MIT successfully designed a computer program to solve it.

The final step in this progression was my downloading the *Greasy Spoon I.Q. Examination* program by Robert Beatty from the ANALOG Computing Atari Users' Group on DELPHI. *I.Q. Exam* is similar to the original marble game, but has 44 marbles. In one form or another, this game has also been called Peg Board, Peg Checkers or Hi-Q. You might recognize it by one of those names. *I.Q. Exam* contains the logic which I adapted for *Marble Magic*.

The object of *Marble Magic* is to jump over and remove marbles successively, until the final marble (hopefully) lands in the center of the board. In this version, the computer acts as the playing board, as well as checking all the moves for validity and keeping score of the number of moves and the number of marbles remaining.

Type in Listing 1, check it with BASIC Editor II on page 54 and save it to disk or cassette. When you run *Marble Magic*, you'll first see a title screen, followed by the board display. The screen displays 32 marbles, with a flashing asterisk indicating the blank space in the center of the board. The number of tries and the number of marbles are displayed near the bottom of the screen. The text bar at the bottom is reserved for instructions, comments, error messages, etc. Although no time limit is used in the game, a simple timer shows you how long you have been playing. Every count is 15 seconds.

How to Play

The marbles are controlled with the

Magic

joystick, but only horizontal and vertical movement is allowed. You must jump into an unoccupied space, but can do so from any direction. Jumping a marble removes it from the board. Use the joystick and the fire button to select and move a marble. After being selected, the marble will be replaced by a flashing asterisk. Move the indicator to a blank space, and press the fire button again to drop the marble.

As you move around the board, your location will be shown by the asterisk or a marble color change. During these moves, the program checks against errors and makes certain that any jump contemplated will be into a blank space. Limits have been set so that you cannot move outside the playing area. At the end of the game, you will be scored on whether you have one marble left in the center, one marble left somewhere else, or on the number of marbles (up to ten) left on the board. Play ends when no more jumps are possible. You can quit and start again by pressing the "Q" key. I have added a little surprise if you get the last marble in the center.

The game's strategy is a little complex, but a fundamental goal is to clear out the corners of the cross without leaving any marbles behind.

The game's strategy is a little complex, but a fundamental goal is to clear out the corners of the cross without leaving any marbles behind. Also, you should arrange your moves so that a jump puts you in a good position for another jump (this is generally, but not absolutely, true). It also helps to look ahead about four or five moves. To get you started, here is the first five moves of a winning pattern: 19.17, 6.18, 13.11, 27.13, 10.12. The moves are based on numbering the holes horizontally, starting from the upper left. The starting blank space in the middle would then be number 17. If you have a low frustration coefficient, the complete winning sequence is given at the end of this article.

How It Works

Marble Magic uses a GRAPHICS 1 screen with a DLI message bar in GRAPHICS 0 for prompts. GRAPHICS 1, for the playfield, is particularly useful because it lets you use different colors for the marbles, the title and the playing board, while at the same time providing enough space to draw the marbles on the playfield plus room for everything else. Redefined characters are used for the marble and the playfield.

The program begins with the break-key-disable routine followed by initialization, and then the installation of the new character set and the display-list modification for the prompt line. This is followed by a brief

credits screen after which the playing board is set up. You can skip the credits screen by changing Line 110 to GOSUB 1210.

In GRAPHICS 1 a PRINT to the screen via Channel 6 places a character on the screen. The redefined "@" character is used for the marble. The choice of a regular or inverse "@" is critical for the decision as to whether the marble has been selected for a move, whether it is present at a board location, etc. This information is given in greater detail in the article "Hidden Graphics" by Gregory Kopp in *Compute!'s Second Book of Atari Graphics*.

Following the drawing of the board, a bucket brigade of GOSUBs starting at Line 130 is used to control the joystick, select a marble, check moves, increment the score and check for the end of the game. Having these at the beginning gives you all the speed you need for a board game such as this.

The main game loop is between Lines 210 and 240. Here the joystick movement is checked. If the joystick is not being moved, the board is checked to ensure that a marble is selected. The locator asterisk (Line 250) is also flashed to indicate an empty space. The PLOT (rather than the PRINT) command for placing the marble is necessary to put the marble in the correct location, since the PRINT after the LOCATE statement repositions the cursor.

If the joystick is moved, the current board position is shown by the change in the marble color. In graphics mode 1, the LOCATE variable (Z in this case) returns an ATASCII character whose corresponding code number is an integer from 0 to 255. Adding 128 changes the marble color or flashes the asterisk.

Movement limits are established in Lines 320 through 340. Once a marble is selected, a move location is chosen followed by error checking beginning at Line 430. The error checking determines any illegal move attempts, such as moving into an occupied space or jumping more than one marble. After the computer checks for errors, the marble count and the number of jumps are totalled in Lines 600 through 640.

Everything is kept in order on the screen and in the message bar with the message fix and screen fix subroutines in Lines 1100 and 1110, respectively.

The game relies on the LOCATE statement to indicate the board position or absence of the marbles (the @ character) and to indicate a change in the marble position (OX=old X, NX=new X, etc.). At the conclusion of a jump, when a marble has been removed, the entire board is checked for its status (using X and Y position coordinates and LOCATE) beginning at Line 660. If further jumps are possible, control is returned to the joystick in the main loop. However, if no marbles remain next to each other, making further jumps impossible, the final score is displayed along with a comment on your game.

May you make all the right moves!

A winning series of moves:

19.17, 6.18, 13.11, 27.13, 10.12, 13.11, 8.10, 1.9, 3.1, 16.4, 1.9, 28.16, 16.4, 4.6, 21.23, 7.21, 24.22, 21.23, 26.24, 33.25, 31.33, 18.30, 33.25, 6.18, 18.30, 30.28, 28.16, 17.5, 15.17, 24.10, 5.17.

Earl Hill was an analytical chemist and statistician prior to retirement. A self-taught BASIC programmer, his interests include utilities, graphics and games. He lives in Erie, Pennsylvania.

LISTING 1: BASIC

```
AE 10 REM ********************************
ZI 20 REM *        MARBLE MAGIC          *
CF 30 REM *        by Earl Hill          *
FV 40 REM *                              *
IL 50 REM *      COPYRIGHT 1989          *
YP 60 REM *    BY ANALOG COMPUTING       *
AK 70 REM ********************************
BF 80 REM
IJ 100 POKE 566,PEEK(566)+12:GOSUB 1550:G
   OSUB 1660:REM BREAK & INITIALIZATION
IC 110 GOSUB 1140:REM CREDITS & GAME BOAR
   D
JN 120 GOSUB MFIX:POSITION (40-LEN(M1$))/
   N2,N0:? #N6;M1$
DK 130 GOSUB 210:REM JOYSTICK CONTROL
AH 140 GOSUB 380:REM SELECT MARBLE
NB 150 GOSUB MFIX:POSITION (40-LEN(M2$))/
   N2,N0:? #N6;M2$
DQ 160 GOSUB 210:REM JOYSTICK CONTROL
KM 170 GOSUB 430:REM CHECK MOVE
SA 180 GOSUB 610:REM INCREMENT COUNT
OG 190 GOTO 660:REM CHECK END OF GAME
IU 200 REM JOYSTICK CONTROL
WK 210 GOSUB SFIX:POKE 752,N1:LOCATE XX+O
   X,YY+OY,Z
IX 220 COLOR N10:PLOT XX+OX,YY+OY
NE 230 POSITION XX+OX,YY+OY:? #N6;CHR$(Z+
   128);
```

```
QK 240 ST=STICK(N0):IF ST=15 AND STRIG(N0
   )=N0 THEN RETURN
JQ 250 IF Z=32 THEN COLOR N10:PLOT XX+OX,
   YY+OY:COLOR 160:PLOT XX+OX,YY+OY
UU 260 KEY=PEEK(764):POKE 764,255:IF KEY=
   47 THEN RUN
CQ 270 TIME=INT((PEEK(18)*65536+PEEK(19)*
   256+PEEK(20))/900):POSITION N0,N3:? #N
   6;TIME
SM 280 IF ST=15 THEN 240
YM 290 POKE 77,N0
NJ 300 SOUND N0,20,N10,N10:FOR D=N1 TO 20
   :NEXT D:SOUND N0,N0,N0,N0
IC 310 POSITION XX+OX,YY+OY:? #N6;CHR$(Z)
   ;
NU 320 NX=OX+(ST=N7)*N2-(ST=N11)*N2
MM 330 NY=OY+(ST=13)*N2-(ST=14)*N2
EA 340 AX=ABS(NX):AY=ABS(NY):IF AX>N6 OR
   AY>N6 OR AX*AY>16 OR (AX=N4 AND AY=N4)
    THEN 360
FO 350 OX=NX:OY=NY
MP 360 GOTO 210
OP 370 REM SELECT MARBLE
VT 380 GOSUB SFIX:LOCATE XX+OX,YY+OY,Z:OZ
   =Z+128
FM 390 IF CHR$(Z)<>"@" THEN GOSUB MFIX:PO
   SITION (40-LEN(M3$))/N2,N0:? #N6;M3$;:
   GOSUB HFIX:GOTO BUZZ
WK 400 POSITION XX+OX,YY+OY:? #N6;" ";:PX
   =OX:PY=OY
DB 410 SOUND N0,20,N10,N10:FOR D=N1 TO 50
   :NEXT D:SOUND N0,N0,N0,N0:RETURN
II 420 REM CHECK MOVE
XG 430 GOSUB SFIX
QM 440 IF PX=OX AND PY=OY THEN GOSUB MFIX
   :POSITION (40-LEN(M4$))/N2,N0:? #N6;M4
   $;:GOSUB PFIX:GOTO BUZZ
SR 450 LOCATE XX+OX,YY+OY,Z:OZ=Z+128
GX 460 IF CHR$(Z)<>"■" THEN 570
WQ 470 IF PX<>OX THEN 520
JT 480 CY=(PY+OY)/N2:IF CY/N2<>INT(CY/N2)
    THEN 570
YR 490 AY=ABS(PY-OY):IF AY<>N4 THEN 570
HX 500 LOCATE XX+OX,YY+CY,Z:POSITION XX+O
   X,YY+CY:? #N6;" ";:IF CHR$(Z)<>"@" THE
   N 570
QK 510 GOTO 580
BE 520 IF PY<>OY THEN 570
GC 530 CX=(PX+OX)/N2:IF CX/N2<>INT(CX/N2)
    THEN 570
WC 540 AX=ABS(PX-OX):IF AX<>N4 THEN 570
OQ 550 LOCATE XX+CX,YY+PY,Z:POSITION XX+C
   X,YY+OY:? #N6;" ";:IF CHR$(Z)<>"@" THE
   N 570
QU 560 GOTO 580
VQ 570 GOSUB MFIX:POSITION (40-LEN(M5$))/
   N2,N0:? #N6;M5$;:GOSUB PFIX:GOSUB HFIX
   :GOTO BUZZ
KG 580 POSITION XX+OX,YY+OY:? #N6;"@";
DS 590 SOUND N0,20,N10,N10:FOR D=N1 TO 50
   :NEXT D:SOUND N0,N0,N0,N0:RETURN
EN 600 REM INCREMENT COUNT
OA 610 COUNT=COUNT+N1
CP 620 POSITION N5,18:? #N6;COUNT;
SU 630 POSITION 18,18:? #N6;32-COUNT;" "
ZK 640 RETURN
NM 650 REM CHECK END OF GAME
MC 660 GOSUB SFIX:EX=N10:EY=9:GOSUB 720
UQ 670 EX=N4:EY=9:GOSUB 720
MS 680 EX=N10:EY=N3:GOSUB 720
MO 690 EX=16:EY=9:GOSUB 720
WQ 700 EX=N10:EY=15:GOSUB 720
PB 710 GOTO 910
OL 720 FOR Y=-N2 TO N2 STEP N2
OE 730 FOR X=-N2 TO N2 STEP N2
BU 740 LOCATE EX+X,EY+Y,Z
WN 750 IF CHR$(Z)<>"@" THEN 880
RY 760 IF (EX+X<N6) OR ((EY<N7 OR EY>N11)
    AND EX+X<N12) THEN 790
AH 770 LOCATE EX+X-N2,EY+Y,Z:IF CHR$(Z)<>
   "@" THEN 790
XA 780 LOCATE EX+X-N4,EY+Y,Z:IF CHR$(Z)="
   " THEN 890
PZ 790 IF (EX+X>14) OR ((EY<N7 OR EY>N11)
    AND EX+X>N8) THEN 820
NF 800 LOCATE EX+X+N2,EY+Y,Z:IF CHR$(Z)<>
   "@" THEN 820
VH 810 LOCATE EX+X+N4,EY+Y,Z:IF CHR$(Z)="
   " THEN 890
MV 820 IF (EY+Y<N5) OR ((EX<N8 OR EX>N12)
    AND EY+Y<N11) THEN 850
PB 830 LOCATE EX+X,EY+Y-N2,Z:IF CHR$(Z)<>
   "@" THEN 850
RW 840 LOCATE EX+X,EY+Y-N4,Z:IF CHR$(Z)="
   " THEN 890
GQ 850 IF (EY+Y>15) OR ((EX<N8 OR EX>N12)
    AND EY+Y>N7) THEN 880
TI 860 LOCATE EX+X,EY+Y+N2,Z:IF CHR$(Z)<>
   "@" THEN 880
QM 870 LOCATE EX+X,EY+Y+N4,Z:IF CHR$(Z)="
   " THEN 890
QO 880 NEXT X:NEXT Y:RETURN
KT 890 POP :GOTO 120
LQ 900 REM END GAME
AJ 910 SC=32-COUNT
AY 920 GOSUB MFIX:POSITION N4,N0:? #N6;"Y
   ou removed all but ";SC;" marble(s)";
AL 930 FOR D=N1 TO 500:NEXT D
KX 940 GOSUB SFIX:IF SC=N1 THEN LOCATE XX
   ,YY,Z:IF CHR$(Z)="@" THEN D$="PERFECT"
   :GOTO 1070
QF 950 IF SC>N10 THEN SC=N10
YA 960 RESTORE 1050
EH 970 FOR N=N1 TO SC:READ D$:NEXT N
VJ 980 SOUND N0,N0,N0,N0:SETCOLOR N4,N0,N
   0:SETCOLOR N2,N8,N4
CN 990 GOSUB MFIX:POSITION N11,N0:? #N6;"
   Your game: ";D$;
GD 1000 FOR DE=1 TO 1000:NEXT DE
MK 1010 GOSUB MFIX:POSITION N11,N0:? #N6;
   "Another game? (Y/N)";:GOSUB 1020:IF A
   N5$="Y" THEN RUN
UO 1020 OPEN #N1,N4,N0,"K:":POKE 764,255:
   POKE 702,64:POKE 694,N0:GET #N1,A:CLOS
   E #N1:IF A<>78 AND A<>89 THEN 1020
CS 1030 AN5$=CHR$(A):IF AN5$="N" THEN POP
    :GRAPHICS 0:END
AL 1040 RETURN
QC 1050 DATA OUTSTANDING,Excellent,very g
   ood,good,average,fair,poor,very poor,s
   ad,horrible
PN 1060 REM SOUND & FIX 'M UP
QG 1070 SOUND N0,50,N8,15:FOR N=N1 TO 20:
   FOR L=N1 TO N10:SETCOLOR N2,N0,L:SETCO
   LOR N4,N0,L:NEXT L:NEXT N:GOTO 980
QM 1080 GOSUB SFIX:POSITION XX+PX,YY+PY:?
    #N6;"@";:RETURN
SW 1090 GOSUB SFIX:POSITION XX+OX,YY+OY:?
    #N6;CHR$(OZ);:RETURN
BK 1100 POKE 88,LO:POKE 89,HI:POKE 752,N1
   :POSITION N0,N0:? #N6;BL$;:RETURN
LX 1110 POKE 88,LO1:POKE 89,HI1:POKE 87,N
```

```
   1:RETURN
SO 1120 SOUND N0,50,N2,N10:FOR D=N1 TO 50
   :NEXT D:SOUND N0,N0,N0,N0:FOR D=N1 TO
   500:NEXT D:POP :GOTO 120
KL 1130 REM CREDITS
NH 1140 POKE 87,N1:POKE 710,112:POKE 712,
   112:POKE 709,200:POKE 752,N1
OT 1150 COLOR 64:PLOT N0,N0:DRAWTO 19,N0:
   DRAWTO 19,19:DRAWTO N0,19:DRAWTO N0,N0
FD 1160 POSITION N2,N4:? #N6;"  Marble Ma
   gic"
TH 1170 POSITION 9,N10:? #N6;"BY"
TB 1180 POSITION N4,15:? #N6;" earl hill"
KK 1190 FOR D=N1 TO 500:NEXT D
NH 1200 COLOR 125:PLOT N0,N0
BT 1210 POKE 756,START/256
HF 1220 REM DO GAME BOARD
LU 1230 POKE 559,N0
RL 1240 POKE 87,N1:POKE 752,N1:SETCOLOR N
   4,N0,N0:SETCOLOR N2,13,N4:SETCOLOR N1,
   13,N12:REM SETCOLOR N3,N4,N8
TG 1250 POSITION N1,N0:? #N6;"** Marble M
   agic **"
JD 1260 X=16:Y=N12:FOR DX=N3 TO X STEP N2
   :FOR DY=N8 TO Y STEP N2
RU 1270 POSITION DY,DX:? #N6;"@":NEXT DY:
   NEXT DX
DO 1280 X=N12:Y=16:FOR DX=N7 TO X STEP N2
   :FOR DY=N4 TO Y STEP N2
SA 1290 POSITION DY,DX:? #N6;"@":NEXT DY:
   NEXT DX
CX 1300 X=N10:Y=16:FOR DX=N8 TO X STEP N2
   :FOR DY=N4 TO Y STEP N2
EO 1310 POSITION DY,DX:? #N6;"■":NEXT DY:
   NEXT DX
JA 1320 X=14:Y=N12:FOR DX=N4 TO X STEP N2
   :FOR DY=N8 TO Y STEP N2
EU 1330 POSITION DY,DX:? #N6;"■":NEXT DY:
   NEXT DX
RA 1340 X=15:Y=N11:FOR DX=N3 TO X STEP N2
   :FOR DY=9 TO Y STEP N2
XM 1350 POSITION DY,DX:? #N6;"■":NEXT DY:
   NEXT DX
EH 1360 X=N11:Y=15:FOR DX=N7 TO X STEP N2
   :FOR DY=N5 TO Y STEP N2
XS 1370 POSITION DY,DX:? #N6;"■":NEXT DY:
   NEXT DX
CU 1380 POSITION N7,N6:? #N6;"+":POSITION
    13,N6:? #N6;",":POSITION N7,N12:? #N6
   ;"$":POSITION 13,N12:? #N6;"'"
ZA 1390 POSITION N7,N2:? #N6;"'":POSITION
    N3,N6:? #N6;"'":POSITION 13,N2:? #N6;
   "$":POSITION 17,N6:? #N6;"$"
BK 1400 POSITION N3,N12:? #N6;",":POSITIO
   N N7,16:? #N6;",":POSITION 13,16:? #N6
   ;"+":POSITION 17,N12:? #N6;"+"
MR 1410 POSITION N8,N2:? #N6;"-----":POSI
   TION N4,N6:? #N6;"---":POSITION 14,N6:
   ? #N6;"---"
HX 1420 POSITION N4,N12:? #N6;"---":POSIT
   ION 14,N12:? #N6;"---":POSITION N8,16:
   ? #N6;"-----"
DE 1430 FOR Y=N7 TO N11:POSITION N3,Y:? #
   N6;"&":NEXT Y:FOR Y=N3 TO N5:POSITION
   N7,Y:? #N6;"&":NEXT Y
RD 1440 FOR Y=13 TO 15:POSITION N7,Y:? #N
   6;"&":NEXT Y:FOR Y=N3 TO N5:POSITION 1
   3,Y:? #N6;"&":NEXT Y
XO 1450 FOR Y=N7 TO N11:POSITION 17,Y:? #
   N6;"&":NEXT Y:FOR Y=13 TO 15:POSITION
   13,Y:? #N6;"&":NEXT Y
KX 1460 POSITION N10,9:? #N6;" "
WQ 1470 POSITION N0,18:? #N6;"Krys"
DS 1480 POSITION N10,18:? #N6;"Marbles"
JP 1490 POSITION N0,N1:FOR X=N0 TO 19:? #
   N6;"#";:NEXT X
RV 1500 POSITION N0,20:FOR X=N0 TO 19:? #
   N6;"%";:NEXT X
QV 1510 POSITION N0,23:FOR X=N0 TO 19:? #
   N6;"#";:NEXT X
YU 1520 POKE 559,34
AS 1530 RETURN
BO 1540 REM INITIALIZATION
EU 1550 DIM D$(12),MVCHR$(28),BL$(40),M1$
   (23),M2$(21),M3$(22),M4$(18),M5$(15),A
   N5$(1)
XW 1560 XX=10:YY=9:PFIX=1080:HFIX=1090:BU
   ZZ=1120:MFIX=1100:SFIX=1110:N0=0:N1=1:
   N2=2:COUNT=N0
KR 1570 N3=3:N4=4:N5=5:N6=6:N7=7:N8=8:N10
   =10:N11=11:N12=12
OL 1580 BL$=" ":BL$(39)=" ":BL$(2)=BL$
NV 1590 M1$="Select marble to move -"
QC 1600 M2$="Select hole to fill -"
IC 1610 M3$="Must select a marble -"
BH 1620 M4$="Marble replaced..."
MZ 1630 M5$="Illegal move!!!"
AX 1640 RETURN
DQ 1650 REM CHARACTER SET MOVE
UP 1660 RESTORE 1670:FOR I=N1 TO 28:READ
   Y:MVCHR$(I)=CHR$(Y):NEXT I
ZP 1670 DATA 104,169,0,133,205,168,169,22
   4,133,206,177,205,145,203
ET 1680 DATA 200,208,249,230,204,230,206,
   165,206,201,228,208,239,96
OH 1690 POKE 559,N0:START=(PEEK(106)-8)*2
   56:POKE 756,START/256
TX 1700 SHIGH=START/256:SLOW=N0:POKE 203,
   SLOW:POKE 204,SHIGH
MV 1710 Z=USR(ADR(MVCHR$))
FJ 1720 REM REDEFINE CHARACTERS
PI 1730 RESTORE 1760
GE 1740 READ X:IF X<N0 THEN 1860
QK 1750 FOR Y=N0 TO N7:READ Z:POKE START+
   X*8+Y,Z:NEXT Y:GOTO 1740
XP 1760 DATA 3,255,0,0,0,0,0,0
JM 1770 DATA 4,0,0,0,248,8,8,8,8
HH 1780 DATA 5,0,0,0,0,0,0,0,255
QU 1790 DATA 6,8,8,8,8,8,8,8,8
BL 1800 DATA 7,0,0,0,15,8,8,8,8
TN 1810 DATA 11,8,8,8,248,0,0,0,0
KP 1820 DATA 12,8,8,8,15,0,0,0,0
EE 1830 DATA 13,0,0,0,255,0,0,0,0
LW 1840 DATA 32,0,60,94,94,78,98,60,0
FJ 1850 DATA -1
VT 1860 POKE 559,34:POKE 756,START/256
HO 1870 REM DISPLAY LIST & SCREEN
EH 1880 POKE 559,N0:GRAPHICS N0
WC 1890 DL=PEEK(560)+PEEK(561)*256+N4
NG 1900 POKE DL-N1,70
PO 1910 FOR X=DL+N2 TO DL+21:POKE X,N6:NE
   XT X
PQ 1920 POKE DL+23,N6:POKE DL+24,N6
AU 1930 POKE DL+25,65:POKE DL+26,PEEK(560
   ):POKE DL+27,PEEK(561)
HA 1940 POKE 756,START/256:POKE 559,34
AG 1950 SCRN=PEEK(88)+PEEK(89)*256
HL 1960 HI=INT((SCRN+21*20)/256):LO=(SCRN
   +21*20)-(HI*256)
SI 1970 HI1=INT((SCRN)/256):LO1=SCRN-(HI1
   *256)
TW 1980 POKE 20,N0:POKE 19,N0:POKE 18,N0
BS 1990 RETURN
```

continued from page 9

state-of-the-art graphics meant slowing program operation and/or cutting the database drastically.

Saying It With Pictures

However, thanks to the development of better programming techniques and more sophisticated program design (spurred on by the demands of the market), we eventually got crude graphics in our adventures—pictures to go with our words. (The "oldtimers" among you will remember the earliest Scott Adams S.A.G.A graphic adventures and the pioneering efforts of Sierra Online in this area.) The graphics weren't much to look at (some were nothing more than line drawings), but, for those who needed or wanted pictures, even line drawings were better than nothing.

Then came graphics-oriented computers, such as the Atari 8-bit machines and the Commodore 64. These accelerated the trend toward graphics in text adventures, and before long almost nobody but Infocom could turn a profit on the text-only adventure market.

It wasn't for lack of trying. Broderbund, for instance, published text-only games previously released by others, as well as original text adventures created by subsidiaries, as late as 1987. But such games died in the marketplace, while lavishly illustrated and animated adventures grabbed the lion's share of the market.

Eventually and inevitably, even Infocom added graphics to its adventures. Some text-adventure purists of my acquaintance were more than a little unhappy with this development, but the conventional computer-game marketing wisdom dictated that only games with graphics could find success in the marketplace.

The Sales "Picture"

But I digress. *Why* are graphics games almost all you see in the marketplace? The direct answer is that they sell better—which is exactly what I'm told by people like Roberta Williams at Sierra Online, Gary Carlston at Broderbund and David Albert at Electronic Arts. These people know what they're doing, to be sure, so we can believe them.

Now, let's move a step or three farther. *Why* do these games sell better? Getting back to the TV/book parallel, I think that the popularity of graphics adventures owes a lot to the same thing that made audio-visual entertainment more popular than books: ease of use.

I have of course ruffled some feathers by now. If you weren't upset by my implied (though not intended) condemnation of TV, many of you are probably sure that I'm calling those who prefer graphic adventures mindless illiterates.

I'm not saying that. But it is a fact that a game with pictures is far easier to play—for most people—than a game that forces you to create your own pictures. And, on an entirely subjective level, the reason such a game is easier to play is the fact that the vast majority of us just don't like to read. ("Us" in this sense is used in the editorial sense—I love to read.)

But this is not the only reason why graphic adventure games sell better than text-only adventures. Actually, easier access is more of a reason for why text-only adventure games not selling as well as graphics games. The reasons why graphic adventure games sell well have to do more with why they exist.

Which brings us to the core of the discussion.

"Because It Was There"

Graphics games originated, in my subjective opinion, not so much due to market demand as the fact that the capability was there. Given a machine with graphics capabilities like the Atari 800 (or, for that matter, the ST), software designers and programmers naturally took advantage of that capability, simply because it was there. There's nothing wrong with that.

"A Picture Can Kill a Thousand Words"

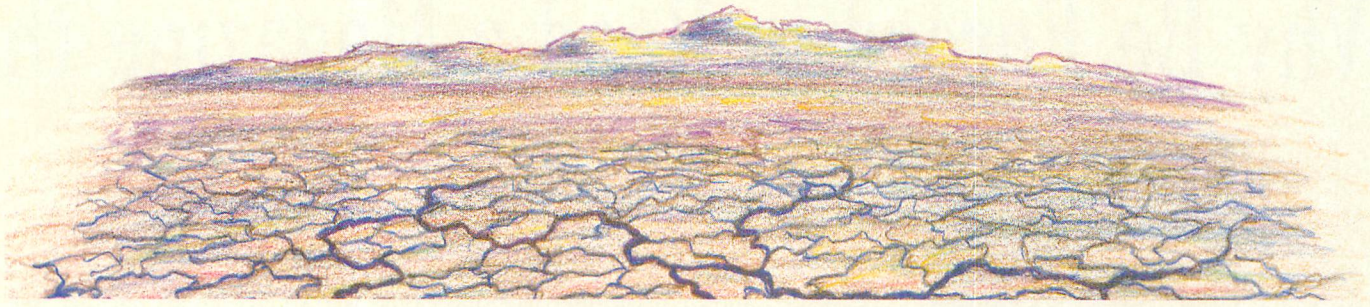
However, fancy graphics can be used to hide poor game design, providing an easy out for game creators. Dazzling graphics mean less pressure to create more challenging games and less need for "minor" details such as plot and character development. This results in games that appear to be brilliant, but which depend more on developed technical abilities than on creativity.

There are more than a few parallels between the movement of adventure gaming toward graphics and the displacement of books and radio by television and movies.

Without naming names and hurting the feelings of a number of game designers, publishers and players of my acquaintance, I'll just say that several of today's leading graphic adventure/role-playing games pale after a couple hours of play, thanks to their repetition and lack of challenge. (Or, if you prefer, lack of creativity and *work* in design.) All the fun is in learning to play the game and exploring the environment; after that, there's little more to do than work your way through the same kinds of challenges time and again, to earn the rewards of seeing more glitzy graphics and picking up points or "gold."

(For those who enjoy repetitiousness of such games, a far from flattering image comes to mind: Imagine a spoiled child demanding candy bars for breakfast.)

The bottom line and summary of my critique is this: The state of the art of game design has gone downhill. The *technology* has advanced, but it's being used as an easy way



out, rather than as a springboard to greater creativity. Thus, in levels of creativity and challenge, computer adventure game design and implementation has gone to seed.

Not *all* graphic adventure games suffer such decadence. There are still adventure gaming challenges out there. But the trend is set, and too many designers, programmers and publishers are following it.

Still, they can't bear all the responsibility; you may remember that one reason such games are published is because our neo-illiterate society encourages this kind of entertainment.

One Designer's Approach

Since I didn't name the games that are the result of the "easy way out," I won't name the games I pick as bearing both state-of-the-art technology, design *and* creativity; it wouldn't be fair. However, I will modestly cite my own text-graphics adventure game, *Gateway*, as an example of the latter. It's out of production (the publisher folded), so I don't have any reservations in using it as an example, especially since it received almost universally positive reviews.

And I'll note that I'm not merely praising my design, but also the sophisticated parser, database and other program elements, which offered far more than the usual four or six branches for a given input at any given point. (The possibilities varied and grew as the characters grew.) The game offered verisimilitude of the real world in terms of a plot and character development on the part of the player, as well as that of other characters in the game. If you acted like a jerk, someone remembered it and acted accordingly later. If you acted accordingly, certain things went your way that wouldn't have otherwise. (Of course, there were rewards for jerks and disappointments for nice guys, too, as in real life.)

The game had graphics in both the Macintosh and ST versions, but these did not carry the story. Instead, they supplemented the descriptions and offered an occasional break from the text. From my viewpoint, they were there for the same reasons that most novels published in the 19th century and the first half of the 20th century carried a score or so illustrations. From the publisher's viewpoint, they were there because graphics adventures were selling better than straight text adventures.

And *Gateway* included all the traditional elements of adventure games: a goal (which game play reveals and which you have the option to pursue or not); scoring; and tasks to perform and objects to find, preparatory to achieving certain goals associated with the quest.

But even with all this, the emphasis was still on creating a recognizable plot with malleable characters—and this includes the player as a character. And I'm not talking simple and unrealistic hit and experience points here, but true personality traits! I did not forego true storytelling in the adventure (which is what I fault most contemporary designers for doing).

I also emphasized a consistency of structure which dictated that nothing existed by gratuitous chance or for the purposes of making the game/story easy to write (a hallmark of good fiction in any media).

Where Do We Go From Here?

All of which describes exactly where graphic adventure game design could—and should—be today. And, considering the power of today's computers, graphic adventure games could be even better than *Gateway*. Unfortunately, graphics make it all too easy to toss plotting, character development, challenge, and *true* creativity out the window. And I don't think we'll see many more of the games that don't use graphics as a crutch until gamers are gorged on the graphics feast and begin demanding more by not automatically snapping up adventure games because they contain graphics. We'll see....

A Better Way

In the meantime, another interesting possibility looms large on the horizon: hypertext fiction. While hypertext is an overused buzz concept (and buzzword), being little more than a system of files linked in relational fashion, it presents enormous potential. Combined with graphics technology and the power and memory of newer computer systems, text that is organized in hypertext fashion and linked to graphics in the same manner can provide the gamer with the best of both worlds: pictures that both entertain the player and support fully developed stories.

In fact, a new company is engaged in creating just such adventures—and I'm certain that they're not the only one. So, more interesting things are on the way. Just wait and while you're waiting, read a few novels.

Michael A. Banks is the author of three science fiction novels, a children's book and sixteen non-fiction books. He's also written some 900 magazine articles, a few dozen short stories, radio scripts and a few catchy advertising slogans. His works include The Odysseus Solution *(a science fiction novel published by Baen Books),* The Modem Reference *(Brady Books/Simon & Schuster) and the Atari ST text/graphics adventure game* Gateway *(Pryority Software).*

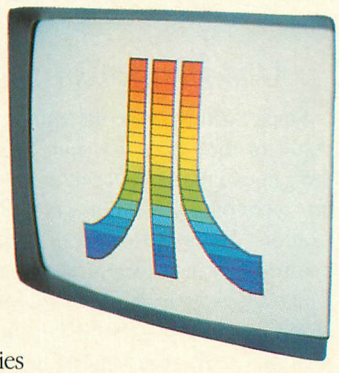
You Own an Atar

Why Be Forced to Read a Magazine that

YOUR ATARI RESOURCE CENTER

ANALOG Computing continues to offer exciting products for you and your Atari Computer. And we're the only magazine for the Atari 8-bit computer line that hasn't allowed its content to be virtually taken over by coverage of the Atari ST. We include only a minimal amount of ST material so that you can stay informed of what's happening with the 8-bit computer's brother.

Whether you own a reliable ol' 400 or 800, a shiny XL, new XE or even an XE Game Machine. . . we offer usable utilities, entertaining educational software, dynamite disk programs and great graphics and games. In fact, our readers still use ANALOG programs that were published over five years ago!



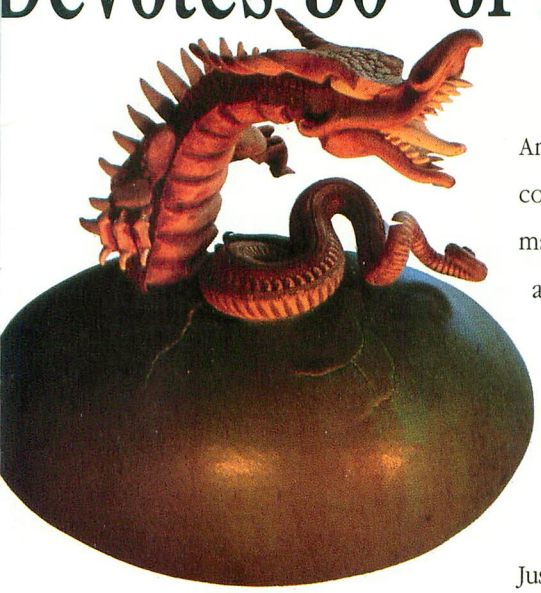
So when software companies turn their heads to other computers, you can turn yours to the one that supports your 8-bit Atari. And that's ANALOG Computing.

ANALOG's Best! Over 88 of ANALOG Computing's best and most requested programs are now available on this series of ten diskettes. The programs are all ready to run and come with complete documentation on the flip side of each floppy diskette. Select from Graphics, Educational, Utilities 1, Utilities 2, Disk Utilities and Games Disks 1, 2, 3, 4 and 5. Only $9.95 each (plus $1.50 shipping per order). Specify disk title when ordering.

Unlock the secrets of your Atari Computer! This handy 16-page pocket reference card covers information you need when programming your 8-bit. Error codes, internal codes, PEEK & POKE locations, machine-language aids, graphic mode specs and BASIC commands with abbreviations are only some of the helpful items at your fingertips. The ANALOG Computing Pocket Reference Card, only $7.95 each! (Plus $1.50 shipping and handling.)

i 6502 Computer.

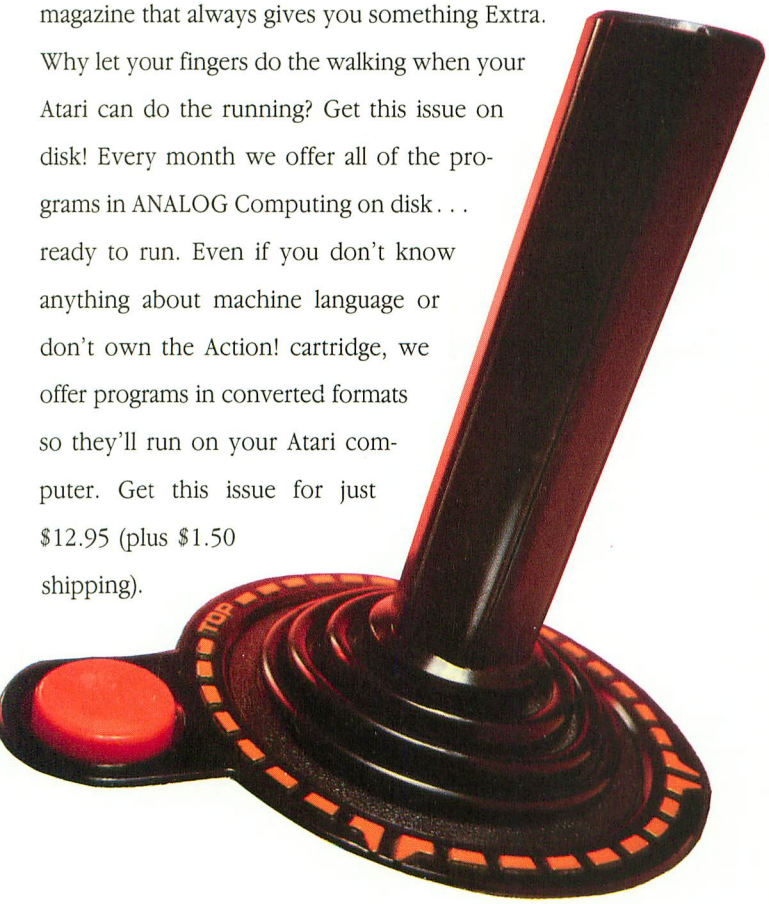
Devotes 50% of Its Pages to the Atari ST?



An Atari 8-bit Extra. While other "Atari" 8-bit magazines just make claims on how they cover your machine, we come through! Over 130 pages of new, never before published material. Programs like Easy Type, Dragon Chase, Pastels, Display List Mod, Tactics, Trivia and Create-a-base are all documented and ready to type in and run. . . all for just $8.95! (Add $1.50 for shipping.)

Get the Extra on disk! This special offer for Extra owners gets you all of the programs in an Atari 8-bit Extra on disk. Avoid typing errors, hours of tedious typing and frustration. Just plug in the disk and you are ready to roll! Two, ready-to-run double-sided floppies, $24.95. (Disks only. Atari 8-bit Extra sold separately. Please add $1.50 for shipping.) From the magazine that always gives you something Extra.

Why let your fingers do the walking when your Atari can do the running? Get this issue on disk! Every month we offer all of the programs in ANALOG Computing on disk. . . ready to run. Even if you don't know anything about machine language or don't own the Action! cartridge, we offer programs in converted formats so they'll run on your Atari computer. Get this issue for just $12.95 (plus $1.50 shipping).



ANALOG COMPUTING

ANALOG COMPUTING OFFICIAL ORDER FORM

Use this coupon to order the most complete up-to-date products specifically designed for your ATARI PC!

ANALOG'S BEST—Graphics Disk	$ 9.95	$ _____
ANALOG'S BEST—Educational Disk	$ 9.95	$ _____
ANALOG'S BEST—Utilities #1	$ 9.95	$ _____
ANALOG'S BEST—Utilities #2	$ 9.95	$ _____
ANALOG'S BEST—Disk Utilities	$ 9.95	$ _____
ANALOG'S BEST—Games #1	$ 9.95	$ _____
ANALOG'S BEST—Games #2	$ 9.95	$ _____
ANALOG'S BEST—Games #3	$ 9.95	$ _____
ANALOG'S BEST—Games #4	$ 9.95	$ _____
ANALOG'S BEST—Games #5	$ 9.95	$ _____
ANALOG COMPUTING—POCKET REFERENCE CARD	$ 7.95	$ _____
ANALOG COMPUTING—8-bit EXTRA	$ 8.95	$ _____
ANALOG COMPUTING—8-bit EXTRA (on disk)	$24.95	$ _____
ANALOG MAGAZINE ON DISK (please specify issue)	$12.95	$ _____
SHIPPING AND HANDLING—add $1.50 for each product ordered		$ _____
TOTAL ORDER		$ _____

☐ Payment Enclosed Charge My ☐ VISA ☐ Master Card

Card # _____ Exp. _____

Signature _____

Name _____

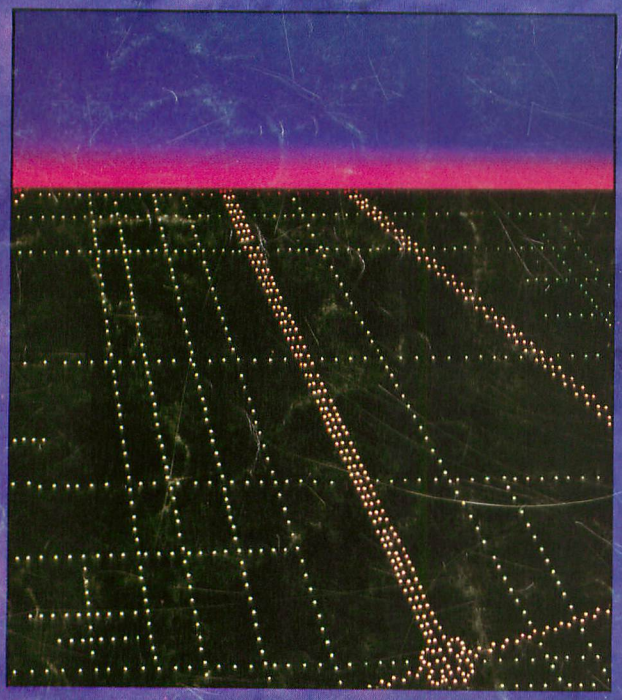
Address _____

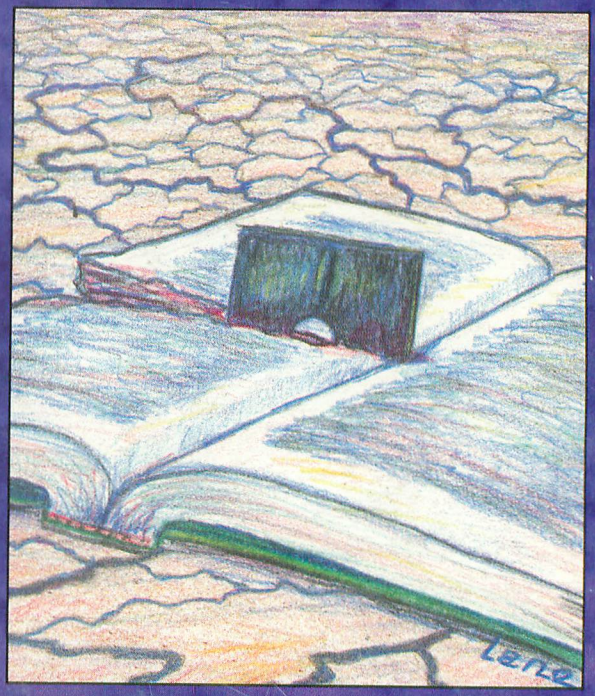
City _____ State _____ Zip _____

Make checks payable to: **LFP, Inc. P.O. Box 67068, Los Angeles, CA 90067.**
Your order will arrive in 4 to 6 weeks — WATCH FOR IT! ZIHYY

California residents add 6.5% sales tax on all orders except back issues.

INSIDE THIS ISSUE:





MORE

BOOT CAMP

PLUS

END USER

DATABASE DELPHI