NO. 29
APRIL 1985

U.S.A. $3.00
CANADA $3.50

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

A.N.A.L.O.G. COMPUTING

ATARI and CES: Part 2

Build your own voice synthesizer

Dragonlord

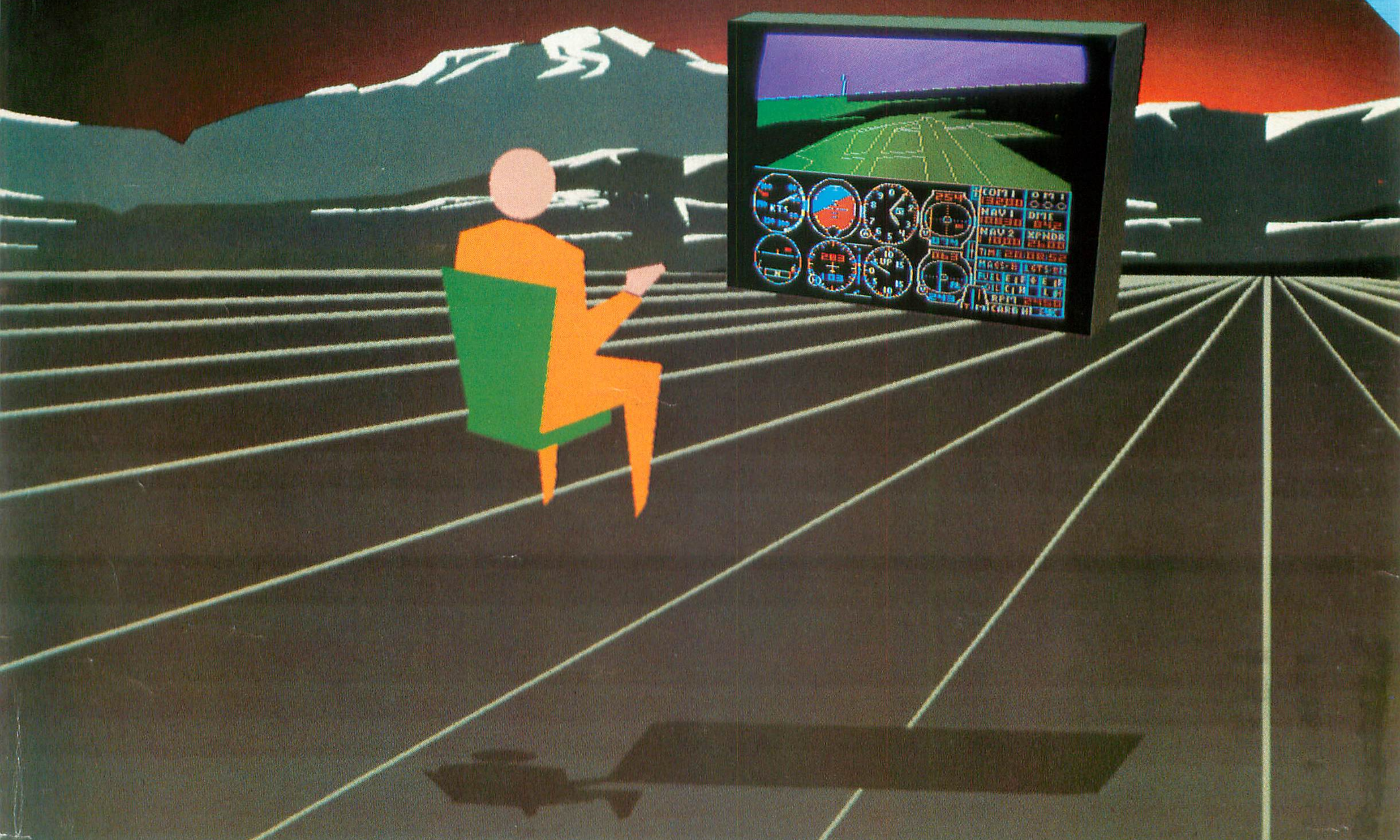
Revive

B Line

[redacted]

For Atari® Computers with 48K memory

Flight Simulator II



Put yourself in the pilot's seat of a Piper 181 Cherokee Archer for an awe-inspiring flight over realistic scenery from New York to Los Angeles. High speed color-filled 3D graphics will give you a beautiful panoramic view as you practice takeoffs, landings, and aerobatics. Complete documentation will get you airborne quickly even if you've never flown before. When you think you're ready, you can play the World War I Ace aerial battle game. Flight Simulator II features include ■ animated color 3D graphics ■ day, dusk, and night flying modes ■ over 80 airports in four scenery areas: New York, Chicago, Los Angeles, Seattle, with additional scenery areas available ■ user-variable weather, from clear blue skies to grey cloudy conditions ■ complete flight instrumentation ■ VOR, ILS, ADF, and DME radio equipped ■ navigation facilities and course plotting ■ World War I Ace aerial battle game ■ complete information manual and flight handbook.

See your dealer . . .

or write or call for more information. For direct orders enclose $49.95 plus $2.00 for shipping and specify UPS or first class mail delivery. American Express, Diner's Club, MasterCard, and Visa accepted.

Order Line: 800 / 637-4983

subLOGIC
Corporation
713 Edgebrook Drive
Champaign IL 61820
(217) 359-8482 Telex: 206995

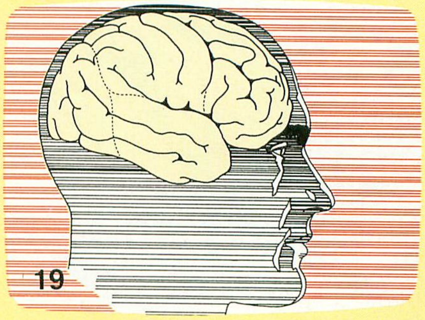
CIRCLE #101 ON READER SERVICE CARD

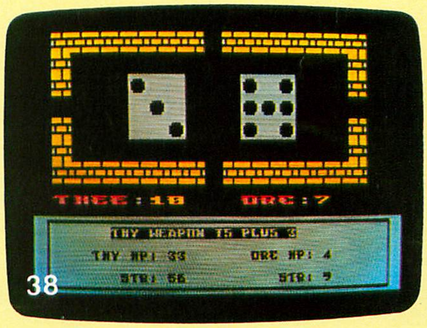
NO. 29 APRIL 1985

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

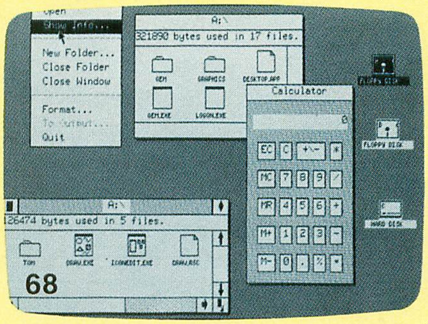
A.N.A.L.O.G. COMPUTING

FEATURES

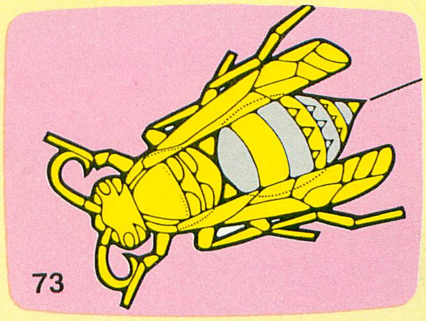




REVIEWS



COLUMNS



ANALOG COMPUTING (ISSN 0744-9917) is published monthly for $28 per year by ANALOG 400/800 Corp., 565 Main Street, Cherry Valley, MA 01611. Second class postage paid at Worcester, MA and additional mailing offices.

For subscription information and service call toll-free:

1-800-345-8112

in Pennsylvania call 1-800-662-2444

ADVERTISING SALES

Gerald F. Sweeney & Associates
P.O. Box 662
New York, NY 10113
(212) 242-3540

ANALOG Computing
Home Office
Michael DesChenes
National Advertising
(617) 892-9230

Address all advertising materials to:
Michael DesChenes — Advertising Production
ANALOG Computing
565 Main Street, Cherry Valley, MA 01611

ANALOG Computing (ISSN 0744-9917) is published monthly for $28 per year by ANALOG 400/800 Corp., 565 Main Street, Cherry Valley, MA 01611, Tel. (617) 892-3488. Second-class postage paid at Worcester, MA and additional mailing offices. POSTMASTER: Send address changes to **ANALOG Computing**, P.O. Box 615, Holmes, PA 19043. No portion of this magazine may be reproduced in any form without written permission of the publisher. Program listings should be provided in printed form. Articles should be furnished as typed copy in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG Computing**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply self-addressed, stamped envelope. U.S.A. newstand distribution by Eastern News Distributors, Inc., 111 Eighth Ave., New York, NY 10011.

WINTER CES: Part 2

by Arthur Leyenberger

At the Summer Consumer Electronics Show held in Chicago in June of 1984, Atari was still trying to turn things around. The theme of their press conference was printed everywhere, even on free T-shirts (one of which I still have), proclaiming "June 3, 1984—the Day the Future Began." Little did we (or they) know that the future was to begin almost a month later, when Jack Tramiel bought Atari from Warner Communications.

After the surprise announcement that Atari was now owned by Tramel Technologies, little or no information came out of the Sunnyvale headquarters of the new Atari Corp. Promises were continually made by Jack Tramiel and every other remaining Atari executive that "you'll see—at CES in January we'll be showing our new computers, and they will knock your socks off."

It came true. The big news at the 1985 Winter CES in Las Vegas was hardware. . . specifically, Atari hardware (see our preview of the new Atari computers in issue 28).

Sure, Coleco was there with a huge exhibit, right across from the Atari booth. Coleco announced just a few days before the show that they were getting out of the home computer business. Seems as if the Adam has been chased out of the home computer Garden of Eden. Anyway, I guess somebody forgot to tell the unfortunate Coleco employees that their company was no longer in the computer business. They spent a lot of their time standing around and talking amongst themselves.

Commodore was there in full strength. Well, almost. They were, after all, missing their past driving force, Jack Tramiel. Nonetheless, they were showing off the Commodore 128 computer, the replacement for the aging 64. They hope to repeat their past 64's success with the new machine.

However, the January *Wall Street Journal* reported Commodore International's earnings falling 94% in the industry's traditionally strong season, the last quarter of 1984. Far fewer Commodore 64s were sold this last Christmas than the previous year. The shocker for Commodore was the fact that as many Atari XLs were sold as 64s—an unprecedented occurrence.

Commodore has just chopped the price of the 64 from $200 to $149, in an attempt to clear out inventory before they begin pushing the 128. The Christmas losses could, in part, be due to the many defections from Commodore's upper management to the new Atari. Also, there has been too much of a lag between the 64 and the new 128 (which, compared to the Atari ST line, is too little, too late).

(continued on page 84)

READER COMMENT

Lost Lucasfilm games.

I am writing to you because I've got two questions to ask you.

Q1: What happened to Atari/ Lucasfilm's **Rescue on Fractalus**?

Q2: What happened to Atari/ Lucasfilm's other, second game for the Atari home computers, **Ballblazer**?

Lucasfilm said they would be out in August 1984. It's now January 1985.

Your friend and forever reader,
James Warren

*Both the Lucasfilm games, **Ballblazer** and **Rescue on Fractalus**, have been bought by Epyx Software, after the contract agreements with the old Atari ran out. They should be out soon. —Ed.*

Printer as display screen.

This letter is in response to the reader who was looking for a POKE that would allow him to use his printer as a display screen.

Try using:

```
POKE 838,166:POKE 839,238
```

This will send anything which normally prints on the screen to the printer. To return to normal printing, use:

```
POKE 838,163:POKE 839,246
```

Sincerely,
Scott Sheck
Gaithersburg, MD

Documentation, please!

I have just finished reading your interview with Mr. Tramiel. . .The success of Commodore with the Vic 20 and the 64 was price, but, in my opinion, the superior documentation contributed greatly to that success.

I own two Commodore 64s and one Vic. The User's Guide coupled with an optional ($14.95) Programmer Guide provide the user with all the data he would ever want to know about the machine, even a schematic for the serious "hacker."

Like many of your readers, when the price of the Atari 800XL went down, I ran out and added to my arsenal of home computers. When I found what documentation came with the system, I was appalled. Therefore, I got on the phone to Atari, only to find out I had to part with another $40.00 for additional documentation. It has been three months now, and still no documentation.

My point is this: Atari can build the best machine for the buck, but if they fail to support it in the form of good documentation, it's goodbye, Atari. The third-party people helped Commodore with their success story—via Commodore's "open book" policy. Why doesn't Atari do the same? Just ask the senior management at Texas Instrument or Coleco what their "closed book" policy got them.

Sincerely yours,
Joseph F. Stoneking
Colorado Springs, CO

Good documentation has always boosted the success of any computer product. We certainly hope that Atari Corp. bears that in mind. . . Remembering that this is a new company, we trust they'll get the bugs out of their information systems soon. —Ed.

Hexidecimals on the menu.

First, I must tell you that I look forward to the arrival of your magazine, and when it comes, all other work around the house ceases temporarily.

Second, I am a big fan of your hexidecimal programs, like: **Retrofire**, **Crash Dive!**, **Bacterion!** and **Fire Bug**. My question is this: is it possible to put more than one of these programs on one side of a disk, then have a routine that—when you boot up—gives you a menu to choose one to RUN?

I was also wondering whether or not a "fix" had been found for **Bacterion!** to prevent a system crash during the game on XL machines.

Lastly, I would like to congratulate Matthew J.W. Ratcliff on his program **Matt*Edit**. It is one of my all-time favorites, and besides being useful, it is fun to use. Keep up the great work!

Sincerely,
Bradly L. Pera
Canoga Park, CA

*So far, we haven't been able to locate the problem with **Bacterion!** but, hopefully, we'll have a fix soon.*

*As for our other machine language games, the program called **Binary File Menu/Loader**, printed in issue 17, will allow you to place several programs on one disk and run them from a menu. —Ed.*

New BBS.

I'm writing to invite your readers to call a new Atari-oriented BBS operating in Anniston, Alabama. For its experimental stage, A.F.I.X. BBS will be on-line from 6 p.m. until 6 a.m. each weekday, and all day Saturdays and Sundays. New files available for download, and plenty of message space is available. The number is (205) 820-2053.

I would also like to make a suggestion. Keeping up with current Atari-oriented BBSs is extremely difficult, because new boards are formed, old boards fall by the wayside, and numbers are changed.

Perhaps **ANALOG Computing** could devote a page each month with a list of Atari BBSs known to be active. When a new BBS comes on-line, or a number changes, the Sysop could notify the magazine.

If readers find a number that is no longer active—or incorrect—they could also notify **ANALOG Computing**. A setup similar to this would be a great service to Atari modem enthusiasts.

Thank you for your excellent magazine. I look forward to receiving it each month.

Sincerely,
C.O. Dickerson
Anniston, AL

We feel that a current BBS listing is important, too. Rather than print an updated list in each issue of ***ANALOG Computing****, we're planning to keep that information on our own bulletin board, starting in the near future.* —*Ed.*

Adam's Adventures for XL.

I've been a 400 owner for three years and an **ANALOG Computing** subscriber almost as long. Recently, I bought an 800XL. . .and found that the loader program included with cassette versions of Scott Adam's Adventures fails to work properly.

I soon discovered that the reason for this lies in the fact that routines called by the loader program have been moved lower in memory in the XL OS. Changing the loader program to the following will allow XL users to continue enjoying cassette versions of Scott Adam's Adventures:

```
10 DATA 104,169,0,133,9,32
,128,198,165,9,240,7,169,8
0,133,2,76,160,198,96
40 FOR I=0 TO 19:READ A:PO
KE 1536+I,A:NEXT I:I=USR(1
536)
```

Sincerely,
Ray Wilmott
Spotswood, NJ

Cassette Compressor problems.

Thank you for a great magazine, which, although I am just a beginner and mainly play games which I type into my Atari, I find. . .very interesting.

At the moment, we are having a bit of difficulty getting **ANALOG Computing** here, but our distributor is trying another source. In issue 24, on page 55, there is a program (**Cassette Compressor**) to reduce the loading time of boot games on cassette (I cannot afford a disk drive yet). I have typed the program in and **C:CHECK**ed it, but I still cannot condense any of the commercial games (i.e., **E.T.**). After going through the recommended steps, the computer just comes up with *BOOT ERROR*.

Could you help by explaining why it will not work, as it would save a lot of time when loading these very long games.

Yours faithfully,
H.C. Langston
Cheshire, England

P.S. I own a 400 (48K) full typewriter keyboard.

First, many commercial games use a "multi-stage" boot process, which ***Cassette Compressor*** *is unable to handle.* ***Cassette Compressor*** *is intended to be used with single-stage boot tapes, such as our machine language games and utilities.*

*Second, commercial programs are usually already made with short interrecord gaps, making **Cassette Compressor** unnecessary.* —*Ed.*

Problem solvers.

I have a problem with my **Bopotron!** program. The platforms don't show up in any color at all, and they only move once in one direction.

Second, there is a problem with my **Robot Raid** program. Whenever I pass the final mission and try to take the next screen, it gives me an *ERROR 141* at Line 230. What is wrong? Can it be fixed?

Third, my **Unicheck** program won't run in its AUTORUN.SYS file on disk. I have even tried putting the FMS.SYS file on the disk with it, but this does not work, either. Something is wrong, but I can't figure out what it is.

Sincerely,
Troy Goodson
Charleston, SC

***Bopotron!** and **Unicheck** both work fine as listed, and the problems you have experienced are most likely errors made typing them in.*

***Robot Raid** simply runs out of DATA for its screens after the third level. The following lines correct this problem, repeating the third level.*

```
860 LV=LV-N2:LEVEL=LEVEL+1
2:IF LEVEL>60 THEN LEVEL=L
EVEL-12:LV=LV+N2
865 RESTORE 19070+LEVEL:FO
R I=N1 TO 48:READ A:MAP(I)
=A:NEXT I
```

—*Ed.*

For Subscribers Only...

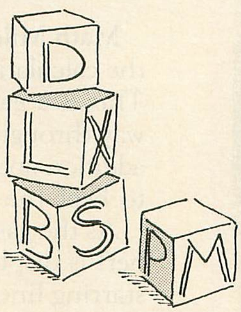
If you're a subscriber to ANALOG Computing, from time to time you'll notice that we will have a bonus bound into our pages. This is one of those times. You'll find the **ANALOG Computing Reference Card** bound in the center of this issue. To use the card, pull it out and fold it on the score lines. We think you'll find it quite a convenience.

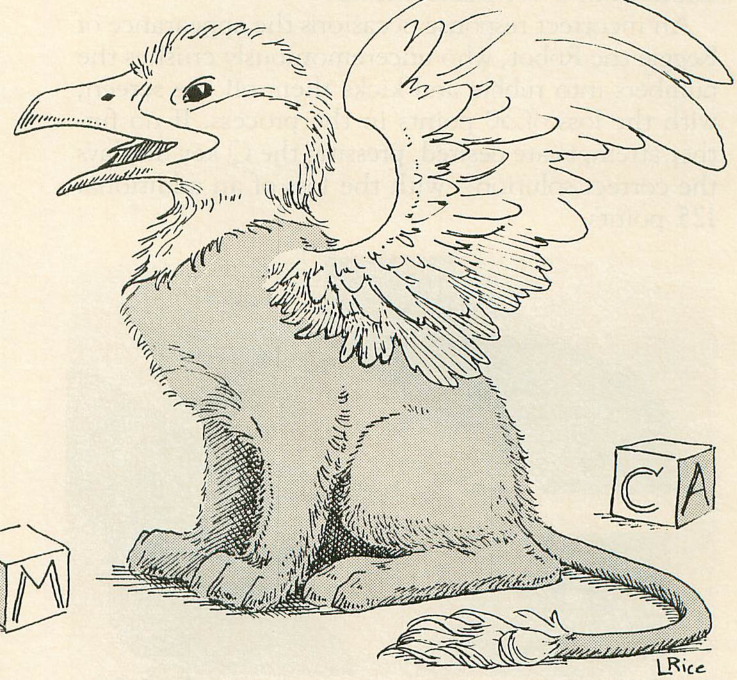
Nonsubscribers—want your own reference card? You can order copies directly from us for $7.95 each. Send your check to:

ANALOG Computing
P.O. Box 23, Worcester, MA 01610

Better still, subscribe *today*.

Griffin's Lair Educational Programs Review





by Braden E. Griffin, M.D.

How many ways are there to develop mathematical skills with educational software? Not many. Some of us fall into the same trap with educational software that we do with game programs. After the first few embellishments of an original concept, the "rip-offs" become so widespread and indistinguishable that they are often ignored.

The "classics" sell because of a unique combination of originality and presentation. Having played a **Lode Runner**, **Jumpman** or **Miner 2049er**, one seldom adds similar games to one's collection.

Occasionally, an unusually imaginative presentation of an erstwhile game model will be successful. This results from either a strikingly innovative approach to the design features or clever, inventive programming techniques which enhance the original. Electronic Arts' **One on One** is an excellent example of how masterful programming can successfully revive an old idea.

It's difficult to be original with the objectives of educational software, particularly in the area of mathematics. The subject matter is pretty standard, and the solutions are rarely creative. The appeal of most math programs rests on their presentation and the ability to initiate an interaction and sustain the interest of the user.

Sparkling graphics should be accompanied by a sound, objective approach to the learning process. Market-conscious software producers may place greater emphasis on the flash and less on the flesh, in order to attract the buyer. In Texas, they call this "all hat and no cattle. . .y'all." This month's hill of beans, if it amounts to that, offers some resourceful means to mastering mathematics.

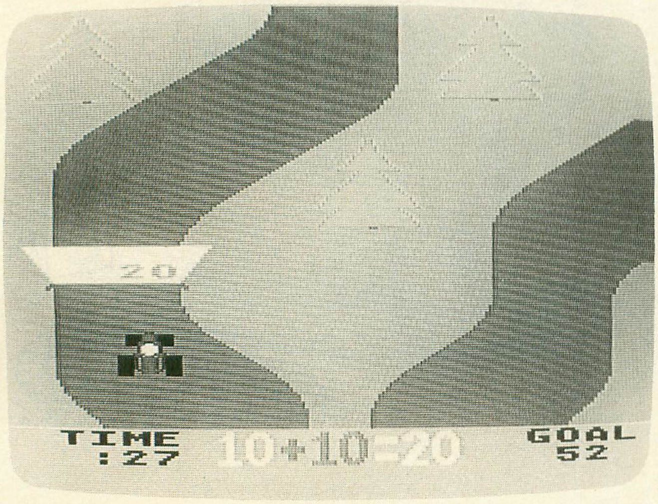
MATH MAGIC
BLAKMAGIC SOFTWARE
3720 Broadmoor
Beaumont, TX 77707
48K Disk (BASIC) $29.95

The development of math skills requires a solid foundation in arithmetic. This foundation results from the memorization of countless number facts. ("Countless numbers". . . is that an oxymoron?)

Sure, there is proof that 6×5=30, but this fact—and many others—must be rapidly accessed from memory for practical application. Flash card drills are frequently used to augment this process. Although basically a video flash card game, **Math Magic** injects a little incentive and fun into this usually tedious task.

From one to six players may participate in the competition. In addition to the four basic arithmetic operations, problems dealing with powers, roots, fractions and decimals are included. A problem is presented on the screen, and if the correct answer is entered, 100 points are scored, eliciting a cheer and a musical salute from a venerable wizard.

An incorrect response occasions the appearance of Reggie the Robot, who unceremoniously crushes the numbers into rubble and kicks them off the screen, with the loss of 50 points in the process. If no further attempts are desired, pressing the Q key displays the correct solution—with the loss of an additional 125 points.



Math Magic.

The feature which distinguishes **Math Magic** from other, similar offerings is the ability to handicap each individual. With a handicapping scale from 0 to 999, elementary age children can successfully compete with high school students and adults.

Also, one may select the largest number to be used in the problems. This number will apply to the individual with the lowest handicap, all others being adjusted accordingly. There is a limited range for the different operations: addition and subtraction - 1 to 999,999; multiplication and division - 1 to 9,999; fractions - 1 to 99; and powers and roots - 0 to 9 for the mantissa.

One has the option of using whole numbers or fractions (or decimals) when working with the four basic operations. A random option allows the computer to choose the type of problem.

The problems can be displayed in a standard or linear format. The answers are entered from the keyboard in a left-to-right fashion. No problem, if one is multiplying 6×3. However, if the problem is 3264×8964, the final answer of 29,258,496 must be entered from left to right, requiring more than just a little thought.

One does not perform the operation on the screen, with the initial four products in the example given being added together. This program is not designed to promote process skills. Instead, it encourages an individual to solve problems "in his or her head" (or "heads". . . sorry about that, Zaphod).

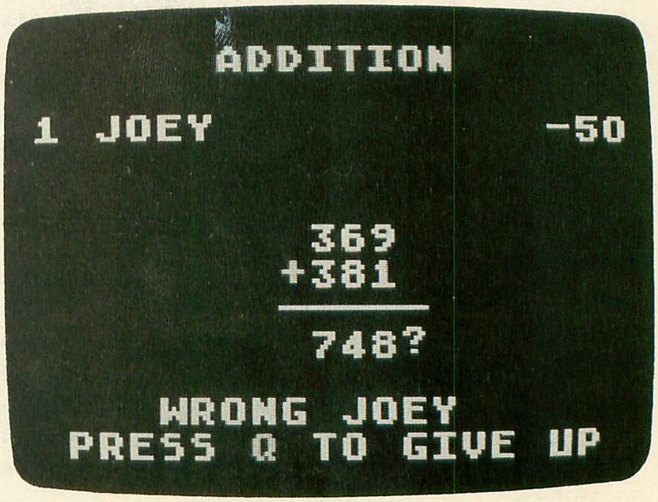
Enjoyable music and graphic enhancements add the right amount of frills to complete the package. The support of fractions, decimals, powers and roots, and the inclusion of the handicapping feature set this product apart from similar programs. **Math Magic** is a solid offering in the educational software market.

MATH MILEAGE
CBS SOFTWARE
A Unit of CBS Inc.
Greenwich, CT 06836
16K Cartridge Joystick $29.95

Math Mileage sounds like a program which teaches the calculation of fuel efficiency. Wrong, STP-breath. This is a game where up to four players wend their way through a race course, using basic math skills in addition and multiplication. Quick thinking is needed to beat the competition.

As the game begins, a randomly selected goal (number) is displayed while the players are sitting at the starting line, engines revving. Using a joystick, a formula race car (I get it!) is driven along a course composed of numerous forks. Over each branch of a fork, a mathematical operation is displayed.

The object is to accumulate points at each crossroad and, eventually, reach one's goal. Let's assume there is a goal of 168. Approaching the first fork, one sees the operation $\times 4$ over one branch and $+8$ over the other. (Once each course is begun, the two operations stay the same.)



Math Mileage.

Starting with 0 points, the first choice is to go to the $+8$ branch. Now the fun begins. One's goal will always be reached no matter which branch is taken, but the challenge is to do it by taking the fewest number of forks. It might seem the best course is to get as high a number as soon as possible, but that may be deceiving.

In our example, if one were to take the $\times 4$ branch second, the total would be 32. If the $\times 4$ route is again taken, a total of 128 is scored, and only by taking five consecutive $+8$ branches will the goal be reached.

However, if after totaling 32 points, the $+8$ branch is chosen (40 points), then the $\times 4$ (160) and finally, the $+8$ branch (168), three fewer forks would be taken. If it sounds complicated, the fault lies with my explanation, not the game. Under my picture in the high school yearbook, it says, "He has the ability to put the minimum amount of thought into the maximum amount of words." Enough said.

Three skill levels allow the user to increase the difficulty as desired. At level 1, players have the choice between only two operations, $+1$ and $+10$. Although little strategy is involved, this serves to emphasize the understanding of place value.

Levels 2 and 3 offer the choice between one addition and one multiplication operation. Level 2 always uses $+10$ for the addition branch, while it is randomly selected from between $+2$ and $+9$ in level 3. Both levels randomly select the multiplication operation from between $\times 2$ and $\times 5$.

Along the course flagmen appear, warning of "hazardous road conditions." The different flag colors alert one to whether either branch may be taken without overshooting the goal (green), one of the two branches will cause one to overshoot the goal (yellow), or if one has already overshot the goal (red).

As each branch of a fork is taken, the car breaks through a billboard showing the accumulated total score to that point. A night driving option is available, doubling the point value, in which the billboards don't show the total. In this case, the total must be kept in one's head. . .or kidney. . .or chewing gum wrapper.

At the end of the race, a scoring summary is displayed. This shows the number of forks taken and the time used for the race just completed, the fewest number of forks that could be taken to reach the goal, and previous race statistics for the course. Each race must be run twice before a new one is attempted. This reinforces the operations involved. A straightforward game play manual and handy reference card provide the necessary documentation.

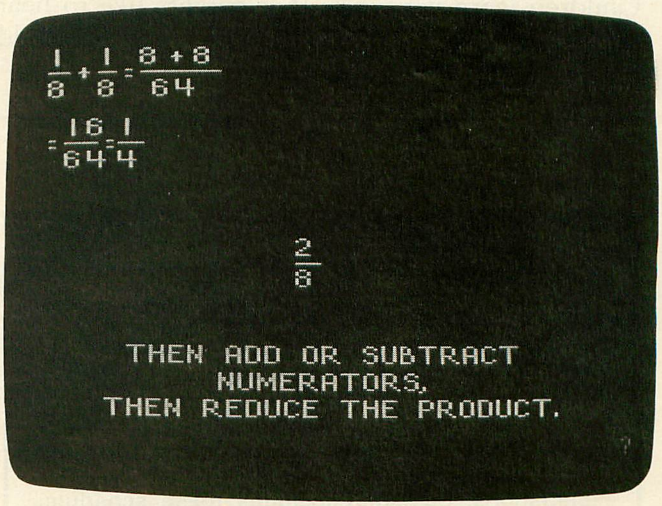
I have reviewed several products from the CBS Software line and found them to be of uniformly high quality. Polished graphics and well-founded educational objectives are partly responsible. Most importantly, their products are designed for children to play, and for children to learn. They're not designed to make *adults*, who spend the $$, think that they're educational.

Complicated, sophisticated, mind-expanding (and boggling) educational software may overwhelm young children. CBS Software knows kids. *You have to crawl before you walk.* In this case, **Math Mileage**'s race car provides first-class transportation.

PLAYFUL PROFESSOR
SCREENPLAY
Intelligent Statements, Inc.
Box 3558
Chapel Hill, NC 27514
48K Disk or Cassette Joystick optional

If limited to just one educational program in my collection, this would be it. I'm including all educational software, regardless of the subject matter.

Playful Professor is an exceptionally well-designed math tutor which will help develop proficiency in the four basic arithmetic operations. Unlike many math programs, it not only provides practice of math facts, it *teaches* them. (Read the previous sentence aloud, using your best John Houseman imitation.) And, unlike conventional tutorials, it's fun. All the necessary math skills required through the first six grades of school are encompassed in this program.



Playful Professor.

Playful Professor contains features which enable two players to compete, each with the ability to select his or her own operation and level of difficulty. For each arithmetic operation, there are four skill levels when dealing with whole numbers (integers) and three levels of difficulty using fractions.

After the selection process is complete, the first problem is displayed on the screen. When solving questions using integers, the problem is worked out on the screen in the same manner as one would with paper and pencil. (Everyone, together now. . ."I *brought* my pencil. . .Gimme something to write on, man.")

As soon as the computer accepts as many numbers as it "knows" it should get, it places a flashing *?* at the appropriate place for the next step. For example, if presented with a multiplication example, each product would be entered below the problem from right to left. The number in the "ones" column is entered first, then the "tens" column, etc. When all of the products have been entered, a line is drawn beneath them, and the products are entered in the usual fashion for the final answer. Problems involving fractions require only the final result, the solution being arrived at without the use of the display screen.

The tutorial portion of **Playful Professor** takes over only if one fails to correctly solve the problem. Ideally, if a student cannot solve a problem in the classroom, the teacher will work it through with a step-by-step explanation. In reality, many students are hesitant to acknowledge their lack of understanding in front of the entire class.

They just plod along, never fully comprehending the material, putting themselves further and further behind. As the years go by, they move on to more complex problems, and the ability to perform the basic math skills is taken for granted. Eventually, they become overwhelmed and, after a period of "hanging on," totally lose interest.

This is not an unusual scenario. More often than not, this pattern goes unrecognized until it's too late. Children learn very early that parents like to hear only good things—particularly, busy parents. If a child admits to having difficulty in school, parents are not only disappointed, but usually respond with a harangue about lack of study, too much TV or some other extraneous reason, not recognizing that the problem goes much deeper.

All the study time in the world won't help if the child doesn't have a grasp of the fundamentals. Concerned parents may be able to help their children by working with them at home. However, most parents aren't teachers, and the frustration of trying to teach what "comes naturally" ends up in a shouting match.

Outside tutors are fine but expensive. Of course, one solution is to assume the child is incapable of functioning above the level of the brain stem, and just give up. . . in which case, he will either become a urologist or a Republican member of the Senate. (The use of the masculine gender in the preceding sentence was intentional. Only rarely does the female of our species sink to such depths.) But I digress.

Using a computer-generated chalk and blackboard, the **Professor** illustrates the correct solution to the problems which have been missed. In the beginning levels, jelly beans are used to demonstrate the underlying principles. The essential rules governing each operation are also displayed. This promotes an understanding of the material, not just memorization.

A simple, common sense approach is used. It's also nonjudgmental. If the child makes the same mistake again, the computer doesn't say, "I just *showed* you how to do that. Weren't you listening?"

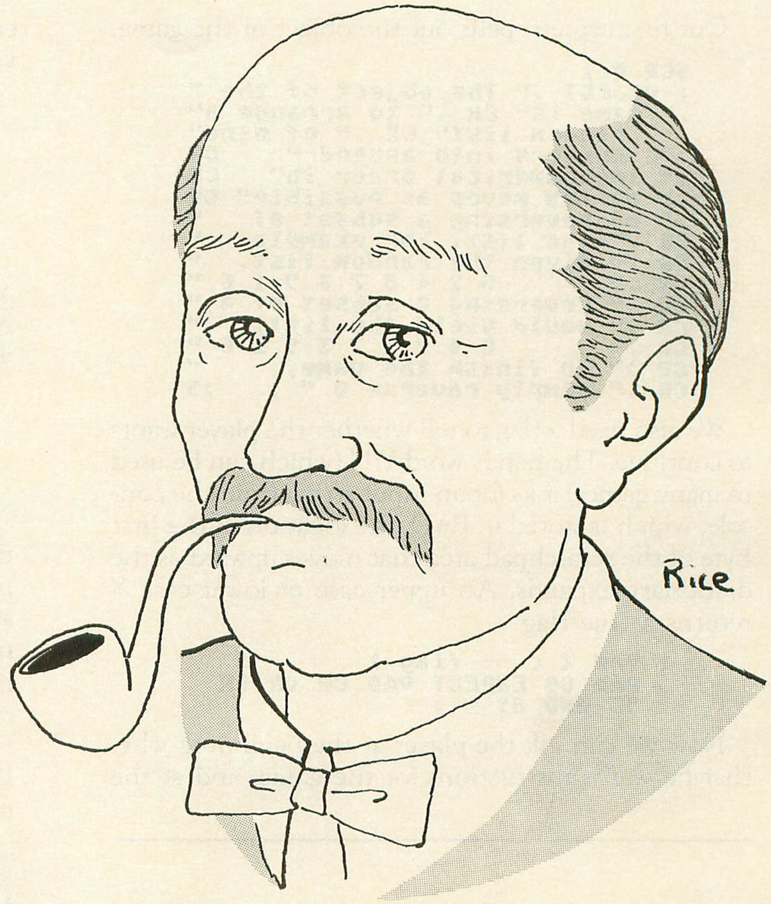
The section on fractions is particularly well done, and a little time spent here with the **Professor** will go a long way toward mastery of this frequently formidable foe. It's not surprising that many children have difficulty with fractions, especially after parents try to explain them. The only LCD dad knows about is in his watch. Mom thinks LCD is some kind of hallucinogen, but she never listens very carefully, anyway.

That's the beef. Where's the relish? Well, it's inside a haunted castle. The player is trapped inside this spooky citadel and must steal the key from a resident ghost to escape. Able to enter only lighted rooms, the ghost must be stalked from room to room.

With each move, a mysterious wind blows through the castle, turning various lights on and off. The player is awarded two moves for each correct answer, or a single move in the case of fraction problems answered correctly but not reduced to the lowest terms. Though simple, the game is enjoyable and exciting, especially when two players vie to exit the castle first.

An extremely thorough manual with an extensive and coherent mathematics review is included. Sure to maintain the interest of children, this program is a "keeper." Kids, don't be surprised if dad and mom get caught brushing up on *their* "Rithmetic," as well. This is the first in a series of educational games from Screenplay. If the others are as good as **Playful Professor**, keep 'em coming. □

Ask Mr. Forth



by Donald Forbes

A short list or array of numbers can make an interesting game, as well as an instructive FORTH demo.

Many of us met FORTH arrays for the first time on page 196 of Leo Brodie's first book, *Starting Forth*, in which he has a lab with five burners to heat various kinds of liquids. Here, "we can make our word *?TOO-HOT* check that all five burners have not exceeded their individual limit."

He defines the five limits with:

```
0 VARIABLE LIMITS 8 ALLOT
```

which sets up five slots for the array, and then stores a number in the first one with:

```
220 LIMITS !
```

and the second one with:

```
340 LIMITS 2 + !
```

and then defines a new word:

```
: LIMIT 2* LIMITS + ;
```

to take the burner number off the stack and store a limiting temperature for the third burner with:

```
170 2 LIMIT !
```

Brodie then defines a new word:

```
: ?TOO.HOT ( burner# temp -- )
  LIMIT @ > IF
  ." Danger - reduce heat " THEN ;
```

which works like this:

```
300 1 ?TOO.HOT ok
350 1 ?TOO.HOT
Danger - reduce heat   ok
```

Kevin McCabe, the Chicago lawyer who wrote *Forth Fundamentals* (the most complete explanation to date of fig-FORTH), also shows (page 119) how to set up an array with four locations with:

```
12 VARIABLE GROUP
24 , 48 , 96 ,
```

and an auxiliary word:

```
: GETGROUP 2* GROUP + @ ;
```

that will retrieve any member of the array with:

```
0 GETGROUP .   12  ok
3 GETGROUP .   96  ok
```

In both cases, the count starts at zero and there is no checking for errors.

"As an exercise in array manipulation," M.P. Burton composed a public domain FORTH version of the number game **Reverse**, which was published in *Forth Dimensions* magazine for January 1982. The object of the game is to arrange a list of numbers (1 through 9) in ascending order from left to right. Moves are made by reversing a subset of the list (from the left). The original game was written in BASIC almost ten years ago by Peter Sessions of People's Computer Company, a nonprofit educational corporation in San Ramon, California.

Our first screen spells out the object of the game.

```
SCR # 1
: OBJECT ." The object of the "
." game is" CR ." to arrange a"
."  random list" CR ." of nine"
."  numbers into ascend-"     CR
." ing numerical order in"    CR
." as few moves as possible" CR
." by reversing a subset of    "
CR ." the list. For example,   "
CR ." given the random list,   "
CR ."       5 2 4 8 7 3 9 1 6 "
CR ." reversing a subset of 4 "
CR ." would yield the list,    "
CR ."       8 4 2 5 7 3 9 1 6 "
CR ." To finish the game,      "
CR ." simply reverse 0 " ;    ;S
```

We also need a flag to tell whether the player wants to continue. The handy word *Y/N* (which can be used in many games) asks for an input string from the console, which is stored in PAD, the location of the first byte of the scratchpad area that moves upward as the dictionary expands. An upper case or lower case *Y* returns a true flag.

```
: Y/N ( c -- flag )
  PAD 80 EXPECT PAD C@ CR CR
  95 AND 89 = ;
```

Now we can ask the player at the beginning whether he wants instructions for the game, and at the end, whether he wants to play again. This code invokes the word *OBJECT*, if requested:

```
: INSTRUCTIONS CR CR 10 SPACES
  ." The game of REVERSE"
  CR CR
  ." Would you like"
  ."   instructions?"
  Y/N IF OBJECT THEN ;
```

Our first task will be to set up a ten-number array, and then initialize it with the numbers *1* to 9 (element *0* will not be used). We can reserve an integer word array with:

```
: DIM ( n --)
  <BUILDS
  (reserve an integer word array)
  1+ 2 * ALLOT
  DOES> ;
```

Defining words in FORTH execute by compiling new word definitions into the dictionary. Two examples are *CONSTANT* and *VARIABLE*, which will store named single-precision numeric values. The far more powerful defining word *:* (pronounced *colon*) permits compilation into the dictionary of word definitions using a "building block" approach. The word *:* (colon) uses *;* (semicolon) as a terminating word. Later execution of a colon-defined word is equivalent to the execution of each of its building blocks.

If you ever wondered about the choice of *:* and *;* as FORTH words, this is the reason. . .When Charles Moore developed FORTH, he started out with:

```
DEFINE: Name something something END;
```

but found it tiresome to keep typing *DEFINE:* and *END;* each time, so he shortened them to save himself (and us!) many keystrokes.

FORTH also allows you to create new high-level defining words. Such words may then be executed to create new classes of words, like numeric arrays or double-precision variables and constants. You will find a discussion of user-defining words in chapter eleven of Leo Brodie's *Starting Forth*. Remember, however, that he uses the series CREATE. . .DOES>. In fig-FORTH, this should be <BUILDS. . .DOES>.

The word *DIM*, which we just created, is such a word. It's a defining word in the form *n DIM xxxx*, which looks for a number on the stack, and then produces an $n+1$-length word array named *xxxx*, which we can then fill with elements *0* through *n*. We do this with:

```
9 DIM ARRAY
```

which reserves a ten-word array.

Now we need a word to store numbers in the array, which we can do with:

```
: A! ( store an array element )
     ( array-value index -- )
  2 * ARRAY + ! ;
```

We can then use a DO. . .LOOP to initialize our array with the numbers *1* to 9, in order:

```
: AINIT ( initialize the array)
  10 1 DO I DUP A! LOOP ;
```

Just as we needed a word to store numbers in the array, we'll need words to fetch the array elements and to place them on the data stack—and, also, to print them. The fetch word, which looks like the store word is:

```
: A@ ( fetch an array element )
  2 * ARRAY + @ ;
```

and the print word becomes:

```
: A. ( print the array )
  CR ." The list is now..."
  CR 6 SPACES
  10 1 DO
  I A@ 3 .R
  LOOP ;
```

To start the game, we must present the player with a scrambled list of numbers. Since the game was written in FORTH as an exercise in array manipulation, we are right in supposing that this code will be the heart of the game. Most games require a random number generator, and this one is no exception.

```
0 VARIABLE SEED
: RND ( random number generator)
      ( range -- rnd# )
  SEED @ 259 * 3 + 32767 AND
  DUP SEED ! 32767 */ ;
```

This is a pseudorandom number generator (there is a difference, as we'll see), courtesy of *Forth Dimensions*. RND generates a number in the range 0 through −1 and is used to scramble the number list.

```
: ASCRAMBLE ( mix up array values)
  1 9 DO
  I RND 1+ (    calculate K       )
  I     A@ ( get    ARRAY(I) value)
  OVER  A@ ( get    ARRAY(K) value)
  I     A! ( store  ARRAY(K)      )
           (        in ARRAY(I)   )
  SWAP  A! ( store  ARRAY(I)      )
           (        in ARRAY(K)   )
  -1 +LOOP ;
```

The code, as you see, counts down the array from 9 to 1 (0 is ignored) and uses RND to calculate a value *K*, with which to swap each of the numbers into a different slot.

Since the object of the game is to get the numbers in the right order (in effect, to neutralize or unscramble the results of ASCRAMBLE), we need a word which will check that ARRAY has been placed in the proper numerical order and, if so, will return a true flag.

```
: ACHECK ( ascending sequence? )
         ( -- flag )
  1 10 1 DO
  I DUP A@ = AND
  LOOP ;
```

This routine puts a *1* (or true flag) on the stack, which remains true (1 AND 1 equals 1) as long as each comparison is true, but switches to false (0 AND 1 equals 0) if any one fails.

Because **Reverse** is an interactive game, we need a way to get input from the player. We must solicit the number of elements in the list that he wants to reverse (which reminds me of page 1 of IBM's tutorial manual on their computer SYSTEM/38: "This manual follows the convention that HE refers to HE or SHE"). The code features a continuous loop, so that, if the player enters any character other than 0 through 9, the program issues an error message and loops back to ask for another number.

```
: GETIN ( get amount to reverse )
        ( -- n )
  BEGIN CR
  ." Reverse how many? "
  PAD 80 EXPECT PAD @ 48 -
  DUP 0< OVER 9 > OR DUP
  IF CR
  ." Only 0 thru 9 allowed. "
  THEN 0= UNTIL CR ;
```

You might notice that the ATASCII code for a 0 is decimal 48, and for a 9 is decimal 57, so that we must subtract 48 from the ATASCII code to get the number we want.

The other crucial part of the game (and the other exercise in array manipulation) is a routine to reverse a leftmost subset of the array, starting with element number 1. The code is similar to ASCRAMBLE, except that it rearranges a part of the array, instead of the complete array from 1 to 9.

```
: AREVERSE ( reverse a subset)
           ( n -- )
  DUP 2 /  ( loop limits are )
  1+ 1     (    1 to [n/2]+1 )
  DO DUP   ( calculate index )
  I - 1+   (          n-I+1 )
  DUP A@
  SWAP  (    get ARRAY[n-I+1])
  I A@  (    get ARRAY[I]    )
  SWAP  ( store ARRAY[I]     )
  A!    (    in ARRAY[n-I+1])
  I     ( store ARRAY[n-I+1])
  A!    (    in ARRAY[I]     )
  LOOP DROP ;
```

We need two more pieces of the puzzle, then we're ready to put the game together. There is a variable:

```
0 VARIABLE MOVES
```

which keeps track of the number of reverses so far. We also need a keypress to start up the random number generator, as follows:

```
: SKEY CR
  ." Please depress any key:"
  KEY SEED ! ;
```

Now we're ready for the game definition, which uses all the previously defined words, mostly in the order in which we defined them.

```
: REVERSE ( play the game )
  125 EMIT SKEY
  INSTRUCTIONS AINIT
  BEGIN
  ASCRAMBLE 0 MOVES !
   BEGIN
    A. GETIN DUP 0=
     IF 1 ELSE
      AREVERSE 1 MOVES +!
       ACHECK
     THEN
   UNTIL
  A. CR ." You made " MOVES @ .
  ." reversals." CR CR
  ." Care to play again? "
   Y/N 0=
  UNTIL
  CR ." Thanks for playing "
  ." REVERSE..." CR CR ; ;S
```

And there we have it. The code will fit in seven or eight screens without crowding. You can end each screen with a *;S* (semi-S or stop), a fig-FORTH run-time procedure that's available for separate use, if needed (**FORTH-79** doesn't specify names for its run-time procedures, which are assumed to be inaccessible to the user). You can then load screens one at a time. If everything loads correctly, you can use a final screen (for example, screen 9) as a LOAD screen *(1 LOAD 2 LOAD 3 LOAD 4 LOAD. . .;S)*, so that 9 LOAD will load the entire game.

How do you play the game to beat a competitor? Mostly by trial and error. There are three different strategies. According to Peter Sessions, there is the algorithmic approach and the heuristic approach. The first is to adopt the strategy of a computer program: move the 9 to the right, then the 8, etc. With two reversals for each of the nine numbers, that should never take more than eighteen moves. Burton's only comment about the game is: "If more than fifteen moves are made to win, you haven't got the hang of the game."

The second approach is by rule of thumb, seeking to take advantage of partial orderings in the list. This is the way most people play.

The third way, of course, is to "cheat." This isn't meant to advocate dishonesty, but merely to make sure ahead of time that, if one plays a game, one is familiar with the rules and plays by them.

The rules aren't always what they seem. There's a story about Alan Mathison Turing (Ph.D., Princeton, 1939), the eccentric British mathematician, who worked as Britain's chief cryptographer and managed to crack the German cypher code during World War II, by building one of the earliest computers (the Germans didn't find out until thirty years later).

He wanted to become proficient with a rifle, so he enrolled in the wartime Home Guard. The form read: "Do you understand you place yourself liable to military law?" Turing, characteristically, answered no instead of yes. Once he became an excellent shot, he stopped attending parades. When the apoplectic colonel called him to task, Turing said, "I am not a soldier. . .look at my form." He had been improperly enrolled. It was typical of Turing at all times to look for the optimal strategy. His life story in the new book by Andres Hodges, *Alan Turing: The Enigma*, reads like a classic Greek tragedy. Although he did more than any one man to win the war for Britain, he died unrecognized (a suicide) in 1954, at age 41.

Notice that, in this game, Burton uses a pseudo-random number generator. These come in two flavors —those that repeat and those that don't. The repeating ones are the most useful in scientific experiments. If you're running a computer simulation of, say, a grocery store, to find out how many shopping carts you need, and use random numbers to model the times between customers' arrivals, you want to be able to repeat the experiment for three, four and five check-out counters using exactly the same numbers.

This game asks a keypress to seed the random number generator. Most people, out of laziness, hit the SPACE BAR. The game will then keep churning out the same sequence of scrambled numbers each time it is called. Someone who took the trouble to work out the answers ahead of time would have a significant advantage.

Of course. the number of possible lists is a large number. For example, the 9 can appear in any of nine slots, then the 8 in any of the remaining slots, and so on. The answer is 9 *times* 8 *times* 7. . .*times* 1, or 9 *factorial* (9!). You can work it out for yourself with a program something like this:

```
: FACTORIAL 1 + 1 1 ROT ROT
  DO I * LOOP ;
```

which gives us:

```
7 FACTORIAL 5040  OK
```

In this particular instance, however, we need some double-precision words:

```
: 2SWAP >R ROT ROT R> ROT ROT ;
: PICK 2* SP@ + @ ;
: ROLL DUP 1= IF DROP ELSE DUP
  1 DO SWAP R> R> ROT >R >R >R
  LOOP 1 DO R> R> R> ROT ROT
  >R >R SWAP LOOP THEN ;
: S->D DUP 0< MINUS ;
: 2@ DUP 2+ @ SWAP @ ;
: 2! SWAP OVER ! 2+ ! ;
: D* OVER 5 PICK U* 6 ROLL
  4 ROLL * + 2SWAP * + ;
```

The new factorial program becomes:

```
: 2VARIABLE
  <BUILDS 4 ALLOT DOES> ;
1. 2VARIABLE PRODUCT
: 2FACTORIAL 1. PRODUCT 2! 1 +
  1 1 ROT ROT DO PRODUCT 2@
  I S->D D* PRODUCT 2! LOOP
  CR PRODUCT 2@ D. ;
```

which will then give us:

```
9 2FACTORIAL 362880  OK
```

Poker players always ask for a new deck in the original wrapping, then look for markings on the backs by flipping the deck and riffling through the cards. We could accomplish something similar by swapping random number generators, of which there are several we can choose from.

This one appears on page 265 of Brodie's book:

```
1 VARIABLE RND HERE RND !
: RANDOM RND @ 31421 * 6927
  + DUP RND ! ;
: CHOOSE ( u1 - u2 )
  RANDOM U* SWAP DROP ;
```

Alan Winfield's *The Complete Forth*, on page 113, has:

```
0 VARIABLE RND   1234 RND !
: RANDOM RND @ 1021 * 41 +
  DUP RND ! ;
```

which places a new random number on the stack and which, he warns, repeats itself every 65535 numbers,

(continued on page 18)

(continued from page 16)

"but that should be no problem here!" The following variation appears on page 65 of *Forth Tools* by Anderson and Tracy:

```
0 VARIABLE SEED 1234 SEED !
: RAND ( -- n )
  SEED @ 5421 * 1+
  DUP SEED ! ;
: RANDOM ( n -- random number )
  RAND SWAP MOD ;
```

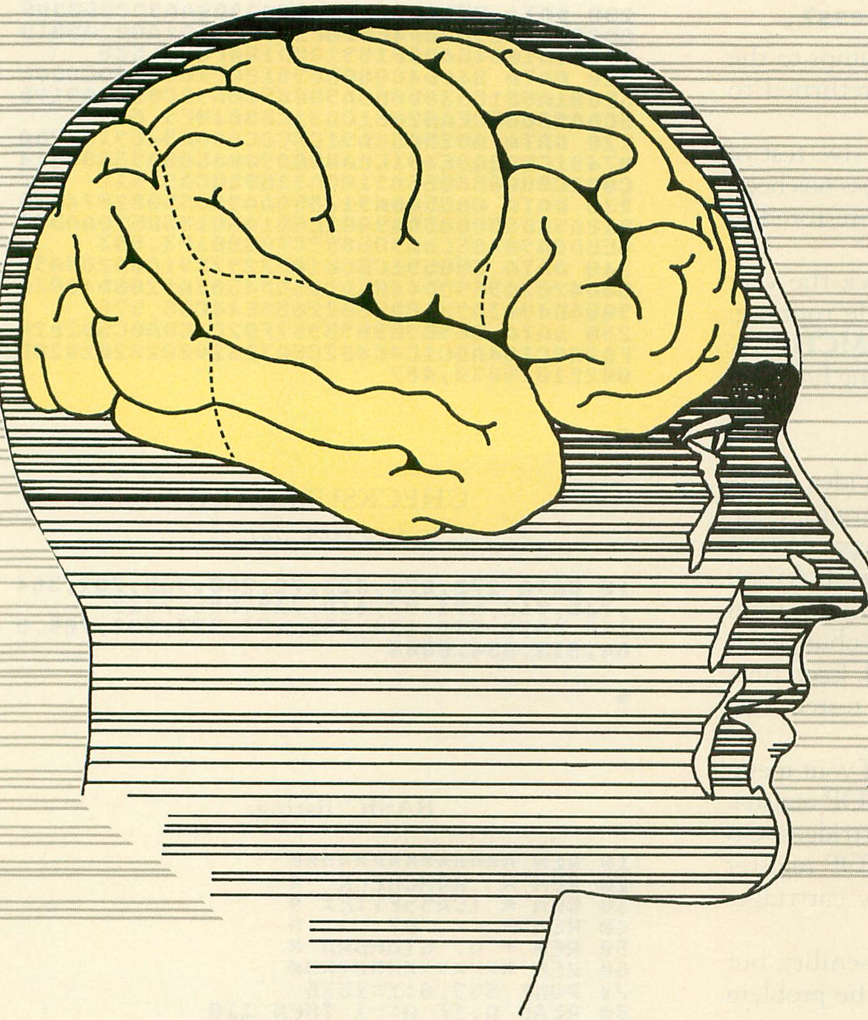
The Atari hardware generates a true random number at location 53770, which you can use like this:

```
: RND# 53770 C@ SWAP /MOD DROP ;
```

or in the more elaborate version that appears on page 41 of Ekkehard Floegel's book, *Forth for the Atari*:

```
0 VARIABLE RND 53770 @ RND !
: RANDOM RND @ 31421 * 6972
  + DUP RND ! ;
: RND# ( n1 -- n2 )
  RANDOM U* SWAP DROP ;
```

You can sum it up for your audience this way. . . Play **Reverse.** Have fun! Check the keyboard and pick a key at random, instead of the SPACE BAR. Don't be lazy! □



RAM CHECK

A memory test for your Atari computer

16K Cassette or Disk, 2.0S DOS

by Angelo Giambra

Like humans, computers sometimes become senile. That is, they lose their ability to remember things accurately. This happens when the random access memory (RAM) becomes faulty. The symptoms can range from programs merely producing strange, unpredictable results to total system lockup.

How can you tell if all the RAM in your system is good? I haven't seen any programs on the market which will give it a thorough workout. Well, now there's **RAMCHECK**.

If you have a disk system, key in the program in Listing 1 and SAVE it to disk. If you use a cassette recorder, key in Listing 2 and SAVE it to cassette. When you RUN the program, the DATA statements will be checked for accuracy. If you're a disk user, read the next paragraph. Otherwise, skip ahead to the paragraph for cassette users.

Disk users.

If there are no errors, a machine language file called **RAMCHECK** will be created on your disk. Here's how to use it.

Remove any cartridges and boot your system from disk. Now load **RAMCHECK** into memory. If you are using Atari 2.0S DOS, use the *L* option. If you have DOSXL, key in *LOAD **RAMCHECK***. Skip the next paragraph and read on.

Cassette users.

If there are no errors, the **RAMCHECK** program will be POKEd into memory and executed immediately. Since this version executes with the BASIC cartridge present, the upper 8K of RAM cannot be tested on a 48K system, because BASIC occupies these addresses.

How it works.

Your screen will begin to cycle through various colors as **RAMCHECK** starts at low memory and works its way up to high memory, checking every address for data integrity. At each memory location, **RAMCHECK** stores, then reads all the values from 0 to 255. If the value read matches the value stored, the test continues; otherwise **RAMCHECK** clears the screen and prints:

RAM ERROR AT ADDRESS <address>

If there are no errors, RAMCHECK jumps to the system warmstart routine, and you are returned to DOS or BASIC.

RAMCHECK performs a non-destructive test of memory. That is, after each memory address has been checked, the contents of that address are restored to their original value.

Interestingly, **RAMCHECK** must check the very memory addresses where it resides—while running. When testing reaches this section, **RAMCHECK** moves itself into lower memory, where testing has been completed, and proceeds.

Go to it!

As **RAMCHECK** runs, you can get an idea of the incredible speed of machine language programs. Consider this: on a 48K machine, 12,582,656 separate tests are performed (256 tests per memory address). As you watch the screen, colors will vary in luminance from dark to bright. Each time the luminance changes, it means 512 memory bytes have been tested. Each time the hue changes, 4K of memory has been tested. BASIC would take hours to do this test.

RAMCHECK will determine the size of your memory when it loads, by checking the RAMTOP register. That's why disk users must remove any cartridges. On a 48K system, the OS points the RAMTOP register to just below the starting address of any cartridges resident.

I hope your system never suffers from senility, but if it does, **RAMCHECK** will diagnose the problem right away! □

Listing 1.

```
10 REM ***************
20 REM *  RAMCHECK   *
30 REM *     by      *
40 REM * A. Giambra *
50 REM ***************
60 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
0,0,10,11,12,13,14,15
70 DIM DAT$(96),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=190:RESTOR
E 200:TRAP 150:? "KCHECKING DATA"
80 TOTAL=0:LINE=LINE+10:POSITION 2,2:?
 "LINE:";LINE:READ DAT$:IF LEN(DAT$)<>
96 THEN 180
90 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
100 FOR X=1 TO LEN(DAT$)-1 STEP 2:D1=A
SC(DAT$(X))-48:D2=ASC(DAT$(X+1))-48:BY
TE=HEX(D1)*16+HEX(D2)
110 IF PASS=2 THEN PUT #1,BYTE:NEXT X:
READ CHKSUM:GOTO 80
120 TOTAL=TOTAL+HEX(D1)+HEX(D2):NEXT X
130 READ CHKSUM:IF TOTAL=CHKSUM THEN 8
0
140 GOTO 180
150 IF PEEK(195)<>6 AND PEEK(195)<>5 T
HEN 180
160 IF PASS=0 THEN OPEN #1,8,0,"D:RAMC
HECK":PASS=2:LINE=190:RESTORE 200:TRAP
 170:? "KCREATING FILE":GOTO 80
170 CLOSE #1:END
180 IF LEN(DAT$)=78 AND LINE=250 THEN
TRAP 160:GOTO 90
190 ? "BAD DATA: LINE ";LINE:END
200 DATA FFFF00307D3078A900AA9D00D09D0
0D29D00D39D00D4E8D0F185008501A002A5018
D1AD0B10048A900189100D100F005,536
210 DATA 84CD4C00066901D0F1689100C8D0E
5E601A501C930D0D8A90085CBA92C85CCA9198
5CDA93085CEA028B1CD91CB8810F9,659
220 DATA A025A56A91CBC8C8C8A94C91CBC8A
97491CBC8A9E491CBA000A9008500A93085014
C002C00068806689100A916850CA9,612
230 DATA 06850DA917850AA906850B2074E46
018A5CD650085D49002E601A50185D520AAD92
0E6D8A98485CBA90685CCA000B1F3,593
240 DATA 300591CBC810F7297F91CBA200A90
B8D4203A9698D4403A9068D4503A9208D4803A
9008D4903A9018DF0022056E44C66,576
250 DATA 069B9B9B9B9B7FD2C1CDA0C5D2D2C
FD2A0C1D4A0C1C4C4D2C5D3D3202020202020E
002E1020030,487
```

•

CHECKSUM DATA.

(see page 32)

```
10 DATA 272,674,323,72,280,965,202,364
,735,912,694,97,438,719,585,7332
160 DATA 596,134,793,174,577,892,766,6
64,613,834,6043
```

•

BASIC listing.

```
10 REM ***************
20 REM *  RAMCHECK   *
30 REM * (CASSETTE) *
40 REM *     BY      *
50 REM * A. GIAMBRA *
60 REM ***************
70 POKE 559,0:I=1536
80 READ A:IF A=-1 THEN 110
90 POKE I,A:I=I+1:CHK=CHK+A
100 GOTO 80
110 I=12288
120 READ A:IF A=-1 THEN 150
130 POKE I,A:I=I+1:CHK=CHK+A
140 GOTO 120
150 READ A:IF A<>CHK THEN POKE 559,34:
? "ERROR IN DATA STATEMENTS":END
160 X=USR(12288)
170 DATA 104,145,0,169,23,133,12,169,6
,133,13,169,23,133,10,169,6,133,11,32,
116,228,96,24
180 DATA 165,205,101,0,133,212,144,2,2
30,1,165,1,133,213,32,170,217,32,230,2
16,169,132,133,203
190 DATA 169,6,133,204,160,0,177,243,4
8,5,145,203,200,16,247,41,127,145,203,
162,0,169,11,141
200 DATA 66,3,169,105,141,68,3,169,6,1
41,69,3,169,32,141,72,3,169,0,141,73,3
,169,1
210 DATA 141,240,2,32,86,228,76,102,6,
155,155,155,155,155,127,210,193,205,16
0,197,210,210,207,210
220 DATA 160,193,212,160,193,196,196,2
10,197,211,211,32,32,32,32,32,32,-1
230 DATA 120,169,0,170,157,0,208,157,0
,210,157,0,211,157,0,212,232,208,241,1
33,0,133,1,160
240 DATA 2,165,1,141,26,208,177,0,72,1
69,0,24,145,0,209,0,240,5,132,205,76,0
,6,105
250 DATA 1,208,241,104,145,0,200,208,2
29,230,1,165,1,201,48,208,216,169,0,13
3,203,169,44,133
260 DATA 204,169,25,133,205,169,48,133
,206,160,40,177,205,145,203,136,16,249
,160,37,165,106,145,203
270 DATA 200,200,200,169,76,145,203,20
0,169,116,145,203,200,169,228,145,203,
```

```
160,0,169,0,133,0,169
280 DATA 48,133,1,76,0,44,-1
290 DATA 31859
```

•

CHECKSUM DATA.

(see page 32)

```
10 DATA 272,674,797,197,690,282,450,16
6,577,494,370,432,930,701,3,7035
160 DATA 346,122,763,404,678,137,734,2
43,603,364,192,98,381,784,5849
```

•

Assembly listing.

```
; **************
; *            *
; *  RAMCHECK  *
; *     by     *
; * A. GIAMBRA *
; *            *
; **************
;
; OS EQUATES
;
RAMTOP    =    $6A      ;TOP OF MEMORY
DOSVEC    =    $ØA      ;DOS MENU VECTOR
DOSINI    =    $ØC      ;DOS INITIALIZATION
CRSINH    =    $Ø2FØ    ;INHIBIT CURSOR
PUTB      =    $ØB      ;PUT CHARACTERS
ICCOM     =    $Ø342    ;OS I/O COMMAND
ICBAL     =    $Ø344    ;BUFFER ADDRESS
ICBLL     =    $Ø348    ;BUFFER LENGTH
CIOV      =    $E456    ;OS I/O ROUTINE
GTIA      =    $DØØØ    ;VIDEO CHIP
POKEY     =    $D2ØØ    ;I/O CHIP
PIA       =    $D3ØØ    ;PORTS
ANTIC     =    $D4ØØ    ;SCREEN DISPLAY
COLBAK    =    $DØ1A    ;BACKGROUND COLOR
LINZBS    =    $ØØ
WARMSTART =    $E474
FRO       =    $D4      ;FP REGISTER
IFP       =    $D9AA    ;INTEGER TO FP
FPA       =    $D8E6    ;FP TO ASCII
INBUFF    =    $F3      ;FP REGISTER
;
; WORK EQUATES
;
HEX3Ø     =    $3Ø
HEX2CØØ   =    $2CØØ
IX        =    $CB      ;WORK INDEX
IX1       =    $CD      ;WORK INDEX
JUMP      =    $4C      ;MACHINE CODE JMP
CR        =    $9B      ;CARRIAGE RETURN
TAB       =    $7F      ;SCREEN TAB
SP        =    $2Ø      ;SPACE
;
          *=   $3ØØØ
;
BEGIN     SEI           ;DISABLE INTERRUPTS
          LDA #Ø
          TAX
CLRCHP    STA GTIA,X    ;CLEAR THE CHIPS
          STA POKEY,X
          STA PIA,X
          STA ANTIC,X
          INX
          BNE CLRCHP
          STA LINZBS    ;1ST MEMORY BYTE
          STA LINZBS+1
          LDY #2        ;START AT 3RD BYTE
LOOPER    LDA LINZBS+1  ;GET PAGE NUMBER
          STA COLBAK    ;STORE IN COLOR
AGAIN     LDA (LINZBS),Y
          PHA           ;SAVE THE CONTENTS
          LDA #Ø
          CLC
LOOP1     STA (LINZBS),Y ;STORE VALUE
          CMP (LINZBS),Y ;SAME VALUE?
          BEQ LOOP2
          STY IX1       ;SAVE Y REGISTER
          JMP ERROR     ;RAM ERROR
;
LOOP2     ADC #1        ;ADD 1
          BNE LOOP1     ;DO IT AGAIN
          PLA           ;RESTORE CONTENTS
          STA (LINZBS),Y
          INY           ;BUMP TO NEXT ADDRESS
          BNE AGAIN
          INC LINZBS+1  ;NEXT PAGE
          LDA LINZBS+1  ;LOAD PAGE NUMBER
INDEX     CMP #HEX3Ø    ;START OF CODE?
          BNE LOOPER    ;NO, KEEP GOING
ENDER     LDA # <HEX2CØØ ;POINT IX TO $2ØØØ
          STA IX
          LDA # >HEX2CØØ
          STA IX+1
          LDA # <LOOPER ;POINT IX1 TO
          STA IX1       ;OUR CODE
          LDA # >LOOPER
          STA IX1+1
          LDY #ENDER-LOOPER
TRANSFER  LDA (IX1),Y   ;MOVE CODE FROM
          STA (IX),Y    ;$4ØØØ TO $2CØØ
          DEY
          BPL TRANSFER
          LDY #INDEX-LOOPER+1
          LDA RAMTOP    ;MEMTOP VALUE
          STA (IX),Y    ;MODIFY COMPARE
          INY           ;POINT TO CODE
          INY           ;AFTER 'ENDER'
          INY           ;LABEL
          LDA #JUMP     ;CHANGE OPERATOR
          STA (IX),Y    ;TO JMP
          INY
          LDA # <WARMSTART
          STA (IX),Y    ;TARGET OF JMP
          INY           ;IS WARMSTART
          LDA # >WARMSTART
          STA (IX),Y    ;MODIFY THE CODE
          LDY #Ø
          LDA #$ØØ      ;NOW POINT LINZBS
          STA LINZBS    ;TO $4ØØØ
          LDA #HEX3Ø
          STA LINZBS+1  ;AND CONTINUE
          JMP $2CØØ     ;THE TEST
;
          *=   $Ø6ØØ
;
ERROR     PLA           ;RESTORE LAST
          STA (LINZBS),Y ;ADDRESS
          LDA # <RETURN ;POINT DOSINI
          STA DOSINI    ;TO AN RTS
          LDA # >RETURN
          STA DOSINI+1
          LDA # <BACK   ;POINT DOSVEC
          STA DOSVEC    ;TO OUR CODE
          LDA # >BACK   ;WARMSTART WILL
          STA DOSVEC+1  ;RETURN THERE
          JSR WARMSTART ;DO WARMSTART
RETURN    RTS
;
BACK      CLC
          LDA IX1       ;GET OFFSET
          ADC LINZBS    ;ADD MEM ADDRESS
          STA FRO       ;PUT IN FP REGISTER
          BCC NOCARRY
          INC LINZBS+1
NOCARRY   LDA LINZBS+1
          STA FRO+1
          JSR IFP       ;CONVERT TO FP
          JSR FPA       ;CONVERT TO ASCII
          LDA # <ADDR   ;POINT IX TO
          STA IX        ;OUR BUFFER
          LDA # >ADDR
          STA IX+1
          LDY #Ø
DD        LDA (INBUFF),Y ;GET THE NUMBER
          BMI OK
          STA (IX),Y    ;PUT IN OUR BUFFER
          INY
          BPL DD
OK        AND #$7F      ;TURN OFF HIGH BIT
          STA (IX),Y    ;SAVE LAST DIGIT
          LDX #Ø        ;CHANNEL Ø (EDITOR)
          LDA #PUTB     ;PUT BYTES
          STA ICCOM
          LDA # <MESG   ;POINT TO OUR
          STA ICBAL     ;MESSAGE
          LDA # >MESG
          STA ICBAL+1
          LDA #32       ;WRITE 32 CHARACTERS
          STA ICBLL
          LDA #Ø
          STA ICBLL+1
          LDA #1        ;TURN OFF CURSOR
          STA CRSINH
          JSR CIOV      ;DO I/O
DX        JMP DX        ;LOOP FOREVER
;
MESG      .BYTE CR,CR,CR,CR,CR,TAB
          .BYTE +$8Ø,"RAM ERROR AT "
          .BYTE +$8Ø,"ADDRESS"
          .BYTE SP
ADDR      .BYTE SP,SP,SP,SP,SP
;
          *=   $Ø2EØ
          .WORD BEGIN
          .END
```

•

RAM Operating System For Atari XLs

64K Disk XL

by Ken Alexander

The Atari XLs advertise 64K of RAM, but, as you know, this is rather misleading. The 6502 processor can only access 64K of memory, and the ROM takes up 14K of this, while the I/O region takes up 2K. This means that 16K of RAM is rendered inaccessible under normal circumstances. What good is RAM that can't be used?

Soon after I purchased my 1200, I called Atari and got their original 1200 information package through the mail. It promised to tell how to access the whole 64K of RAM. All that it said on the subject was: "PIA PORTB at 54017 ($D301) is used to control the memory management. Bit0 controls access to the OS ROM and is normally enabled (bit=1). Setting the bit to 0 will disable the 14K of OS ROM in the region $C000-$CFFF and $D800-$FFFF and enable the RAM. Unless another OS has already been provided, the system will crash on the next interrupt." To disable the ROM, you must first provide another OS in the RAM. . . but, to provide another OS in the RAM, you must first disable the ROM. Fun.

Then the translator disk came out, so it had to be possible somehow. The translator puts the equivalent of a 400/800 OS into an XL, and it is highly recommended for XL owners. It not only allows XLs to run many Atari programs that they couldn't at first, but also gives them a modifiable OS.

Many minor changes can be made, and a redefined character set can be made by directly altering the main set at $E000-$E3FF. It also frees the 4K block of memory from $C000 to $CFFF that was wasted ROM in the 800, giving you 4K of RAM that is absolutely protected. (Actually, the region from $CF00-$CFFF isn't available, because residing here are interrupt handlers that allow for some hardware differences between the XLs and the 400/800.)

Unfortunately, changes to the OS can't be made until the translator has done its work, and these must be small enough to fit in, or it's very easy to foul something up. Recovery is difficult, because RESET causes total lockup.

I disassembled everything, but I couldn't find the translator program. Apparently it wipes itself out after changing the OS. I wanted to know how the translator worked, so I could make my own OS and then put it in all at one time.

At first, I tried creating a new OS in RAM from $4000-$7FFF and writing a program that would move it up piece by piece. It used a vertical blank interrupt (VBI) routine that would, each sixtieth of a second, disable the ROM, move 256 more bytes of the OS and re-enable the ROM before the VBI returned.

After all that, it didn't work. I finally reached Atari's toll-free number and got through to someone who knew about it—and who actually called me back in the evening to talk about it. He sent me the program in Listing 1.

In accordance with true Atari policy, they never told the public about this. It's so simple that my superior brain failed to think of it. With it, you can turn your XL OS into RAM, or be Frankenstein—create your very own OS, and this will give it life. □

Listing 1.
BASIC listing.

```
10 REM RAM OS BASIC MAKER
20 REM
30 OPEN #1,8,0,"D:RAMOS.OBJ"
40 TRAP 60
50 READ A:PUT #1,A:GOTO 50
60 END
100 DATA 255,255,0,6,101,6,32,70,6,177
,203,145,205,200,208,249,230,204,230,2
06,202,224,48,208,3,32,91
110 DATA 6,224,0,208,233,120,169,0,141
,14,212,169,254,141,1,211,32,70,6,177,
205,145,203,200,208,249,230
120 DATA 204,230,206,202,224,48,208,3,
32,91,6,224,0,208,233,88,169,64,141,14
,212,0,169,0,133,203,169
130 DATA 192,133,204,169,0,133,205,169
,64,133,206,162,64,160,0,96,160,8,230,
204,230,206,202,136,208,248,96
```

•

CHECKSUM DATA.
(see page 32)

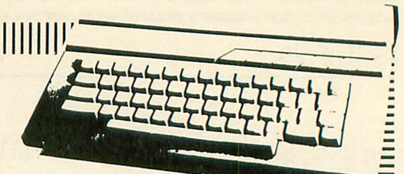
```
10 DATA 566,253,776,630,300,253,290,73
6,396,110,4310
```

Assembly listing.

```
;  "RAMOS"
;
;  This routine moves the 1200XL or
;  800XL OS ROM to RAM from $4000
;  to $7FFF, disables the ROM, then
;  moves the OS back to its proper
;  address in what is now RAM.
;
;  It can also be used to move an
;  alternate OS into place after
;  it is created at $4000-$7FFF.
;  Just Go to DISABLE at $61A
;  instead of the start at $600.
;
;  The screen will flicker as the
;  charset is disabled, and then
;  Viola! You have a RAM OS. Tell
;  all your friends.
;
;
;  equates
;
ROM      =    $CB       ;page 0 ROM index
RAM      =    ROM+2     ;page 0 RAM index
OSROM    =    $C000     ;start of OS ROM
OSRAM    =    $4000     ;start of RAM area
NMIEN    =    $D40E     ;ANTIC NMI disable
PORTB    =    $D301     ;PIA OS ROM disable
;
         *=   $0600
;
         JSR INIT       ;init move addresses
;
;move ROM to RAM
;
MOV1     LDA (ROM),Y
         STA (RAM),Y
         INY
         BNE MOV1
         INC ROM+1
         INC RAM+1
         DEX
         CPX #$30
         BNE MOV1A
         JSR SKIP       ;skip I/O region
MOV1A    CPX #0
         BNE MOV1
;
DISABLE  SEI            ;disable IRQ's
         LDA #0
         STA NMIEN      ;disable NMI's
         LDA #$FE
         STA PORTB      ;disable OS ROM
         JSR INIT
;
;move RAM to former ROM region
;
MOV2     LDA (RAM),Y
         STA (ROM),Y
         INY
         BNE MOV2
         INC ROM+1
         INC RAM+1
         DEX
         CPX #$30
         BNE MOV2A
         JSR SKIP
MOV2A    CPX #0
         BNE MOV2
;
ENABLE   CLI            ;IRQ's
         LDA #$40
         STA NMIEN      ;NMI's
         BRK            ;that's all folks
;
INIT     LDA # <OSROM
         STA ROM
         LDA # >OSROM
         STA ROM+1
         LDA # <OSRAM
         STA RAM
         LDA # >OSRAM
         STA RAM+1
         LDX #$40
         LDY #0
         RTS
;
SKIP     LDY #8
SKIP1    INC ROM+1
         INC RAM+1
         DEX
         DEY
         BNE SKIP1
         RTS
         .END
```

POWER WITHOUT THE PRICE AT. . .COMPUTER CREATIONS

ATARI®



ATARI HARDWARE

COMPUTERS		COMPUTERS
Atari 800 XL	CALL	Atari 130 ST
Atari 65 XE	FOR NEW	Atari 520 ST
Atari 130 XE	PRICES	

DISK DRIVES

Atari 1050	CALL	Atari SF 324 (3½" floppy disk 250K)
Indus GT (Free Software)	FOR	Atari SF 354 (3½" floppy disk 500K)
Astra 2001	NEW	Atari SH 317 (3½" hard disk 10 MB)
Happy Enhancement for Atari 810 and 1050 Drives	PRICES	

GENERIC DISKS AT FANTASTIC PRICES!

GENERIC DISKS AS LOW AS 94¢ea.

Generic 100% Defect-Free/Guaranteed.

Includes sleeves, labels, write protect tabs, reinforced hub rings, lifetime warranty. (2 boxes minimum)

DISKETTES (2 bx. minimum) 10 per box		SS/SD	SS/DD	DS/DD
	2 boxes	11.99	13.99	15.99
	3-9 boxes	10.50	12.50	14.50
	10+boxes	9.40	11.49	13.99

★ ★ DEALER INQUIRIES INVITED ★ ★

PRINTERS

Atari XTM 201 (Non-impact Dot Matrix 20 CPS)	
Atari XTC 201 (Color, non-impact Dot Matrix 20 CPS)	
Atari XDM 121 (Daisy Wheel Letter Quality 12 CPS)	CALL
Atari XMM 801 (Dot Matrix, Impact 80 CPS)	FOR
Atari 1025 (Dot Matrix, Impact CPS)	NEW
Atari 1027 (Letter Quality)	
Atari ST 504 (Color Dot Matrix, non-impact 50 CPS)	PRICES
Atari 5 DM 124 (Daisy Wheel Letter Quality, 12 CPS)	
Atari SMM 804 (Dot Matrix, impact 80 CPS)	

STAR MICRONICS PRINTERS

SG-10 (80 column)	
SG-15 (136 column)	CALL
SD-10 (80 column)	FOR
SR-10 (80 column)	PRICES
SR-15 (136 column)	
Powertype Daisywheel	

EPSON PRINTERS

Epson RX-80+ (80 column	249.00
Epson RX-80+ FT (80 column)	319.00
Epson FX-80+ (80 column)	429.00
Epson Rx 100+ (135 column)	419.00
Okidata 92P	440.00

PRINTER INTERFACE CABLES

MPP-Microprint	49.95
MPP-1150 Parallel Printer Interface	59.95
Microbits Microstuffer	109.95

PRINTER RIBBONS

Gemini Printers (Black/Blue/Red/Purple)	3.00
Epson Printers	6.95

OTHER HARDWARE

Atari 1010 Program Recorder	54.00
Atari CX 77 Touch Tablet	CALL
Atari Light Pen	CALL
Microbits 64K (600XL) Expansion	69.00
Ram Rod XL w/ Omniview for 800 XL	99.00
Omniview for 800 XL	49.00
B.I. 80 Column Display Adaptor	CALL

MONITORS

Atari XC 141 (14" Composite Color)	
Atari XM 148 (12" Monochrome, 80 column, low resolution)	
Atari SM 124 (12" Monochrome, 80 column, high resolution)	CALL FOR
Atari SC 1224 (12" RGB Color)	PRICES
Sanyo 12" Green Screen	84.00
Sanyo 12" Amber Screen	89.00
Sanyo Color Screen, 13"	219.00
Sanyo 9" Green Screen	69.00
Monitor Cable	9.00

MODEMS

Atari 1030 Direct 300 Band Connect	CALL
Atari XM 301 Direct Connect 300 Band	FOR PRICES
MPP-1000E Modem	114.00
Signalman Mark XII Modem with R-Verter	299.00
Mark X with R-Verter	169.00

DISKETTE/CARTRIDGE/ CASSETTE FILES

Flip 'N' File 10	3.95
Flip 'N' File 15	6.95

SOFTWARE

ATARI

Logo (R)	68
Prog. 1, 2 or 3 (C)	18
Assembler Editor (R)	27
Macroassembler (D)	19
Microsoft Basic II (R)	45
Basic Cart. (R)	5
AtariWriter (R)	35
Family Finances (D)	18
Home Filing Mgr. (D)	21
Telelink I (R)	21
Visicalc (D)	45
Juggles House (C/D)	16
Juggles Rainbow (C/D)	16
Mickey in the Great Outdoors (D)	21
Skywriter (R)	16
Atari Music I or II	19
Speed Reading	27
Conv. Languages (ea.)	21
Video Easel	19
Type Attack	16
Asteroids	10

EPYX

Moreta	25
Dragonriders of Pern	25
The Right Stuff	25
The World's Greatest Baseball Game	25
FBI	25
Impossible Mission	25
Summer Games II	25
Summer Games	25
Pitstop II	25
Ballblazer	25
Rescue on Fractalus	25
Scrabble	25

ACCESS

Beach Head (D)	23
Beach Head II (D)	23
Raid Over Moscow (D)	23

ACTIVISION

Decathalon (R)	25
Pitfall II (R)	25
Space Shuttle (R)	25
Ghost Busters (D)	25

AMERICAN EDUCATIONAL

Spelling Grades 2 thru 8 (D)	28
Reading Comprehension (D)	28

AVALON HILL

Jupiter Mission (D)	38
Quest of Space Beagle (D)	38

BATTERIES INCLUDED

Paperclip (D)	59

BRODERBUND

Printship (D)	29
Spelunker (D)	19
Stealth (D)	19
Whistler's Brother (D)	19

CBS

Call for items and prices

CONTINENTAL

Home Accountant (D)	44

DATASOFT

Bruce Lee	23
Dallas Quest	23
Conan	25
Letter Wizard w/speller	CALL

ELECTRONIC ARTS

Archon (D)	29
Pinball Construction (D)	29
M.U.L.E. (D)	29
Murder/Zinderneuf (D)	29
One on One (D)	29
Archon II (D)	29
Financial Cookbook (D)	37
Music Construction (D)	29
Realm/Impossibility (D)	29
Hard Hat Mack	25
AXIS Assasin	25

INFOCOM

Cut Throats (D)	23
Deadline (D)	29
Enchanter (D)	23
Hitchhiker's Guide to the Galaxy (D)	23
Infidel (D)	26
Planetfall (D)	23
Sea Stalker (D)	23
Sorcerer (D)	26
Starcross (D)	29
Suspect (D)	29
Suspended (D)	29
Witness (D)	23
Zork I (D)	23
Zork II or III (D)	26
Invisiclues Hint Books	7

MICROPROSE

F-15 Strike Eagle (D)	23
Mig Alley Ace (D)	23
Solo Flight (D)	23

OSS

Action (R)	59
Action Tool Kit (D)	25
Basic XL (R)	59
DOS XL (D)	25
MAC/54 (R)	59
MAC/65 Tool Kit (D)	25

ORIGIN

Ultima III (D)	39

SCARBOROUGH

Mastertype	25

SIERRA ON LINE

Ultima I	23
Ultima II	39

SPINNAKER

Call for items and prices

SYNAPSE

Quasimodo	15
Alley Cat	15
Syn-File +	38
Syn-Calc	38
Syn-Trend	38
Syn-Comm	29
Syn-Stock	29

TRONIX

S.A.M.	39
Chatterbee	25

To order call TOLL FREE
1-800-824-7506

ORDER LINE ONLY

No extra charge for MasterCard or Visa

COMPUTER CREATIONS, Inc.

P.O. Bo 292467 - Dayton, Ohio 45429

For information call: (513) 294-2002 (Or to order in Ohio)

Order Lines Open 8:30 a.m. to 8:00 p.m. Mon.-Fri.; 10 a.m. to 4:00 p.m. Sat. (Eastern Standard Time). Minimum $10 per order. Orders under $100 add $3.00. C.O.D. (add $2.50). All prices are subject to change without notice. Call toll free number to verify prices and availability of product. Actual freight will be charged on all hardware. Software and accessories add $4.00 shipping and handling in Continental United States. Actual freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico. Ohio residents add 6% sales tax. NO CREDITS! Return must have authorization number.

CIRCLE #113 ON READER SERVICE CARD

MAXI COPY

16K Disk

by Grant Albrecht

MaxiCopy is a very useful utility for the Atari computer. It allows you to take full advantage of available memory when copying disk files. This helps to minimize or eliminate the need for disk swapping when more than one file on a disk has to be copied, but a duplicate disk is not desired.

MaxiCopy is written entirely in assembly language for optimum speed. The program demonstrates some useful routines for data input/output, as well as an effective error trapping routine.

Using it.

To use **MaxiCopy**, simply load the binary file. The program will autostart and begin by prompting the user to enter a filename. You don't have to enter the device prefix. **MaxiCopy** assumes drive 1. Note: minor modifications could permit flexibility in drive selection, but the program's greatest advantage is gained with a single-drive system.)

Enter the names of the files you wish to copy. The program will accept as many as sixteen filenames, after which it will sound the console buzzer to alert you to initiate copying.

If you wish to copy fewer than sixteen files, then simply press RETURN in response to the filename prompt, to terminate input. The program will prompt you to insert source and destination disks as required. When the copy is completed, the option is offered to start over or to return to DOS. □

Listing 1.
BASIC listing.

```
10 REM *** MAXICOPY MAKER ***
20 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
0,0,10,11,12,13,14,15
30 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 110:? "CHECKING DATA"
40 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 150
50 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
60 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X)
)-48:D2=ASC(DAT$(X+1))-48:BYTE=HEX(D1)
*16+HEX(D2)
70 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 40
80 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
90 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
THEN 40
100 GOTO 150
110 IF PEEK(195)<>6 THEN 150
120 IF PASS THEN CLOSE #1:END
130 ? "INSERT DISK WITH DOS, PRESS RET
URN";:INPUT DAT$:OPEN #1,8,0,"D:MAXICO
PY.OBJ"
140 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 110:GOTO 40
150 ? "BAD DATA: LINE ";LINE:END
1000 DATA FFFF002102214431 3A1021122144
```

```
1000 DATA FFFF0021022144313A1021122144
313A2021222144313A3021322144313A402142
2144313A5021522144313A60,544
1010 DATA 21622144313A7021722144313A80
21822144313A9021922144313AA021A2214431
3AB021B22144313AC021C221,990
1020 DATA 44313AD021D22144313AE021E221
44313AF021F22144313A002302230000002323
2323006423CD260000000000,828
1030 DATA 007D20202020202020204D41584943
4F50592062792047 2E20416C627265636874 1D
9B46696C6520746F20636F70,213
1040 DATA 793F9B496E7365727420736F7572
6365206469736B3A9B496E73657274206465 73
74276E206469736B3A9B4D6F,644
1050 DATA 726520746F20636F70793F9BFD9B
FD425245414B204B4559204142 4F52549B2080
26208426A96AA223202326A0,794
1060 DATA 69A9009900238810FAA900A22185
B086B1AD300238E9018D6423AD3102E9018D65
23A2008E0223A98BA2232023,79
1070 DATA 2618A90365B0A6B1208E261007C0
80D0E94CBC25A003B1B0C99BF01520EF25AE02
23E8E01090D08E0223A9CFA2,423
1080 DATA 23202326A9018D0023A999A22320
2326A900A222208E26AE002320E025AE0023A9
CE85B2A92785B38E0023A5B2,547
1090 DATA 9D2323A5B39D3323AD642338E5B2
8D6623AD6523E5B38D6723D0034C2E25AD2323
F010A9008D2323A9029D0223,20
1100 DATA CE01234CB624A9018D6923A90420
A626300DA210A9072064261 05AC088F01420FB
25AE0023FE1223A9009D4323,959
1110 DATA 9D53234CF024AE0023AD58039D43
23AD59039D5323BD0223C9029003FE02232080
26AE0023BD43231865B285B2,37
1120 DATA BD532365B385B320EF25AE0023EC
0223B01EEE0023AE00234C7224AE0023FE0223
AD58039D4323AD59039D5323,203
1130 DATA EE2323A9ADA223202326A900A222
208E26A9028D6923EE0123AE012320E025AE01
23BD1223D03EBD0223C902B0,288
1140 DATA 10A90820A626100920FB25AE0123
FE1223AE0123BD1223D01FBD232385B2BD3323
85B3BD43238D6623BD53238D,444
1150 DATA 6723A220A90B20642630D0AE0123
BD0223F009C9039008A9009D0223208426AE01
23EC0023B0034C4125BD0223,52
1160 DATA D005EC0223F00A4C5324A9D1A223
202326A9C1A223202326A900A222208E2630F0
AD0022C959D0034CE2236C0A,355
1170 DATA 00A9F085B0A92085B120EF25CAD0
FA60A5B018691085B09002E6B16084D4A90085
D520AAD920E6D8A000B1F330,476
1180 DATA 080980995E26C8D0F4995E26A93B
A226202326A5B0A6B18D44038E4503A200A909
8D4203A9FF8D48038E49034C,970
1190 DATA 56E47D202020202020202020202020
2020202020204E6F7420636F706965642020C5D2
D2CFD2A0B0B0B01C1C9B9D42,106
1200 DATA 03A5B29D4403A5B39D4503AD6623
9D4803AD67239D49034C56E4A210D002A220A9
0C9D42034C56E48D44038E45,531
1210 DATA 03A200A9058D4203A9FF8D48038E
49034C56E48D6823AD69230A0A0A0AAAA9039D
4203AD68239D4A03A9009D4B,476
1220 DATA 03A5B09D4403A5B19D45034C56E4
E002E102E2230000000000000000000000000000
00000000000000000000000000,723
```

•

CHECKSUM DATA.
(see page 32)

```
10 DATA 571,957,808,428,727,554,599,55
3,272,698,610,21,850,33,162,7843
1000 DATA 49,123,42,366,643,551,635,71
0,531,858,517,761,698,717,588,7789
1150 DATA 571,531,741,756,445,683,898,
884,5509
```

•

Listing 2.
Assembly listing.

```
; ********************************
; *                              *
; *         MAXICOPY 2.1         *
; *                              *
; * Prog to copy multiple files. *
; *      By Grant Albrecht       *
; *          84/07/18            *
; *                              *
; ********************************
;
        *=  $2100
;
; Equates
;
ICCMD   =   $Ø342
ICADR   =   $Ø344
ICLEN   =   $Ø348
ICAUX   =   $Ø34A
DLIST   =   $Ø23Ø
COPEN   =   3
CGBIN   =   7
CGTXT   =   5
CPBIN   =   11
CPTXT   =   9
CCLOS   =   12
OPINP   =   4
OPOUT   =   8
GOADR   =   $Ø2EØ
DOSLOC  =   $ØA
CIO     =   $E456
EOL     =   $9B
EOF     =   136
ZPG     =   $BØ
ZP2     =   $B2
FRØ     =   $D4
IFP     =   $D9AA
FASC    =   $D8E6
INBUFF  =   $F3
;
; Reserved memory
;
FNAMES  .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
        .BYTE "D1:"
        *=  *+13
;
BUFF    *=  *+256
RDONE   .BYTE Ø        ;# files read
```

```
WDONE    .BYTE 0       ;# files written
COUNT    .BYTE 0       ;# files to copy
PART     *=  *+16      ;Only part read?
EFLG     *=  *+16      ;Error flag
FLAG     .BYTE 0
;
SLO      *=  *+16      ;Start addr of
SHI      *=  *+16      ;file data.
;
LLO      *=  *+16      ;Length of
LHI      *=  *+16      ;file.
;
EOM      .WORD 0       ;End avail mem
FLEN     .WORD 0
MODE     .BYTE 0
CHAN     .BYTE 0
;
;
; Messages used by program
;
MS1      .BYTE $7D,"        MAXICOPY"
         .BYTE " by G. Albrecht",EOL
MS2      .BYTE "File to copy?",EOL
MS3      .BYTE "Insert source "
         .BYTE "disk:",EOL
MS4      .BYTE "Insert dest'n "
         .BYTE "disk:",EOL
MS5      .BYTE "More to copy?",EOL
MS6      .BYTE $FD,EOL ;(bell char)
MS7      .BYTE $FD,"BREAK KEY ABORT",EOL
;
; Actual code begins here
;
INIT     JSR CLOSE1   ;Close files,
         JSR CLOSE2   ;(just in case).
         LDA # <MS1
         LDX # >MS1
         JSR PRINT    ;Print title.
;
         LDY #105
         LDA #0
LP1      STA RDONE,Y ;Initilize
         DEY          ;memory.
         BPL LP1
;
         LDA # <FNAMES
         LDX # >FNAMES
         STA ZPG      ;(Pointer to
         STX ZPG+1    ; filenames)
;
         LDA DLIST
         SEC
         SBC #1       ;Establish top
         STA EOM      ;avail memory
         LDA DLIST+1  ;as 1 page below
         SBC #1       ;display list.
         STA EOM+1
;
; The following loop accepts
; filenames until just [RETURN]
; is pressed or sixteen names
; have been entered.
;
         LDX #0
LP2      STX COUNT
ASK      LDA # <MS2   ;Ask for
         LDX # >MS2   ;filename.
         JSR PRINT
;
         CLC
         LDA #3
         ADC ZPG      ;Put the name
         LDX ZPG+1    ;into FNAME
         JSR INPUT    ;buffer.
;
         BPL OK       ;Input Ok?
         CPY #128     ;BREAK key?
         BNE ASK      ;No, retry.
         JMP ABORT    ;Yes, abort.
;
OK       LDY #3
         LDA (ZPG),Y
         CMP #EOL     ;Just [RETURN]?
         BEQ OUT      ;Yes input done.
         JSR INCZPG   ;No.
         LDX COUNT
         INX
         CPX #16      ;Got 16 names?
         BCC LP2      ;No, get more.
;
         STX COUNT    ;Yes.
         LDA # <MS6   ;Alert user by
         LDX # >MS6   ;sounding bell.
         JSR PRINT
;
OUT      LDA #1       ;Reset counter
         STA RDONE    ;for # read.
;
SRC      LDA # <MS3   ;Ask for source
         LDX # >MS3   ;disk.
         JSR PRINT
         LDA # <BUFF  ;Wait for user
         LDX # >BUFF  ;to press
         JSR INPUT    ;[RETURN].
;
         LDX RDONE
         JSR ZLOOP    ;Set FNAME pntr.
         LDX RDONE
         LDA # <MEM   ;Point to start
         STA ZP2      ;of available
         LDA # >MEM   ;memory.
         STA ZP2+1
LP4      STX RDONE
         LDA ZP2      ;Keep the addr
         STA SLO-1,X  ;where this
         LDA ZP2+1    ;file is saved.
         STA SHI-1,X
;
         LDA EOM      ;Calculate max
         SEC          ;number of bytes
         SBC ZP2      ;that can be
         STA FLEN     ;read into mem.
         LDA EOM+1    ;And use this
         SBC ZP2+1    ;as file length
         STA FLEN+1   ;when reading.
;
         BNE READ     ;If nil, then
         JMP WRITE    ;write first.
;
READ     LDA FLAG     ;File open?
         BEQ OVER1    ;No, skip this.
         LDA #0       ;Yes, so...
         STA FLAG     ;Reset flag.
         LDA #2       ;Set flag so
         STA PART-1,X ;write routine
         DEC WDONE    ;will append.
         JMP OVER2    ;Skip file open.
;
OVER1    LDA #1
         STA CHAN     ;Open file on
         LDA #OPINP   ;channel 1 for
         JSR OPEN     ;input.
         BMI ER1
;
OVER2    LDX #$10     ;Retrieve bytes
         LDA #CGBIN   ;from the file.
         JSR PUTGET
         BPL NEND     ;No errors?
         CPY #EOF     ;EOF reached?
         BEQ REND     ;Yes.
;
ER1      JSR ERROR    ;Report error,
         LDX RDONE    ;then continue.
         INC EFLG-1,X ;Flag it.
         LDA #0
         STA LLO-1,X  ;Set file length
         STA LHI-1,X  ;to zero.
         JMP RCLOSE   ;And skip it.
;
REND     LDX RDONE
         LDA ICLEN+16
         STA LLO-1,X  ;Save the length
         LDA ICLEN+17 ;of the file.
         STA LHI-1,X
         LDA PART-1,X ;Part written?
         CMP #2
         BCC RCLOSE   ;No.
         INC PART-1,X ;Yes, now fini.
;
RCLOSE   JSR CLOSE1
         LDX RDONE
         LDA LLO-1,X  ;Adjust memory
         CLC          ;pointer to
         ADC ZP2      ;next avail
         STA ZP2      ;space.
         LDA LHI-1,X
         ADC ZP2+1
         STA ZP2+1
;
         JSR INCZPG   ;Next filename.
         LDX RDONE
         CPX COUNT    ;Finished read?
         BCS WRITE    ;Yes, now write.
;
         INC RDONE    ;No, increment
         LDX RDONE    ;counter and
         JMP LP4      ;keep reading
;
;
NEND     LDX RDONE    ; Only part of
         INC PART-1,X ;file read.
         LDA ICLEN+16 ;Save file
         STA LLO-1,X  ; length.
         LDA ICLEN+17
         STA LHI-1,X
         INC FLAG     ;Set flag.
;
WRITE    LDA # <MS4   ;Ask for dest'n
         LDX # >MS4   ;disk.
         JSR PRINT
         LDA # <BUFF  ;And wait for
         LDX # >BUFF  ;user to press
         JSR INPUT    ;[RETURN].
;
         LDA #2       ;Use channel 2
         STA CHAN     ;for write.
WMORE    INC WDONE    ;(# written)
         LDX WDONE
         JSR ZLOOP    ;Set FNAME pntr.
         LDX WDONE
         LDA EFLG-1,X ;Err from read?
         BNE WCLOSE   ;Yes,skip write.
;
         LDA PART-1,X
         CMP #2       ;Part written?
         BCS WOVER2   ;Yes, dont open.
;
         LDA #OPOUT   ;Open new file
         JSR OPEN     ;for output.
         BPL WOVER2   ;Error on open?
;
ER2      JSR ERROR    ;Disp err msg,
         LDX WDONE    ;and set flag.
         INC EFLG-1,X
;
WOVER2   LDX WDONE
         LDA EFLG-1,X ;Any error?
         BNE WCLOSE   ;Yes skip write.
;
         LDA SLO-1,X  ;Point to start
         STA ZP2      ;of file data.
         LDA SHI-1,X
         STA ZP2+1
         LDA LLO-1,X  ;Get saved
         STA FLEN     ;file length.
         LDA LHI-1,X
         STA FLEN+1
;
         LDX #2*16    ;Send the bytes
         LDA #CPBIN   ;to the disk.
         JSR PUTGET
         BMI ER2      ;Error on write?
;
WCLOSE   LDX WDONE
         LDA PART-1,X
         BEQ WCL
         CMP #3       ;Still part
         BCC NCLOSE   ;read/write?
         LDA #0       ;No, finished.
         STA PART-1,X
WCL      JSR CLOSE2
;
NCLOSE   LDX WDONE    ;Wrote all files
         CPX RDONE    ;that were read?
         BCS WEND     ;Yes.
         JMP WMORE    ;No, write more.
;
WEND     LDA PART-1,X
         BNE BAK
         CPX COUNT    ;Finished?
         BEQ FINI     ;Yes.
BAK      JMP SRC      ;No, keep going.
;
ABORT    LDA # <MS7   ;BREAK key abort
         LDX # >MS7   ;message.
         JSR PRINT
;
FINI     LDA # <MS5   ;Ask if more
         LDX # >MS5   ;files to copy.
         JSR PRINT
         LDA # <BUFF  ;Accept input.
         LDX # >BUFF
         JSR INPUT
         BMI FINI     ;Retry if error.
         LDA BUFF
         CMP #'Y      ;More to copy?
         BNE EXIT     ;No.
         JMP INIT     ;Yes, restart.
EXIT     JMP (DOSLOC) ;Indirect jump.
```

```
;
; Establish pointer to FNAMES
;
ZLOOP    LDA # <[FNAMES-16]
         STA ZPG
         LDA # >[FNAMES-16]
         STA ZPG+1
LP5      JSR INCZPG
         DEX
         BNE LP5
         RTS
;
; Increment pointer to FNAMES
;
INCZPG   LDA ZPG
         CLC
         ADC #16
         STA ZPG
         BCC INCX
         INC ZPG+1
INCX     RTS
;
; Display error message
;
ERROR    STY FRØ        ;Store error #
         LDA #Ø         ;in floating
         STA FRØ+1      ;point register.
;
         JSR IFP        ;Convert to
         JSR FASC       ;ASCII format.
;
         LDY #Ø
EL1      LDA (INBUFF),Y
         BMI EL2        ;Copy error #
         ORA #128       ;to message.
         STA ERR+35,Y
         INY
         BNE EL1
EL2      STA ERR+35,Y
;
         LDA # <ERR     ;Display error
         LDX # >ERR     ;message...
         JSR PRINT
         LDA ZPG        ;with name
         LDX ZPG+1      ;of file.
;
; Display message to screen
;
PRINT    STA ICADR
         STX ICADR+1
         LDX #Ø
         LDA #CPTXT
         STA ICCMD
         LDA #255
         STA ICLEN
         STX ICLEN+1
         JMP CIO
;
; Message to display error
; (space reserved for filename)
;
ERR      .BYTE $7D,"                    "
         .BYTE "Not copied
         .BYTE +$80,"ERROR ØØØ"
         .BYTE $1C,$1C,EOL
;
; Puts/Gets bytes to/from disk
;
PUTGET   STA ICCMD,X
         LDA ZP2
         STA ICADR,X
         LDA ZP2+1
         STA ICADR+1,X
         LDA FLEN
         STA ICLEN,X
         LDA FLEN+1
         STA ICLEN+1,X
         JMP CIO
;
; Close file 1 or 2
;
CLOSE1   LDX #$10
         BNE CLOSE
CLOSE2   LDX #$20
CLOSE    LDA #CCLOS
         STA ICCMD,X
         JMP CIO
;
; Accept input from keyboard
;
INPUT    STA ICADR
         STX ICADR+1
         LDX #Ø
         LDA #CGTXT
         STA ICCMD
         LDA #255
         STA ICLEN
         STX ICLEN+1
         JMP CIO
;
; Same as BASIC
; OPEN #CHAN,MODE,Ø,BUFF
;
OPEN     STA MODE
         LDA CHAN
         ASL A
         ASL A
         ASL A
         ASL A
         TAX
         LDA #COPEN
         STA ICCMD,X
         LDA MODE
         STA ICAUX,X
         LDA #Ø
         STA ICAUX+1,X
         LDA ZPG
         STA ICADR,X
         LDA ZPG+1
         STA ICADR+1,X
         JMP CIO
;
         *=  *+256      ;(Safety buffer)
;
; Useable memory starts here
;
MEM      =   *
;
         *=  $2E2
         .WORD INIT     ;(Autorun file)
         .END
```

EXTENDED CALCULATIONS

16K Cassette or Disk

by Donny Cherf

I'm sure that there have been many times when you've needed to figure up some arithmetic calculations that have exceeded the maximum length your computer will accurately display. And then, you end up with an answer similar to *4.8374953E23*. This is the computer's (and most calculators') way of handling these large numbers.

For those of us who have never seen this type of representation, it's called scientific notation and, in this example, it means *4.8374953 times 10 to the 23rd power*, or *483,749,530,000,000,000,000,000*. All you have to do is move the decimal point over twenty-three places and put 0s in the spaces.

It happens to be a very convenient way to handle large numbers but, unfortunately, it's not accurate toward the latter part of the number. Sure, the first digits are accurate, but what about the last ones? How it is possible to arrive at a 24-digit number that is accurate to the final digit?. . .especially on computers that give only *10*-digit accuracy, not to mention hand calculators?

Well, it is possible. One method, which I'm sure many of you have tried at one time or another, is to actually hand multiply (gasp!) the equation. And how many out there are *that* sure of their multiplication of these numbers that they need never go back and recheck their work? So you really end up multiplying the two numbers twice.

Another method.

An easier way is to multiply the two numbers in segments that will not surpass the computer's readout. Here is an example.

Presume that you want to multiply two 8-digit numbers accurately to the last digit. We'll let the two numbers be 86273482 and 54856358. First, let's set these up as we would a multiplication problem.

$$\begin{array}{r} 86273482 \\ \underline{54856358} \end{array}$$

There is the equation. I'll perform this as if the computer or calculator has only an 8-digit display. It must be agreed by all that an *n*-digit number multiplied by an *m*-digit number yields a number that is *n* times *m* digits long (or *n* times *m* minus 1, if the numbers don't carry over).

To clarify, a 4-digit number times a 5-digit number will produce a 9-digit number at most or, perhaps, an 8-digit number if the multiplication doesn't carry over into the ninth digit. Try it if you're having difficulty understanding.

Since we've seen that a 4- and a 5-digit number yield a result that's already over the 8-digit limit, we'll set our working limits to 4-digit segments.

Let's rewrite the equation now, with periods separating the numbers into 4-digit components.

$$\begin{array}{r} 8627.3482 \\ \underline{5485.6358} \end{array}$$

It's easy from here on out. All we have to do is multiply the segments together while keeping track of where to place them below the line.

First, multiply the two rightmost segments together. So that's 6358 (from the lower number) times 3482 (from the upper number). That yields 22138556. Now, we simply position it under the line.

```
    8627.3482
    5485.6458
    ---------
    2213.8556
```

As you can see, I'm going to leave the numbers underneath separated by carets, also. Next, let's multiply the lower right segment, 6358, times the upper left one, 8627. This gives us 54850466. All we have to do now is figure out where it goes. This isn't difficult, either.

When we multiplied 6358 times 8627, we were actually multiplying 6358 times 86270000. We left the 0s out of the equation, because our computer's display can hold only eight digits, supposedly. By eliminating these 0s temporarily, we are able to multiply this accurately.

So now, for visibility's sake, and since we already have the result figured up, let's put those 0s back. We take 54850466 and attach the four 0s to the end, giving us 548504660000.

```
         8627.3482
         5485.6358
         ---------
         2213.8556
    5485.0466.0000
```

Now let's multiply the lower left segment, 5485, by the upper right segment, 3482. This yields 19098770. And, just as I explained above, the lower number is actually 34820000, so let's attach the 0s to the end of the result again. We get 190987700000.

```
         8627.3482
         5485.6358
         2213.8556
    5485.0466.0000
    1909.8770.0000
```

For our last multiplication, we take 8627 times 5485 to arrive at 47319095. This time, though, both numbers have extra 0s that we've ignored for the multiplication. The two numbers are actually 86270000 and 54850000; the result is—gulp!—4731909500000000. We have to add all eight 0s to the end of our result, so the calculation will be correct. Let's place that into our problem now.

```
              8627.3482
              5485.6358
              ---------
              2213.8556
         5485.0466.0000
         1909.8770.0000
    4731.9095.0000.0000
```

(continued on next page)

(continued from page 31)

Have you noticed that this is very similar to regular multiplication, except that we're doing groups of digits instead of single digits?

Finally, all that's left to do is add up all the digits as if this were a normal multiplication problem—which it is, pretty much.

```
              8627.3482
              5485.6358
              ---------
              2213.8556
         5485.0466.0000
         1909.8770.0000
    4731.9095.0000.0000
    -------------------
    4732.6490.1449.8556
```

That's it. We now know that 86273482 multiplied by 54856358 equals 4,732,649,014,498,556, accurate to sixteen decimal places. And, once you know what to do, it isn't that difficult to accomplish.

Even better.

So where is all this leading us, you might ask. Well, there is an easier method to multiply out these huge, sometimes massive numbers.

Oh, no! Another lecture! Ah, but this example will be short. Just type in Listing 1, NUMMULTR (remember to SAVE it) and RUN it. At the prompts, enter both numbers to be multiplied and let the computer do the rest.

When it has finished its computations, it will print the answer—accurate to the last digit. It will only accept three lines of input, which limits your numbers to 119 digits each, giving a result 237 or 238 digits long. But, for most, this will be enough.

For equations that need more, you'll have to write up a routine for inputting additional digits with more prompts, plus you must dimension the strings A$ and B$ to higher values. The maximum number of digits that the final result can be will depend on how much memory you have. With the 48K Atari 800, the maximum is about 28000 digits.

Listing 2 is similar in nature. It's useful for the many times that you've needed to simply double numbers into infinity. Entitled NUMDUBLR, it has the ability to (beginning with *1*) double itself until it is a 28000-digit (or more) number.

This program works by placing a *1* in the very last position of a string and then doubling the contents of the string, one character at a time, from the end of the string to the beginning. It pulls each digit out of the string, doubles it, checks to see if there is a carry, and then places the result back into the same string.

It does have an added routine that's necessary to check if a *1* is carried over to a 9. If so, then it will

WHAT IS CHECKSUM DATA?

Most program listings in **ANALOG Computing** are followed by a table of numbers appearing as DATA statements, called "CHECKSUM DATA." These numbers are to be used in conjunction with **D:CHECK** and **C:CHECK** (which appeared in **ANALOG Computing** issue 16 and the **ANALOG Compendium**) or with **Unicheck** (from issue 24).

D:CHECK and **C:CHECK** (written by Istvan Mohos and Tom Hudson) and **Unicheck** (by Tom Hudson) are designed to find and correct typing errors when readers are entering programs from the magazine. For those readers who would like copies of these articles, you may send for back issue 16 or 24 ($4.00 each) or the **ANALOG Compendium** ($14.95 plus $2.00 shipping and handling from:

ANALOG Computing
P.O. Box 615
Holmes, PA 19045

continue to check to see if it carries over to any more 9s.

Ad infinitum (or close).

And that's it. You can now amaze your friends by performing calculations more immense (almost) than can be imagined. □

Listing 1.

```
100 DIM A$(119),B$(119),T$(9)
110 INPUT A$,B$
120 A=LEN(A$):B=LEN(B$):S=A+B
130 DIM S$(S)
140 S$="0":S$(S)="0":S$(2)=S$
150 A1=A-3:IF A1<1 THEN A1=1
160 B1=B-3:IF B1<1 THEN B1=1
170 T=VAL(A$(A1,A))*VAL(B$(B1,B))
180 T$=STR$(T):T=LEN(T$)
190 T$=STR$(VAL(T$)+VAL(S$(S-T+1,S)))
200 K=1:IF LEN(T$)>T THEN K=2
210 S$(S-T+1,S)=T$(K)
220 IF K=1 THEN 280
230 K=0
240 C=0:V=VAL(S$(S-T-K,S-T-K))+1
250 IF V=10 THEN V=0:C=1
260 S$(S-T-K,S-T-K)=STR$(V)
270 IF C=1 THEN K=K+1:GOTO 240
280 IF B1>1 THEN B=B-4:S=S-4:GOTO 160
290 IF A=1 THEN ? "Answer: ";S$:END
300 B=LEN(B$):A=A-4
310 S=LEN(S$)-LEN(A$)+A:GOTO 150
```

•

CHECKSUM DATA.

(see page 32)

```
100 DATA 84,80,553,795,105,645,657,991
,248,795,643,808,539,212,902,8057
250 DATA 108,658,836,860,824,620,967,4
873
```

•

Listing 2.

```
10 REM C=(MAXIMUM OF MEMORY MINUS 50)
20 REM DELETE ALL REMS BEFORE ? FRE(0)
30 REM TO FIND MAXIMUM MEMORY
35 C=FRE(0)-50:IF C>32767 THEN C=32767
40 DIM A$(C):A$="0":A$(C)="0":A$(2)=A$
:A$(C)="1":J=0:K=0
50 REM ENTIRE DOUBLING ROUTINE
60 FOR I=0 TO J:A=2*VAL(A$(C-I,C-I))+K
:K=0:IF A<10 THEN 80
70 A=A-10:K=1:IF I=J THEN J=J+1:A$(C-I
-1,C-I-1)="1"
80 A$(C-I,C-I)=STR$(A):NEXT I:K=0:? A$
(LEN(A$)-J,LEN(A$)):IF J<=C THEN 60
```

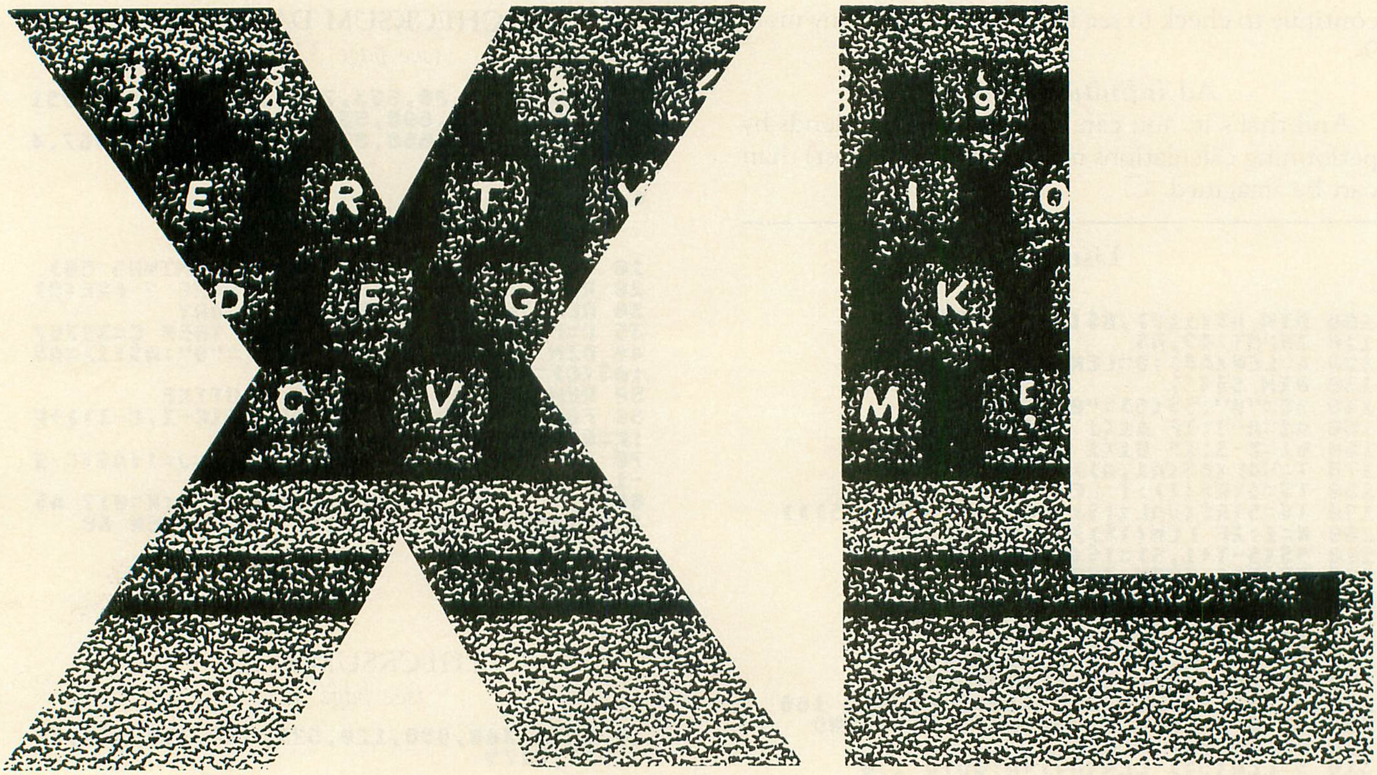
•

CHECKSUM DATA.

(see page 32)

```
10 DATA 340,898,120,521,937,354,107,98
9,913,5179
```

•



COMPATIBILITY

by Dwight Stanley

When Atari released their new XL line of computers, they gave us a tremendous gift, namely "upward compatibility." Contrary to the practice of some computer manufacturers, Atari chose to recognize the retail base of existing Atari owners. I'd like everyone to stand up with me and applaud Atari on their insight for this compatibility.

Hey! You're not standing. How come? You say you know of a program that won't run on the XLs. You say you're angry at Atari and that they let you down! Now, wait just a minute—you're making *me* angry.

Why should I be? I own an 800 and have never owned an XL. My ire is up because I believe that: we have the best personal computer on the market; Atari has, in fact, honored us in making the XLs compatible; and most problems of incompatibility are the fault of programmers, not Atari. Do you realize that Atari is one of the few personal computer manufacturers that has released documented source codes for the Operating System and DOS?

Now, how do you feel? More pleased with Atari, I hope. In this article, I want to pass on some information to make your machine language programming compatible with the XLs.

I'm not a professional programmer, nor do I profess to know all. However, by being an active computer hobbyist for over four years on the Atari computers, I have gathered bits and pieces of information from hundreds of magazines and books. From countless hours of reading, I have found Atari's basic Rules of Compatibility. They are:

(1) Respect all memory below page 7 ($0700), and

(2) Make no illegal calls to the OS.

First, let me say that the majority of problems with incompatible software is the result of sloppy programming. I know of many programs that have made one or two stupid jumps to the OS which aren't compatible. I also know of many programs that abuse lower memory and also wind up incompatible. So, what can be done?...

. . .*with incompatible commercial software?* You should write to the manufacturer of the software and let them know how you feel. Unless you speak up, you won't be heard.

. . .*with incompatible magazine software?* I believe magazines should be responsible for ensuring compatibility of their programs. Most magazines are young and need a solid reader base. They will surely lose out if they ignore the XLs.

. . .*with incompatible public domain software?* Take the software to your user's group or club. Either with a more experienced member or even the whole group, look at the software. Find what is wrong, fix it and then send a letter to your favorite magazine to help others.

For budding machine language programmers, I suggest you read the following and make sure you understand it. Also, purchase a copy of the *Atari Technical Users Notes* and/or *De Re Atari*. If you can gain just one piece of information, they're well worth it.

When you make your program—and it works—take it to a store, club or friend and try it on an XL. Don't quit until it works. I guarantee you'll appreciate it!

I cannot stress how important it is to all of us that we recognize the XLs and do all we can to help our new brothers and sisters succeed. Each and every one of us is responsible for the future of Atari and our hobby. If we lose just one person who throws up their hands and says, "Forget it—Atari isn't what it says it is," then we all will suffer.

And, who knows. . .soon many new hardware and software goodies for the XLs could leave us oldtime 400 and 800 owners in the *incompatible cold.*

Respect all memory below page 7 ($0700).

According to the *Atari Technical Users Notes*, page 0 addresses below $80 are reserved for computer use. That leaves 128 bytes for your use, which *should* be enough. If you need more of page 0, I suggest that less frequently used variables be stored in higher addresses and moved to page 0 when needed.

Although there may seem to be empty locations, stay out; they may not be empty for long. Some programmers have used seemingly empty locations in page 2 ($0200). Most of these locations are used in the XLs. Stay out!

Even more locations seemed empty above the device handlers in page 3 ($0300), but guess what? Ah, you guessed! They're now used, too. Page 4 is still an input buffer, and page 5 is still reserved.

Now, page 6. It's still reserved as a somewhat-free RAM. However, use it only if needed. That still leaves $0700 to $BFFF for your program. . .whew!

Make no illegal calls to the OS.

If you see a routine in the OS that you just *have* to use, my best advice is to rewrite the code in your program. I've seen many programs, even in magazines, go bye-bye on the XL for less than 20 bytes. Those could have easily been included by the programmer. *Less than 20 bytes!* What difference could that be in your program?

That doesn't mean you can't use the OS to help your program; just do it right. Below I'll explain how. Please make sure you understand it.

There are many vectors into the OS. A vector is an address guaranteed not to change. So far, none have. There are two types of vectors, and both are easy to use.

The first group is straight jump vectors. You can simply jump to the vector, and it will jump to the appropriate code. For example:

```
LDA #$05   ;This could be
STA THIS   ;your code
JMP $E477  ;Go do coldstart
```

Control will be passed to different locations on the old 800s than it will on the XLs, but we don't have to care.

Some examples of jump vectors are:

DISKINV	$E453	—	disk interface
CIOV	$E456	—	central input-output vector
SIOV	$E459	—	serial input-output vector
SETVBV	$E45C	—	routine for setting VBI vectors
SYSBVB	$E45F	—	VBI routine vector
WARMSV	$E474	—	warmstart vector
COLDSV	$E477	—	coldstart vector
XITVBV	$E462	—	exit VBI vector

The second set of usable guaranteed vectors is the device handler routines. These may be used to get a key from the keyboard or to print a letter to the screen, etc.

All handlers are 16 bytes ($10) in length and start at $E400. Each handler has eight entries in the following order:

$E400	—	$E401	OPEN VECTOR
$E402	—	$E403	CLOSE VECTOR
$E404	—	$E405	GETBYTE VECTOR
$E406	—	$E407	PUTBYTE VECTOR
$E408	—	$E409	GETSTAT VECTOR
$E40A	—	$E40B	SPECIAL VECTOR
$E40C	—	$E40E	INIT VECTOR (JUMP)
$E40F			FILLER BYTE

Above is the ROM location for the (E:) editor device. The base addresses for other handlers are as follows:

E:	—	$E400	Editor device
S:	—	$E410	Screen device
K:	—	$E420	Keyboard device
P:	—	$E430	Printer device
C:	—	$E440	Cassette device

Okay? Now, the method of using these vectors that I'll show you still isn't the proper way of doing things, but it will make your programs XL-compatible, so it's better than illegal jumps to the OS.

An example of use would be in a situation where you wish to get a key from the keyboard. The base address is $E420, and the GETBYTE routine is +$04. So we'll use the vector at $E424 - $E425. The key pressed will be returned in the A register.

```
10 ;This would be your program
20 ;code and we want to go get a key
30 ;
40    JSR GET.KEY
50    BRK          ;your code would
60                 ;continue here
70 ;
100 GET.KEY
110    LDA $E425   ;This routine prepares
120    PHA         ;to jump into the O.S.
130    LDA $E424   ;by loading vectors
140    PHA         ;on the stack to jump
150    RTS         ;NOW GOTO O.S. vector!
160                ;this will return to
170                ;your code at line 50
```

Please note the GET.KEY subroutine. This loads the vectored address into the stack, and when the program gets to the return from subroutine, it "returns" to the appropriate code in the OS. When the OS routine issues a return from subroutine, control is passed back to your program. If it doesn't make sense, consider the following BASIC routine:

```
10 GOSUB 100: REM get.key
20 STOP
100 GOSUB O.S.
110 RETURN
```

This is exactly what happens in the machine language code and is intended only to clarify the routine. These vectors are best used from the machine language level only.

Other uses would be to get a key from the E: device (editor), or put a byte to the E: device. You could also put or get a byte from the S: device (screen), or put a byte to the P: device (printer). All of these uses pass information using the A register.

I've seen all of these locations improperly used, where the above method of using the vectors would have made the program XL-compatible.

ATASCII conversion, etc.

A few other notes before I finish. I've seen programs use an OS table which is located at $FEFE on the old 800s for ATASCII conversion. Once again, these programs do not function properly on the XLs, because this table has been moved.

Now, this table is 192 bytes long, so I could see why a programmer wouldn't *like* typing this into his work. However, if it had been included, there would be no worries of incompatibility. Atari has recognized the need for a vector for this table and has included it on the XLs as a pointer in $79-$7A.

This means that programs which used codes like *CMP $FEFE,Y* can be written as *CMP ($79),Y.* A routine may be added to see if the addresses in $79 and $7A are zero (as in an 800) and, if they are, then stuff

$FE in both addresses. This program would run on either machine.

The last tidbit I've found is a way to load a small machine language program from cassette (magazine programs, for example) on the XLs, without having control passed to BASIC. You don't have to hold the OPTION key, only the START key. This header will not remove BASIC, but control will be passed to the tape program and co-reside with BASIC in 40K.

```
10 LOCATION
20    .BYTE 0
30    .BYTE SIZE ;# of 128 byte blocks
40    .WORD LOCATION
50    .WORD INIT
60 ;
70    LDA #0
80    STA #$06
90    STA #$07
100   LDA #<START
110   STA $0A
120   STA $0C
130   LDA #>START
140   STA $0B
150   STA $0D
160   CLC
170 INIT
180   RTS
190 ;
200 START
210 ; Program continues
220 ; from here on
```

So there it is, in a nutshell. I hope you'll make a serious effort to support the XLs and Atari. □

24K Cassette or 32K Disk

DRAGONLORD

by Clayton Walnum

It was bound to happen sooner or later. The reports had been coming in from all over the countryside... livestock slaughtered, homes burned, princesses kidnapped. But the most horrifying tales of all, those that left one numb with shock, told of the near extinction of blueberry pie. From town after town came ghastly accounts of the mouth-watering treats being snatched from windowsills as fast as they were set out to cool. The county was in an uproar.

And now the dragon had come to Dellwood.

The mayor tapped a finger thoughtfully on his forehead and tried to keep a smile frozen on his lips. All around him, angry citizens stomped and frowned.

"I lost two prize cattle!" bellowed Babbit Costowitz.

"My barn burned to the ground!" yelled Loodchuck Allison.

"My hens stopped layin'," moaned Chip Monk.

"And I ain't makin' no more blueberry pies!!" screamed Widow Taccoon. The town hall plunged into silence so suddenly that the windows rattled in their frames. No more blueberry pie?

"Can't help it," mumbled the widow, acutely aware of all the horrified eyes focused on her. "He keeps stealin' 'em!"

All attention turned to the mayor. He stood, still smiling, his demeanor confident. The people visibly relaxed. Here was a great leader of men. Here was the town guardian, unwavering in strength, unsurpassed in courage, a veritable fount of knowledge. He gazed into his people's hopeful faces and spoke profound words of wisdom and encouragement.

"I...uh...well...hmmmmmm...Any suggestions?"

*"Call the **Dragonlord**!"* they yelled in unison.

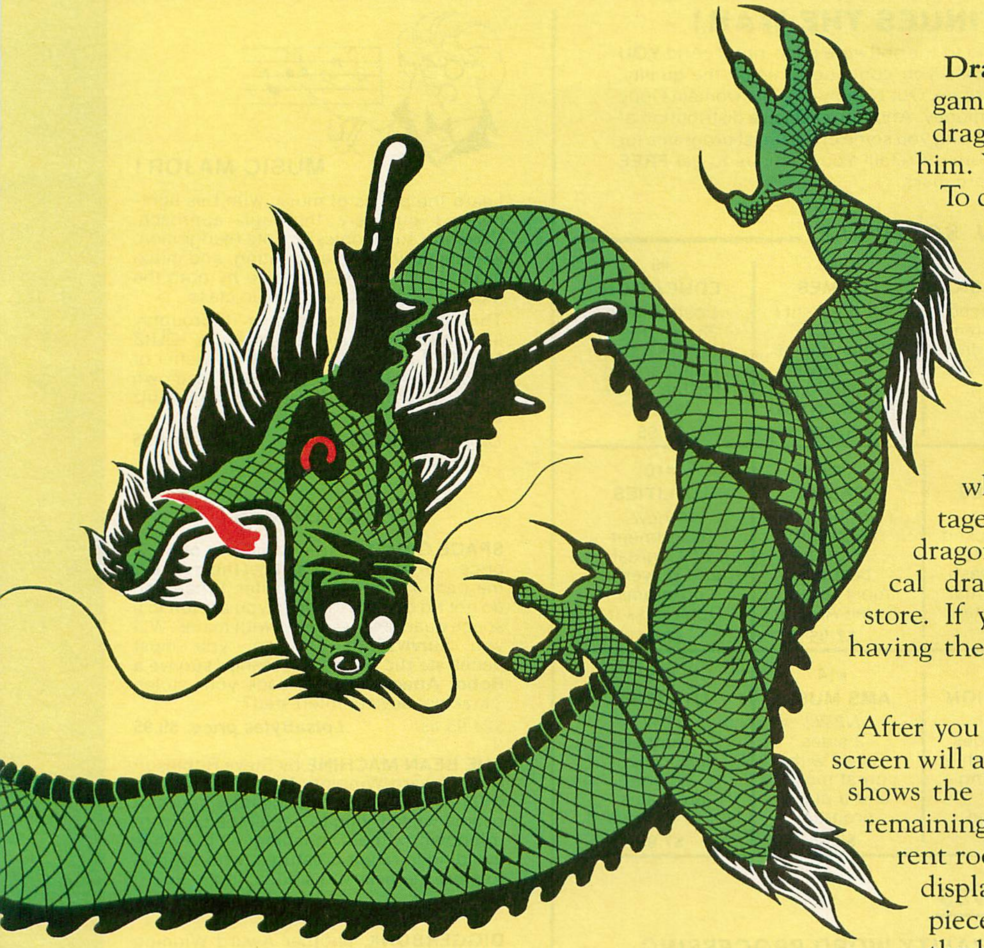
"Uh...oh, yes...of course!...Hmmmmmm."

It wasn't too long before the **Dragonlord** arrived in Dellwood with nothing but the armor on his back, a skin of spring water and a week's supply of blueberry pie. Within the hour, he was standing with the mayor in Babbit Costowitz's meadow examining a rather large mound of dragonflop.

"Yep, you got yourself a big'un here," he said as he tucked away his tape measure. "Judging by the size of this, I'd say..."

But the mayor wasn't listening. His attention was fixed on a huge object that was plummeting from the sky at an alarming rate. "Good grief! It's the dr... dr...dr...dr..."

"Dragon," finished the hero, while helping the poor



man reclose his jaw. "Don't you worry. I've got quite a reputation. Most dragons would rather just buzz off than tangle with me."

He strode to where the colossal beast was settling to the ground amidst a whirlwind of dust and straw, and looked up into its emerald, catlike eyes. "Say, why don't you just get lost before I have to get rough with you?"

"*Ha!*" snorted the dragon.

Of course, you know what happens when a dragon snorts. It took the village blacksmith three days to peel off the **Dragonlord**'s welded armor.

"The doc said it will take a month for your burns to heal," said the mayor.

"I can live with that," said the **Dragonlord**.

"Your armor is completely ruined."

"I can live with that."

"Every hair on your body has been burned off."

"I can live with that."

"He took your blueberry pie."

"*What?* Why, that confounded dr...dr...dr...dr..."

"Dragon," finished the mayor, not unsarcastically, while helping the fried hero reclose his jaw.

But **Dragonlords** are tough. With sixty gold pieces from the town treasury, he soon set off to shop for supplies. He was going dr...dr...dr...dr...

(Dragon!)

(Thanks.)

...dragon hunting.

Playing Dragonlord.

Dragonlord is a fantasy adventure board game for one player. The object is to find the dragon in the dungeon maze, then capture him. You must manage to stay alive, of course. To do this, you mustn't allow your hit points or strength to drop to zero. You must make sure that you have plenty of pie to eat, and you must fight and slay the many orcs that will try to keep you from your goal.

Throughout the dungeon you will discover spells, serums, treasures and teleporter devices. There is also a thief who is more than happy to take advantage of unwary adventurers. To capture the dragon, you must tame him with the magical dragon brew you can purchase in the store. If you stumble upon the dragon without having the brew, he'll kill you instantly.

The status screen.

After you enter your character's name, the status screen will appear. The top left portion of the screen shows the number of hit points, strength and pie remaining. At the bottom left, you'll see your current room and all available exits. The top right displays the number of spells, serums and gold pieces you're carrying. The small window at the bottom right indicates if you are carrying dragon brew. It will turn blue when you have purchased the concoction.

At the bottom center is the command window. Use the joystick to move the cursor, then press the trigger to finalize your choice.

Movement.

When you choose MOVE, the four main compass directions will appear in the command window. Use the joystick to pick the desired direction, then press the trigger. The screen will change to show the room you have moved to, as well as any item you may have found.

Dragonlord.

If there's an orc in the room, a scoreboard will be drawn and a die will start rolling. When you press your trigger, the die will stop and your score will appear. The score is based on the roll of the die, your weapon and your strength.

A second die will then appear. This is the orc's attack. The computer will stop this die and show the orc's score. One to five hit points are subtracted from the loser's score. Note that, each time you roll, one

LOTSABYTES CONTINUES THE WAR!

WAR on high prices! We're going to put an end to the software price 'ripoff'. And **YOU** can help! Just keep those orders coming while you continue to enjoy the **quality, quantity, selection** and **low prices** that you deserve. Our National Public Domain Copy Service will save you time, tedious work, and money. And our **exclusive** distribution of **sharply discounted** commercial programs will bring you some of the finest programs for the lowest possible price, usually 50% and more off retail! You continue to get **FREE BONUSES** with each purchase of three or more disks.

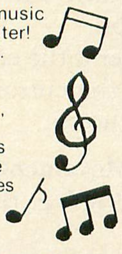
PUBLIC DOMAIN SOFTWARE

#1 **GAMES**	#2 **UTILITIES**	#3 **AMS MUSIC**	#4 **GAMES**	#5 **EDUCATION**
Two full disk sides packed with over 25 games including some Arcade quality. **$7.95**	25 powerful programs to help you get the most out of your Atari computer. **$7.95**	25 Advanced Musicsystem files including a new Player program. 2 sides. **$7.95**	All different! 14 more better games on 2 disk sides. Some Arcade types. **$7.95**	Loaded with 28 programs on 2 disk sides Fun learning for the whole family. **$7.95**
#6 **AMS MUSIC**	#7 **GAMES**	#8 **UTILITIES**	#9 **GAMES**	#10 **UTILITIES**
25 all-time favorites with a Player program. Two sides. **$7.95**	Two disk sides packed with 14 more great games. Some Arcade types. **$7.95**	17 more power-packed utilities to help unleash full potential of your Atari. **$7.95**	Two full sides filled with 17 of the best and most recent. Some Arcade. **$7.95**	A new assortment of 17 great and powerful programs Don't miss it! **$7.95**
#11 **GAMES**	#12 **ADVENTURES**	#13 **EDUCATION**	#14 **AMS MUSIC**	#15 **UTILITIES**
NEW! Our newest. 2 sides filled with great games. **$7.95**	*NEW!* 2 full disk sides filled with text adventures. **$7.95**	*NEW!* 2 disk sides filled with something for everyone. **$7.95**	*NEW!* 2 sides filled with great music and a player program. **$7.95**	*NEW!* Another assortment of fine programs. Not to be missed. **$7.95**

LotsaBytes EXCLUSIVES

ADVANCED MUSICSYSTEM II
by LEE ACTOR

Allows you to create music with your Atari computer! All new machine code.

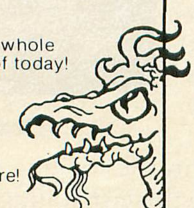


* Control over pitch duration, envelope dynamic level, meter, tempo and key.
* 4 independent voices
* 5½ octaves per voice
* Save up to 8200 notes
* Custom DOS
* FULL instructions
* 24K disk

Originally $29.95 **Only $14.95**

ORIGINAL ADVENTURE
by Bob Howell

For all Atari computers. The Original Colossal Cave Adventure faithfully reproduced from the 'main-frames'. This is the one that launched the whole Adventure craze of today!



* Two mazes
* 130 rooms
* Deadly Dragons
* Nasty Dwarves
* Tenacious Troll
* The Pirate & More!
* 86 coded hints
* SAVE/RESUME
* 40K disk or 32K tape

Originally $24.95 **Only $14.95**

QUALITY WORD PROCESSING

ESI WRITER! At last a brand-new Word Processor that has more features and is easier to use than anything else available for the Atari. Easy for the beginner to use, it asks questions and remembers the answers. ESI WRITER is so sophisticated that it has many more features we don't even have room to mention! Works with ANY Atari.

* Reads any text file * Built in Help screen
* Very fast! * Works with ANY printer
* Instant top, bottom or text location without scrolling!
* Every printer feature * DISK ONLY (Any Atari)
* Search and replace * Block move text
* Page eject/start * Set margins/lines etc.
* Full justification * Print headers etc.
* Block delete etc. * Change video color
* Over 50 pages of docs and tutorials

TRUST US ON THIS ONE! YOU WILL LOVE IT!

Originally $49.95 ***LotsaBytes* price $19.95**

** * FREE BONUSES * **

Now for each 3 disks ordered you may choose any 1 of the following disks FREE!!

... buy 3 - get 1, buy 6 - get 2, buy 9 get 3 ...

a. The Atari XL TRANSLATOR DISK that enables XL owners to use most 400/800 software. **FREE!!**

-- or --

b. An all different AMS MUSIC disk with Player. **FREE!!**

--or --

c. Your choice of one of the P.D. disks -- #1, #2, #3, #4, #5, #6, #7, #8, #9, or #10 (specify one) **FREE!!**

MUSIC MAJOR!

Learn the basics of music with this light-hearted but very thorough approach. Covering such topics as note recognition, key signatures, note counting, and much more, it is designed for use by both the individual student and music class.

This program includes a thoroughly illustrated manual and offers a QUIZ MASTER utility that allows the teacher or the self-taught student to create their own A-B-C-D type tests, with a sample quiz included.

Originally $39.95 ***Only $14.95***

* * * *

GREAT GAMES!

SPACE GAMES: Three games for **one** low price!. In **Aliens** you can't get them all and the pace keeps getting faster. When you do get rid of most of them, you are left in a space quadrant peppered with mines. Will you **Survive**? If you do, you must penetrate the alien's spaceship, survive a **Robot Attack**, and get back your stolen 'cloaking' device! Interested?

$24.95 list ***LotsaBytes price:* $9.95**

THE BEAN MACHINE by Steve Robinson is an Award Winning Arcade game that will drive you crazy balancing a series of beams while trying to get all the beans to roll down, without touching, all the while avoiding 'strange creatures' who drop in to steal the beans. It's addicting!

$24.95 list ***LotsaBytes price:* $9.95**

DIGGERBONK, another Award Winning game by Steve Robinson, challenges you to find your way through a continuously scrolling maze while avoiding some really strange creatures. Along the way you will need to Bonk some of them, but watch out for the bombs.

$24.95 list ***LotsaBytes price:* $9.95**

GUESS WHAT'S COMING TO DINNER lets you try to manuever a snake through 7 levels if you can keep it from starving or being electrocuted. Lots of surprises! One or two players.

$24.95 list ***LotsaBytes price:* $9.95**

* * * *

CREATIVE LEARNING ADVENTURES

Ages 4 to 10 — Disk only

1. Hours of **educational** fun playing 3 exciting creative adventures with a friendly alien learning about our planet Earth. Hand/eye co-ordination, drawing, and music skills are emphasized.

$24.95 list ***LotsaBytes price:* $12.95**

2. Four challenging learning games that are the favorites of our friendly alien. Helps your child to develop logical reasoning ability.

$24.94 list ***LotsaBytes price:* $12.95**

3. These 3 Fun-Day learning games will help with intellectual development, hand/eye co-ordination, logic, spatial, and analytical abilities.

$24.95 list ***LotsaBytes price:* $12.95**

LOTSABYTES

Full 100% Replacement guarantee. Any disk found to be defective will be replaced free and we will also refund your return postage. All orders shipped by First Class U.S. Mail. Add $1.95 shipping and handling for 1 to 5 disks. Add $2.95 for 6 to 12 disks. California residents add 6% sales tax. Outside of U.S.A. and Canada add 15%. U.S. Funds only. We accept checks or Money Orders. Sorry, no COD or Charge Cards. Allow three weeks for personal checks to clear.

15445 Ventura Blvd., Suite 10H, Sherman Oaks, CA 91413

CIRCLE #122 ON READER SERVICE CARD

strength point is lost. There's no way to avoid a battle. You must fight to the death. Each time you kill an orc, you will find some gold.

If you happen upon a teleporter, you'll be magically moved to a randomly-selected room. If you don't have the dragon brew, there is a one in eight chance that the selected room will contain the dragon. And, if you stumble upon the dragon without the brew, you'll find yourself in a very hot situation.

If you bump into the thief, he will steal half your gold.

Besides the above, you may find gold, spells or serums. Serums, when taken, restore a portion of your strength points. Whenever your strength falls below twenty, you will automatically drink one (if you have any on hand).

Each time you move, you lose one strength point.

All items are placed randomly throughout the dungeon each time you start a new game, and will not move as you play. When you enter a room, you'll either pick up any object there or begin the necessary action. When you leave a room, it will be empty. The only exceptions are the teleport rooms. You will be teleported each time you enter one, even if you've already been there before. When you enter an "empty" room, there is a one in four chance that an orc will follow you.

Casting spells.

Casting a spell allows you to move instantly to any room of your choice, with no decrease in strength. If you choose the CAST command (assuming that you have at least one spell), a number will appear in the command window. Use the joystick to increase or decrease the number. When the room number you want appears, press the trigger. You will be magically teleported there.

Note that there are seven rows of eleven rooms each. The rooms are numbered from left to right, starting with room number one in the upper left and ending with room number seventy-seven in the lower right.

The map.

When you are not sure of your whereabouts, or would like to see the rooms you've searched, use the MAP command. Viewing the map is a "free" command. It doesn't decrease your strength. When you are through with the map, press the trigger. You will return to the status screen.

Spending your gold.

The first thing you should do at the beginning of a game is hightail it to the store. Without purchasing at least some pie, you won't last very long in the

(continued on next page)

dungeons. You should be aware, however, that every trip to the store will cost you two strength points.

When you get to the store, you'll see a message scroll across the top of the screen. If you're lucky, there will be a sale in progress! Press the trigger, and the store menu will appear. Choose the department you want. The items available in that department will then appear.

Make your selection and press the trigger. The cost of the item will be subtracted from your gold. The gold you have remaining will appear briefly above the menu window. You may now buy something else or exit the store.

In the magic department, you may purchase spells or dragon brew, or take a trip to see the gypsy. The gypsy will give you a directional clue—your position relative to that of the dragon. This will not only help you find the dragon, but will make it easier to avoid until you can afford the brew.

The health department sells pie, lodging and medical assistance. You need pie to keep up your strength; if it runs out, your strength points will decrease twice as fast. An alarm will warn you when the pie is gone.

If you wish to restore some of your strength, you may take lodging for the night. To restore hit points, go see the doctor.

In the weapons department, you may purchase a dagger, a short sword or a long sword. The dagger will add one point to your attack score; the short sword will add two; and the long sword will add three. You can carry only one weapon at a time.

You may exit a department without making a purchase by choosing the EXIT command. When you are finished shopping, you leave the store in the same way.

Ending the game.

Dragonlord is over either when you capture the dragon or when you're dead, at which time you'll see your score. You get one point for each move, two points for finding a useful item, five points for killing an orc and one hundred points for capturing the dragon. If you captured the dragon, you will also be awarded bonus points based on the number of moves you made. The fewer the moves, the higher the bonus.

Now, go forth and slay the dragon. □

BASIC listing.

```
1 REM     DRAGONLORD
2 REM by Clayton Walnum
3 REM
10 N1=1:N2=2:N3=3:N4=4:N5=5:N6=6:N7=7:
N8=8:N9=9:N10=10
20 N11=11:N12=12:N13=13:N14=14:N15=15:
N16=16:N17=17:N18=18:N19=19:N20=20:N21
=21:N77=77:GOTO 2260
30 RESTORE :FOR X=N0 TO 27:READ A:POKE
 ADR(R$)+N1+X,A:NEXT X
40 FOR X=CHBASE+264 TO CHBASE+479:READ
 A:POKE X,A:NEXT X
50 FOR X=CHBASE+128 TO CHBASE+207:READ
 A:POKE X,A:NEXT X
60 FOR X=CHBASE+24 TO CHBASE+103:READ
A:POKE X,A:NEXT X
70 FOR X=CHBASE+216 TO CHBASE+223:READ
 A:POKE X,A:NEXT X
80 FOR X=N1 TO N77:READ A:RM(X)=A:I(X)
=N0:NEXT X:I(39)=N7
90 READ A:IF A=-N1 THEN 3030
100 X=INT(RND(N0)*N77)+N1:IF I(X) THEN
 100
110 IF A=N1 THEN DR=X
120 I(X)=A:GOTO 90
130 ROOM=INT(RND(N0)*N77)+N1:IF  NOT B
 THEN X=INT(RND(N0)*N8)+N1:IF X=N1 THE
N ROOM=DR
140 FOR X=N1 TO N5:FOR Y=N0 TO N16:SOU
ND N0,Y*N2,Y,N4:POKE 708,Y*N10:FOR A=N
1 TO N5:NEXT A:NEXT Y:NEXT X
150 POKE 559,N0:SOUND N0,N0,N0,N0:GOTO
 950
160 SOUND N0,N10,N12,N8:FOR X=N1 TO N2
0:NEXT X:SOUND N0,N0,N0,N0:RETURN
170 SOUND N0,100,N12,N8:FOR X=N1 TO 10
0:NEXT X:SOUND N0,N0,N0,N0:RETURN
180 RT=RM(R):COL=(R-INT(R/N11)*N11)*N3
:IF INT(R/N11)=R/N11 THEN COL=33
190 ROW=INT(R/N11)*N3:IF INT(R/N11)=R/
N11 THEN ROW=ROW-N3
200 RETURN
210 R=ROOM:GOSUB ROOMPOS:C=COL:RW=ROW:
R=DR:GOSUB ROOMPOS:POSITION 34,N14:Q=N
0:D=N0
220 Q=N1*(COL<C AND ROW<RW)+N2*(COL>C
AND ROW<RW)+N3*(COL>C AND ROW>RW)+N4*(
COL<C AND ROW>RW)
230 D=N1*(COL=C AND ROW<RW)+N2*(COL>C
AND ROW=RW)+N3*(COL=C AND ROW>RW)+N4*(
COL<C AND ROW=RW)
240 IF Q=N1 THEN D=N1:IF RND(N0)<0.5 T
HEN D=N4
250 IF Q=N2 THEN D=N2:IF RND(N0)<0.5 T
HEN D=N1
260 IF Q=N3 THEN D=N3:IF RND(N0)<0.5 T
HEN D=N2
270 IF Q=N4 THEN D=N3:IF RND(N0)<0.5 T
HEN D=N4
280 RETURN
290 ROW=R:POSITION COL,ROW:? ">"
300 IF STRIG(N0)=N0 THEN 300
310 IF STRIG(N0)=N0 THEN GOSUB SND1:CH
=ROW-R+N1:RETURN
320 ST=STICK(N0):IF ST<>N14 AND ST<>N1
3 THEN 310
330 RO=ROW:ROW=ROW+(ST=N14)*-N1+(ST=N1
3)*N1:IF ROW<R THEN ROW=R+N3
340 IF ROW>R+N3 THEN ROW=R
350 POSITION COL,RO:? " ":POSITION COL
,ROW:? ">":FOR X=N1 TO 25:NEXT X:GOTO
310
360 FOR X=N1 TO 300:NEXT X:RETURN
370 N=N0:S=N0:E=N0:W=N0:RT=RM(ROOM)
380 ON RT GOTO 390,400,410,420,430,440
,450,460,470,480,490,500,510,520,530
390 N=N1:E=N1:E$="  ,,":RETURN
400 N=N1:S=N1:E$=" , ,":RETURN
410 N=N1:W=N1:E$=" ,, ":RETURN
420 S=N1:E=N1:E$=",  ,":RETURN
430 S=N1:W=N1:E$=",,  ":RETURN
440 E=N1:W=N1:E$=", , ":RETURN
450 N=N1:E=N1:S=N1:E$="   ,":RETURN
460 N=N1:E=N1:W=N1:E$="  , ":RETURN
470 S=N1:E=N1:W=N1:E$=",   ":RETURN
480 N=N1:S=N1:E=N1:W=N1:E$="    ":RETU
RN
490 N=N1:E$=" ,,,":RETURN
500 S=N1:E$=",, ,":RETURN
510 E=N1:E$=", ,,":RETURN
520 W=N1:E$=",,, ":RETURN
530 N=N1:S=N1:W=N1:E$=" ,  ":RETURN
540 REM *** STATUS SCREEN ***
550 IF STR>N20 OR SM<N1 THEN 580
560 GRAPHICS N17:POKE 756,CHSET:POSITI
ON N1,N6:? #N6;"You drink a serum"
570 STR=STR+INT(RND(N0)*N15)+N15:SM=SM
-N1:GOSUB DELAY2
580 GRAPHICS N0:POKE 559,N0:POKE N77,N
0:POKE 756,CHSET:POKE 752,N1:POKE 709,
N8:POKE 710,N0
```

```
590 DL=PEEK(560)+256*PEEK(561)+N4
600 POKE DL-N1,70:POKE DL+N2,N6:POKE D
L+N21,N6:POKE DL+22,N6:POKE DL+23,N6:P
OKE DL+24,N6
610 POKE DL,N0:POKE DL+N1,STTUS:POKE 8
8,N0:POKE 89,STTUS
620 POSITION N9,N3:? N$:POSITION N9,N5
:? HP;" ":POSITION N9,N7:? STR;" ":POS
ITION N9,N9:? PIE;" "
630 POSITION 30,N3:? W$(WN*N7-N6,WN*N7
);:IF W=N2 THEN ? " "
640 POSITION 30,N5:? SPL;" ":POSITION
30,N7:? SM;" "
650 POSITION 30,N9:? G;" ":POSITION N1
,N20:? "                  "
660 POSITION N10,N14:? ROOM;" ":POSITI
ON N10,N15:? "         ←←←←←←←←←";
670 POSITION N21,N13:? " COMMAND ":POS
ITION 22,N14:? " MOVE":POSITION 22,N15
:? " MAP  "
680 POSITION 23,N16:? "CAST":POSITION
22,N17:? " STORE"
690 GOSUB DIR:POSITION N10,N15
700 IF N THEN ? "N ";
710 IF S THEN ? "S ";
720 IF E THEN ? "E ";
730 IF W THEN ? "W ";
740 IF B THEN POSITION 33,N14:? "[[[["
750 POSITION N21,N19:? "THE MAGICAL ST
ICK ":POSITION N21,N20:? "PICKS THY CO
MMAND":POKE 559,34
760 IF PIE=N0 THEN FOR X=N1 TO N4:FOR
Y=240 TO N0 STEP -N20:SOUND N0,Y,N10,N
4:NEXT Y:NEXT X:SOUND N0,N0,N0,N0
770 COL=22:R=N14:GOSUB CHOOSE:ON CH GO
TO 790,1450,1540,1660
780 REM *** MOVE ***
790 POSITION N21,N13:? "     N    ↓←←←←
←←←     ↓←←←←←←←W   E↓←←←←←←←    ↓←←←←
←  S  "
800 POSITION 22,ROW:? " ":POKE 766,N1:
POSITION 25,N15:? "↑"
810 POSITION N21,N19:? "
   ":POSITION N21,N20:? "
   "
820 POSITION N1,N20:? "PICK THY DIRECT
ION":TURN=TURN+N1:CH=N1
830 IF STRIG(N0)=N0 THEN 830
840 IF STRIG(N0)=N0 THEN GOSUB SND1:PO
KE 766,N0:GOTO 900
850 ST=STICK(N0):POSITION 25,N15:IF ST
=N14 THEN ? "↑":CH=N1
860 IF ST=N13 THEN ? "↓":CH=N2
870 IF ST=N7 THEN ? "→":CH=N3
880 IF ST=N11 THEN ? "←":CH=N4
890 GOTO 840
900 IF CH=N1 AND N THEN ROOM=ROOM-N11:
GOTO 950
910 IF CH=N2 AND S THEN ROOM=ROOM+N11:
GOTO 950
920 IF CH=N3 AND E THEN ROOM=ROOM+N1:G
OTO 950
930 IF CH=N4 AND W THEN ROOM=ROOM-N1:G
OTO 950
940 SOUND N0,150,N12,N8:FOR X=N1 TO N2
0:NEXT X:SOUND N0,N0,N0,N0:POKE 766,N1
:GOTO 840
950 POKE 88,N0:POKE 89,MAP:R=ROOM:GOSU
B ROOMPOS:POSITION COL,ROW:? RM$(RT*N1
7-N16,RT*N17)
960 STR=STR-N1-N1*(PIE=N0):IF STR<N1 T
HEN GOTO DEAD
970 CNT=CNT+N1:IF CNT=N3 THEN CNT=N0:P
IE=PIE-N1:IF PIE<N0 THEN PIE=N0
980 GOSUB DIR
990 GRAPHICS N0:POKE 559,N0:POKE 756,C
HSET:POKE 710,N0:POKE 709,N12:POKE 752
,N1:DL=PEEK(560)+256*PEEK(561)+N4
1000 POKE DL-N1,71:POKE DL+N2,N7:POKE
DL+N3,N7:POKE DL+N4,N7:POKE DL+N5,N7:P
OKE DL+N6,N7:POKE DL+N7,N7
1010 POKE DL+N8,N6:POKE DL+N9,N6:POKE
DL+N10,N6:POKE DL+N18,65:POKE DL+N19,P
EEK(560):POKE DL+N20,PEEK(561)
1020 POKE 82,N0:POSITION N1,N0:? ",,,,
,,,,,,,,,,,,,    ,                   ,"
1030 ? " ,                       ,  ,
      , ,                    ,   ,
     ,"
1040 ? " ,,,,,,,,,,,,,,,,,,"
1050 POSITION N9,N0:? E$(N1,N1):POSITI
ON 37,N1:? E$(N2,N2):POSITION N9,N3:?
E$(N3,N3):POSITION N21,N1:? E$(N4,N4)
1060 IF I(ROOM)=N7 THEN X=INT(RND(N0)*
N3)+N1:IF X=N1 THEN I(ROOM)=N2
1070 IF I(ROOM)>N2 THEN POKE 559,34:GO
TO 1350
1080 IF I(ROOM)=N2 THEN POKE 710,N8:PO
KE 709,N0:POKE 559,34:POSITION 28,N1:?
 "orc":FOR X=N1 TO 50:NEXT X:GOTO 1130
1090 POKE 559,34:POSITION 26,N1:? "DRA
GON!":FOR Y=N1 TO N8:FOR Z=N1 TO 30:PO
KE 710,Z*N8:POKE 712,Z*N8-N6
1100 SOUND N0,Z,N6,N8:NEXT Z:NEXT Y:SO
UND N0,N0,N0,N0
1110 GRAPHICS N18:POKE 756,CHSET:IF B
THEN POSITION N1,N1:? #N6;"THE TOWN RE
JOICES!":I(DR)=N0:SC=SC+100:GOTO 2180
1120 GOTO DEAD
1130 CHP=N5:CSTR=N10:POSITION N1,N4:?
"thee:      orc:"
1140 POSITION N1,N5:? "┌──────────────
─────────────────────┐   |
                       |"
1150 ? " |
         |  |     THY HP:          ORC H
P:            |"
1160 ? " |
         |  |        STR:             ST
R:            |"
1170 ? " └──────────────────────────────
────────┘";
1180 PL=WN-N1:B$="PLUS  ":C$=STR$(PL):
C$=CHR$(ASC(C$)+128):B$(N6,N6)=C$
1190 POSITION N8,N6:? "THY WEAPON IS "
;B$:POSITION 28,N1:? "     "
1200 POSITION N14,N8:? HP;" ":POSITION
 31,N8:? CHP;" ":POSITION N14,N10:? ST
R;" ":POSITION 31,N10:? CSTR;" "
1210 D1=INT(RND(N0)*N9)+N1:POSITION N5
,N1:? D$(D1*N9-N8,D1*N9-N6):POSITION 2
5,N1:? D$(D1*N9-N5,D1*N9-N3)
1220 POSITION N5,N2:? D$(D1*N9-N2,D1*N
9):SOUND N0,150,N2,N8:SOUND N0,N0,N0,N
0
1230 IF STRIG(N0) THEN 1210
1240 D1=D1-INT(N4-STR/N20)+PL:POSITION
 N6,N4:? D1;" ":STR=STR-N1-N1*(PIE=N0)
:IF STR<N1 THEN GOTO DEAD
1250 FOR X=N1 TO N20:D2=INT(RND(N0)*N9
)+N1:POSITION N11,N1:? D$(D2*N9-N8,D2*
N9-N6)
1260 POSITION 31,N1:? D$(D2*N9-N5,D2*N
9-N3):POSITION N11,N2:? D$(D2*N9-N2,D2
*N9):SOUND N0,150,N2,N8
1270 SOUND N0,N0,N0,N0:NEXT X:POSITION
 N15,N4:? D2;" ":CSTR=CSTR-N1:IF CSTR<
N1 THEN 1330
1280 H=INT(RND(N0)*N5)+N1:IF D1>=D2 TH
EN SC=SC+N1:GOTO 1310
1290 FOR X=N0 TO 255 STEP N5:SOUND N0,
X,N10,N4:NEXT X:SOUND N0,N0,N0,N0:HP=H
P-H:IF HP<N1 THEN GOTO DEAD
1300 GOTO 1200
1310 FOR X=250 TO N0 STEP -N10:SOUND N
0,X,N10,N4:NEXT X:SOUND N0,N0,N0,N0:CH
P=CHP-H:IF CHP<N1 THEN 1330
1320 GOTO 1200
1330 X=INT(RND(N0)*N5)+N4:GRAPHICS N0:
POKE 756,CHSET:POKE 710,N0:POKE 752,N1
:POSITION N14,N8:? "HE IS DEAD!"
1340 POSITION N11,N12:? "YOU FOUND ";X
;" GOLD!":G=G+X:I(ROOM)=N7:SC=SC+N11:G
OSUB DELAY2:GOTO 550
1350 IF I(ROOM)=N3 THEN POSITION 25,N1
:? "teleport":SC=SC+N1:GOTO TEL
1360 COL=N10:IF I(ROOM)=N7 THEN POSITI
ON 27,N1:? "empty":GOTO 1420
1370 POSITION N5,N1:? "you found":SC=S
C+N1
1380 IF I(ROOM)=N4 THEN POSITION N6,N2
:? "a spell":SPL=SPL+N1
```

```
1390 IF I(ROOM)=N5 THEN X=INT(RND(N0)*
N5)+N4:POSITION N3,N2:? X;" gold piece
s":G=G+X
1400 IF I(ROOM)=N6 THEN POSITION N6,N2
:? "a serum":SM=SM+N1
1410 IF I(ROOM)=N8 THEN POSITION N5,N2
:? "the theif":G=INT(G/N2):FOR Z=N1 TO
 N8:GOSUB SND2:NEXT Z
1420 SC=SC+N1:POSITION COL,N7:? #N6;"P
RESS THY TRIGGER"
1430 IF STRIG(N0) THEN 1430
1440 GOSUB SND1:I(ROOM)=N7:GOTO 550
1450 GRAPHICS N0:POKE 756,CHSET:POKE 7
52,N1:POKE 710,N0:POKE 711,54:POKE 712
,112
1460 DL=PEEK(560)+256*PEEK(561)+N4:POK
E DL+22,N6:POKE DL+23,N6:POKE DL+24,N6
:POKE DL,N0:POKE DL+N1,MAP
1470 POKE 88,N0:POKE 89,MAP:POKE 559,3
4
1480 R=ROOM:GOSUB ROOMPOS
1490 POSITION COL+N1,ROW+N1:? "•":FOR
X=N1 TO N10:NEXT X
1500 POSITION COL+N1,ROW+N1:? " ":FOR
X=N1 TO N10:NEXT X
1510 IF STRIG(N0)=N0 THEN GOSUB SND1:G
OTO 550
1520 GOTO 1490
1530 REM *** CAST SPELLS ***
1540 IF SPL>N0 THEN SPL=SPL-N1:ROOM=N1
:GOTO 1560
1550 GOSUB SND2:POSITION COL,ROW:? " "
:GOTO 770
1560 POSITION N21,N13:? "          ↓←←←
←←←←     ↓←←←←)   ↓←←←←       ↓←←←←      "
1570 POSITION N21,N19:? "
    ":POSITION N21,N20:? "
    ":POSITION N5,N20:? "WHAT ROOM?"
1580 IF STRIG(N0)=N0 THEN 1580
1590 IF STRIG(N0)=N0 THEN 1630
1600 ST=STICK(N0):ROOM=ROOM+(ST=N14)*N
1+(ST=N13)*-N1:IF ROOM>N77 THEN ROOM=N
1
1610 IF ROOM<N1 THEN ROOM=N77
1620 POSITION 25,N15:? ROOM;" ":FOR X=
N1 TO N10:NEXT X:GOTO 1590
1630 SOUND N0,248,N10,N4:SOUND N1,255,
N14,N4:SOUND N2,246,N14,N4:SOUND N3,24
3,N10,N4
1640 POKE ADR(R$)+24,26:D=USR(ADR(R$)+
N1,N2):FOR X=N0 TO N3:SOUND X,N0,N0,N0
:NEXT X:GOTO 950
1650 REM *** STORE ***
1660 STR=STR-N2-N2*(PIE=N0):IF STR<N1
THEN GOTO DEAD
1670 S=INT(RND(N0)*N10)+N1:IF STR>90 T
HEN S=N1
1680 IF S<N6 THEN SL=N0:S$(36,75)="---
---- WELCOME ALL ADVENTURERS! -------"
:GOTO 1730
1690 S$(36,75)="---- SPECIAL TODAY:
 PERCENT OFF! ----"
1700 IF S<N9 THEN S$(56,57)="10":SL=0.
1
1710 IF S=N9 THEN S$(56,57)="20":SL=0.
2
1720 IF S=N10 THEN S$(56,57)="30":SL=0
.3
1730 GRAPHICS N0:POKE 559,N0:POKE 710,
N0:POKE 756,CHSET:DL=PEEK(560)+256*PEE
K(561)+N4
1740 POKE DL-N1,70:FOR X=N2 TO N6:POKE
 DL+X,N6:NEXT X:FOR X=N9 TO N20:POKE D
L+X,N6:NEXT X:COL=N6:R=N7
1750 POKE 752,N1:POKE 82,N0:TURN=TURN+
N1:POKE DL,N0:POKE DL+N1,STORE:POKE 88
,N0:POKE 89,STORE:POKE 559,34
1760 FOR X=N1 TO 80:B$=S$(X,37+X):POSI
TION N1,N3:? B$;:FOR Y=N1 TO N8:NEXT Y
1770 IF STRIG(N0)=N0 THEN POP :GOSUB S
ND1:GOTO 1790
1780 NEXT X:GOTO 1760
1790 COL=N6:R=N6:ROW=N6:POSITION N1,N3
:? "
    "
1800 POSITION N7,N6:? "          ↓←←←←←←
←←      ↓←←←←←←←←←←         ↓←←←←←←←←←
    "
1810 POSITION N10,N3:? "YOU HAVE ";G;"
 GOLD.":GOSUB DELAY2
1820 POSITION N10,N3:? "CHOOSE THE DEP
ARTMENT"
1830 POSITION COL,ROW:? " ":POSITION N
7,N6:? "magic ↓←←←←←←health ↓←←←←←←←←←
weapons↓←←←←←←←exit    "
1840 GOSUB CHOOSE:POSITION COL,ROW:? "
 ":POSITION N9,N3:IF ROW<N9 THEN ? "
MAKE THY PURCHASE   "
1850 ON CH GOTO 1860,1930,2020,2110
1860 POSITION N7,N6:? "spells↓←←←←←←gy
psy ↓←←←←←←←brew   "
1870 GOSUB CHOOSE:ON CH GOTO 1880,1900
,1890,1820
1880 COST=N10:GOSUB 2070:SPL=SPL+N1:GO
TO 2090
1890 COST=60:GOSUB 2070:B=N1:GOTO 2090
1900 COST=N20:GOSUB 2070
1910 GOSUB CLUE:POSITION N7,N6:? "
  ↓←←←←←←←      ↓←←←←←←     ↓←←←←←←     "
1920 POSITION 27,N7:? DIR$(D*N5-N4,D*N
5):GOSUB DELAY2:POSITION 27,N7:? "
 ":GOTO 2090
1930 POSITION N7,N6:? "pie  ↓←←←←←←lodg
ing↓←←←←←←←←doctor "
1940 GOSUB CHOOSE:ON CH GOTO 1950,1990
,1960,1820
1950 COST=N10:GOSUB 2070:PIE=PIE+N5:GO
TO 2090
1960 COST=N15:GOSUB 2070
1970 X=INT(RND(N0)*N10)+N15:HP=HP+X:IF
 HP>50 THEN HP=50
1980 POSITION N1,N3:? "THE DOCTOR HAS
HEALED THEE -- ";X;" HP --":GOSUB DELA
Y2:GOTO 2090
1990 COST=N15:GOSUB 2070
2000 X=INT(RND(N0)*N15)+N20:STR=STR+X:
IF STR>100 THEN STR=100
2010 POSITION N1,N3:? "   A GOOD NIGHT
S SLEEP! -- ";X;" STR --   ":GOSUB DELA
Y2:GOTO 2090
2020 POSITION N7,N6:? "dagger↓←←←←←←←s
sword↓←←←←←←←←l sword"
2030 GOSUB CHOOSE:ON CH GOTO 2040,2050
,2060,1820
2040 COST=N10:GOSUB 2070:WN=N2:GOTO 20
90
2050 COST=30:GOSUB 2070:WN=N3:GOTO 209
0
2060 COST=50:GOSUB 2070:WN=N4:GOTO 209
0
2070 POSITION COL,ROW:? " ":GD=G:G=INT
(G-(COST-(SL*COST))):IF G<N0 THEN 2140
2080 P=N1:RETURN
2090 FOR X=N1 TO N2:SOUND N0,N3,N0,N8:
FOR Y=N1 TO N8:NEXT Y:SOUND N0,N0,N0,N
0
2100 FOR Y=N1 TO N3:NEXT Y:NEXT X:FOR
X=N14 TO N0 STEP -0.3:SOUND N0,N5,N10,
X:SOUND N1,N20,N10,X:NEXT X:GOTO 1790
2110 IF  NOT P THEN 2150
2120 POSITION N1,N3:? "    I THANK THE
E FOR THY PURCHASE!"
2130 FOR X=N14 TO N0 STEP -0.3:SOUND N
0,30,N10,X:NEXT X:GOSUB DELAY2:P=N0:GO
TO 2150
2140 POSITION N1,N3:? "         THY PURS
E IS TOO MEAGER! ":GOSUB SND2:GOSUB DE
LAY2:G=GD:POP :GOTO 1790
2150 POKE 559,N0:POSITION N3,N3:? "
                                  "
2160 POSITION N7,N6:? "press↓←←←←←← thy
  ↓←←←←←←←joystick↓←←←←←←←←button":GOTO
 550
2170 GRAPHICS N18:POKE 756,CHSET:POSIT
ION N2,N1:? #N6;"YOU ARE DEAD!"
2180 POSITION N2,N3:? #N6;"DO YOU WANT
":? #N6;"  TO PLAY AGAIN?"
2190 IF I(DR)=N0 THEN SC=SC+(200-TURN)
2200 POSITION N4,N7:? #N6;"SCORE: ";SC
2210 OPEN #N1,N4,N0,"K:":GET #N1,A:CLO
SE #N1:IF A=ASC("N") THEN POKE 82,N2:E
ND
2220 IF A<>ASC("Y") THEN 2210
2230 POSITION N4,N10:? #N6;"one moment
"
```

```
2240 RESTORE 2940:HP=50:STR=100:SPL=N0
:PIE=N1:CNT=N0:B=N0:WN=N1:G=60:ROOM=39
:SC=N0:SM=N0:TURN=N0:GOTO 80
2250 REM *** INITIALIZE ***
2260 RTOP=PEEK(106)-N21:POKE 106,RTOP:
GRAPHICS N0:DL=PEEK(560)+256*PEEK(561)
+N4:POKE DL+N4,N7:POKE DL+N5,N6
2270 POKE DL+N8,N6:POKE DL+N9,N6:POKE
710,N0:POKE 752,N1:MAP=RTOP+N1:STTUS=R
TOP+N6:STORE=RTOP+N11
2280 POSITION N4,N3:? "dragonlord":POS
ITION N8,N6:? "by":POSITION 22,N6:? "c
layton walnum"
2290 POSITION N10,N20:? "ONE MOMENT PL
EASE"
2300 DIM RM(N77),I(N77),RM$(255),MV$(N
20),N$(N9),W$(28),DIR$(N20),S$(120),B$
(40),CL$(40)
2310 DIM R$(N1),R(N5),C$(N1),E$(N4),D$
(81)
2320 S$(N1)=" ":S$(120)=" ":S$(N2)=S$
2330 S$(N1)=" ":S$(120)=" ":S$(N2)=S$:
RESTORE 3370:FOR X=N1 TO 40:READ A:CL$
(X)=CHR$(A):NEXT X
2340 WN=N1:PIE=N1:CNT=N0:G=60:SPL=N0:H
P=50:STR=100:B=N0:ROOM=39:SM=N0:SC=N0:
TURN=N0
2350 ROOMPOS=180:CHOOSE=290:DIR=370:CL
UE=210:DEAD=2170:DELAY2=360:SND1=160:S
ND2=170:TEL=130
2360 RM$="┌┴┐↓←←←| ├↓←←←└─┘┌┴┐↓←←←| |↓
←←←└┬┘┌┴┐↓←←←┤ |↓←←←└─┘┌─┐↓←←←| ├↓←←←└
┬┘┌─┐↓←←←┤ |↓←←←└┬┘"
2370 RM$(86)="┌─┐↓←←←┤ ├↓←←←└─┘┌┴┐↓←←←
| ├↓←←←└┬┘┌┴┐↓←←←┤ ├↓←←←└─┘┌─┐↓←←←┤ ├↓
←←←└┬┘┌┴┐↓←←←┤ ├↓←←←└┬┘"
2380 RM$(171)="┌┴┐↓←←←| |↓←←←└─┘┌─┐↓←←
←| |↓←←←└┬┘┌─┐↓←←←| ├↓←←←└─┘┌─┐↓←←←┤ |
↓←←←└─┘┌┴┐↓←←←┤ |↓←←←└┬┘"
2390 W$="NONE   DAGGER S SWORDL SWORD"
:DIR$="northeast southwest "
2400 D$="++++;++++;+++++++++;;+++;+++;;+
;+++;+;;+;+;;+;;+;;+;;+;;+;;+;;+;;+;;;
+;;;;;;;;;;;;;"
2410 CHSET=RTOP+N17:CHBASE=CHSET*256:P
OKE 203,N0:POKE 204,CHSET:POKE 205,N0:
POKE 206,224
2420 FOR X=N1 TO N20:READ A:MV$(X)=CHR
$(A):NEXT X:D=USR(ADR(MV$)):POKE 756,C
HSET:GOTO 30
2430 DATA 104,104,104,162,255,160,58,2
02,208,9,136,208,6,170,202
2440 DATA 138,208,241,96,142,10,212,14
2,22,208,24,144,235
2450 DATA 0,248,60,102,102,126,231,0
2460 DATA 0,252,102,252,102,102,124,0
2470 DATA 0,254,102,112,240,102,62,0
2480 DATA 0,248,108,102,230,108,124,0
2490 DATA 0,254,96,124,224,102,126,0
2500 DATA 0,254,99,252,108,96,96,0
2510 DATA 0,254,102,96,238,102,62,0
2520 DATA 0,118,54,126,54,54,118,0
2530 DATA 0,126,24,60,24,24,126,0
2540 DATA 0,14,6,6,246,102,60,0
2550 DATA 0,230,110,248,120,110,102,0
2560 DATA 0,240,96,224,96,118,126,0
2570 DATA 0,227,119,255,107,99,119,0
2580 DATA 0,230,118,126,126,110,246,0
2590 DATA 0,254,102,230,102,102,126,0
2600 DATA 0,252,102,238,124,96,112,0
2610 DATA 0,254,102,230,102,108,118,0
2620 DATA 0,252,102,230,124,108,118,0
2630 DATA 0,254,96,126,6,102,126,0
2640 DATA 0,126,216,24,24,24,28,0
2650 DATA 0,230,102,230,102,126,124,0
2660 DATA 0,230,102,230,102,126,24,0
2670 DATA 0,227,99,235,127,119,99,0
2680 DATA 0,230,102,60,60,102,118,0
2690 DATA 0,230,102,60,24,56,56,0
2700 DATA 0,254,108,24,48,102,126,0
2710 DATA 170,170,170,170,170,170,170,
170
2720 DATA 0,254,102,238,118,102,126,0
2730 DATA 0,24,120,120,24,24,126,0
2740 DATA 0,124,102,76,24,54,126,0
2750 DATA 0,126,206,24,12,102,62,0
2760 DATA 0,12,28,60,108,254,12,0
2770 DATA 0,254,96,252,6,102,124,0
2780 DATA 0,124,224,124,230,102,126,0
2790 DATA 0,254,102,12,24,48,48,0
2800 DATA 0,60,102,60,230,102,254,0
2810 DATA 0,254,102,254,6,12,56,0
2820 DATA 0,60,102,219,195,0,0,0
2830 DATA 24,28,6,54,54,6,28,24
2840 DATA 24,56,96,108,108,96,56,24
2850 DATA 24,24,12,198,230,62,0,0
2860 DATA 0,0,62,230,198,12,24,24
2870 DATA 24,24,48,99,103,124,0,0
2880 DATA 0,0,124,103,99,48,24,24
2890 DATA 0,0,0,195,219,102,60,0
2900 DATA 255,255,255,255,255,255,255,
255
2910 DATA 247,247,0,189,189,0,247,247
2920 DATA 255,255,195,129,129,129,195,
255
2930 REM *** ROOM DATA ***
2940 DATA 4,6,6,9,6,9,6,9,6,6,5
2950 DATA 1,9,9,8,5,2,4,8,9,9,3
2960 DATA 12,11,2,4,3,2,1,5,2,11,12
2970 DATA 7,6,10,10,6,10,6,10,10,6,15
2980 DATA 11,12,2,1,5,2,4,3,2,12,11
2990 DATA 4,8,8,9,3,2,1,9,8,8,5
3000 DATA 1,6,6,8,6,8,6,8,6,6,3
3010 DATA 1,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,3,3,3,3,4,4,4,4,4,5,5,5,
5,5,6,6,6,6,6,6,6,7,7,7,7,7,7
3020 DATA 7,7,7,7,7,7,7,7,7,7,7,7,7,7,
7,7,7,7,7,7,7,7,7,7,7,7,7,7,8,-1
3030 POKE 87,N0:POKE 88,N0:POKE 89,MAP
:A=USR(ADR(CL$),N15)
3040 POSITION N18,N9:? RM$(154,170)
3050 POSITION 26,N21:? "a map of":POSI
TION N4,22:? "thy progress";
3060 POKE 89,STTUS:POSITION N5,N0:? "d
ragonlord"
3070 POSITION N0,N1:? "┌──────────────
──────────────────┬┐"
3080 POSITION N0,N2:? ")###############
####'|)########################'"
3090 POSITION N0,N3:? "% NAME:
    $|% WEAPON:         $"
3100 POSITION N0,N4:? "%
    $|%                 $"
3110 POSITION N0,N5:? "%    HP:
    $|% SPELLS:         $"
3120 POSITION N0,N6:? "%
    $|%                 $"
3130 POSITION N0,N7:? "%   STR:
    $|%  SERUM:         $"
3140 POSITION N0,N8:? "%
    $|%                 $"
3150 POSITION N0,N9:? "%   PIE:
    $|%    GOLD:        $"
3160 POSITION N0,N10:? "(*************
****&|(****************&"
3170 POSITION N0,N11:? "└─────────────
──────────────────────┘"
3180 POSITION N0,N12:? "┌─────────────
─────┐┌──────────┐┌────────┐"
3190 POSITION N0,N13:? ")##############
######'||  COMMAND  ||)######'"
3200 POSITION N0,N14:? "%  ROOM:
    $||    MOVE    ||%      $"
3210 POSITION N0,N15:? "% EXITS:
    $||    MAP     ||(****&"
3220 POSITION N0,N16:? "(*************
****&||    CAST    |└────────┘"
3230 POSITION N0,N17:? "└─────────────
─────┘|    STORE   |":POSITION N20,N18:?
"└──────────┘"
3240 POKE 89,STORE:POKE 82,N0:POSITION
 N1,N0:? ",,,,,,,,,,,,,,,,,,,,,,,   ,
          ,  ,  ye olde shoppe  ,"
3250 POSITION N21,N1:? ",
   ,  ,,,,,,,,,,,,,,,,,,,,,,"
3260 POSITION N4,N5:? ",,,,,,,,,,,,,,
    ,            ,       ,   press   ,
          ,"
3270 ? "      ,    thy      ,        ,
    ,        ,  joystick ,        ,
    ,"
3280 ? "      ,   button  ,        ,
    ,        ,,,,,,,,,,,,,,"
```

```
3290 POSITION N2,N11:? "MAGIC         H
EALTH         WEAPONS         SPELLS: 10
 5 PIE   : 10  DAGGER  : 10"
3300 ? "  GYPSY  : 20   LODGING: 15  5
SWORD: 30   BREW   : 60   DOCTOR : 15
L SWORD: 50"
3310 GRAPHICS N0:POKE 756,CHSET:POKE 7
52,N1:POKE 710,160
3320 POSITION N2,N10:? "WHAT WILL BE T
HY NAME FOR THIS QUEST?"
3330 POSITION N14,N13:? "----------←←←←
←←←←←←";:INPUT N$
3340 FOR X=N1 TO N9:IF N$(X,X)<>"-" TH
EN NEXT X:GOTO 550
3350 TRAP 3030:N$=N$(N1,X-N1):POP :GOT
O 550:TRAP 40000
3360 REM *** CL$ DATA ***
3370 DATA 104,104,104,133,205,165,88,1
33,203,165,89,133,204,162,0,169,0,168,
145,203,200,208,251,232,228,205
3380 DATA 240,11,24,165,204,105,1,133,
204,240,234,208,232,96
3390 REM *** MV$ DATA ***
3400 DATA 104,162,4,160,0,177,205,145,
203,200,208,249,230,206,230,204,202,20
8,242,96
```

•

CHECKSUM DATA.

(see page 32)

```
1 DATA 303,273,991,703,317,844,392,368
,499,372,465,771,238,819,436,7791
130 DATA 16,407,617,441,292,82,517,583
,844,68,34,989,990,999,9,6888
280 DATA 607,336,734,603,164,657,664,4
91,776,822,885,0,27,26,8,6800
430 DATA 59,996,548,579,573,52,155,157
,130,163,588,691,164,914,325,6094
580 DATA 786,127,135,549,784,372,735,7
6,908,54,174,157,532,555,502,6446
730 DATA 577,860,328,664,310,357,310,7
8,65,506,771,107,525,767,631,6856
880 DATA 774,750,265,270,55,81,881,924
,951,576,143,808,989,399,367,8233
1030 DATA 254,631,665,665,687,260,366,
994,31,873,945,381,107,796,727,8382
1180 DATA 550,505,359,151,411,320,843,
285,882,427,90,293,707,463,709,6995
1330 DATA 396,267,130,818,458,336,459,
862,730,404,336,528,621,657,20,7022
1480 DATA 924,197,227,785,735,734,411,
999,376,868,722,710,718,884,107,9397
1630 DATA 251,360,891,84,123,279,560,6
13,618,850,511,415,708,832,588,7683
1780 DATA 857,200,758,592,293,874,717,
848,626,958,944,774,203,10,837,9491
1930 DATA 540,969,888,218,207,157,221,
109,877,684,884,439,67,77,951,7288
2080 DATA 320,575,208,12,366,623,307,3
40,355,363,338,590,695,632,365,6089
2230 DATA 265,714,474,368,129,335,862,
82,763,134,376,52,737,224,770,6285
2380 DATA 860,934,384,580,395,874,12,9
11,160,900,187,942,974,733,891,9737
2530 DATA 622,588,158,758,968,193,164,
944,175,182,906,612,157,892,787,8106
2680 DATA 684,606,720,325,196,872,877,
874,642,905,185,624,715,630,427,9282
2830 DATA 645,834,633,635,654,659,430,
401,277,414,107,647,636,685,163,7820
2980 DATA 687,638,599,449,474,462,120,
790,974,374,309,16,87,916,93,6988
3130 DATA 2,99,763,986,673,574,960,685
,809,932,199,144,37,951,99,7913
3280 DATA 864,802,153,861,865,110,794,
446,41,827,282,84,660,6789
```

•

Assembly listing.

```
; ***************************
; *     MEMORY CLEANER      *
; ***************************
;
; A=USR(ADR,PAGES)
;
; ADR=ADDRESS OF ROUTINE
; PAGES=NUMBER OF PAGES TO BE CLEARED
;
        *=  $5000
;
; EQUATES
;
SAVMSC  =   $58
PAGES   =   $CD
TMP     =   $CB
;
; INITIALIZE
;
        PLA             ;# OF ARGUMENTS
        PLA             ;HI-BYTE: IGNORE
        PLA             ;# OF PAGES
        STA PAGES
        LDA SAVMSC      ;GET SCREEN ADDRESS
        STA TMP         ;AND STORE IT
        LDA SAVMSC+1    ;IN TEMP WORK AREA
        STA TMP+1
;
; MAIN PROGRAM
;
        LDX #0          ;ZERO PAGE COUNTER
NEXT    LDA #0          ;LOAD WITH BLANK
        TAY             ;ZERO INDEX
CLEAR   STA (TMP),Y     ;CLEAR ONE BYTE OF MEMORY
        INY             ;INCREMENT INDEX
        BNE CLEAR       ;GO CLEAR NEXT BYTE
        INX             ;INCREMENT COUNTER
        CPX PAGES       ;ALL DONE?
        BEQ END         ;YES
        CLC             ;NO, MOVE
        LDA TMP+1       ;UP ONE PAGE
        ADC #1
        STA TMP+1
        BEQ NEXT        ;GO TO CLEAR
        BNE NEXT        ;NEXT PAGE
END     RTS             ;BACK TO BASIC
```

•

Talk to
ANALOG Computing

We're happy to announce that three members of our staff can now be regularly found on CompuServe. If you're a CompuServe member, you can contact Tom Hudson, Charles Bachand or Art Leyenberger by leaving a message on the Atari SIG, which can be accessed by typing GO PCS-132 at any menu page.

The Atari SIG has logged over 100,000 calls—with over 60,000 messages posted! They have a staff of highly competent SYSOPs, headed up by Ron Luks, who are more than happy to help you. Their program database contains well over a megabyte (that's one million bytes, folks!) of Atari programs that can be downloaded into your computer.

So, if you need to get in touch with **ANALOG Computing**, you can now do it through CompuServe. Our user numbers are:

Tom Hudson 70775,424
Charles Bachand 73765,646
Art Leyenberger 71266,46

SUSPECT
by David Lebling
INFOCOM, INC.
55 Wheeler Street
Cambridge, MA 02138
(617) 492-1031
48K Disk $39.95

by Ray Berube

Infocom has released a new text adventure for its mystery series. It is titled **Suspect** and is not for the inexperienced player. Retailing for $39.95, it is certainly a chock-full adventure for the money. However, as varied and detailed as it is, **Suspect** doesn't satisfy as well as some of Infocom's earlier mystery adventures.

Written by David Lebling, co-author of **Zork** and a mainstay of creative imagination at Infocom, **Suspect** invites you to play the role of a newspaper reporter attending a "chic" Halloween costume party. As expected, your hostess is murdered with your lariat (you *had* to come dressed as Roy Rogers), and you suddenly become the primary **Suspect**. The familiar character of Sergeant Duffy is on hand to arrest you if you can't solve the crime by discovering the identity of the real murderer.

According to Lebling, "**Suspect** combines the rich texture of **The Witness** with the complicated plot structure that distinguishes Infocom mysteries." It is also supposedly characterized by the dry humor familiar to Infocom fans.

Well, before addressing these claims in detail, I'd like to take a moment to comment on one of **Suspect**'s best features—its new packaging. There was a time when interested computer owners could spot an Infocom game from across the crowded software store. Its packaging was unique and very often beautiful, as with the fantasy game **Enchanter**. However, Infocom discovered a few drawbacks with implementing innovative packaging, namely: poor dealer space utilization and customer dissatisfaction.

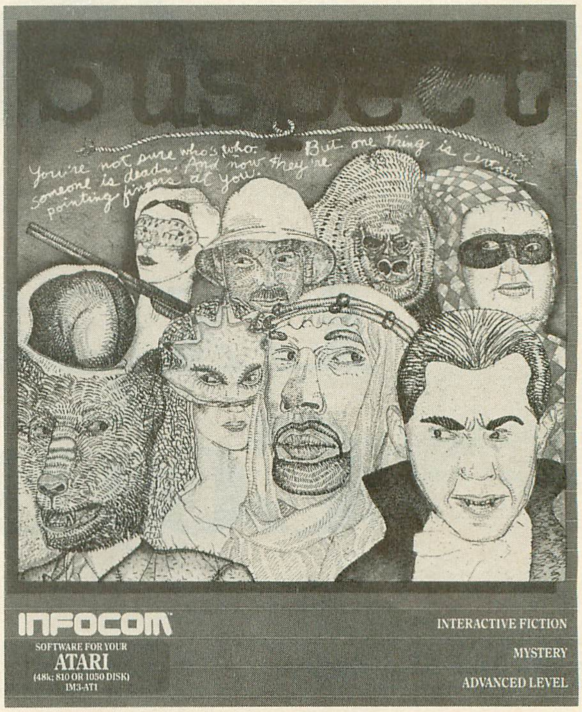
It looked pretty, but you couldn't stack it, shelve it, or put it in a drawer (the frisbee of **Starcross** springs to mind). It only caught your attention as a buyer if the dealer would display it. So some of Infocom's games began to experience sales declines, because of dealer reluctance to allot space displaying the unusual boxes.

What to do? The wizards at Infocom got together and came up with a solution both practical and attractive. The change to a more uniform shape for the package, with a distinctive design for each game, has helped some Infocom titles, especially the **Zork** series.

Suspect has arrived in a new box, compact and functional. It serves well as a storage unit for the game's components and it stacks neatly on the dealer's shelf. So, from now on, look for Infocom in a classy book-sized package. The days of **Suspended**'s mask are gone (and I still haven't found the right opportunity to wear my **Suspended** mask).

Now, on to the game itself. The opening moves reveal a cast of very well-detailed characters, who can be interacted with most effectively. However, trying to map or chart their movements from ballroom to hallway to outside to office is quite a challenge. Not to mention that some of these characters are only known to you by their costume. Considerable care and patience is required to chart this opening.

The inexperienced player should take heed! **Suspect** requires careful mapping and charting of characters to keep things straight and to give you *any* chance to solve its riddles. The play is very user friendly, as with all Infocom's games. You can't fault a parser which allows you to communicate on a nearly conversational level with the game elements.



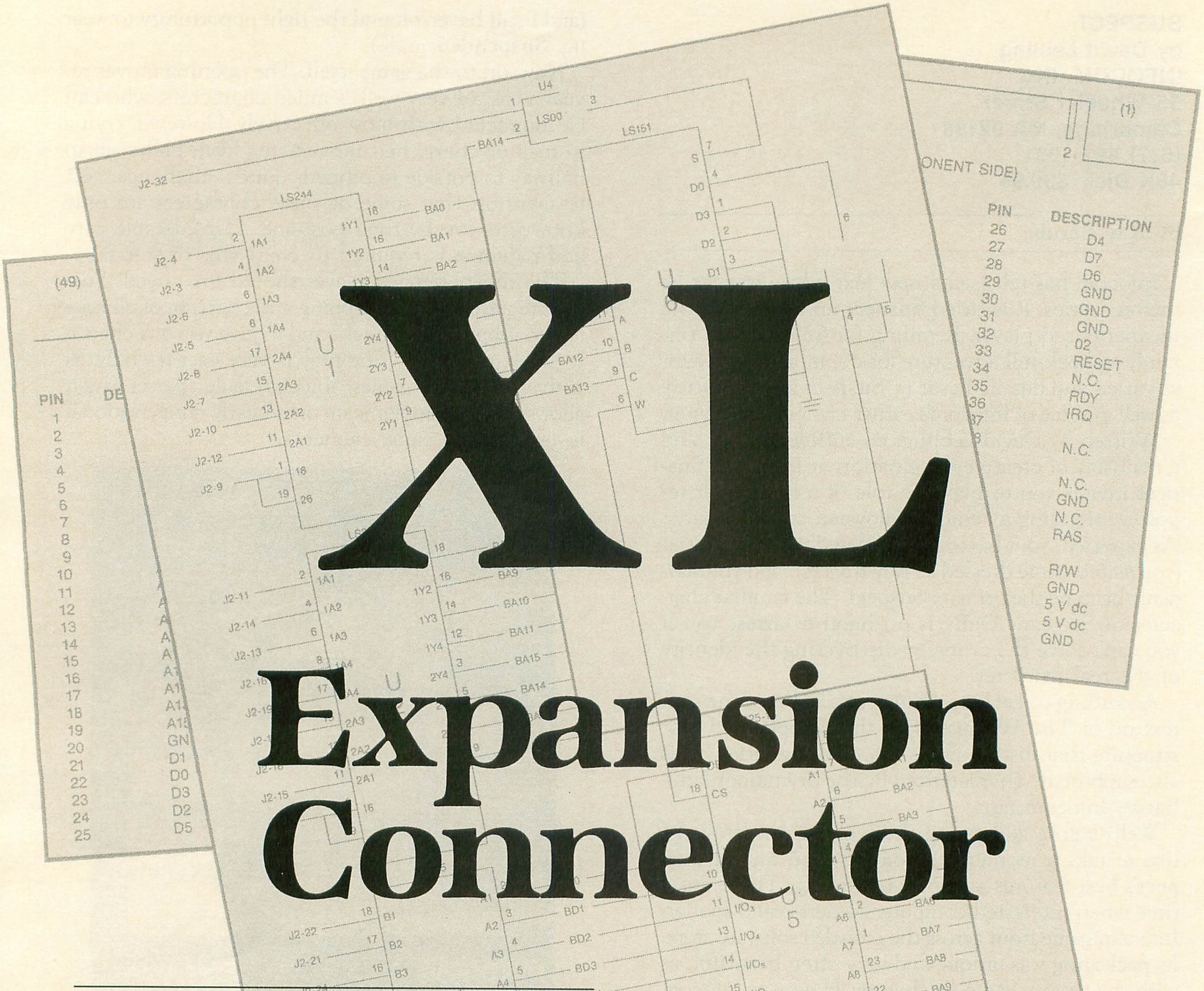
Suspect.

In fact, technically, this is probably Infocom's most complex and truly "interactive" adventure thus far. There is a whole rogues' gallery of characters with whom you can converse and interact. This reviewer could not find any negative aspects to report on concerning the programming of the adventure.

The complaints I have are involved more with the tone and style of **Suspect** than with its mechanics. From the moment you open the package and read "Murder and Modern Manners" (the game's expository text), you get the feeling this whole adventure promises to be a joke—at your expense.

The idea of finding oneself **Suspect**ed of murder and then being forced to find the real killer is a good one.

(continued on page 52)



XL Expansion Connector

by Michael Alan Barton

The Atari 600XL home computer provides 320 by 192-bit mapped graphics, 16K of RAM, built-in BASIC interpreter and external access to the microprocessor's bus. The latter is referred to as the "expansion connector." The expansion connector is not documented by Atari's Owner's Guide or other printed material supplied with the XL package.

This article describes a project which had the following as goals: (1) determine the pinout of the expansion connector; (2) describe the characteristics of the system's clock; and (3) design and construct a simple memory interface to verify the findings in goals 1 and 2.

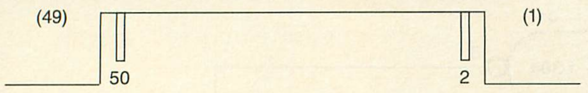
The 600XL is based on a modified version of the popular 6502 8-bit microprocessor. The microprocessor's bus is divided into three sections. They are: the address, data and control busses. Access to each of these is available at the expansion connector.

Expansion connector pinout.

The first goal of the project was to determine the pinout of the expansion connector. This connector is an integral part of the XL's printed circuit board.

There are twenty-five contact fingers on each side of the printed circuit board. The contact fingers are on 0.100 inch centers. A 50-pin card-edge connector is required to mate with the expansion connector.

The expansion connector is designated as J2. The contact numbering and description are given in Figure 1. As shown, the even-numbered contacts are on the top side of the connector. They are numbered 2 through 50. When looking at the connector from the back side of the computer, number 2 will be on the left side. The odd-numbered contacts are on the bottom side of the expansion connector. Number 1 is directly under number 2, and number 49 is directly under 50.



TOP (COMPONENT SIDE)

PIN	DESCRIPTION	PIN	DESCRIPTION
1		26	D4
2	GND	27	D7
3	A1	28	D6
4	A0	29	GND
5	A3	30	GND
6	A2	31	GND
7	A5	32	02
8	A4	33	RESET
9	GND	34	N.C.
10	A6	35	RDY
11	A8	36	IRQ
12	A7	37	
13	A10	38	N.C.
14	A9	39	
15	A12	40	N.C.
16	A11	41	GND
17	A14	42	N.C.
18	A13	43	RAS
19	A15	44	
20	GND	45	R/W
21	D1	46	GND
22	D0	47	5 V dc
23	D3	48	5 V dc
24	D2	49	GND
25	D5	50	

Figure 1. Atari 600XL Expansion Bus (J2).

The complete address bus is available at the expansion connector. The address bus is designated A0 through A15. The microprocessor communicates with an external device by placing the binary address on Lines A0 through A15 for the device. The most significant bit (MSB) is A15. The least significant bit (LSB) is A0. External devices connected to the expansion connector are required to decode the address. Each device is assigned an address or range of addresses. When an external device determines that it is being addressed by the CPU (microprocessor), it will transmit data to or receive data from the microprocessor on the data bus.

The data bus consists of eight lines. They are designated D0 through D7. The MSB is D7, and the LSB is D0. Since 8 bits of data are transferred on the bus during any one bus cycle, the 600XL is considered an 8-bit microcomputer.

Several other lines make up the control bus. We'll discuss the two most important ones here. They are the phase 2 clock (02) and Read/Write signal (R/W).

The R/W line is held high (2.4 to 5.0 volts) by the CPU whenever it is attempting to read data from the data bus. The R/W line transitions to the low level (0.0 to 0.4 volts) when the CPU is writing data to external devices.

During the transitions of the R/W, address and data lines, the information on the bus is invalid. Therefore, peripheral devices must be signaled by the CPU when the bus lines are stable and valid data exist. The 02 clock provides this function of signaling good data. While the 02 clock is low, the address bus lines are in the transition phase. As soon as the 02 clock goes high, the address bus and R/W lines present valid data to the peripherals. While the 02 clock is high, the data bus lines make their transitions. The data bus is signaled to be valid by the high to low transition of the 02 clock.

The phase 2 clock of the Atari 600XL computer is found on pin 32 of J2. The clock frequency is 1.8 MHz. A single clock cycle takes 550 nanoseconds. The 02 clock is low for 300 nanoseconds and high for the remaining 250 nanoseconds of the clock period.

The address and R/W lines transition to new states approximately 100 nanoseconds after the 02 line transitions from the high to low level. The fall time for 02 is approximately 15 nanoseconds, and its rise time is 50 nanoseconds. The fall and rise times were measured from the 10% to 90% levels of the clock signal.

The 02 clock's period is a measure of the speed at which the computer can execute instructions. An immediate mode ADD instruction takes two clock periods to complete. This corresponds to 1.1 microseconds. Many 6502 machine language instructions require only two clock cycles. The average is probably closer to three for many programs. Depending on the coding, the 600XL will execute instructions at a rate in the range of 300,000 to 900,000 instructions per second.

Numerous ground pins (designated GND) are provided on J2. A 5-volt DC power source is found on J2-48. Use J2-48 for supply voltages for small projects only.

Buffering the interface.

A system is required to interface a project to the computer's expansion connector. One method is to use a 6- to 9-inch ribbon cable assembly. The computer's end requires a 50-pin card-edge connector with contacts on 0.100-inch centers. If the project is done using wire wrapping, then a 50-pin socket connector works well on the other end. A double row 50-pin wire wrap header is required on the project's plugboard when the socket connector is used.

To complete the interface, the address and data bus should be buffered. This will improve the electrical noise immunity of the project's interface. Also, the computer will be protected against short circuits or other miswiring problems.

Figure 2 shows a memory expansion project with U1, U2 and U3 being the buffered interface. The TTL circuits in the 74LS244 work well for address buffers. U3 is an Octal Bus Transceiver. The direction of data flow through the transceiver is controlled by the CPU's R/W line. The R/W line is connected to the DIR input of U3.

The interface (shown on page 50) should be used with all projects that are designed for the 600XL's expansion bus.

Memory expansion project.

The expansion connector pin designations were determined by tracing the interconnections of ICs on

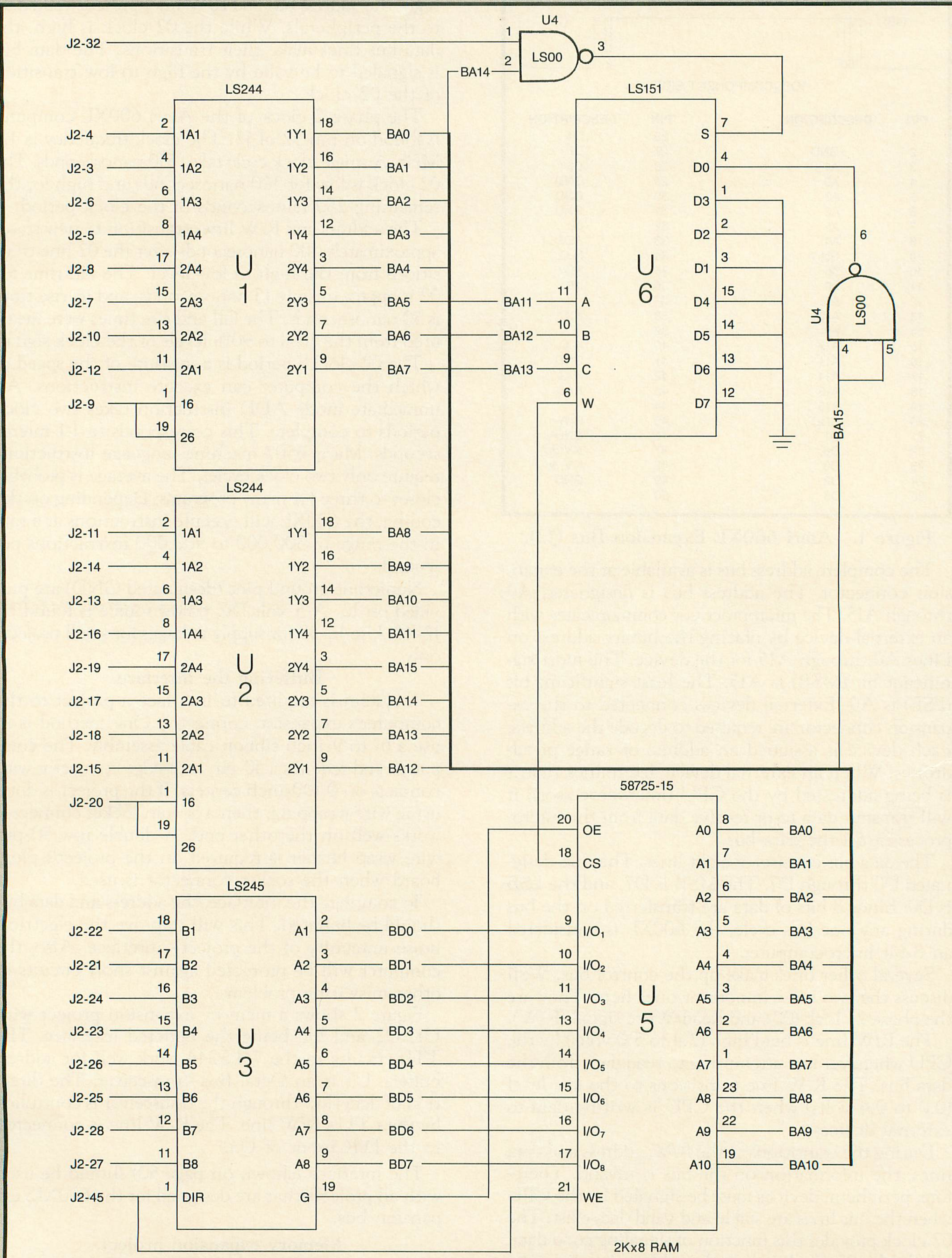


Figure 2. Atari 600XL 2K Memory Expansion Schematic.

the XL's printed circuit board. A partial schematic for the XL was produced. The schematic provided the information necessary to determine pin designations for most of J2, as shown in Figure 1. The following memory circuit was designed and built to verify the findings of goals 1 and 2.

The project provides 2K of additional volatile memory for the XL computer. It consists of three ICs. Two TTL circuits were used for decoding the address lines. The address was decoded for location 4000 hex and occupies 2K. A 24-pin 2K by 8-bit static RAM chip was used for the memory circuit. When the computer is turned on, this RAM will be used by the XL system for video memory. The memory circuit is shown in Figure 2. The power connections to each IC are shown in Figure 3.

ITEM NO.	VCC	GND
U1	20	10
U2	20	10
U3	20	10
U4	14	7
U5	24	12
U6	16	8

NOTE: PLACE 0.1 uF CAP ACROSS Vcc AND GND OF U5.

Figure 3. V cc and Ground Connections.

The RAM chip was wired directly to the buffered data bus and low order buffered address lines BA0 through BA10. The high order address bits are decoded by U4 and U6 to provide a chip select (CS) signal to U5. The CS signal is synchronized with the 02 clock by the first NAND gate of U4.

The CS line will go low, thus selecting the 2K RAM chip only if the following conditions exist.

BA15 = 0
BA14 = 1
BA13 = 0
BA12 = 0
BA11 = 0
02 = 1

The memory expansion project was built using wire wrap sockets on a plugboard. It took four hours to wire, using hand wire wrap tools.

After checking the wiring, the board was tested by plugging the project onto the 600XL's expansion connector. The power switch was turned to on, and operation verified by simply looking at the TV monitor.

As mentioned earlier, this extra memory is used for video memory by the computer. An assumption was made that, if the video was working, then the RAM card was okay. However, if the system didn't recognize the memory, then this assumption would not be correct! To verify that the Atari did, indeed, recognize the additional memory, it was simply asked with the *PRINT FRE(0)* statement. The answer was *15374*, which is 2048 bytes more than without the expansion card.

Conclusion.

You may remember from goal 2 that Atari's clock is relatively fast, compared to the more typical 1 MHz CPUs found in other home computers. Care must be exercised when selecting components for projects. Their access times must be short (i.e., 450 nanosecond memory chips won't function properly with the 600XL computer).

The memory expansion project for goal number 3 verified the information discovered in goals 1 and 2.

I hope the J2 pinout listing and design project will be useful to the reader, as an aid in designing individual projects for the expansion bus. Such projects could include digital I/O cards and A/D or D/A converters. □

Mr. Barton is the Manager of Software Products for Soft Systems Engineering, Inc. of York, Pennsylvania. He holds a Bachelor's Degree in Electrical Design Engineering Technology and enjoys designing hardware projects for home computers in his spare time.

(continued from page 47)

A real sense of danger, excitement and a little paranoia is possible. So **Suspect** has a good idea, but fails to take that idea seriously.

Throughout the adventure, references to other familiar mysteries are forever cropping up: for example, empty window box seats and *Arsenic and Old Lace*. Somehow, this adventure would be more fun, realistic and intriguing from a mystery point of view *without* these "dry humor" intrusions.

All of this leads me to a point that I feel I must make at this time, concerning Infocom's current crop of adventures. Are these wizards of the text adventure afraid to be serious for a change? There seems to be a cynicism underlying their adventures, from **Zork** to **Infidel** and now **Suspect**. I haven't had the opportunity to play **Cutthroats** as yet, but its packaging seems to suggest more of the same.

Perhaps it's time for Infocom to consider dividing the task of writing text adventures. There is no question that they have the technical know-how. I can't think of a software company that is even near to challenging their technical skills. Maybe they need an infusion of new blood from the creative writing end of text adventures. **Seastalker** and **The Hitchhiker's Guide to the Galaxy** were collaborative efforts, and it may be time to do more games along this line.

The future of text adventures is bright, and its brightest star is Infocom. If anyone is to succeed in bringing us more exciting adventure, I'm sure they will be a prime source.

I don't recommend the novice adventurer buy **Suspect**. He or she will be more frustrated than entertained by this game. I recommend **The Witness**, if you want to get a taste for mystery.

If you're a seasoned gamer, and you aren't upset by constant kidding, then **Suspect** may be just your cup of tea. It is certainly complex, detailed and imaginative. I just wished it took itself a little more seriously, both as a game and an example of truly interactive fiction. □

SUPERPRINTER PACKAGES

- Gemini 10X and U-Print A 319.00
- Gemini 10X and Apeface XLPS 319.00
- Panasonic 1091 and U-Print A 364.00
- Panasonic 1090 and U-Print A 279.00
- Legend 880 and U-Print A 309.00
- Prowriter and U-Print A 385.00

No additional ship. charges on printer packages in Cont. USA

Bring the trivia craze home with **P.Q. The Party Quiz Game** for the Atari 800 & 800XL (disk only) 49.95

ATARI

ATARI is a trademark of ATARI, INC.

Atari Inc. has cut all hardware and software prices. Please call for latest prices!!

PRINTERS

- Axiom AT-550 ... 279.00
- Epson ... Call
- Prowriter I ... 309.00
- Riteman ... Call
- Silver Reed ... Call
- Toshiba 1340 ... Call
- Toshiba 1351 ... Call
- Silver Reed ... Call
- Legend 880 ... 239.00
- Panosonic 1090 . 219.00
- Panosonic 1091 . 285.00

star micronics • inc

THE POWER BEHIND THE PRINTED WORD.

- Gemini 10X .. 245
- Gemini 15X .. 389
- Delta 10X 339
- Delta 15X 499
- Radix 10X 549
- Radix 15X 629
- Powertype 329

ATARI MODEM SPECIAL

- Hayes 300 & R-Verter 239.00
- Hayes 1200 & R-Verter ..... Call

No additional shipping for Modem packages in Cont. USA

MOSAIC

- 48K RAM 99.00
- 64K RAM/400 .. 149.00
- 64K RAM/800 + Cable Kit #1 .. 169.00
- 64K Expander for 600 XL 99.95

DISK DRIVES

- Indus GT ..... Call
- Percom ..... Call
- Astra 2001 ..... Call

MODEMS

- Hayes Amart Modem 300 ..... Call
- Mark II 79.00
- Mark VII/Auto Ans/ Auto Dial ..... Call
- Mark XII/1200 Baud .. Call
- MPP 1000 C ..... Call
- R-Verter Modem Adaptor 39.95

INTERFACES

- Aid interfast I ..... Call
- Ape Face XLPS ..... Call
- R-Verter Modem Adaptor 39.95
- MPP 1150 ..... Call
- U-Print A ..... Call

☆Call for prices on joysticks, printer cables, blank floppy disks, and other computer accessories.☆

A T A R I   S O F T W A R E

ACCESSORIES

- Ape-Link 29.95
- Gemini 10X 8K Upgrade .... Call
- Koala Pad-D 69.95
- Koala Pad-Cart 74.95
- Humpty Dump-D 29.95
- Monitors ..... Call
- Compuserve Starter 27.95
- Vidtex 29.95
- EIS Subscription Kit 64.95
- Analog Compendium 9.95
- Atari Assembler 14.95
- Compute's Machine Lang/Beg 12.95
- Inside Atari Dos 19.95
- Mapping the Atari 14.95
- Compute's 1st-3rd Books Atari-EA 12.95
- Printer Stand 15.95
- Omnimon 82.95
- Omniview 80 ..... 39.95
- Printer Ribbons ..... Call
- Ramrod XL ..... Call
- WICO Joysticks ..... Call
- MPP 64K Printer Buffer ..... Call
- U-Print 16, 32, or 64K Buffer . Call
- TAC III Joystick 12.95
- Starfighter Joystick 9.95
- Ramrod XL 99.95
- Muppet Keys (XL Only)-D . 54.95

ACTIVISION

- Decatholon-Cart 29.95
- Beamrider-Cart 29.95
- Hero-Cart 29.95
- Pitfall II-Cart 29.95
- River Raid-Cart 29.95
- Designer Pencil-Cart 29.95
- Space Shuttle- 29.95
- Lone Ranger-Cart 29.95
- Zenji-Cart 29.95

AVALON HILL

Call for items and prices

BRODERBUND

- Bank Street Writer-D 49.95
- Loderunner-D 23.95
- Mask of the Sun-D 27.95
- Operation Whirlwind-D ... 27.95
- Spelunker-D 20.95
- Stelth-D 20.95
- Whistler's Brother 20.95
- Print Shop-D 34.95
- Serpent's Star-D 27.95

CBS SOFTWARE

Call for items and prices

DATASOFT

- Bruce Lee-D/T 27.95
- Micropainter-D 23.95
- Lost Tomb-D/T 23.95
- Letter Wizard + Spell-D .. 54.95
- Conan the Barbarian-D/T . 27.95
- Mr. Do-D/T 27.95
- Dig Dug-D 20.95
- Pole Position-D 20.95
- Pacman-D 20.95

DISKETTES

- Dysan ..... Call
- Verbatim ..... For
- Certron ..... 10 pak
- Elephant ..... &
- Maxell ..... Quantity
- Memorex ..... Pricing.
- Ultra Magnetics .....
- BASF .....
- Wabash .....

ELECTRONIC ARTS

- Archon-D 29.95
- Pinball Construction-D 29.95
- M.U.L.E.-D 29.95
- Murder/Zinderneuf-D 29.95
- One on One-D 29.95
- Archon II-D 29.95
- Financial Cookbook-D 37.95
- Music Construction-D 29.95
- Realm/Impossibility-D ... 29.95
- Hard Hat Mack-D 29.95

EPYX

Call for items and prices

INFOCOM

- Deadline-D 29.95
- Enchanter-D 23.95
- Infidel-D 29.95
- Planetfall-D 24.95
- Sorcerer-D 29.95
- Starcross-D 29.95
- Suspended-D 29.95
- Witness-D 29.95
- Sea Stalker-D 24.95
- Cutthroats-D 24.95
- Suspect-D 24.95
- Hitchiker-D 24.95
- Zork I-D 24.95
- Zork II or III-D 27.95

MINDSCAPE

Call for items and prices

MICROPROSE

- Chopper Rescue-D/T 23.95
- Floyd/Jungle-D/T 23.95
- Hellcat Ace-D/T 23.95
- Mig Alley Ace-D/T 23.95
- NATO Commander-D/T .. 23.95
- Solo Flight-D/T 23.95
- Spitfire Ace-D/T 23.95
- Wingman-D/T 23.95
- Air Rescue I-D/T 23.95
- Challenger-D/T 23.95
- F-15 Strike Eagle-D/T 23.95

MISCELLANEOUS ATARI

- Diskey-D 34.95
- Ultra Disassembler-D 34.95
- Codewriter-D 39.95
- Star League Baseball D/T 23.95
- Star Bowl Football-D/T ... 23.95
- Master Type-D/Cart 27.95
- Flight Simulator II-D 37.95
- S.A.M.-D 41.95
- Castle Wolfenstein-D 20.95
- Compuserve Starter Kit 27.95
- Home Accountant-D 49.95
- Megafont II-D 19.95
- Monkey Wrench II-C 37.95
- Movie Maker-D 37.95
- Ultima III-D 41.95
- Jupiter Mission-D 34.95
- Boulder Dash-D/T 20.95
- Scraper Caper-Cart 34.95
- Miner 2049'er-Cart 34.95
- Spy Hunter-Cart/D 29.95
- Tapper-Cart/D 29.95
- Up 'N Down-Cart/D 29.95
- Beach Landing-D 27.95
- Pic Builder-D 27.95
- Astrochase-D 20.95
- Flip-Flop-D/T 20.95
- Sargon II-D/T 16.95
- Odesta Chess-D 49.95
- Millionaire-D 27.95
- Spy vs. Spy-D 23.95
- Adventure Writer-D 27.95
- MMG Basic Compiler-D ... 69.95
- Summer Games-D 27.95
- Pitstop II-D 27.95
- Gateway to Apshai-Cart .. 27.95
- Montezuma's Revenge-D .. 27.95
- Dragon/Pern-D 27.95
- Adventure Master-D 34.95
- Get Rich Series-D 34.95

MISC. ATARI (cont'd.)

- MPP Modern Driver-D 19.95
- Microfiler-Cart 34.95
- Microcheck-D 34.95
- Mr. Do's Castle-Cart 34.95
- Frogger II-Cart 34.95
- Net Worth-D 54.95
- Stickybear-D 27.95
- Windham Classics-D 19.95
- Omnitrend Universe-D 69.95
- Space Beagle-D 23.95
- Adventure Writer-D 41.95
- Beachhead-D 23.95
- Letter Perfect/Spell-D 74.95
- Harcourt/Bruce S.A.T.- ... 59.95
- S.A.G.E. Graphics Editor-D 39.95
- Strip Poker-D 23.95
- Millionaire-D 27.95
- Scroll of Abadon-D 23.95
- Ultima IV-D 41.95
- Raid Over Moscow-S 27.95
- Micro-League Baseball-D . 29.95
- Paper Clip-D ..... Call
- Home Pak-D 37.95
- Ultima II-D 41.95

OPTIMIZED SYSTEMS

- Action-Cart 69.95
- Basic XL-Cart 69.95
- MAC/65-Cart 69.95
- MAC/65 Tool Kit-D 27.95
- Action Tool Kit-D 27.95
- DOS XL-D 27.95
- Action Aid-D 27.95
- C65-D 59.95
- Handy-Writer-D ..... Call
- Postal Tool-D 41.95
- Print Tool-D 41.95

SCHOLASTIC

Call for items and prices

SCREENPLAY

Call for items and prices

SEGA

Call for items and prices

SIERRA ON-LINE

Call for items and prices

SPINNAKER

- Adventure Creator-D 22.95
- Aerobics-D 27.95
- All in the Color Caves-C .. 22.95
- Alphabet Zoo-Cart 22.95

SPINNAKER (cont'd.)

- Delta Drawing-Cart 22.95
- Pacemaker-Cart 22.95
- Fraction Fever-Cart 22.95
- Grandma's House-D 19.95
- Kids on Keys-Cart 22.95
- Kindercomp-Cart 22.95
- Search/Amazing Thing-D . 22.95
- Snooper #1-D 22.95
- Snooper #2-D 22.95
- Story Machine-Cart 22.95
- Trains-D 22.95

SSI

- Carrier Force-D 41.95
- Combat Leader-D/T 27.95
- Cosmic Balance II-D 27.95
- Cosmic Balance-D 27.95
- Broadsides-D 27.95
- War in Russia-D 55.95
- 50 Mission Crush-D 27.95
- Questron-D 34.95
- Rails West-D 27.95
- Bomb Alley-D 41.95
- Computer Ambush-D 41.95
- Galactic Adventures-D 41.95
- Computer Baseball-D 27.95
- Reforger 88-D 41.95
- Objective Kursk-D 27.95
- Breakthrough/Ardennes/D 41.95
- Field of Fire-D 27.95
- Imperial Galactium-D 27.95

SYNAPSE

- Air Support-D/T 23.95
- Alley Cat-D/T 16.95
- Blue Max-D/T 23.95
- Dimension X-D/T 23.95
- Drelbs-D/T 23.95
- Electrician-D/T 23.95
- Encounter-D/T 16.95
- Fort Apocalypse-D/T 23.95
- Necromancer-D/T 23.95
- New York City-D/T 23.95
- Pharoah's Curse-D/T 23.95
- Quasimodo-D/T 23.95
- Rainbow Walker-D/T 23.95
- Relax Stress Reduction Sys. . 79.95
- Shamus Case II-D/T 23.95
- Zepplin-D/T 23.95
- Synfile-D 34.95
- Syncalc-D 34.95
- Syntrend-D 34.95
- Synchron-D 27.95
- Syncomm-D 27.95
- Synstock-D 27.95

D-Disk T-Cassette Cart-Cartridge

To Order Call Toll Free
800-558-0003

For Technical Info, Order Inquiries, or for Wisc. Orders -
414-351-2007

NEW LOWER PRICES

ComputAbility™

NEW PRODUCTS

NO SURCHARGE FOR MASTERCARD OR VISA

Est. 1982

ORDERING INFORMATION. Please specify system. For fast delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 2 weeks to clear. Charges for COD are $3.00. School Purchase Orders welcome. In CONTINENTAL USA, include $3.00 shipping per software order. Include 3% shipping on all Hardware orders, minimum $3.00. Mastercard & Visa please include card # and expiration date. WI residents please add 5% sales tax. HI, AK, FPO, APO, Canadian orders — add 5% shipping, minimum $5.00. All other foreign orders, please add 15% shipping, minimum $10.00. All goods are new and include factory warranty. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call 414-351-2007 to obtain an RA# or your return will NOT be accepted for replacement or repair. Prices and availability are subject to change without notice.

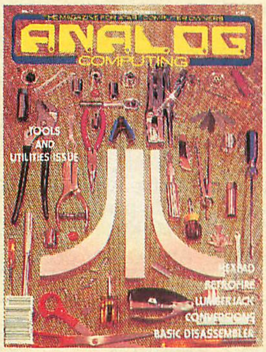
COMPUTABILITY
P.O. Box 17882
Milwaukee, WI 53217

ORDER LINES OPEN
Mon-Fri 11 AM - 7 PM CST
Sat 12 PM - 5 PM CST

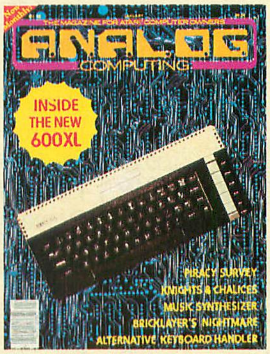
CIRCLE #125 ON READER SERVICE CARD

BACK ISSUES

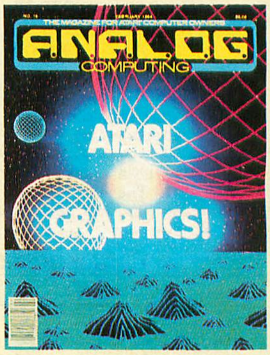
ISSUE 13 • Fine Scrolling Part 1 • Roundup • Space Assault • Observational Astronomy • CIO Routines



ISSUE 14 • Fine Scrolling Part 2 • Disassembler in BASIC • Hexpad • Lumberjack • Retrofire!

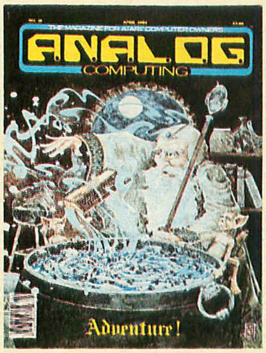


ISSUE 15 • Fine Scrolling Part 3 • Knights & Chalices • Music Synthesizer • Bricklayer's Nightmare • Keyboard Handler

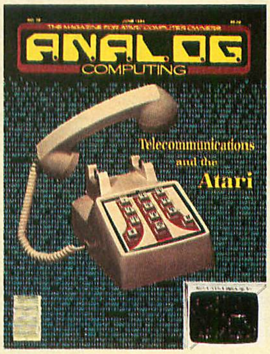


ISSUE 16 • Fine Scrolling Part 4 • Create-A-Font • Bar Chart Subroutine • Shooting Stars • 3-D Object Rotation

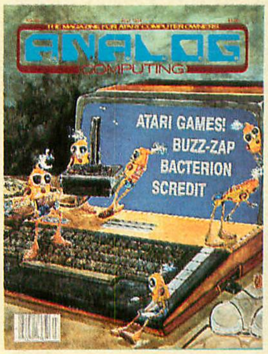
ISSUE 17 • Planetary Defense • Disk Miser • Live without DOS • Boot Camp • D:CHECK2



ISSUE 18 • Crash Dive • File'em • Munch'in Climb'in • H:BUG • Adventure Reviews

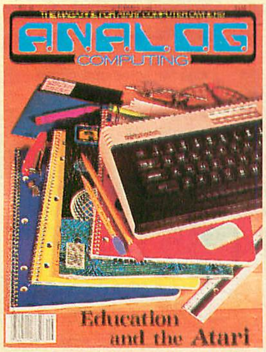


ISSUE 19 • Battle in the B-Ring • A Look at Modems • Bulletin Board Systems • Siege • Touch-Tone® Dialer

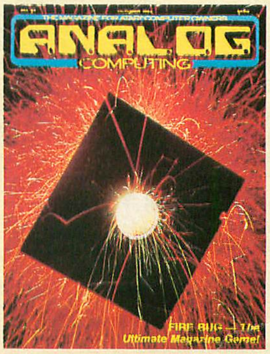


ISSUE 20 • Buzz-zap! • Scredit • AlterDOS • Bacterion! • ConTEXT

ISSUE 21 • Selecting your Perfect Printer • Matt∗Edit • Graph E's • Spy Plane • Printer Survey



ISSUE 22 • Math Attack • Micro-Puzzler • Typing Evaluator • Air Attack • Mathman • The Reading Program



ISSUE 23 • Fire Bug • Minicomp • Dark Horse • Climber • P/M Creator/Animator

ISSUE 24 • Circuit Database • Bopotron! • XL-DOS • Cassette Compressor • Race in Space • Unicheck

All back issues $4.00 each

Send check or money order to:
ANALOG Computing Back Issues
P.O. Box 615, Holmes, PA 19043

Issues 25 and 26 are also available, along with a limited number of issues 2 and 11.

MasterCard and VISA orders call:
1-800-345-8112
in PA. 1-800-662-2444

REVIVE

A disk file recovery utility

16K Disk

by Philip Altman

All disk users have had the unpleasant experience of mistakenly deleting a wanted file. With a backup copy, there's no problem. But what if you haven't been so careful? There's no simple way to recover a deleted file with Atari DOS. In this article, I present **Revive**, a machine language modification for DOS 2.0, which gives you the power to rescue a scratched file with ease. **Revive** works with all Atari computers using Atari DOS 2.0.

Atari disk structure.

In order to understand how this program works, we will need to discuss some aspects of the Atari disk format. A typical single-density DOS 2.0 disk is organized into 720 sectors (1 to 720) of 128 bytes each. Certain sectors are reserved for the system, so not all are available for storing program data. The first three sectors are the boot sectors, which are read by the operating system when the computer is turned on.

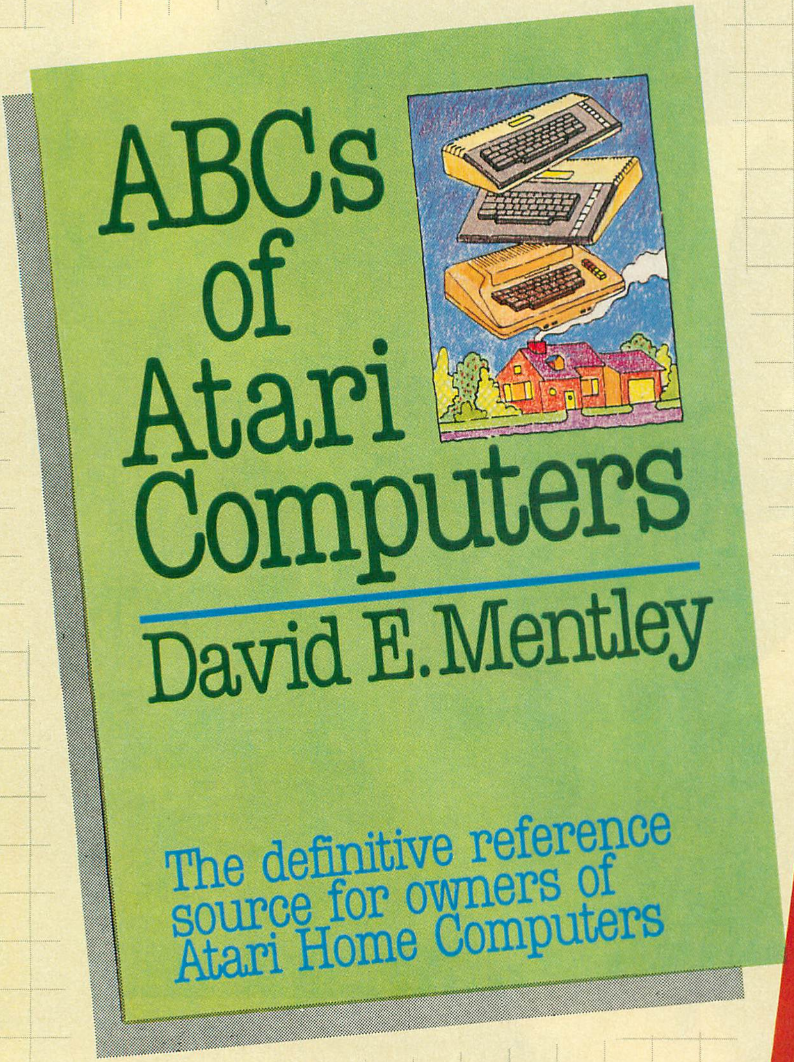
Beginning with sector 361, eight sectors are allocated to the disk directory, which can hold up to sixty-four files. Each directory entry is 16 bytes long and, along with the filename, contains information about the file type, its length in sectors and the starting sector number. One more sector, the Volume Table of Contents (VTOC, sector 360), is reserved for system use. Here, DOS keeps track of which sectors are already in use (i.e., assigned to files in the directory), and which are free.

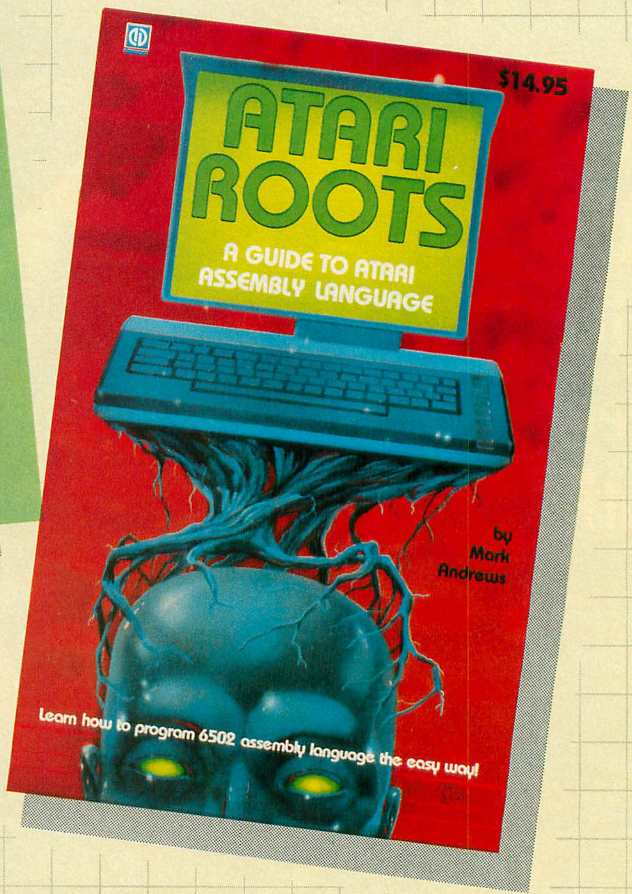
The remaining 707 sectors (sector 720 is unused by DOS) are data sectors. Each contains 125 data bytes (0-124) and 3 control bytes. These tell which file the sector belongs to, the number of bytes in the sector and which sector number comes next in the file.

(continued on page 56)

FREE!

Your choice of Atari guides...





ABCs of Atari Computers

by David E. Mentley Sugg. Retail **$14.95**

Full of useful information without being overly technical, *ABCs* addresses a broad spectrum of Atari topics. Arranged in alphabetical order, many definitions are clarified with accompanying programs. Written in clear, concise terms, this book can be used by beginners and experts alike.

OR

Atari Roots

by Mark Andrews Sugg. Retail **$14.95**

The easiest guide ever to learning Atari assembly language. Using a "hands on" approach, you'll get started programming this fast, efficient language quickly. This much needed book teaches assembly language using the Atari Assembler Editor and the popular MAC/65 Assembler. Programs will work on all Atari computers.

When you order a new subscription to **ANALOG Computing**, you'll receive your choice of *ABCs of Atari Computers* or *Atari Roots* ***FREE***. Our supply is limited, so send in your subscription card today!

Mail in the special subscription envelope inserted in this magazine, with your selection checked,

OR

CALL TOLL FREE
(800) 345-8112
in Pennsylvania
(800) 662-2444

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

P.O. BOX 615 HOLMES, PENNSYLVANIA 19043

What happens when you delete a disk file? After processing the filename, DOS searches the disk directory for a match. If the file is found, DOS sets a bit in the directory entry indicating that the file is deleted. Then DOS reads each sector in the file, finds it in the VTOC, frees it for re-use and increments the free sector count. The directory search continues for the next match. If one is found, the process is repeated until no further matching entries are found.

How it works.

You've probably already figured out how **Revive** works. The program simply reverses the DELETE process.

First, the DOS directory search routine is instructed not to ignore deleted entries when testing for a filename match. When a match is found, the file is restored to active status. Then, each VTOC sector bit for the file is reallocated as each sector in the file is read, and the free sector count is decremented.

Revive has one important limitation. It cannot reliably be used if any data has been written to the disk after a file was mistakenly deleted. The reason is that some of the sectors freed when the file was deleted may subsequently have been allocated to the new data file. **Revive** may then also assign these sectors to the rescued file.

Reviving.

Get started by carefully typing in Listing 1. After proofreading the program, SAVE it to disk. Insert a disk in drive 1 and RUN the program. A binary file, D1:REVIVE, will then be written.

For those interested in assembly language, the source code in MAC-65 format is presented in Listing 2.

Revive is designed to be loaded from the DOS 2.0 utilities menu with the load binary file (option L.) command. You'll see a modified menu with the **Revive** command (option D.) highlighted in inverse. Enter the filename to **Revive**, according to DOS 2.0 syntax.

The menu reappears when the task is complete. **Revive** responds with *ERROR 168* (command invalid) if a file isn't found or hasn't been deleted. Return to the standard menu by selecting option M. (run at address) and entering *179F* (make sure there is a DOS.SYS-DUP.SYS disk in drive 1). Enter *E477* instead, if you want to reboot the system. ☐

Listing 1.
BASIC listing.

```
10 REM REVIVE LOADER by P.ALTMAN 10/84
20 ? "K":TRAP 130:OPEN #1,8,0,"D:REVIV
E":? "WRITING BINARY FILE":?
30 PUT #1,255:PUT #1,255:PUT #1,54:PUT
 #1,31:PUT #1,73:PUT #1,31
40 FOR I=1 TO 20:READ D:PUT #1,D:NEXT
I
50 PUT #1,181:PUT #1,31:PUT #1,186:PUT
 #1,31
60 FOR I=1 TO 6:READ D:PUT #1,D:NEXT I
70 PUT #1,13:PUT #1,35:PUT #1,18:PUT #
1,35
80 FOR I=1 TO 6:READ D:PUT #1,D:NEXT I
90 PUT #1,247:PUT #1,33:PUT #1,148:PUT
 #1,34
100 FOR I=1 TO 500:TRAP 110:READ D:PUT
 #1,D:NEXT I
110 PUT #1,224:PUT #1,2:PUT #1,225:PUT
 #1,2:PUT #1,119:PUT #1,32
120 ? :? "DISK WRITE SUCCESSFULLY COMP
LETED":GOTO 140
130 ? "DISK ERROR!"
140 CLOSE #1:END
150 DATA 77,111,100,105,102,105,101,10
0,32,98,121,32,80,46,65
160 DATA 76,84,77,65,78
170 DATA 210,197,214,201,214,197
180 DATA 82,69,86,73,86,69
190 DATA 169,49,72,169,244,72,160,2,18
5,124,29,136,201,58,208
200 DATA 248,185,124,29,201,65,144,2,1
69,49,41,3,133,33,32
210 DATA 100,17,32,158,14,169,4,141,87
,15,32,33,15,169,27
220 DATA 141,87,15,176,8,172,5,19,185,
1,20,48,3,76,191
230 DATA 18,169,66,153,1,20,173,7,19,1
0,10,157,129,19,185
240 DATA 4,20,157,137,19,185,5,20,157,
138,19,32,113,16,32
250 DATA 146,16,174,1,19,32,36,16,169,
81,141,241,16,169,56
260 DATA 141,249,16,169,233,141,250,16
,141,1,17,32,197,16,169
270 DATA 17,141,241,16,169,24,141,249,
16,169,105,141,250,16,141
280 DATA 1,17,32,23,16,144,212,32,149,
16,32,155,18,208,5
290 DATA 169,1,32,27,18,76,119,32
```

•

CHECKSUM DATA.

(see page 32)

```
10 DATA 268,207,681,909,153,887,30,891
,187,371,625,614,604,125,549,7101
160 DATA 727,515,48,701,281,101,774,90
,64,294,673,705,837,777,6587
```

•

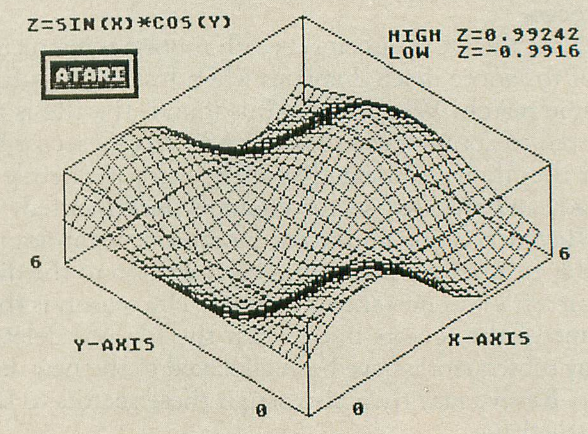
Listing 2.
Assembly listing.

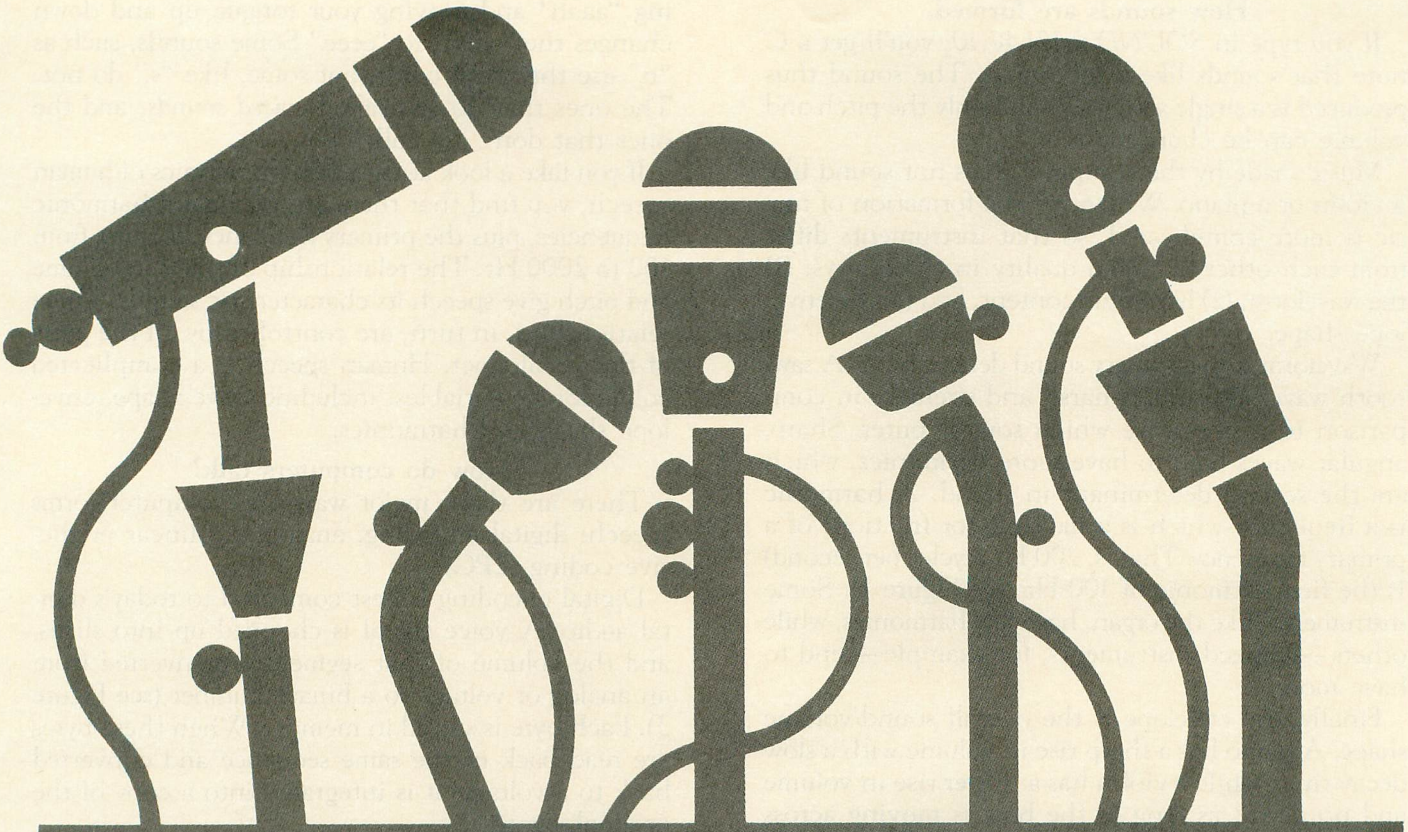
```
; ************************
; *       REVIVE         *
; *                      *
; * File Rescue Utility  *
; *                      *
; *   by Philip Altman   *
; *                      *
; ************************
;
; *** DOS EQUATES ***
;
ICDNOZ   =   $21     ;DRIVE #
FNDCODE  =   $ØE9E   ;FNAME DECODE
SFDIR    =   $ØF21   ;DIR SEARCH
RDNSO    =   $1Ø17   ;READ SECTOR
WRTDIR   =   $1Ø71   ;WRITE DIRECTORY
RDVGO    =   $1Ø92   ;READ VTOC
WRVTOC   =   $1Ø95   ;WRITE VTOC
FRESECT  =   $1ØC5   ;FREE/ALLOC SEC
SETUP    =   $1164   ;INIT DOS I/O
DD1      =   $121B   ;REWRITE BT SECS
TSTDOS   =   $129B   ;TST FOR DOS.SYS
ERDVDC   =   $12BF   ;INVALID CMD
CURFCB   =   $13Ø1   ;CURRENT FCB
CDIRD    =   $13Ø5   ;DIR DISPLACEMNT
SFNUM    =   $13Ø7   ;CURRENT FILE #
FCBFNO   =   $1381   ;JUSTIFIED FILE#
FCBCSN   =   $1389   ;CURRENT SECTOR
;
; *** DUP EQUATES ***
;
PAR      =   $1D7C   ;PARAMETER AREA
MENU     =   $2Ø77   ;DUP ENTRY
CIOER    =   $31F6   ;CIO ERR RET ADR
;
         *=  $1F36
;
         .BYTE "Modified by P.ALTMAN"
;
         *=  $1FB5
;
         .BYTE +$8Ø,"REVIVE"
;
         *=  $23ØD
;
         .BYTE "REVIVE"
```

```
;
        *=  $21F7
;
        LDA # >CIOER-2 ;STACK CIO
        PHA            ;ERROR RETURN
        LDA # <CIOER-2 ;ADRS FOR DOS
        PHA
        LDY #2         ;FIND ":" IN
FD      LDA PAR,Y      ;FILE NAME
        DEY
        CMP #':
        BNE FD
        LDA PAR,Y      ;TEST DRIVE ID
        CMP #'A
        BCC GOTNUM     ;BR IF NUMERIC
        LDA #'1        ;DEFAULT=#1
GOTNUM  AND #3         ;CONV FROM ASCII
        STA ICDNOZ     ;SAVE DRIVE #
        JSR SETUP      ;INIT DOS I/O
        JSR FNDCODE    ;DECODE FNAME
        LDA #4         ;DON'T SKIP
        STA $0F57      ;DELETED FNAMES
        JSR SFDIR      ;SEARCH DIR
        LDA #$1B       ;FOR DELETD FILE
        STA $0F57      ;RESTORE CODE
        BCS REVXT      ;BR IF ABSENT
        LDY CDIRD      ;CUR DIR DISPL
        LDA $1401,Y    ;MAKE SURE WAS
        BMI DELFIL     ;DELETED FILE
REVXT   JMP ERDVDC     ;ELSE ERROR
DELFIL  LDA #$42       ;MAKE GOOD
        STA $1401,Y    ;DOS 2 FILE
        LDA SFNUM      ;GET FILE #
        ASL A          ;SHIFT LEFT X2
        ASL A
        STA FCBFNO,X   ;SAVE FOR ID
        LDA $1404,Y    ;GET FIRST FILE
        STA FCBCSN,X   ;SECTOR #
        LDA $1405,Y    ;MAKE IT CURRENT
        STA FCBCSN+1,X ;SECTOR #
        JSR WRTDIR     ;FIX DIRECTORY
        JSR RDVGO      ;READ VTOC
        LDX CURFCB     ;RESTORE FCB PTR
        JSR $1024      ;GET CUR SEC PRM
SECLP   LDA #$51       ;CHANGE FRESECT
        STA $10F1      ;SO VTOC BYTES
        LDA #$38       ;RE-ALLOCATED
        STA $10F9      ;TO FILE
        LDA #$E9
        STA $10FA
        STA $1101
        JSR FRESECT    ;UPDATE VTOC
        LDA #$11       ;RESTORE FRESECT
        STA $10F1      ;IN CASE ERROR
        LDA #$18       ;IN SECTOR READ
        STA $10F9
        LDA #$69
        STA $10FA
        STA $1101
        JSR RDNSO      ;READ NXT SECTOR
        BCC SECLP      ;DO UNTIL EOF
        JSR WRVTOC     ;FIX VTOC
        JSR TSTDOS     ;WAS IT DOS.SYS?
        BNE EXIT
        LDA #1         ;SHOW DOS THERE
        JSR DD1        ;REWRITE BT SECS
EXIT    JMP MENU       ;BACK TO DUP
;
        *=  $02E0
        .WORD MENU
        .END
```

•

NEXT ISSUE:
ZPLOTTER





Cheep Talk

Build your own speech synthesizer

16K Cassette or Disk

by Lee Brilliant, M.D.

What do *2001*, *Star Trek* and *Star Wars* all have in common? Among other things, they all have that wonderful invention of the future, the talking computer! *2001* had HAL, the monotone malevolent; *Star Trek*'s counterpart was decidedly female; and from *Star Wars*, we have the proper-English-speaking C3-PO and his "binary"-speaking (don't *you* speak binary?) pal, R2-D2.

Now, your very own Atari home computer can be in the same class as these. All you need is a few hours and a few dollars to enter the world of **Cheep Talk**. Why a do-it-yourself speech synthesizer when there are several excellent speech products on the market already? Well, aside from the personal satisfaction of building it yourself, add-on devices are expensive in money, and software speakers are expensive in RAM. For under forty dollars, you can build your own **Cheep Talk**—it's easy to use and occupies little memory.

First, a little theory.

The subject of speech synthesis is very complex, and, though I make no pretense of this being a complete discourse on the subject, you need to know some language theory.

The English alphabet has twenty-six letters in it and about forty separate sounds called *phonemes*. The reason that there are more sounds than letters is that many letters have multiple sounds, or that multiple letters make single sounds. For example, the letter *g* can be hard, as in *go*, or soft, as in *gem*.

Actually, there are hundreds of variations of these phonemes, depending on how finely you can divide sounds. A sound such as a hard *g* can have slight differences in intonation and duration, depending on its location within a word and which vowels it associates with. These variations of phonemes are called *allophones*. **Cheep Talk** uses fifty-nine allophones.

How sounds are formed.

If you type in *SOUND 0,121,10,10*, you'll get a C note that sounds like a toy organ. The sound thus produced is a single sine wave, and only the pitch and volume can be changed.

Music made by the computer does not sound like a violin or a piano. Why not? The formation of music is more complicated, so that instruments differ from each other in sound quality in three ways: (1) the waveform, (2) harmonic content, and (3) the envelope shape.

Waveform is the primary sound determinant. A sawtooth waveform sounds harsh and sirenlike in comparison to a sine wave which sounds purer. Sharp, angular waves tend to have more harmonics, which are the second determinant in sound. A harmonic is a frequency which is a multiple (or fraction) of a primary frequency. That is, 200 Hz (cycles per second) is the first harmonic of 100 Hz (see Figure 1). Some instruments, like the organ, have few harmonics, while others—stringed instruments, for example—tend to have more.

Finally, the envelope is the overall sound/volume shape. A piano has a sharp rise in volume with a slow decay time, while a violin has a slower rise in volume and no decay, as long as the bow is moving across the string.

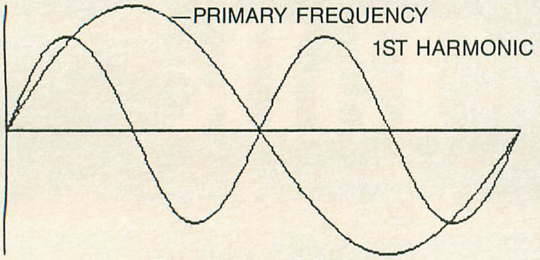


Figure 1. Harmonics.

Human speech is created in much the same manner as music. The vocal cords provide a primary wave shape called the *glottal pulse* (see Figure 2).

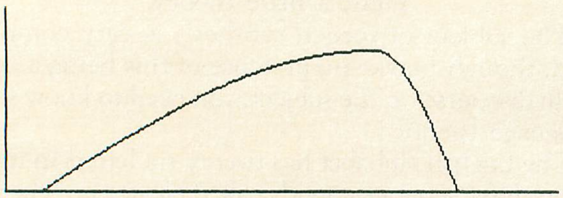


Figure 2. Glottal pulse.

The male pitch is around 141 Hz, while the female pitch is around 233 Hz. As with musical instruments, the tonal quality of a voice depends on the harmonics, which are created by the cavities of the vocal tract, including the sinuses, mouth, tongue, throat, etc. You can alter the sounds of speech by altering the size of these cavity resonators.

For example, say "aaah" while opening and pursing your lips. Doing this changes the "aaah" to "oh." Saying "aaah" and moving your tongue up and down changes the "aaah" to "eeee." Some sounds, such as "b," use the vocal cords, but some, like "s," do not. The ones that do are called *voiced sounds*, and the ones that don't are called *voiceless*.

If you take a look at the entire harmonics of human speech, you find that there are two major harmonic frequencies, plus the primary frequency, ranging from 100 to 2000 Hz. The relationship of these in volume and pitch give speech its characteristic sounds. These relationships, in turn, are controlled by all the parts of the vocal tract. Human speech is a complicated collection of variables, including wave shape, envelope shape and harmonics.

How do computers talk?

There are three major ways the computer forms speech: digital encoding, analog and linear predictive coding (LPC).

Digital encoding is best compared to today's digital audio. A voice signal is chopped up into slices, and the volume of that segment is converted from an analog or voltage to a binary number (see Figure 3). Each byte is stored in memory. When these bytes are read back in the same sequence and converted back to a voltage, it is integrated into a copy of the original signal.

To be of good quality, the signal must be sampled at a rate double the highest frequency to be used. If we cut off speech at 5000 Hz, then the sample rate should be at least 10000 bytes per second! At that rate, the word *hello* could use 5 to 10K of memory!

An example of this type of speech reproduction (not true synthesis) is the phone company's automated operators. While the reproduction quality is excellent, the vocabulary is limited. SAM (Software Automated Mouth) is a software example of this technique. It drives the sound channels directly, in machine language (see *De Re Atari*, section 7-21). Instead of a vocabulary of fixed words, SAM's is made up of phonemes which can be hooked together to give a large vocabulary. Nevertheless, it's still memory hungry.

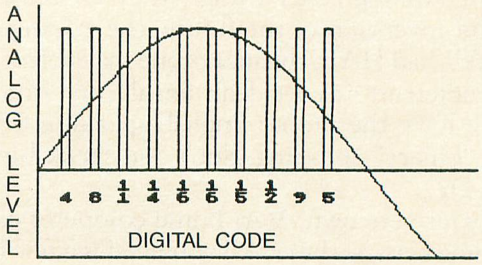


Figure 3.

Analog speech is based on a low frequency pulse generator for voiced sounds and a white noise generator for voiceless sounds. These are passed through tunable filters to select which harmonics will go on to the amplifier, much like today's multichannel audio equalizers.

The pitch and volume of the generators can also be varied, giving about nine separate adjustments which need to be continuously varied to create reproductions of voice. It would be difficult, indeed, for a human to twiddle all those knobs fast enough to make intelligible speech.

Instead, the computer can feed the processor appropriate parameters. Of course, the more frequently you feed the parameters, the more accurately you can define the sounds, and the more intelligible the speech. Also. . .the more memory is used. Thus, when a series of computer-defined parameters is passed to the processor, it sounds like speech.

LPC is somewhat "between" the two previous techniques. In structure, it is similar to analog synthesis using parameters to control 12-stage filters, volume controls and oscillator frequencies. Speech produced is better than analog, because the sample rate is higher. But, despite the high sample rate, computer memory requirements are very low.

This feat is accomplished by an onboard microprocessor and 8 to 16K of built-in ROM. The processor calculates most of the control settings based on complicated formulae in its ROM. So, while the control parameters may be updated from ROM every twenty or so milliseconds, the onboard microprocessor is calculating hundreds of intermediate values.

Your Atari doing the same thing in BASIC *would* take five to ten minutes to calculate what the speech processor does every twenty milliseconds. In this manner, the LPC system greatly improves the sample rate and the quality of speech, without the memory overhead of the digital system.

The SPO256-AL2 speech processor used in **Cheep Talk** is of the LPC type, as is the Texas Instruments **Speak 'n Spell**. LPC speech is not truly synthetic, because the control settings are obtained from digitally encoded speech patterns, and so are modeled after an original source.

The SPO256-AL2 does not have a fixed vocabulary; rather, it has a set of allophones programmed into its ROM which can be strung together to make words. Vocabulary is almost unlimited.

Building Cheep Talk.

Construction is fairly straightforward. Except for the 22pf capacitors, all parts are available at Radio Shack. The manufacturers of the SPO256-AL2 call for a 3.12 MHz crystal which can be specially ordered, but I used a 3.579 MHz color TV crystal without any difficulty. Be sure you get the right speech chip; Radio Shack carries two versions of the SPO256 speech processor.

The 28-pin socket and all components should be soldered in place *before* the integrated circuit is installed. Wiring can be done on perf board with point-to-point wiring or on a printed circuit. If you're into making your own board, then use the pattern shown.

After all parts except the chip are installed, wire in the joystick plugs. If you can obtain a couple of cords from old joysticks, and if they have all six wires called for, then use them. Otherwise, use 9-pin plugs and 6-conductor ribbon cable made by splitting the 25-conductor cable (see parts list).

Pinouts.

Connections for the joystick plugs are simple. If you are using the printed circuit, then the connections are numbered 1 to 4, plus 7 and 8. These correspond to the pin numbers on the plugs.

Figures 4, 5 and 6 show the actual pin connections. The views of the plugs are looking at the end which interfaces with the computer, so that the cable goes away from you. The term *port A* corresponds to joystick plug 1; *B* is plug 2.

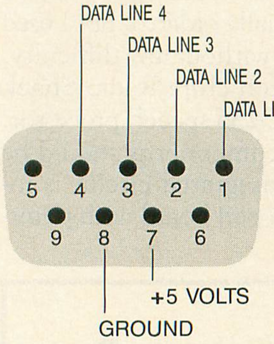


Figure 4. Joystick plug 1.

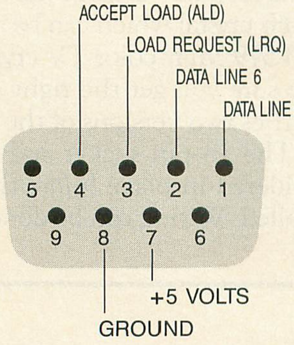
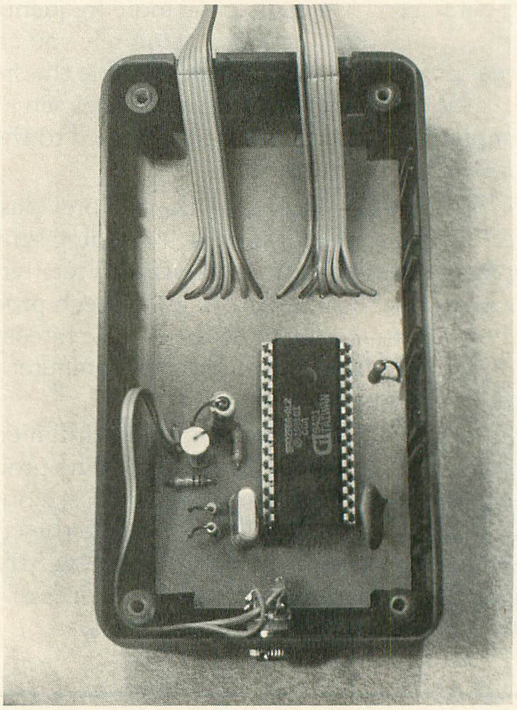


Figure 5. Joystick plug 2.

The hoods for these plugs will not fit the computer without your removing the tabs at the ends, which hold the plug. Therefore, you need to use small flathead machine screws to secure the plugs to the hoods.

Double check the order of your pin connections and verify that no solder has bridged any connections on the circuit board. Mount the circuit board and earphone jack in the box and attach the audio output leads to the jack. Make an opening for the joystick cables and label the plugs, so you put them in the right sockets.



Internal arrangement of the Cheep Talk voice synthesizer.

Triple check all wiring, then install the chip. *Note* that there is a small dot on the top of the chip over pin 1. Locate the chip properly and install in the socket by pushing gently and evenly, being sure not to bend any pins. While digital chips are very forgiving of wrong wiring, they give up when faced with static electricity. So ground yourself before handling the chip and leave it in its black conductive foam until the last minute. Close up the case.

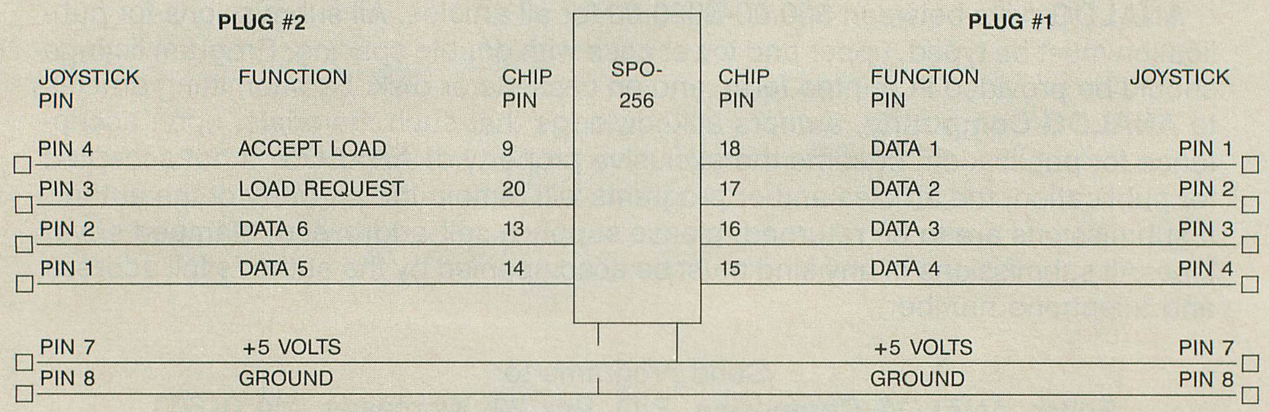


Figure 6. Pinout schematic.

Once assembled, plug in the joystick cables and connect up to an amplifier. I use a small unit sold by Radio Shack, which has its own amplifier and speaker in a compact case. You can also connect **Cheep Talk** to your Atari's cassette audio input line, so it will play through the TV.

To do this, either put a clip lead on pin 11 of the serial jack, or open the plug on your serial cable and solder a wire to the number 11 connector, bringing it out the back of the plug. Attach it to the center lead of a miniature phone plug and connect to **Cheep Talk**. No ground wire is needed.

If the volume isn't enough, jump C3 to pin 24 of the IC and eliminate C2 and R2. Stay away from stereo equipment or plug-in amplifiers, unless they're properly grounded or isolated. Failure to heed this warning may fry your computer! Power for **Cheep Talk** comes from your Atari through the joystick ports.

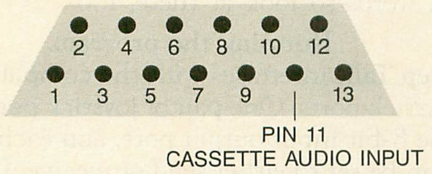


Figure 7. Serial plug.

Using Cheep Talk.

To use your new toy, just type in the short program and run it with **Cheep Talk** installed. If everything is right, the computer should talk to you and say, "Hello. This is **ANALOG Computing** magazine **Cheep Talk**." If it doesn't, recheck all wiring, especially the order of the wires in the joystick plugs and, finally, your program typing. If it works, onward!

To make **Cheep Talk** say what you want, we must return to language theory briefly. The SPO256-A2 has fifty-nine allophones (or speech sounds), and five pauses numbered 0 to 63. The trick is to know which ones to use.

Table 1 contains a listing of the allophones by phonetic groupings. Don't worry, it's not that hard. To code a word into data statements, you first need to break the word down to its sounds. Spelling has *no bearing* on sounds! For example, you will find no letter *c*. Instead, you use *s* if it is a soft sound, and *k* if it is hard.

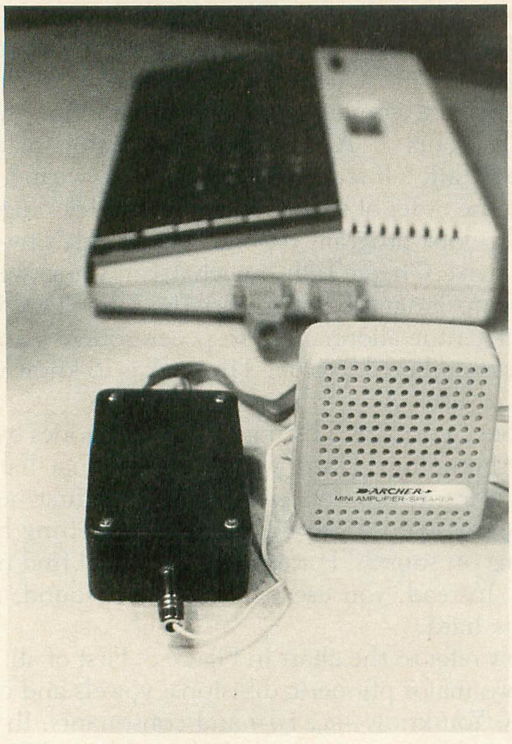
Now refer to the chart in Figure 8. First of all, there are two major phonetic divisions: vowels and consonants. You know—*a,e,i,o,u* and consonants. In turn, there are three types of vowels: long (like in *bE*), short (like in *bEd*) and diphthongs or blends (like in *bEAr*).

(continued on next page)

Table 1. Allophone listing.

PAUSES				
	0	10 ms.	(PA1)	Before *p, t, k, ch* and between words
	1	30 ms.	(PA2)	Same
	2	50 ms.	(PA3)	Same
	3	100 ms.	(PA4)	Between clauses and sentences
	4	200 ms.	(PA5)	Same
VOWELS				
SHORT:				
*	a	26	(AE)	bAt, mAp
*	e	7	(EH)	sEt, tEnt
*	i	12	(IH)	sIt, kItten
*	o	24	(AA)	pOt, mOp
*	u	15	(AX)	Up, lApel, trUck
LONG:				
	a	20	(EY)	cAke, grEAt, grAte
	e	19	(IY)	spEAk, pEEk, pEOple, pennY
	i	6	(AY)	kIte, skY, mIght
	o	53	(OW)	gO, snOW, clOse, zOne
	u	22	(UW1)	After *y* sounds: yOUth
		31	(UW2)	In monosyllable words: twO, tOO, tO, shOE, fOOd
DIPHTHONGS (BLENDS)				
R COLORED:				
	ar	59	(AR)	fARm, gARment
	air	47	(XR)	hAIR, stARE
	er	51	(ER1)	stIRring, fURniture, lettER
		52	(ER2)	In monosyllable words: bIRd, fERn, bURn
	ear	60	(YR)	hEAR, pEER, IRresistible
	or	58	(OR)	fORturn, stORe
OTHERS:				
*	aw	23	(AO)	AWful, sOng, tALk
	oi	5	(OY)	vOIce, tOY
	ow	32	(AW)	sOUnd, dOWn
*	uh	39	(UH)	cOOkie, fUll
	ul	62	(EL)	littLE, angLE, gentLE
CONSONANTS				
STOPS — VOICED:				
	b	28	(BB1)	Final position, between vowels and in blends with other consonants: riB, fiBer, BLend
		63	(BB2)	Initial position before a vowel: Bat
	d	21	(DD1)	Final position: saiD, enD
		33	(DD2)	Initial position and in blends with other consonants: Down, DRain
	g	36	(GG1)	Before the vowels *ā, ē, ĕ, ĭ, ear*, and *air*: Gear, Gift, Gate
		61	(GG2)	Before the vowels *ō, oi, ū, ŭ, uh*, and in blends with other consonants: Gun, GReen
		34	(GG3)	Before the vowels *ă, ī, ŏ, ar, er, or, aw, ow*, blends in the middle of words, and final position: Gap, aGRee, peG
CONSONANTS continued				
STOPS — UNVOICED:				
	p	9	(PP)	Please, amPle, triP
	t	17	(TT1)	Before *s* and in final position blends with other consonants: gaTeS, STop
		13	(TT2)	ALl other positions
	k	42	(KK1)	Before vowels *ā, ă, ē, ĕ, ī, ĭ, ŭ, er, air, ear*, and initial blends with other consonant: Cute, Kit, Came, sCream, Clown
		41	(KK2)	Final position and final blends with other consonants: speaK, cliCK, taSK
		8	(KK3)	Before vowels *ar, aw, ō, oi, or, ū, uh*, and in initial blends with consonants: Coin, Collide, sCream, Clown
FRICATIVES — VOICED:				
	v	35	(VV)	Vest, proVe, eVen
	dh	18	(DH1)	Initial *th* sound: THis, THey, THen
		54	(DH2)	Final position and between vowels: baTHing
	z	43	(ZZ)	Zoo, phaSE
	zh	38	(ZH)	beiGE, pleaSure
FRICATIVES — VOICELESS:				
*	f	40	(FF)	Find, Finger
*	th	29	(TH)	THin, wiTH
*	s	55	(SS)	Sit, Single
	sh	37	(SH)	SHirt, wiSH
	h	27	(HH1)	Before vowels *ā, ă, ē, ĕ, ĭ, ear, air*: Hat, Hair
		57	(HH2)	Before vowels *ō, oi, ū, uh, aw, or, ar*: Harm, Hoist, Home
	wh	48	(WH)	WHite, WHim, tWenty
RESONANTS				
	w	46	(WW)	We, War, langUage
	r	14	(RR1)	Initial position: Red, Robot, WRite
		39	(RR2)	In initial blends with consonants: BRown, GRease. In the middle, use R-colored vowels.
	l	45(LL)		Like, heL-Lo, steeL
	y	49	(YY1)	In blends: bEAuty, comp(Y)uter, c(Y)ute
		25	(YY2)	Initial position: Yes, Yarn
NASALS				
	m	16	(MM)	Milk, alarM, aMple
	n	11	(NN1)	Before vowels *ā, ă, ē, ĕ, ī, ĭ, ū, ŭ, er, ear, ow, air*, and final consonant blends: Name, Now, Nervous, earN, turN
		56	(NN2)	Before vowels *ō,ŏ,oi, or, ar, uh*: Note, North, Noise
	bg	44	(NG)	striNG, aNGer
JH-CH				
	jh	10	(JH)	fuDGe, inJure
	ch	50	(CH)	CHurch, CHeer, feaTure, matCH

* *These sounds can be doubled for long sounds.*



The finished Cheep Talk synthesizer.

Consonants are either stops, which are short, explosive sounds like *buh* or *guh*, fricatives like *v*, resonants like *r*, or nasals like *m*, which you can't say with your nose pinched. Stops and fricatives can be voiced or voiceless, depending on whether the vocal cords are used—as in *b* or whispered like *p*. There are two consonant blends which do not fit: *ch* and *j*, because they are both fricatives and voiced stops.

To use Table 1, you must break your words down into sounds. Decide if each is a vowel or consonant, and which type, then look it up in the table. Some sounds have several versions, such as *g*, so read the short rules next to the sound. Once you have the right sound, find its number and place it in order in the data statement. Also included are the phonetic code and some example words.

Let's try "hello." The sounds are *h, e, l, o*. *H* is a consonant, a fricative type, which is voiceless. Its number is 27. There are two *h* sounds. Number 27 goes with the short *e* (code EH), while 57 goes with other vowels. The *e* is a short vowel whose number is 7; *l* is a resonant consonant numbered 45; and, finally *o* is a long vowel numbered 53. So "hello" is really 27,7,45,53. The dictionary that comes with the chip adds an extra vowel before the *o*. Some experimentation may be needed. Pauses are sometimes used within words so look at these, too.

Running the program.

Cheep Talk interfaces with the computer through the joystick ports. One pair of joystick ports make up a single 8-bit input/output port, and each bit of this port can be set to either send or receive. Line 10 sets this up for joysticks 1 and 2 (you can use locations 54017 and 54019 for joysticks 3 and 4 on the Atari 400s or 800s).

The first 7 bits of the port are set to transmit, and the eighth bit to receive. The sixty-four allophones and pauses use the first 6 bits to code them in binary. By dropping from +5 volts to 0, bit 7 tells the processor that an allophone number is on the first six lines. The processor accepts the load and begins

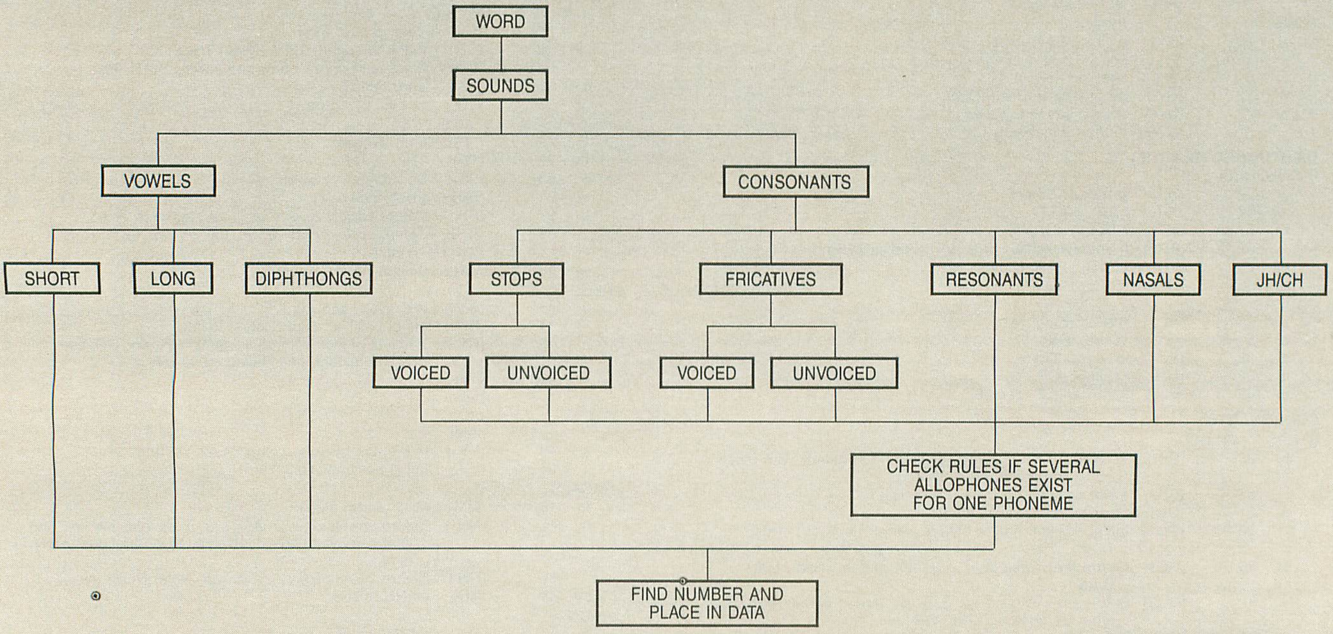


Figure 8. Logic chart.

PARTS LIST

#	Value	Radio Shack #
C1	.1 μf 50-volt capacitor	272-1069
C2	.02 μf 50-volt capacitor	272-1066
C3	1 μf 16-volt capacitor	272-1434
C4,C5	22 pf capacitors	
IC1	SP0256-AL2 speech chip	276-1784
J1	miniature phone jack	274-251
R1	100K ¼-watt resistor	271-1347
R2	33K ¼-watt resistor	271-1341
R3	10K ¼-watt resistor	271-1335
XTAL1	3.579 MHz TV crystal	272-1310
MISCELLANEOUS		
2	9-pin female D plugs	276-1538
2	hoods for plugs	276-1539
1	ribbon cable	278-772
1	plastic case	270-222
1	circuit board	276-162
1	28-pin DIP socket	276-1997
OPTIONAL		
Amplifier		277-1008

talking. At the same time, it sets its load request (LRQ) line to high, until it's done making its sound.

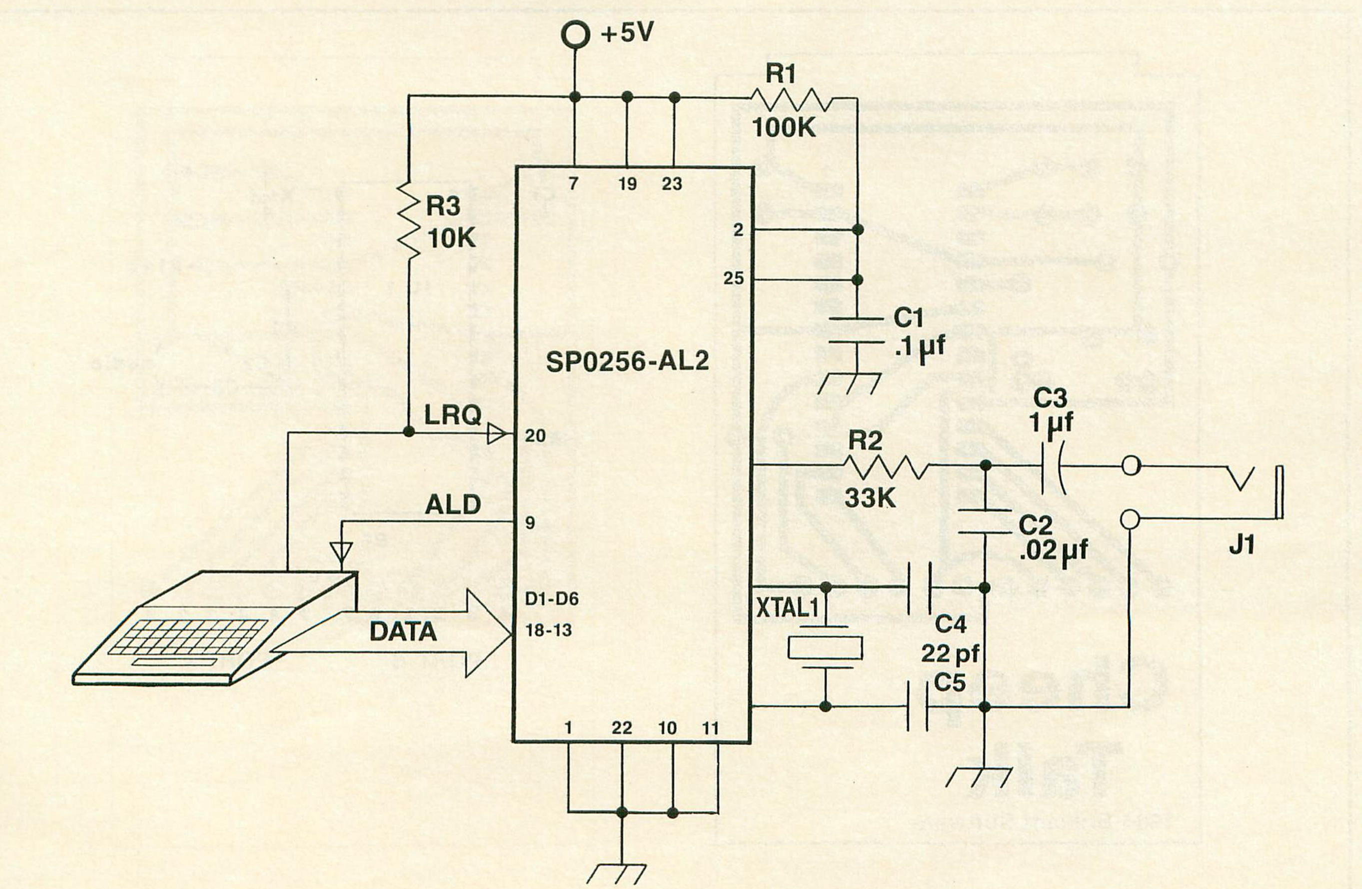
Program Line 130 reads LRQ on bit number 8 and keeps the Atari from forcing a new load until the processor is done. When finished, LRQ drops low, and your computer responds by loading the next allophone address and strobing bit 7. This "handshaking" keeps things in order.

One final word—as with most other new areas of exploration, it takes a lot of practice to be good. Don't get discouraged if, at first, **Cheep Talk** is hard to grasp; just keep at it. If nothing else, you might understand why English is one of the most difficult languages to master. . .especially to those who speak German or Spanish, where there are twenty-six letters and only twenty-six phonemes!

The uses for **Cheep Talk** are many. Consider the possibilities of programs for the unsighted, instructional tutorials, verbal instructions for your programs, or interactive games. Beam me up, Scotty! □

(Program listings and circuit board design start on page 66)

Reference: *Electronically Speaking: Computer Speech Generation*. John P. Cater, publ. Howard W. Sams & Co. 1983.



Cheep Talk Schematic.

Listing 1.

```
100 P=PEEK(54018):POKE 54018,P-4:POKE
54016,127:POKE 54018,P
110 TRAP 500
120 FOR S=1 TO 99:READ D
130 IF PEEK(54016)>128 THEN 130
140 POKE 54016,D+64:POKE 54016,D:NEXT
S
290 DATA 27,7,45,15,53,4,4,18,12,55,55
,55,3,12,43,3,26,11,15,15,45,24,24,1,3
4,3,8,24,16,0,9,22,13,12,44,55,3
300 DATA 50,19,1,9,3,13,23,23,1,41,0
500 END
```

•

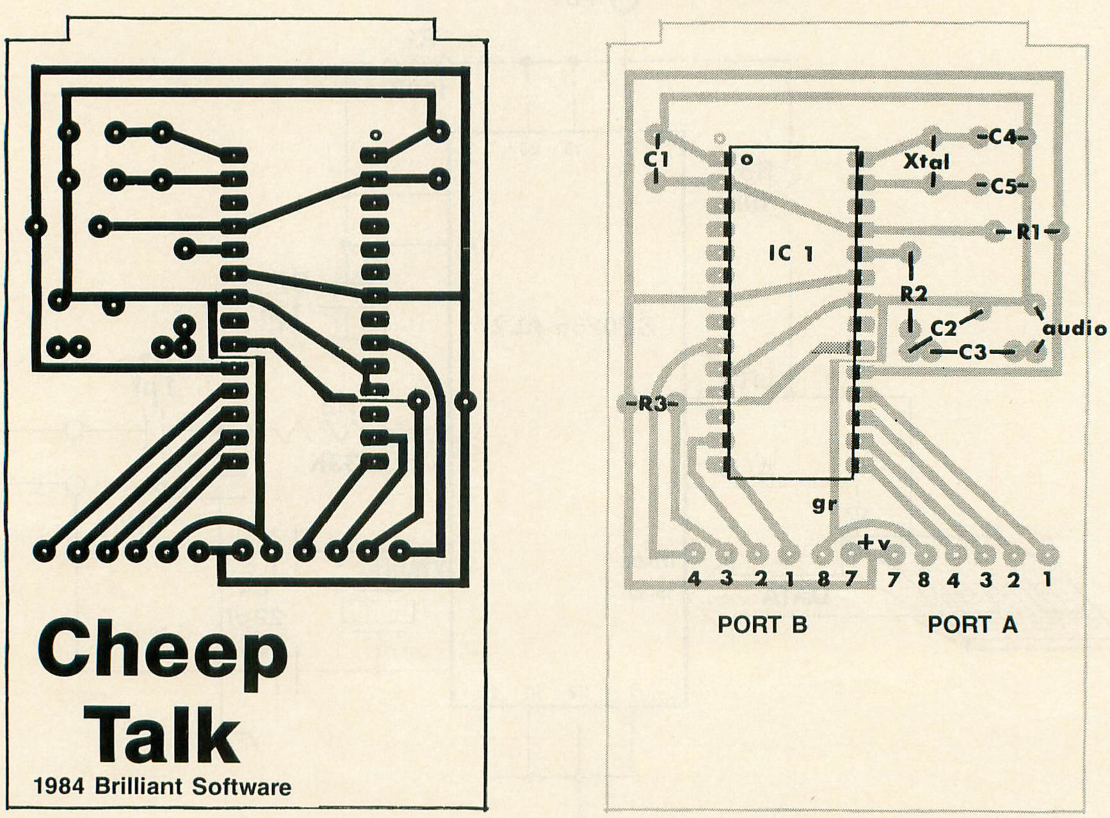
CHECKSUM DATA.
(see page 32)

```
100 DATA 46,701,119,904,847,970,960,34
,4581
```

•

Listing 2.
First Words.

```
0 REM ************************
1 REM * FIRST WORDS          *
2 REM * by Lee Brilliant MD. *
3 REM * for CHEEP TALK       *
4 REM * 1984                 *
5 REM ************************
10 GOTO 100
20 READ N
25 FOR S=1 TO N:READ D
30 IF PEEK(54016)>127 THEN 30
35 POKE 54016,D+64:POKE 54016,D:NEXT S
:A=1^1:POKE 54016,64:POKE 54016,0:RETU
RN
48 DATA 4,43,60,53
49 DATA 3,46,15,11
50 DATA 2,13,31
51 DATA 4,29,14,19
52 DATA 3,40,40,58
53 DATA 5,40,40,6,35,2
54 DATA 7,55,55,12,12,2,41,55
55 DATA 7,55,55,7,7,35,12,11
56 DATA 3,20,2,13
57 DATA 4,11,24,6,11
65 DATA 1,20
66 DATA 2,63,19
67 DATA 3,55,55,19
68 DATA 2,33,19
69 DATA 1,19
70 DATA 4,7,7,40,40
71 DATA 2,10,19
72 DATA 4,20,1,2,50
73 DATA 2,24,6
74 DATA 3,10,20,20
75 DATA 3,42,7,20
76 DATA 3,7,7,62
77 DATA 3,7,7,16
78 DATA 3,7,7,11
79 DATA 1,53
80 DATA 2,9,19
81 DATA 3,42,49,31
82 DATA 1,59
83 DATA 4,7,7,55,55
84 DATA 2,13,19
85 DATA 2,49,31
```



Cheep Talk
Printed Circuit Board Layout.

```
86 DATA 2,35,19
87 DATA 7,33,15,1,63,62,49,31
88 DATA 6,7,7,2,41,55,55
89 DATA 2,46,6
90 DATA 2,43,19
95 DATA 18,14,23,44,3,42,19,4,4,17,39,
6,4,15,2,36,7,7,11
100 P=PEEK(54018):POKE 54018,P-4:POKE
54016,127:POKE 54018,P
110 X=7:Y=3
120 GRAPHICS 18:POKE 712,44:GOSUB 500
125 POKE 16,64:POKE 53774,64
150 OPEN #1,4,0,"K:"
160 GET #1,K
170 IF PEEK(694)=128 THEN POKE 694,0:K
=K-128
180 IF PEEK(702)<>64 THEN POKE 702,64:
K=K-32
200 IF K>47 AND K<59 THEN 230
210 IF K>64 AND K<91 THEN 230
220 SOUND 1,100,10,10:A=1^1:SOUND 1,20
0,10,10:A=1^1:SOUND 1,0,0,0:RESTORE 95
:GOSUB 20:POKE 764,255:GOTO 160
230 POSITION X+2,Y+2:? #6;CHR$(K)
240 RESTORE K:GOSUB 20
250 X=INT(RND(0)*16):Y=INT(RND(0)*8):?
 #6;"K":GOSUB 500:POKE 764,255:GOTO 16
0
500 POKE 712,4+16*INT(RND(0)*16):POKE
708,16*INT(RND(0)*16)+10
510 COLOR 42:PLOT X,Y:DRAWTO X+4,Y:DRA
WTO X+4,Y+4:DRAWTO X,Y+4:DRAWTO X,Y:RE
TURN
```

•

CHECKSUM DATA.
(see page 32)

```
0 DATA 1,415,675,809,718,11,435,332,13
2,511,895,198,179,859,162,6332
52 DATA 177,733,157,314,966,264,585,89
6,199,893,603,411,865,396,681,8140
74 DATA 155,983,154,160,153,611,706,17
8,598,444,879,895,893,194,799,7802
89 DATA 709,878,763,46,485,948,639,273
,568,348,319,543,525,861,291,8196
240 DATA 309,458,155,119,1041
```

•

MOVING?

DON'T MISS A SINGLE ISSUE.

Let us know your new address right away. Attach an old mailing label in the space provided and print your new address where indicated.

QUESTION ABOUT YOUR SUBSCRIPTION?

Check the appropriate boxes below:

☐ New subscription. Please allow 4-8 weeks for your first copy to be mailed.
☐ Renewal subscription. Please include a current address label to insure prompt and proper extension. ☐ 1-year $28.00. This rate limited to the U.S.A. and its possessions. ☐ Payment enclosed or ☐ Bill me.

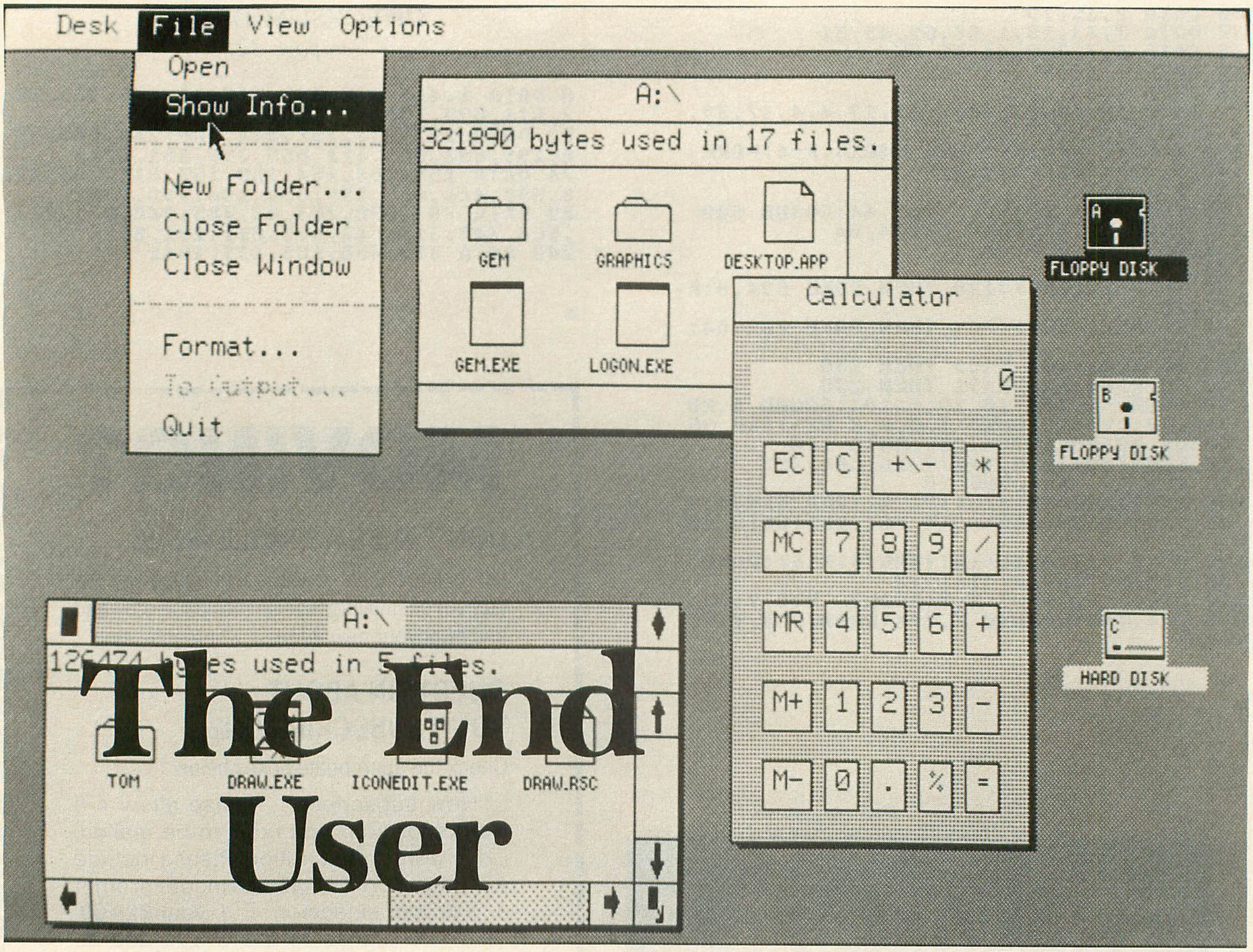
MAIL TO: ANALOG COMPUTING
P.O. Box 615, Holmes, PA 19043

Name ____________________
Address ____________________
City ____________ State ____________ Zip ____________

ATTACH LABEL HERE

(If label is not handy, print OLD address in this space.)

The End User



GEM screen © 1984 Digital Research.

by Arthur Leyenberger

Welcome to the first installment of **The End User**. This will be a regular column in the pages of **ANALOG Computing** that will bring you news, information, application tips, short reviews of useful or significant products, or whatever else seems of interest to me or you. We may even do a little gossiping and philosophizing.

ANALOG Computing already brings you excellent programs and program-related information each month. My goal is to make the **End User** column a place where you can find information on how to get the best from your Atari computer, regardless of how experienced you are, or how much equipment you have. I welcome your input on what topics you would like to see covered in the coming months. Send correspondence to the address given at the end of this column.

The entire line of Atari's new ST computers, based on the Motorola MC68000 microprocessor, will use the new Graphics Environment Manager (GEM) software from Digital Research. The Atari 130ST and 520ST computers use a proprietary operating system developed jointly by Atari and Digital Research. According to Sam Tramiel, Atari chose the GEM interface because it represents the most advanced microcomputer technology and it makes personal computers easy to learn and use.

GEM software eliminates the need for cryptic operating system commands by presenting the user with a pictorial representation of a desk's surface. Familiar icons of disks and a wastebasket appear on the desktop, while folders and documents appear in user-controlled windows. By simply moving the mouse and clicking its button, the user can open a file, run a particular software program, delete a file or do anything else allowed by the operating system.

In order to better understand the GEM operating environment, I am pleased this month to present an exclusive interview conducted with Rob LaTulipe from Digital Research. Rob is a Product Line Manager with DR and was gracious enough to talk about their affiliation with Atari and the new GEM operating system.

AL: *Digital Research has created the Graphics Environment Manager (GEM). Please tell us about it.*

RL: Let me first differentiate between our software and the other programs being shown on the ST family. We basically worked on two software projects. First, the TOS operating system which is a proprietary implementation for the Atari ST computers.

Now GEM, as a graphics user interface, is a portable operating system extension. We have been working on GEM for well over a year as a graphics user interface for MS-DOS and PC-DOS. Written primarily in C, we were able to easily take that technology and recompile it, port it over, for the TOS 68K environment on the ST family.

So those were the two projects. . .one, to create an Operating System (OS) for the 68K chip to Atari specifications; and two, take the technology that we were already developing for our commercial use in the DOS world and move that over to the new environment.

AL: *How does the GEM operating system differ from the Apple Macintosh operating system environment?*

RL: In terms of what is in the Mac, there is an OS and there is a graphics user interface capability. Here there is TOS, or on a PCjr for example, MS-DOS, and a graphics user interface embodied in GEM. Other than there being an analogy between software pieces that essentially do the same thing, there is not a great deal of similarity in terms of the call systems.

AL: *Does GEM put a large burden on the Motorola 68000 microprocessor, or does it use a lot of the support chips that the Atari ST computer has which, perhaps, the Mac and the IBM don't have?*

RL: GEM itself is not hardware dependent. That is, it is truly portable. It takes advantage of hardware, in the sense that at the core of GEM is what is called the virtual device interface, which is a concept that has been developed over a period of several years.

Digital Research was actually the first company to ship a VDI in the microcomputer world (our GSX product) two years ago, so we were a pioneer there. In fact, IBM now has their own VDI as well, so it is a common concept in the MS-DOS world, and is the subject of much ANSI (American National Standards Institute) committee activity.

AL: *What, actually, does a VDI perform within the computer?*

RL: The purpose of the VDI is to allow a programmer and software to deal with an abstract conceptual concept of space, within which graphics are drawn, and let software device drivers translate what is program into actual device output.

For example, our GEM draw product is written in a 16K by 16K unit coordinate system where drawings can be made, and yet that can be translated by loading in a particular software device driver and can be output to a screen or a printer or a polaroid palette camera (which we have a driver for in our IBM retail product). Programmers need not concern themselves with the actual physical device. They would load in a workstation as part of the programming code to identify what kind of device it is that the graphics are currently being written to.

AL: *So it is really designed with portability being the major thrust of the software system?*

RL: Yes. That's how we were able to move it quickly over to TOS when the time came.

GEM itself is not hardware dependent. . . it is truly portable.

AL: *I understand Commodore has GEM running on their IBM clone computer slated for the European market. Are there different "flavors" of GEM, or is the Commodore implementation of GEM the same as that seen on the new Atari ST computers?*

RL: Before I answer that, let me spell out the various components of GEM. GEM is systems software in terms of our product called the Graphics Environment Manager. It is strictly systems software that extends the OS.

What you see, then, be it Atari's Logo product or the desktop application for the ST computers, are mainly applications. The metaphor that allows end users to visually look at a desktop, rather than having to think about OS commands, is itself an application. It is not GEM itself, but rather the GEM desktop application.

What you see, in fact, is something where, because we have a system of resource and image files supporting the main program, you can literally swap a file in a few seconds and go from a French desktop to an English desktop, or a desktop with a certain selection of icons to one with an entirely different set of icons. So, what you see visually is extremely malleable and can vary from time to time. In terms of the icon set that Digital Research will ship with its DOS products, Atari has the choice of using our set or their own set—that may look completely different.

AL: *It sounds like Atari can use whatever set of icons they want to and may, in fact, change the icons as they market the computer for different countries. The application is really like a second layer of the user interface, and one with which the user interacts. Is that basically correct?*

RL: The GEM desktop application is a special application on the Atari ST computers because it is designated, within the GEM system software, as being the primary application. But someone else, another OEM, could do a different kind of metaphor that they want all applications to return to when it's finished executing, and designate that as the primary application. That would be equally valid.

It is just that we ourselves have felt strongly that the key functional application in any work session is something that will manage and interact with your disk files. So that is what the GEM desktop does.

AL: *So does that mean that, if I have an Atari ST computer at home and, say, a Commodore PC using GEM at the office, I am going to see the same desktop in both cases?*

RL: It will depend on the company involved. Digital Research is not involved in the decision on the part of either of those companies—or any other OEM, for that matter—how they implement the desktop metaphor.

AL: *Could it be the same desktop application, or is that what you are licensing to the hardware manufacturer?*

RL: Digital Research will license to OEMs, and are actively doing it now in the MS-DOS world, a certain desktop with a certain look. The OEMs are welcome to use that as is, and there may be some good marketing reasons to do so, but they are also more than welcome to change the look of it in terms of changing the icons or that resource file. So Atari could choose to be the same or be different, depending upon their marketing orientation.

AL: *So functionally it may be the same, but it may appear different?*

RL: Yes, exactly. All GEM applications can take advantage of what Atari is calling drop-down menus, which designates a different—and, we feel, better—way of functioning than what we have seen in other graphics user interfaces.

AL: *What are the major components of a graphics user interface?*

RL: The graphics user interface theory, which has been in the process of evolution for the past twenty years, has four components: overlapping, scrollable windows; forms; icons; and menus. Apple did not invent it with their Mac, but they were the first to popularize it.

AL: *As an IBM PC user, can I go out today and buy GEM for my PC?*

RL: GEM is geared primarily to software application solutions, so the answer is yes, but what you will really be doing is going out and buying a productivity application like GEM Draw.

> *Apple did not invent it (the graphics user interface). . . but they were the first to popularize it.*

There is a new product for the 8-bit Atari XE computers called Infinity. It is an integrated product from Matrix. Matrix is committed to GEM for the ST family, and if they decide to market it in the MS-DOS world—which is their prerogative, if they buy the license—you could buy, say, GEM Infinity for the IBM.

Digital Research will also market GEM and assorted GEM applications, so end users will have a lot of ways to get GEM desktop in the IBM world. We will provide the drivers for whatever computer is being used to run the application.

AL: *Do you see GEM as being a new standard?*

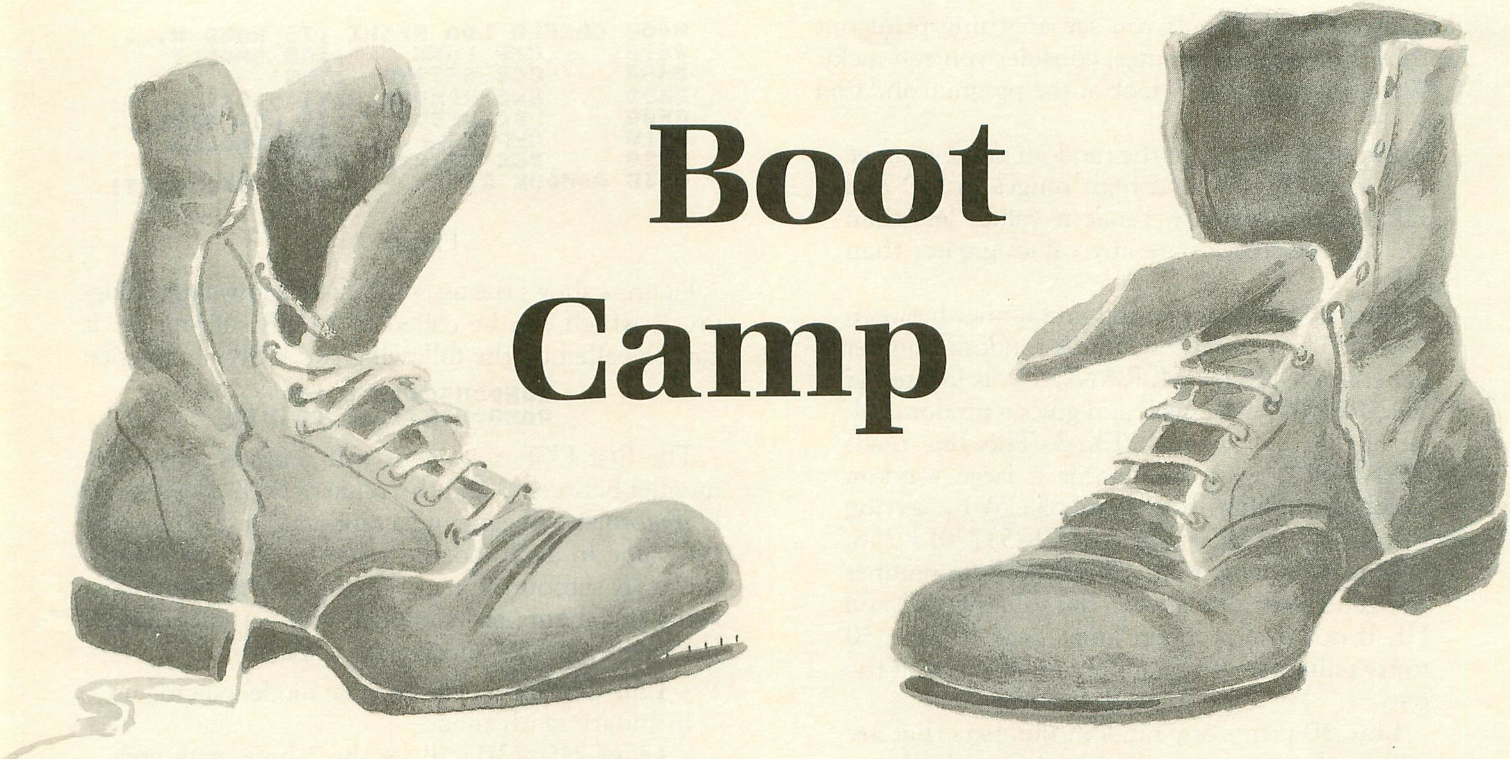
RL: That is our goal, certainly.

That's it for this month. I would like to thank Rob for enlightening us on GEM and what the next generation Atari computers will be like. Next time, I'll talk about how to take pictures of your television or monitor.

Until next month, remember—*you* are the **End User**! □

The End User
c/o ANALOG Computing
P.O. Box 23
Worcester, MA 01603

Boot Camp



by Tom Hudson

This issue, we conclude our coverage of the BASIC USR function, a handy statement that puts the speed and power of machine language to use in BASIC programs. We've looked at single- and multiple-argument operation, modifying strings, examining and changing system memory, and setting precision timers.

This issue, we're going to look at a USR function that will generate random numbers within specified ranges. This can be done in several different ways, with varying degrees of speed. We'll also see that you shouldn't always accept the first solution you come up with, since there may be one which is more efficient.

Random ramblings.

At one time or another, we've all used random numbers. Whether in games or statistical analysis, random numbers have an important function in computing.

Would you like it if your computer chess program made the same moves every game? I wouldn't—the games would get too predictable, and the chess disk would be quickly relegated to the "outdated program" pile. BASIC's random number function, RND(n), produces random numbers between zero and one, and usually works fine for most applications.

Just for fun, assume that we're simply not happy with BASIC's RND function, and want one that's more versatile. We want a function that will return a random integer value between two given numbers, or if only one parameter is given, between zero and that value. We could write the function as a BASIC subroutine, but top speed is essential. We need to write a USR subroutine.

Hats off!

The first method most people would come up with is what I call "pulling numbers out of a hat." Simply stated, you get a random number, and if it's in the range you want, you use it. If not, you reach into the hat and try another. This method works fine, but there's one big drawback: speed.

```
1 REM *** RANDOM NUMBERS ***
2 REM
3 REM NUMBERS OUT OF A HAT (BAS)
4 REM SLOW WAY TO GET THE JOB DONE!
5 REM
10 ? "ENTER RANDOM # RANGE (LO,HI)"
20 TRAP 10:INPUT LO,HI
30 RAND=PEEK(53770)+PEEK(53770)*256
40 IF RAND<LO OR RAND>HI THEN 30
50 ? RAND:GOTO 30
```

Figure 1.

Figure 1 is the BASIC version of pulling numbers out of a hat. Type in the program and RUN it. You will be asked for a random number range. Type in:

```
0,65535
```

and press RETURN. You will see the program happily print out random numbers ranging from 0 to 65535, at BASIC's top speed. All's well, right? Wrong!

Press BREAK and RUN the program again. This time, when prompted for the random number range, type:

```
350,355
```

and press RETURN. If you see anything print out within three or four minutes, consider yourself lucky.

What happened? Let's look at the program and find out.

Lines 10-20 accept the random number range and store the low and high ranges in LO and HI, respectively. Any random values less than LO are rejected, as are any values greater than HI.

Line 30 generates a random number between 0 and 65535, using the Atari's random number generator, RANDOM. RANDOM is located at $D20A (53770 decimal) and gives a random value of 0 to 255 when PEEKed. This line reads RANDOM twice and builds a large random number (ranging from 0 to 65535) by setting RAND to PEEK(53770) + PEEK(53770) * 256.

Line 40 checks to see if the random number just generated falls between the values in LO and HI. If not, the program loops back to Line 30 to try pulling another random number out of the hat.

Line 50 prints any random numbers that are within the range specified by LO and HI.

Now can you see why this program works so slowly? When a large range (such as 0-65535) is specified, there is a better chance of the random number falling into that range. When a smaller range is given, the odds of picking a random number in that range can drop drastically, making the program take virtually forever.

"Aha," you say, "I'll just write this routine in assembly language and speed it up. Assembly language fixes everything!" Let's see what happens.

```
0100 LOWL  =  $CB       ;LOW LIMIT
0110 LOWH  =  $CC
0120 HIGHL =  $CE       ;HIGH LIMIT
0130 HIGHH =  $CF
0140 RESLO =  $D4       ;BASIC'S RESULT
0150 RESHI =  $D5
0160 RANDOM = $D20A     ;RANDOM # 0-255
0170 ;
0180       *=  $0600    ;ROUTINE START
0190 ;
0200       CLD          ;CLEAR DECIMAL
0210       LDA #0       ;SET DEFAULT
0220       STA LOWL     ;LOW RANGE
0230       STA LOWH     ;VALUE
0240       PLA          ;GET #ARGS
0250       CMP #1       ;ONE ARGUMENT?
0260       BEQ PULLHI   ;YES!  GET HI
0270       PLA          ;PULL LOW HI
0280       STA LOWH     ;AND SAVE IT
0290       PLA          ;PULL LOW LO
0300       STA LOWL     ;AND SAVE IT
0310 PULLHI PLA         ;PULL HIGH HI
0320       STA HIGHH    ;AND SAVE IT
0330       PLA          ;PULL HIGH LO
0340       STA HIGHL    ;AND SAVE IT
0350 GETRND LDA RANDOM ;GET RANDOM #
0360       STA RESLO    ;SET LO BYTE
0370       LDA RANDOM   ;GET RANDOM #
0380       STA RESHI    ;SET HI BYTE
0390       CMP HIGHH    ;TOO BIG?
0400       BCC CHEKLO   ;NO, TOO SMALL?
0410       BNE GETRND   ;TOO BIG!
0420       LDA RESLO    ;IS LOW BYTE
0430       CMP HIGHL    ;TOO BIG?
0440       BCC CHEKLO   ;NO, TOO SMALL?
0450       BNE GETRND   ;TOO BIG!
0460 CHEKLO LDA RESHI ;IS RAND #...
0470       CMP LOWH     ;TOO SMALL?
0480       BCC GETRND   ;YES!
0490       BNE RANDOK   ;IT'S OK!
0500       LDA RESLO    ;IS LOW BYTE...
0510       CMP LOWL     ;TOO SMALL?
0520       BCC GETRND   ;YES!
0530 RANDOK RTS         ;IT'S OK, EXIT!
```

Figure 2.

Figure 2 shows the assembly code equivalent of Figure 1, which can be called as a USR subroutine. It can be called by the following two USR statements:

```
RAND=USR(1536,HI)
RAND=USR(1536,LO,HI)
```

The first USR statement will generate a random number between 0 and the value of arg1. The second USR format will generate a random number between the value in arg1 and the value in arg2. Obviously, the USR subroutine must be able to determine how many arguments are supplied, and act accordingly. Let's see how this subroutine works.

Line 200 clears the decimal mode, placing us in binary math mode.

Lines 210-230 will set the 2-byte work area LOWL-LOWH to 0. This ensures that, if there is only one argument, the low range will default to 0.

Line 240 pulls the number of arguments off the stack.

Line 250 compares the number of arguments to 1. If there is only one argument, we will want to go get the high range value.

Line 260 branches to PULLHI if the number of arguments if equal (BEQ) to 1. This will cause the computer to pull just one argument from the stack.

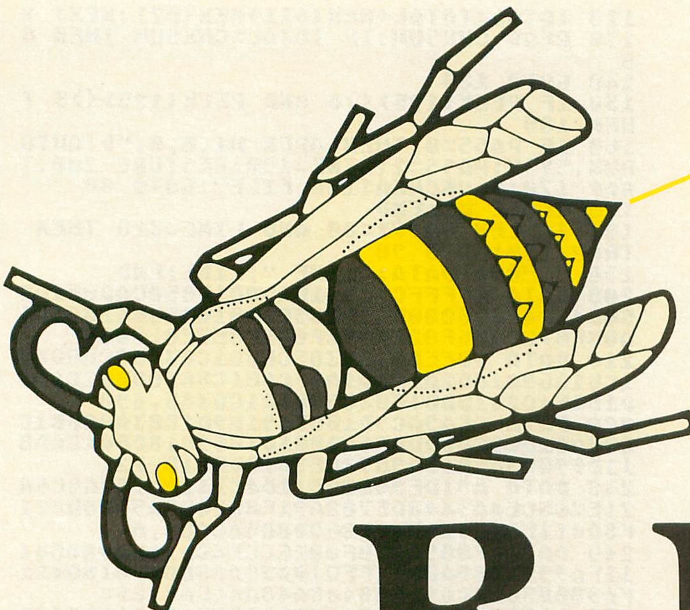
Lines 270-300 pull and store the low limit for the random number. If there are two arguments, this is the first.

Lines 310-340 pull and store the high limit for the random number. Of course, if there's only one argument, specifying a range from 0 to arg1, this is the one that will be pulled, and the low limit (set in Lines 220-240) will be 0.

Lines 350-380, labeled GETRND, generate a random number between 0 and 65535, placing it in the locations RESLO and RESHI. As you should know by now, RESLO and RESHI ($D4 and $D5) are the locations used to send values to BASIC from the USR subroutine. The random number is built by simply loading the accumulator twice, placing each random byte into the RESLO and RESHI locations.

At this point, I would like to discuss an important function in assembly language: comparisons. We've already seen how single-byte values can be compared easily, using the CMP instruction. Since we're using

(continued on page 78)



B-Line

16K Disk, 2.0 DOS

by Angelo Giambra

The designers of Atari BASIC faced a difficult challenge: cram a reasonably powerful BASIC interpreter into an 8K ROM cartridge. Despite the constraints imposed, they produced a really fine product.

But to get all that power into a ROM cartridge, many desirable features had to be forgone—features like line resequencing, mass line deletes and automatic line numbering. While all of these features are timesaving, I find that automatic line numbering is the one I most frequently miss. So I did something about it.

The **B-Line** program creates an AUTORUN.SYS file on your disk which, when booted with your BASIC cartridge, will make keying in BASIC programs a snap.

Loading B-Line.

Key in the program in Listing 1 carefully. When you're finished, SAVE it to disk, then RUN it. **B-Line** will check the DATA statements for accuracy and inform you of any errors. If there are none, it will proceed to create the AUTORUN.SYS file.

Now, power off your computer and turn it back on again. This will allow the AUTORUN.SYS file to load and execute. Finally, the *READY* prompt will appear, and you will be back to old, familiar BASIC. Well, not quite.

Using it.

BASIC will now accept a new command. The syntax of the new command is:

```
AUTO <base>,<increment>
```

The base and increment values are optional. Here are some examples.

If you key in the following:

```
AUTO
```

BASIC will begin numbering lines at 10 and increment each succeeding line with the default value of 10.

Or you may key in this:

```
AUTO 920
```

This will cause BASIC to begin numbering lines at 920 and increment each succeeding line by the default, 10.

Finally, you may key in something like this:

```
AUTO 350,20
```

Now BASIC will begin numbering lines at 350 and increment each succeeding line by 20.

To turn off auto-sequencing mode, simply press the BREAK key.

B-Line will not go away if you press SYSTEM RESET, so you needn't worry about losing it. If you do want to deactivate it, however, key in DOS. The system will perform a warmstart, and you'll be returned to BASIC. The next time you key in DOS, you'll be transferred to the DOS menu normally.

How it works.

B-Line works by installing a new handler for the editor into memory and a vertical blank interrupt (VBI) routine to handle the line sequencing. The handler first passes control to the OS editor getbyte routine. It then examines the input buffer to see if *AUTO* has been entered.

If not, control passes back to BASIC normally. But if *AUTO* has been entered, the routine scans the remainder of the buffer looking for the base and increment values. If it finds them, it processes them; otherwise, it substitutes the default values for the base and increment. It then signals the VBI routine to begin line sequencing.

This routine waits for you to press the RETURN key, then figures out the next line number and prints it on the screen.

Some bothersome timing problems had to be overcome. Whenever you key a line into BASIC, a lot of things happen. BASIC must parse the line and, if necessary, move portions of the code around in memory to make room for the new line. The VBI routine had to somehow *know* when BASIC was finished with all this, since, if it printed up the next line number to the screen too soon, BASIC got all fouled up.

I discovered that, whenever BASIC is finished examining a line of input, it stores a hex C2 in a variable called PROMPT. The handler takes advantage of this by zeroing out PROMPT whenever it sees the RETURN key pressed. The VBI routine then waits for PROMPT to equal C2 again, before printing the next line.

The B-ottom Line.

B-Line will prove to be a real timesaver for you. I think you're going to wonder how you ever got along without it. . .especially when you're keying in all those great magazine games! □

Listing 1.

```
10 REM ****************
20 REM *   B-LINE     *
30 REM *     by       *
40 REM * A. Giambra *
50 REM ****************
60 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
0,0,10,11,12,13,14,15
70 DIM DAT$(96),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=190:RESTOR
E 200:TRAP 150:? "KCHECKING DATA"
80 TOTAL=0:LINE=LINE+10:POSITION 2,2:?
 "LINE:";LINE:READ DAT$:IF LEN(DAT$)<>
96 THEN 180
90 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
100 FOR X=1 TO LEN(DAT$)-1 STEP 2:D1=A
SC(DAT$(X))-48:D2=ASC(DAT$(X+1))-48:BY
TE=HEX(D1)*16+HEX(D2)
110 IF PASS=2 THEN PUT #1,BYTE:NEXT X:
READ CHKSUM:GOTO 80
120 TOTAL=TOTAL+HEX(D1)+HEX(D2):NEXT X
130 READ CHKSUM:IF TOTAL=CHKSUM THEN 8
0
140 GOTO 180
150 IF PEEK(195)<>6 AND PEEK(195)<>5 T
HEN 180
160 IF PASS=0 THEN OPEN #1,8,0,"D:AUTO
RUN.SYS":PASS=2:LINE=190:RESTORE 200:T
RAP 170:? "KCREATING FILE":GOTO 80
170 CLOSE #1:END
180 IF LEN(DAT$)=80 AND LINE=320 THEN
TRAP 160:GOTO 90
190 ? "BAD DATA: LINE ";LINE:END
200 DATA FFFFFC1CAB1DAD0A1D850CAD0B1D8
50D4C74E4200000A9FC850AA91C850BA200BD1
A03F075C945F02EC953F005E8E8E8,706
210 DATA D0EEBD1B0385CBBD1C0385CCA006B
1CB1869018D2A1F8D1A1FC8B1CB69008D2B1F8
D1B1F4C231DBD1B0385CBBD1C0385,698
220 DATA CCA9AC9D1B03A91D9D1C03A000B1C
B99AC1DC8C00DD0F6ADB01D1869018DBA1DADB
11D69008DBB1DA9B938E9018DB01D,721
230 DATA A91DE9008DB11D4C231DA907A0C8A
21E205CE4A9448DE702A91F8DE802A9008D421
F8D411F60B91DB41E20000008C080,613
240 DATA F009C99BF00EE6CF4CD11DA9008D4
11FA99B2860AD411FF010A90A85D0A9018D421
FA90085C24C091E98488A48A4CFA9,684
250 DATA 9B998005A000B98005D9351FD008C
8C004D0F320111E68AA68A8A90085CFA99B286
0B98005C99BF075C8A200B98005C9,614
260 DATA 2CF00BC99BF0079D391FC8E810EE9
D391FA90085F220BF1E98482000D820D2D9A5D
485CBA5D585CC68A8B98005C99BF0,716
270 DATA 32C8A200B98005C930900BC93AB00
79D391FC8E810EEA9209D391FA90085F220BF1
E2000D820D2D9A5D485CDA5D585CE,654
280 DATA 4C991EA90A85CDA90085CE4C991EA
90A85CB85CDA90085CC85CE2051DA38A5CBE5C
D85CBA5CCE5CE85CCA90A85D0A900,776
290 DATA 85C2A9018D41B51E381F1F8D421FA
99B8D800560A93985F3A91F85F460AD421FF05
2A5C2F04EA5D0F005C6D04C1F1FA9,679
300 DATA 008D421FA5CB1865CD85CBA5CC65C
E85CCA5CB85D4A5CC85D520AAD920E6D8A000B
1F31005297F4C0B1F99391FC810F1,705
310 DATA 99391F8C431FEE431F20221FA9202
000002051DA4C62E4A0009848B9391F2000006
8A8C8CC431FD0F0604155544F411F,544
320 DATA 601F000000A9FC850AA91C850BA50
C8D0A1DA50D8D0B1DA909850CA91D850D4C141
DE002E102441F,497
```

•

CHECKSUM DATA.

(see page 32)

```
10 DATA 272,470,323,72,280,965,202,364
,735,912,694,97,438,719,585,7128
160 DATA 177,134,773,174,140,150,101,8
37,886,732,25,916,286,45,84,5460
310 DATA 406,865,1271
```

•

Assembly listing.

```
;  ******************
;  *                *
;  *    B-LINE      *
;  *                *
;  *      by        *
;  *                *
;  *  A. GIAMBRA    *
;  *                *
;  ******************
;
;
        *=  7420
;
; OS EQUATES
;
HATABS   =   $031A
LBUFF    =   $0580
FASC     =   $D8E6
IFP      =   $D9AA
AFP      =   $D800
```

```
FPI         =   $D9D2
INTLBF      =   $DA51
VBXIT       =   $E462
SETVB       =   $E45C
WARMSTART   =   $E474
MEMLO       =   $02E7
FRO         =   $D4
INBUFF      =   $F3
CIX         =   $F2
DOSVEC      =   $0A
DOSINI      =   $0C
;
; WORK VARIABLES
;
CR          =   $9B       ;CARRIAGE RETURN
COMMA       =   $2C
SPACE       =   $20
PROMPT      =   $C2       ;BASIC PROMPT CHAR
NUM         =   $CB       ;BASE NUMBER
INCR        =   $CD       ;INCREMENT
CHARCOUNT   =   $CF       ;COUNTER
INDEX       =   $CB       ;WORK INDEX
TIMER       =   $D0       ;VB TIMER
;
;
DOS         LDA RESET+1
            STA DOSINI   ;RESTORE DOSINI
            LDA RESET+2
            STA DOSINI+1
            JMP WARMSTART ;DO WARMSTART
RESET       JSR $FFFF    ;CHANGED TO JSR TO DOSINI
            LDA # <DOS
            STA DOSVEC   ;POINT DOSVEC
            LDA # >DOS   ;TO OUR ROUTINE
            STA DOSVEC+1
LOOKUP      LDX #0
SEARCH      LDA HATABS,X ;EXAMINE HANDLER
            BEQ FINI     ;ENTRY
            CMP #'E      ;IS IT THE EDITOR
            BEQ FOUNDED
            CMP #'S      ;IS IT THE SCREEN
            BEQ FOUNDSCR
CONT        INX          ;GO TO NEXT
            INX          ;ENTRY
            INX
            BNE SEARCH
FOUNDSCR    LDA HATABS+1,X ;STORE SCREEN
            STA INDEX    ;HANDLER ADDRESS
            LDA HATABS+2,X
            STA INDEX+1
            LDY #6       ;OFFSET TO
            LDA (INDEX),Y ;PUTBYTE VECTOR
            CLC
            ADC #1       ;ADD 1
            STA PUTCHAR+1 ;MODIFY OUR JSR
            STA MAKESPACE+1
            INY
            LDA (INDEX),Y
            ADC #0
            STA PUTCHAR+2
            STA MAKESPACE+2
            JMP CONT
;
FOUNDED     LDA HATABS+1,X ;LOAD VECTOR
            STA INDEX    ;PUT IN INDEX
            LDA HATABS+2,X
            STA INDEX+1
            LDA # <HANDLER ;POINT HANDLER
            STA HATABS+1,X ;TO OUR TABLE
            LDA # >HANDLER
            STA HATABS+2,X
            LDY #0
MOVE        LDA (INDEX),Y ;GET OS TABLE
            STA HANDLER,Y ;MOVE TO OUR
            INY          ;LOCAL TABLE
            CPY #13      ;ARE WE DONE?
            BNE MOVE
            LDA HANDLER+4 ;GET GETBYTE
            CLC          ;VECTOR
            ADC #1       ;ADD 1
            STA GETB+1   ;MODIFY OUR
            LDA HANDLER+5 ;JSR
            ADC #0
            STA GETB+2
            LDA # <GETB  ;POINT TABLE TO
            SEC          ;OUR OWN ROUTINE
            SBC #1
            STA HANDLER+4
            LDA # >GETB
            SBC #0
            STA HANDLER+5
            JMP CONT
;
FINI        LDA #7
            LDY # <INTERRUPT ;FIRE UP VB
            LDX # >INTERRUPT ;ROUTINE
            JSR SETVB
            LDA # <ENDP
            STA MEMLO    ;POINT MEMLO
            LDA # >ENDP  ;TO END OF OUR
            STA MEMLO+1  ;PROGRAM
            LDA #0
            STA ENDLINE  ;CLEAR WORK
            STA SWITCH   ;REGISTERS
            RTS
;
HANDLER     *=  *+13     ;HANDLER TABLE
;
GETB        JSR $FFFF    ;JSR TO OS
            PHP          ;GETBYTE ROUTINE
            CPY #$80     ;BREAK PRESSED?
            BEQ BRK      ;YES
            CMP #CR      ;CARRIAGE RETURN?
            BEQ PROCESS  ;YES, PROCESS IT
            INC CHARCOUNT ;COUNT CHARACTER
            JMP NOBRK
;
BRK         LDA #0       ;SHUT OFF AUTONUM
            STA SWITCH
            LDA #CR
NOBRK       PLP          ;RESTORE STATUS
            RTS
;
PROCESS     LDA SWITCH   ;AUTONUM ACTIVE?
            BEQ AROUND   ;NO, PARSE LINE
            LDA #10      ;SET VB TIMER
            STA TIMER
            LDA #1       ;SET ENDLINE FLAG
            STA ENDLINE
            LDA #0       ;CLEAR BASIC'S
            STA PROMPT   ;PROMPT REGISTER
            JMP XIT
;
AROUND      TYA          ;SAVE X AND Y
            PHA
            TXA
            PHA
            LDY CHARCOUNT
            LDA #CR      ;STORE A RETURN
            STA LBUFF,Y  ;IN BASIC BUFFER
            LDY #0
COMPARE     LDA LBUFF,Y  ;COMPARE BUFFER
            CMP TABLE,Y  ;TO 'AUTO'
            BNE EXIT
            INY
            CPY #4       ;ARE WE DONE?
            BNE COMPARE
            JSR AUTONUM  ;DO AUTONUM
EXIT        PLA          ;RESTORE X AND Y
            TAX
            PLA
            TAY
XIT         LDA #0       ;CLEAR CHARCOUNT
            STA CHARCOUNT
            LDA #CR      ;RESTORE ACCUM
            PLP          ;RESTORE STATUS
            RTS
;
AUTONUM     LDA LBUFF,Y  ;LOAD NEXT CHAR
            CMP #CR      ;CARRIAGE RETURN?
            BEQ DEFAULT  ;YES, NO PARAMS
            INY
            LDX #0
GETNUM      LDA LBUFF,Y  ;GET BASE NUMBER
            CMP #COMMA   ;DONE?
            BEQ GETNEXT  ;YES
            CMP #CR      ;DONE?
            BEQ GETNEXT  ;YES
            STA BUFF,X   ;SAVE CHARACTER
            INY          ;IN BUFF
            INX
            BPL GETNUM
GETNEXT     STA BUFF,X   ;SAVE CHAR
            LDA #0       ;RESET CIX
            STA CIX
£           JSR          TYA
            PHA          ;SAVE Y
            JSR AFP      ;ASCII TO FP
            JSR FPI      ;FP TO INTEGER
            LDA FRO      ;GET BASE
            STA NUM      ;STORE IN NUM
            LDA FRO+1
            STA NUM+1
            PLA
            TAY          ;RESTORE Y
            LDA LBUFF,Y  ;GET NEXT CHAR
            CMP #CR      ;RETURN?
            BEQ SETINCR  ;YES, NO INCR
            INY
            LDX #0
NEXTNUM     LDA LBUFF,Y  ;LOAD NEXT CHAR
            CMP #'0      ;IS IT BETWEEN
            BCC CONVERT  ;0 AND 9?
            CMP #':
            BCS CONVERT
            STA BUFF,X   ;YES, SAVE IT
            INY
            INX
            BPL NEXTNUM
CONVERT     LDA #SPACE
            STA BUFF,X   ;STORE A SPACE
            LDA #0
            STA CIX      ;RESET CIX
            JSR POINT    ;INIT INBUFF
            JSR AFP      ;ASCII TO FP
            JSR FPI      ;FP TO INTEGER
            LDA FRO
            STA INCR     ;STORE INCR
            LDA FRO+1
            STA INCR+1
            JMP START
;
SETINCR     LDA #10      ;SET INCR AT
            STA INCR     ;THE DEFAULT (10)
            LDA #0
            STA INCR+1
            JMP START
;
DEFAULT     LDA #10      ;SET BASE AT
            STA NUM      ;THE DEFAULT (10)
            STA INCR
            LDA #0
            STA NUM+1
            STA INCR+1
START       JSR INTLBF
            SEC
            LDA NUM      ;LOAD BASE
            SBC INCR     ;SUBTRACT INCR
            STA NUM
            LDA NUM+1
            SBC INCR+1
            STA NUM+1
            LDA #10      ;SET VB TIMER
            STA TIMER
            LDA #0       ;RESET BASIC'S
            STA PROMPT   ;PROMPT REGISTER
            LDA #1
            STA SWITCH   ;AUTONUM ACTIVE
            STA ENDLINE  ;SET ENDLINE
            LDA #CR      ;RETURN A CR TO
            STA LBUFF    ;BASIC
            RTS
;
POINT       LDA # <BUFF  ;POINT INBUFF
            STA INBUFF   ;TO OUR BUFFER
            LDA # >BUFF
            STA INBUFF+1
            RTS
;
; VB INTERRUPT ROUTINE
;
INTERRUPT LDA ENDLINE ;LINE ENTERED?
            BEQ EXITVB   ;NOT YET
            LDA PROMPT   ;DID BASIC FINISH
            BEQ EXITVB   ;WITH IT YET?
            LDA TIMER    ;WAIT TEN JIFFIES
            BEQ OK
            DEC TIMER
            JMP EXITVB
;
OK          LDA #0       ;RESET ENDLINE
            STA ENDLINE
            LDA NUM      ;GET BASE
            CLC
            ADC INCR     ;ADD INCREMENT
            STA NUM      ;SAVE IT
            LDA NUM+1
            ADC INCR+1
            STA NUM+1
            LDA NUM      ;GET BASE
            STA FRO      ;PUT IN FP REGISTER
            LDA NUM+1
            STA FRO+1
            JSR IFP      ;INTEGER TO FP
            JSR FASC     ;FP TO ASCII
            LDY #0
```

```
WRITEIT  LDA (INBUFF),Y ;GET A BYTE
         BPL WRITENUM
         AND #$7F       ;TURN OFF HIGH BIT
         JMP DONE
;
WRITENUM STA BUFF,Y     ;SAVE IT
         INY
         BPL WRITEIT
DONE     STA BUFF,Y     ;SAVE IT
         STY LENGTH     ;SAVE LENGTH
         INC LENGTH     ;INCREMENT LENGTH
         JSR WRTF       ;PUT NUM ON SCREEN
         LDA #SPACE
MAKESPACE JSR $FFFF     ;PRINT A SPACE
         JSR INTLBF     ;INIT INBUFF
EXITVB   JMP VBXIT
;
WRTF     LDY #0
LP       TYA
         PHA            ;SAVE Y
         LDA BUFF,Y
PUTCHAR  JSR $FFFF      ;WILL POINT TO SCREEN
         PLA            ;PUTBYTE ROUTINE
         TAY
         INY
         CPY LENGTH     ;DONE?
         BNE LP
         RTS
;
TABLE    .BYTE "AUTO"
BUFF     *=  *+8
SWITCH   .BYTE 0
ENDLINE  .BYTE 0
LENGTH   .BYTE 0
ENDP
BEGIN    LDA # <DOS     ;POINT DOSVEC
         STA DOSVEC     ;TO OUR ROUTINE
         LDA # >DOS
         STA DOSVEC+1
         LDA DOSINI     ;GET DOS INIT VECTOR
         STA RESET+1    ;MODIFY JSR
         LDA DOSINI+1
         STA RESET+2
         LDA # <RESET   ;POINT DOSINI
         STA DOSINI     ;TO OUR RESET LOGIC
         LDA # >RESET
         STA DOSINI+1
         JMP LOOKUP

         *=  $02E0

         .WORD BEGIN
```

Boot Camp

(continued from page 72)

2-byte values here, we must learn how to perform multiple-byte comparisons.

A 2-byte comparison is not very different from a single-byte comparison. The obvious difference is that there are now 2 bytes to be compared instead of 1. What may not be obvious is that we must compare the high-order bytes first, then the low-order bytes. Figure 3 is a flowchart of possible comparison outcomes.

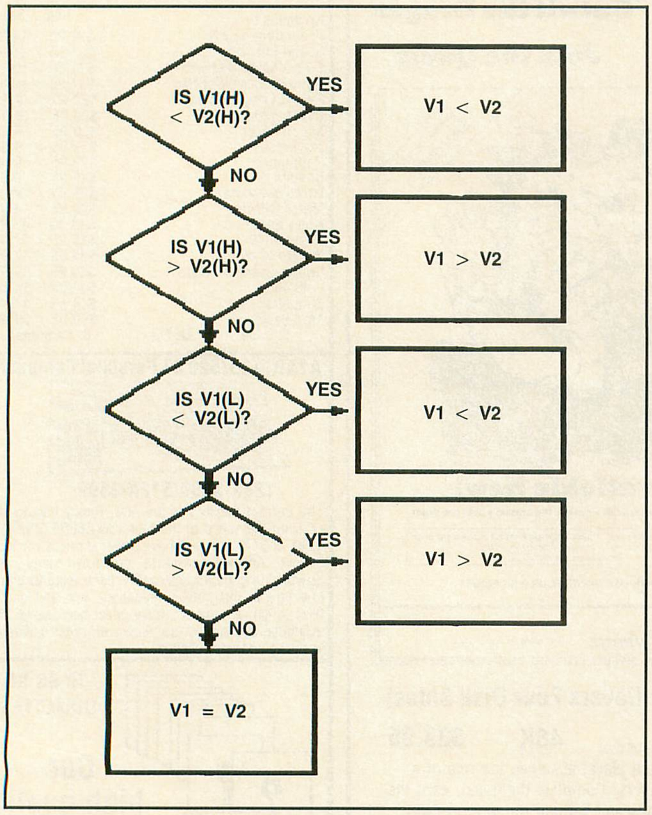


Figure 3.

In Figure 3, we are comparing the values V1 and V2. V1 and V2 are both 2-byte values, and their high-order and low-order portions are designated H and L, respectively.

As you can see, there are three possible outcomes in any comparison: greater than, less than, and equal to. The flowchart is fairly straightforward, showing the step-by-step procedure for comparing any two 2-byte values. Note that, since the high-order bytes are the most significant bytes, they are compared first. After all, if the high byte of V1 is greater than that of V2, V1 is greater than V2, no matter what the low bytes of the values contain.

Note, however, that if the high-order bytes of V1 and V2 are equal, we *must* compare the low bytes to complete the comparison properly. Figure 4 shows the assembly code equivalent of Figure 3.

```
10    LDA V1H
20    CMP V2H
30    BCC V1LTV2
40    BNE V1GTV2
50    LDA V1L
60    CMP V2L
70    BCC V1LTV2
80    BNE V1GTV2
90    BEQ V1EQV2
```

Figure 4.

The first operation in Figure 4 is the actual CMP operation on the high-order bytes of V1 and V2, in Lines 10-20. At this point, the CARRY and ZERO flags are set according to the comparison results. If V1 is greater than V2, the carry flag is set to 1. If V1 is less than V2, the carry flag is cleared. If V1 and V2 are equal, the zero flag *and* the carry flag are set.

Next, the computer branches to V1LTV2 (V1 Less Than V2) if the carry flag is cleared (BCC V1LTV2).

The next operation is somewhat tricky. Since an equal condition sets the carry flag as well as the zero flag, we BNE (Branch Not Equal) to V1GTV2 (V1 Greater Than V2). This insures that we will only branch to V1GTV2 when V1 is greater than V2. The program will fall through to the next instruction if V1H is equal to V2H.

At this point, we know that the high bytes of V1 and V2 are equal, and we have to compare the low-order bytes. This happens if V1=$4F00 and V2=$4F9B, V1=$007F and V2=$0020, etc.

Lines 50-60 compare the low bytes of V1 and V2, just as the high bytes were compared. Now we're ready to finish the 2-byte comparison.

Line 70 branches if the carry flag is clear (BCC) to V1LTV2. Remember that if the carry is clear after a compare, the accumulator value (V1L, in this case) is less than the byte it was compared to (V2L).

Line 80 branches if the compare was not equal (BNE) to V1GTV2. Once again, this branch operation is used instead of BCS, because an equal condition also sets the carry flag. In this case, since the BNE is used after a BCC instruction, the BNE can be considered a kind of "branch if greater than" instruction.

Line 90 branches to V1EQV2 using the BEQ instruction. At this point, we know V1 equals V2, since the high bytes are equal, and the low bytes are equal.

Multiple-byte comparisons can be somewhat confusing at first, but we'll be using them often in **Boot Camp** programs, and you'll soon feel comfortable with them. Now, let's return to our "walk-through" of the first random number program.

Line 390 compares the accumulator (which contains the high byte of the random number) to HIGHH, the high byte of the upper random number limit. This is the start of a 2-byte comparison to see if the random number we just built is greater than the upper random number limit.

Line 400 branches if the carry is clear (BCC) to CHEKLO. If the carry is clear, we know that the high byte of the random number is less than the high byte of the upper limit, and we can go on to check the random number to see if it is less than the lower limit.

Line 410 branches if not equal to GETRND, since a not-equal condition (the same as "branch if greater than," when used after a BCC instruction) means that the random number is greater than the upper random number limit, and we have to reach into the hat for another random number.

Lines 420-430 compare the low byte of the random number to the low byte of the upper limit. At this point, we know that the high-order byte of the random number is the same as that of the upper limit, so we need to compare the low-order bytes to complete the comparison operation.

Line 440 branches if the carry is clear (random < limit) to CHEKLO. We now know that the random number is less than the upper limit, and must check to see if it is above the lower limit.

Line 450 branches if not equal (random > limit) to GETRND, since this shows that the random number is greater than the upper limit.

Lines 460-470, labeled CHEKLO, begin the process of comparing the random number to the lower limit. The high value of the random number (RESHI) is loaded into the accumulator and compared to LOWH, the high byte of the lower random number limit.

Line 480 branches if the carry flag is clear (random < lower limit) to GETRND, because the random number is less than the lower limit.

Line 490 branches if not equal (random > lower limit) to RANDOK, since this indicates that the random number is greater than the lower limit.

Lines 500-510 compare the low byte of the random number to the low byte of the lower random number limit. This is done only when the high bytes of the random number and low limit are equal.

Line 520 branches if the carry flag is clear (random < lower limit) to GETRND to try another random number. If this branch is not taken, we know that the random number is greater than or equal to the lower limit, and is acceptable.

Line 530 returns to BASIC when the random number is greater than or equal to the lower limit, and less than or equal to the upper limit. The random number is in BASIC's return area (RESLO and RESHI), ready to be used by the BASIC program.

Now that we've completed the random number subroutine ("hat" version), let's use it in a BASIC program. Figure 5 shows the subroutine installed in a BASIC program.

```
1 REM *** RANDOM NUMBERS ***
2 REM
3 REM NUMBERS OUT OF A HAT (ASM)
4 REM FASTER THAN BASIC, STILL SLOW
5 REM
10 DIM D(319):FOR X=0 TO 319:D(X)=192:
NEXT X
20 GRAPHICS 24:SETCOLOR 2,0,0:COLOR 1
30 FOR X=1536 TO 1598:READ N:POKE X,N:
NEXT X:RAND=1536
40 A=USR(RAND,0,319)
50 D(A)=D(A)-1:PLOT A,D(A)
60 GOTO 40
100 DATA 216,169,0,133,203,133,204,104
,201,1,240,6,104,133,204,104,133,203,1
04,133,207,104,133,206,173
110 DATA 10,210,133,212,173,10,210,133
,213,197,207,144,10,208,240,165,212,19
7,206,144,2,208,232,165,213
120 DATA 197,204,144,226,208,6,165,212
,197,203,144,218,96
```

Figure 5.

After typing in Figure 5, RUN it. In a few seconds (required to initialize the program), you will see a graphic representation of the random numbers being generated by the subroutine. The program is generating random numbers between 0 and 319, and plotting them on a graphics 8 screen, each value plotting in the appropriate X column. Like our first BASIC program, this looks fine, doesn't it?

Stop the program by pressing BREAK and change Line 40 to read:

```
40 A=USR(RAND,100,101)
```

This will change the random number range from 0-319 to 100-101, a much smaller range. After changing the program, RUN it. See how much more slowly the columns grow? Even in ultra-fast machine code, the "hat" method has speed problems. What can we do to fix this problem? Our next program will show a technique which works just fine.

Who *was* that masked program?

One of the many nice things about assembly language is the degree of control you have over the computer. You can rewrite I/O routines, alter the display with control structures known as "interrupts," and manipulate data in many useful ways. We're going to use this latter feature to help us write a better, faster random number generator.

The reason our first random number subroutine didn't work as fast as we wanted was that it was taking every number that came along and checking to see if it was in the specified range. Sooner or later, a number comes along that fits, but we don't want to wait that long. If you're interviewing people for a nuclear physicist's job, you don't want to talk to everyone in the state of New York, so you place a classified ad listing the qualifications—to limit the number of people you have to interview. That's just what we're going to do, only we'll do it with numbers.

Inside the computer, all numbers are stored in *binary* format, a series of on or off *bits*. Using a technique called "masking," we'll preprocess the random numbers, making a match in the range we want more likely.

Here's how it works. First, we get and store the limits of random number values, say, from 200 to 1580. Next, we find the difference between these "endpoint" values, which, in this case, is 1580−200, or 1380. Knowing this range makes the random number generation much easier, since we only have to generate a number from 0 to 1380, then add the low limit of 200 to it.

The real "meat" of this technique lies in masking the "raw" random number, so that it will be more likely to fall into the specified range. We take the binary representation of 1380 and make a mask that stops at the highest bit, like this:

```
1380: 00000101 01100100
MASK: 00000111 11111111
```

Next, we build a 2-byte random number from the RANDOM location, then AND it with the mask, like so:

```
RANDOM: 11001011 01101001 = 52073
  MASK: 00000111 11111111
        -----------------
RESULT: 00000011 01101001 = 873
```

As you can see, the original random number 52073, has been masked down to 873, which is within our range of 1380. We then add 200 (the low limit of our random number) to the previous result, giving a final random number of 1073.

It is possible for the masked random number to exceed our range, but if that happens, we merely try the operation again. In any case, it's much faster than the "hat" method.

```
0100 LOWL =   $CB          ;LOW LIMIT
0110 LOWH =   $CC
0120 HIGHH = $CE           ;HIGH LIMIT TEMP
0130 RANGEL = $CF          ;RANDOM # RANGE
0140 RANGEH = $D0
0150 RESLO = $D4           ;BASIC'S RESULT
0160 RESHI = $D5
0170 RANDOM = $D20A        ;RAND # REGISTER
0180 ;
0190        *=  $0600      ;SUBROUTINE START
0200 ;
0210        CLD            ;BINARY MATH!
0220        LDA #0         ;INITIALIZE...
0230        STA LOWL       ;LOW RANGE...
0240        STA LOWH       ;DEFAULT (0)
0250        PLA            ;GET # OF ARGS
0260        CMP #1         ;1 ARGUMENT?
0270        BEQ ARG2       ;YES!
0280        PLA            ;PULL AND STORE
0290        STA LOWH       ;LOW RANGE
0300        PLA
0310        STA LOWL
0320 ARG2 PLA              ;PULL AND STORE
0330        STA HIGHH      ;HIGH RANGE (HI)
0340        PLA            ;PULL HIGH LO
0350        SEC            ;SUBTRACT...
0360        SBC LOWL       ;LOW LIMIT...
0370        STA RANGEL     ;FROM...
0380        LDA HIGHH      ;HI LIMIT...
0390        SBC LOWH       ;AND GET THE...
0400        STA RANGEH     ;RANDOM RANGE!
0410        LDA #$FF       ;INIT LOW MASK
0420        STA LOMASK
0430        LDX #0         ;START W/HI BIT
0440 HILOOP LDA BITS,X ;GET TEST BIT,
0450        AND RANGEH     ;IS IT ON?
0460        BNE GOTHLM     ;YES! GET HI MASK
0470        INX            ;NEXT BIT
0480        CPX #8         ;DONE ALL 8?
0490        BNE HILOOP     ;NO, LOOP BACK.
0500        STA HIMASK     ;ZERO HI MASK,
0510        TAX            ;ZERO X REGISTER
0520 LOLOOP LDA BITS,X ;GET TEST BIT,
0530        AND RANGEL     ;IS IT ON?
0540        BNE GOTLLM     ;YES! GET LO MASK
0550        INX            ;NEXT BIT
0560        CPX #8         ;DONE ALL 8?
0570        BNE LOLOOP     ;NO, LOOP BACK
0580        STA LOMASK     ;ZERO LOW MASK
0590        BEQ RNDIT      ;NOW GET RAND#!
0600 GOTHLM LDA MASKS,X ;GET MASK,
0610        STA HIMASK     ;SAVE IT,
0620        JMP RNDIT      ;AND GET RAND#!
0630 GOTLLM LDA MASKS,X ;GET MASK
0640        STA LOMASK     ;AND SAVE IT.
0650 RNDIT LDA RANDOM ;GET RANDOM #
0660        AND HIMASK     ;MASK IT OFF,
0670        STA RESHI      ;SAVE HI RAND#
0680        LDA RANDOM     ;GET ANOTHER,
0690        AND LOMASK     ;MASK W/LOW MASK
0700        STA RESLO      ;AND SAVE IT.
0710        LDA RESHI      ;IS HI BYTE
0720        CMP RANGEH     ;>LIMIT?
0730        BCC LOWOK      ;NO, BOTH OK!
0740        BNE RNDIT      ;TOO BIG, RETRY!
0750        LDA RESLO      ;IS LOW BYTE
0760        CMP RANGEL     ;>LIMIT?
0770        BEQ LOWOK      ;NO, IT'S =, OK!
0780        BCS RNDIT      ;TOO HIGH, RETRY!
0790 LOWOK LDA RESLO ;NOW FINISH...
0800        CLC            ;BY ADDING...
0810        ADC LOWL       ;BASE VALUE...
0820        STA RESLO      ;TO RANDOM...
0830        LDA RESHI      ;NUMBER...
0840        ADC LOWH       ;AND RETURN...
0850        STA RESHI      ;TO BASIC!
0860        RTS
0870 ;
0880 ;DATA ITEMS
0890 ;
0900 BITS .BYTE $80,$40,$20,$10
0910      .BYTE $08,$04,$02,$01
0920 MASKS .BYTE $FF,$7F,$3F,$1F
0930       .BYTE $0F,$07,$03,$01
0940 LOMASK .BYTE 0
0950 HIMASK .BYTE 0
0960        .END
```

Figure 6.

Figure 6 shows the assembly code for the random number masking method. Let's walk through it together, finding out how it works.

Line 210 clears the decimal mode, to ensure that we're working with binary arithmetic. This is absolutely essential in this program, since we'll be doing addition.

Lines 220-310 retrieve the low random number limit, just as in Figure 2. Once again, if only one argument is sent by BASIC, the low limit will default to 0.

Lines 320-330 pull and store the high byte of the upper range limit temporarily.

Lines340-400 pull the low byte of the upper limit, then subtract the low limit from the upper limit, giving the range of values. This number is stored in the locations RANGEL and RANGEH.

Lines 410-420 initialize the low byte mask to $FF (11111111 binary).

Lines 430-500 make up a loop which scans the high byte of the range to find the first "on" bit. This is done by using the BITS table at Lines 900-910. The X register is used to index each byte in the bits table, which is, in turn, ANDed with RANGEH. If the result of the AND operation is nonzero, the bit is on, and the program branches to GOTHLM to select the proper mask for the high byte. If no bits are on in the high byte of the range, the HIMASK mask is set to 0. Three typical bytes and their associated masks are shown in Figure 7.

```
HI BYTE: 10110001
   MASK: 11111111

HI BYTE: 00110100
   MASK: 00111111

HI BYTE: 00000000
   MASK: 00000000
```

Figure 7.

Lines 510-590 perform the same function as Lines 430-500, except that they find the highest bit in the low byte of the range. This code is only performed if no bits were found in the high byte of the range. If no bits are on in the low byte, the mask is set to 0, and the program will branch to RNDIT, where a random number will be generated.

Lines 600-620 load the appropriate high-byte bit mask from the MASKS table, placing it in the location HIMASK, then jump to RNDIT, to generate a random number.

Lines 630-640 load the mask for the low byte of the random number from the masks table. This byte is placed in LOMASK.

Lines 650-700 load random bytes from the location RANDOM, mask them with the LOMASK and HIMASK masks, and place them in the RESLO and RESHI bytes. Remember, we must still compare this number to the random number range to be sure it's not too big, before returning to BASIC.

Lines 710-780 perform a 2-byte comparison operation RESLO & RESHI and RANGEL & RANGEH. If the random number generated is greater than the range, the program loops back to RNDIT to try again.

Lines 790-850 are executed when the random number generated is acceptable. They add the random value to the low range limit, placing it back into RESLO and RESHI. At this point, the subroutine is finished, and we have a random number between the specified upper and lower limits.

Line 860 returns to BASIC with the RTS instruction.

Lines 900-930 are .BYTE directives which set up the bits and masks tables. These are used in Lines 440-640 to set up the appropriate data mask values. Note that each table is made up of 8 bytes, and that each byte of the masks table is the mask for the corresponding byte of the bits table.

Lines 940-950 are the storage locations for the high and low byte masks.

Figure 8 is a BASIC program with the "masking" random number subroutine. Type it in and RUN it.

```
1 REM *** RANDOM NUMBERS ***
2 REM
3 REM MASKING METHOD
4 REM FASTER THAN THE HAT!
5 REM
10 DIM D(319):FOR X=0 TO 319:D(X)=192:
NEXT X
20 GRAPHICS 24:SETCOLOR 2,0,0:COLOR 1
30 FOR X=1536 TO 1687:READ N:POKE X,N:
NEXT X:RAND=1536
40 A=USR(RAND,0,319)
50 D(A)=D(A)-1:PLOT A,D(A)
60 GOTO 40
100 DATA 216,169,0,133,203,133,204,104
,201,1,240,6,104,133,204,104,133,203,1
04,133,206,104,56,229,203
110 DATA 133,207,165,206,229,204,133,2
08,169,255,141,150,6,162,0,189,134,6,3
7,208,208,26,232,224,8
120 DATA 208,244,141,151,6,170,189,134
,6,37,207,208,19,232,224,8,208,244,141
,150,6,240,15,189,142
130 DATA 6,141,151,6,76,88,6,189,142,6
,141,150,6,173,10,210,45,151,6,133,213
,173,10,210,45
140 DATA 150,6,133,212,165,213,197,208
,144,10,208,232,165,212,197,207,240,2,
176,224,165,212,24,101,203
150 DATA 133,212,165,213,101,204,133,2
13,96,128,64,32,16,8,4,2,1,255,127,63,
31,15,7,3,1
160 DATA 0,0
```

Figure 8.

Once again, you will see the random numbers selected graphically represented by columns on your screen. As you can see, the subroutine returns random values quickly. Now stop the program with the BREAK key and change Line 40 to read:

```
40 A=USR(RAND,100,101)
```

RUN the program again. See how fast columns 100 and 101 grow? Seeing is believing: the masking method of generating random numbers gives much faster results than the "hat" method, even when the random number range is small.

Don't change that dial!

Next issue, we'll delve into new areas of assembly language programming on the Atari personal computers. Until then, study these program examples to increase your understanding. Remember, if you get stuck, you can contact Charles Bachand or me on CompuServe, or by writing. □

Boot Camp
c/o ANALOG Computing
P.O. Box 23
Worcester, MA 01603

SUFFERING FROM
Atari Writer
plus Printer Woes?

Rx: PRINTER FORMATTED DATA DISK

- EACH PRINTER DATA DISK WILL GIVE YOU **ALL** AVAILABLE OPTIONS POSSIBLE WITH YOUR PRINTR & ATARI WRITER
- NOW YOU CAN EASILY UNDERLINE, SUPER/-SUBSCRIPT, ELONGATE, DOUBLE COLUMN PRINT, CHANGE FONTS, ETC.

AVAILABLE FOR

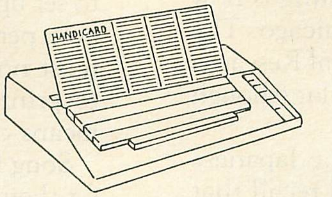
DOT MATRIX PRINTER $14.95	DAISY WHEEL PRINTER $24.95
BMC - C. ITON - EPSON - GEMINI - LEGEND - OKIDATA - PANASONIC - ATARI - RITEMAN - CITIZEN *PLUS OTHERS*	SILVER REED PANASONIC JUKI - DIABLO - OLYMPIA SMITH CORONA *PLUS OTHERS*

SPECIFY PRINTER MAKE & MODEL
Send Check or Money Order to:

AT A GLANCE™
Dept. 3, 86 Ridgedale Avenue
Cedar Knolls, NJ 07927
(NJ Residents add 6% Sales Tax)
Dealer Inquiries Invited

CIRCLE #128 ON READER SERVICE CARD

NEED HELP?
Use Handicards!™
Quick Reference Instructions for Atari® Programs



- Organized commands
- Easy to read
- Durable plastic (11" x 4 1/4")
- Use on or off computer

Now available for:
ATARIWRITER®
Beginning BASIC
(with Error Codes)

Only $8.95 ea. postpaid
(NY residents add sales tax)
®Trademark of Atari Corp.

Handi Publishing Inc.
P.O. Box 453
Ardsley, NY 10502

CIRCLE #135 ON READER SERVICE CARD

ATTENTION ATARI DISK DRIVE OWNERS
Back up your valuable software.

THE CHIP

THE CHIP with Archiver/Editor Software for the Atari 810 and 1050 disc drives. Includes Disassembler & Sector Editor. Includes Custom Formatter and Mapper. Backs up virtually any disk.
$129.95 POST PAID
Available soon for other Atari compatible disc drives.

HAPPY 810 ENHANCEMENT

Complete with Warp Speed software package. Plug-in installation — no soldering. Backs up any disc.
Regular Price $249.95
LIMITED SPECIAL OFFER $199.95
Soon available for other disc drives.

HAPPY OWNERS

Update your enhancement with Happy Version Archiver/Editor. Makes Happy drives compatible with the chip.

$39.95 POST PAID

All software for the Atari DISCOUNTED 30% or MORE.
Discounts on all Atari compatible hardware.
Send for free brochure on any of the above or for details on our software discounts.

SOUTHERN SOFTWARE
A DIVISION OF SOUTHERN SUPPLY COMPANY
1879 RUFFNER ROAD BIRMINGHAM, AL 35210
24 HOUR PHONE 205-956-0986
Order before 11 A.M. for same day shipping.

CIRCLE #136 ON READER SERVICE CARD

Need something interesting to do with that left over hamburger? How about a dessert for someone on a diet? The recipe you need is only seconds away with:

THE COMPUTER GOURMET

The Computer Gourmet is your complete kitchen helper. Twelve servings too many? A quick keystroke and your serving size is adjusted to the number you need. Want a shopping list of ingredients? No problem!

The Computer Gourmet even comes with its own complete set of recipes! (And its very easy to add your own.)

Automate your kitchen with
The Computer Gourmet

Available on disk for Atari™ computers (requires 48K).
Send $29.95 plus $2.00 for postage to:

NEW HORIZONS SOFTWARE
P.O. Box 180253
Austin, Texas 78718
Or, for more information, call (512) 445-1767.

New Horizons
Expanding Your Life

Contact us for information on all our products for Atari Computers.
Dealer inquiries invited. Atari is a trademark of Atari, Inc.

CIRCLE #137 ON READER SERVICE CARD

SUPER 3D PLOTTER

THIS IS IT!! AN INCREDIBLE 3D ANIMATION AND DISPLAY PACKAGE FOR ALL ATARI COMPUTERS WITH 48K RAM AND ONE DISK DRIVE. THIS PROGRAM WILL ALLOW YOU TO CREATE DETAILED 3D LINE DRAWN IMAGES, IN FULL SCREEN, HI-RES COLOR! LAY OUT AN ENTIRE CITY OR DESIGN COMPLEX OBJECTS WITH THE AID OF A HIGHLY VERSITILE SCREEN EDITOR. THEN, SIT BACK AND BE AMAZED AS YOU MOVE AND ROTATE YOUR CREATION, AT AN UNBELIEVABLE 3-6 SCREEN UPDATES PER SECOND!! THIS IS THE KIND OF ANIMATION THAT WAS CONSIDERED IMPOSSIBLE FOR AN ATARI. BELIEVE IT!! THIS PROGRAM ALLOWS MORE REAL TIME CONTROL THAN ANY CURRENTLY AVAILABLE 3D SYSTEM FOR ANY 8 BIT MACHINE:

- FULL ROTATION AND ROTATIONAL OFFSET CONTROL ON ANY AXIS!
- CHANGE VIEWER DISTANCE, MAGNIFY, AND ALTER PERSPECTIVE AT WILL!
- SPEED AND ACCELERATION CONTROL IN ALL DIRECTIONS.
- ALL MOVEMENT EXCEPTIONALLY SMOOTH AND ACCURATE
- DUMP A SUPER HI-RES (640 X 384) IMAGE TO EPSON AND MOST OTHER DOT MATRIX PRINTERS.
- CHANGE OR ADD TO YOUR CREATIONS AT ANY TIME. FULL DISK STORAGE CAPABILITY FOR YOUR FINISHED WORK.

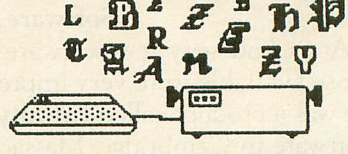
AND MUCH MORE!

FOR MORE INFO ON THIS MASTERPIECE, SEND S.A.S.E. TO:

RANDOLPH CONSTAN
23 BROOK PLACE
E. ISLIP NY,11730

CIRCLE #138 ON READER SERVICE CARD

Font Downloader



Design characters or symbols with a joystick. Save them & download sets to an **Okidata 92 or Epson FX80** printer. Comes with a library of fonts Requires 48k Atari®. On disk. Send $29.95 (check or money order-Specify printer) to:

Designer S. Inc.
7543 W. Hillsborough
Tampa, Fl. 33615
Visa & Mastercard accepted
Phone **(813)-884-6640**
Hours 10:00am to 9:30pm

MasterCard VISA

CIRCLE #139 ON READER SERVICE CARD

WINTER CES: Part 2

(continued from page 4)

Unfortunately, Commodore wasn't showing their recent acquisition, the Amiga. Its unveiling is now scheduled for June's Summer CES in Chicago. The latest word is that it will not use the Digital Research GEM operating system, but rather a similar (though not compatible) operating system.

It seemed as if there were dozens of the Japanese MSX computers in attendance. You will recall that MSX was announced just about three years ago, heralded as the beginning of a new wave of home computers. As it turns out, a lot has happened in the low-end home computer market in the last two years. What once looked like a shoo-in is now not so likely to be an automatic success.

MSX was to bring a standard to the low-end computer business. That may have been fine two years ago, but now that Commodore and Atari are the only two players in the low-end market, it's virtually impossible for a newcomer to break in. Sorry MSX, you missed your market window.

What of Apple and IBM? Both were no-shows at this CES, but some interesting rumors were circulating. Apple is understandably a little nervous about its overpriced Macintosh computer, especially in light of the new Atari ST line.

If the information I've received is correct, Apple will have announced significant price reductions by the time you read this. Apple might be selling the 128K Macintosh for about $1495 and the 512K Macintosh (the fat Mac) for about $2495.

Software.

Atari had very few software titles on display, but those available were very impressive. The most notable was a product called **Infinity**, developed by Matrix Software in Cambridge, Massachusetts. **Infinity** is an integrated software product that has a word processor, spreadsheet, relational database and telecommunications programs.

Infinity uses a technique called virtual memory for its disk storage, so that the size of any document is dependent on the amount of storage available on the disk, rather than on how much RAM the computer has. It will be available for the Atari XE and ST computers. Price is expected to be $50 for the XE version and $79 for the ST version. It's said that **Infinity** will even run on the Atari 800 and XL computers.

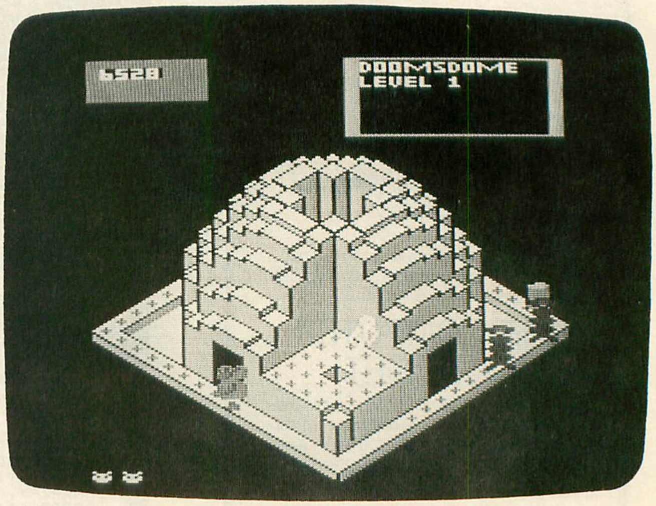
Another new Atari program is **Shopkeeper**. This is a modular program for small business use. It will sport inventory, accounting and electronic cash register functions. The first module, available in the first quarter of 1985, is essentially an electronic cash register emulator. It will keep track of inventory as sales are made, compiling a daily report which is directly transferable to the general ledger module (to be available in the second quarter). Atari said that there would be six modules in the series.

Still another Atari software title is the **Silent Butler**. This home financial program balances multiple checking and credit card accounts. It allows the user to set up various categories for financial tracking and even permits easy maintenance of tax deductions. The most novel aspect of this new program is its ability to print on your own ordinary personal checks by means of a plastic holder.

Song Painter is Atari's music construction program for their current line. It allows the user to place standard notation musical notes on the screen by using joystick-controlled, self-explanatory icons. Because of these features, this program is easy to use and will allow even musical novices to tinker with making music.

Atari finally announced that the **Plato** cartridge would become available in the first quarter of 1985. However, Control Data (not Atari) will be marketing this telecommunications learning aid. The official name of this product is the **Learning Phone**, and it will retail for under $50.

Other new software products include the **Atari Tutorial**, a cartridge which explains the XL parallel bus, ROM, RAM, graphics, and so on. **Crystal Castles** and **Mario Bros.** are both cartridges based on the coin-op games.

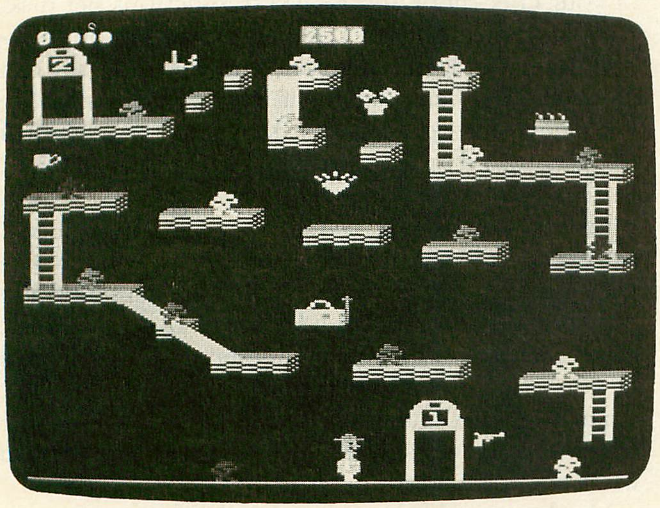


Crystal Castles.

Batteries Included, the Canadian firm most noted for its Commodore products, was showing some new Atari software. They haven't been in the Atari market long, but their recent product introductions for the Atari have been strong. The integrated software package called **HomePak** contains a word processor, database and terminal program. (See the reviews of **HomeTerm** and **HomePak** in **ANALOG Computing**'s issues 25 and 28, respectively.) Voted the 1984 bargain of the year by *Infoworld*, **HomePak** lists for $50.

Another recent release for the Atari is **B/Graph** (reviewed in issue 15). Although first published over two years ago by In Home Software, **B/Graph** had been unavailable to Atari owners for almost a year. It offers the capability to do high quality charts, graphs and statistics for business or educational use. **B/Graph** lists for $70.

The major news coming from Batteries Included is that their **Paper Clip** for the Atari is almost completed. **Paper Clip** is an extremely powerful word processor. It offers dual text windows, print previewing, dozens of printer drivers, macro capability and a host of useful features. Two of the features I particularly like are the automatic word count command and the two-letter reversal command for correcting mistakes like *hte*. **Paper Clip** will sell for about $80.



Bounty Bob Strikes Back.

Big Five Software has announced the sequel to their very popular **Miner 2049er**. Called **Bounty Bob Strikes Back**, this new game provides more of the same type of hopping, jumping and climbing activity that made the original game such a success. **Bounty Bob** is available for the Atari computer now, for a rather steep $49.95. It comes on a bank-selecting 40K ROM cartridge. The new game features improved graphics and sound, and over twenty screens. The title screen and the high score screen are very clever and amusing. Bill Hogue and Curtis Mikolyski, its programmers, have done an excellent job.

Broderbund had only one significant product announcement for Atari owners. The **Print Shop** will be available for the Atari computer by the time you read this. The **Print Shop** allows you to make greeting cards for family and friends, stationery for personal or business use, banners, signs, notices, advertising flyers. . .whatever you want.

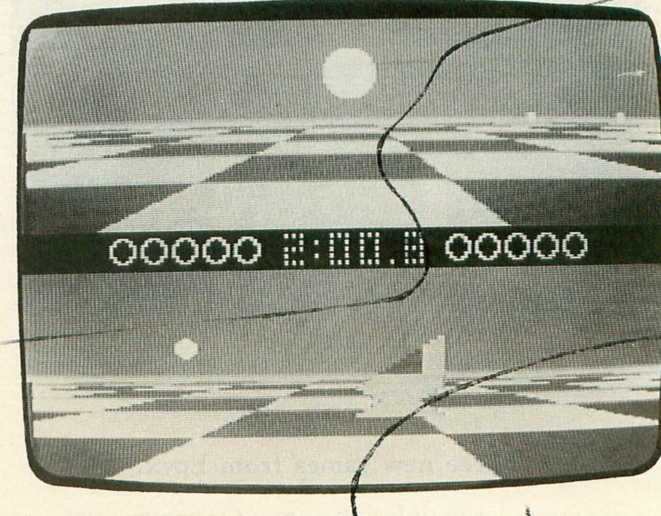
The program is menu-driven and easy to use. I've seen the results, and they're quite good. The **Print Shop** provides eight type styles in various sizes, dozens of already created pictures and symbols, a text and graphic editor to allow you to create your own designs, and numerous border designs and pattersn for use in your printed output. . .all for $45.

CBS Software had what seemed to be dozens of new titles but few, if any, for the Atari computer. **Dr. Seuss Fix-up the Mix-up Puzzler** is an electronic jigsaw puzzle that features six favorite Dr. Seuss characters, including the Cat in the Hat. Each puzzle is randomly designed and has five skill levels. The more advanced levels divide the picture into more and smaller pieces. The list price for this educational game is $30, and it will be available in March of 1985.

No other titles for the Atari were announced by CBS, but recent educational games include: **Sesame Street Astro Grover**, a counting and adding game that provides learning and fun for children aged 3 to 6; **Sesame Street Learning Go Round**, a letter recognition and simple spelling game for youngsters 3 to 6; and the excellent **Success with Math** series of self-paced math tutorials for children in grades 6 through 12.

Electronic Arts, less than two years old, currently has about fifteen software titles for the Atari. In their first year of existence, they supported Atari computers with over a dozen software titles, including such greats as **Pinball Construction Set, M.U.L.E., Seven Cities of Gold** and **Archon**. Unfortunately, Electronic Arts announced no new titles for Atari at the CES, although they did have some interesting new Commodore 64 material 64. Hopefully, we'll see these ported over to the Atari sometime this year.

Epyx had some surprising news. Lucasfilm's **Ballblazer** and **Rescue on Fractalus**, originally developed over eight months ago for the old Atari, will be distributed on disk by Epyx, rather than on cartridge by Atari. Both games are said to contain additional features not found on the originals.



Ballblazer.

Ballblazer is a futuristic two-player fantasy sport game, played at high speeds on a split screen showing both players' points of view. **Rescue on Fractalus** is a space action strategy game, featuring excellent

3-D animation. The player flies a fighter spacecraft to the planet Fractalus, to rescue downed pilots, battle enemy saucers and destroy gun emplacements.

In addition to announcing these Lucasfilm games, Epyx reported that they had signed a deal with Lucasfilm to bring out two future games for the Atari and other computers. The current disks will retail for about $35.

Epyx introduced four of their own action/strategy games. **Summer Games II** is a follow-up to their previous Olympic-style game. New events in **Summer Games II**: fencing, cycling, equestrian competition, kayaking and others. **Two-on-Two Sports** is a sports game with truly cooperative team play. Enthusiasts are offered the chance to play against the computer, as a team, in four popular sports—volleyball, soccer, football and baseball. Players can also face each other individually or in pairs, and a single player can compete with a single computer opponent.

Epyx's **FBI** takes the fun approach to simulations. Players are challenged to pass the Bureau's qualification tests. You can try your skill on the combat pistol range, attempt the obstacle course, challenge your memory by constructing "mug" shots, or take a general examination on your knowledge of criminology and more. Pass the test and you can become a "G-Man."

The fourth new action/strategy game from Epyx is called **The Right Stuff**. It's billed as a state-of-the-art flight simulation game. You are in the cockpit of a World War II Spitfire in the Battle of Britain. With joystick in hand, you take off, intercept the enemy in aerial dogfights and then land. Get those goggles, silk scarfs and leather jackets out for some seat-of-the-pants flying.

Three new games from Epyx.

Two popular mainframe computer games were also introduced by Epyx. **Empire** allows you to strategize your way to global domination, and the mission in **Rogue** is to make your way through a maze of seemingly never-ending dungeons, returning with the "Amulet of Yendor."

In **Moreta: Dragonlady of Pern**, the sequel to **Dragonrider of Pern**, players must fight off a dreaded disease. This adventure role-playing game follows the characterizations in Anne McCaffrey's novels.

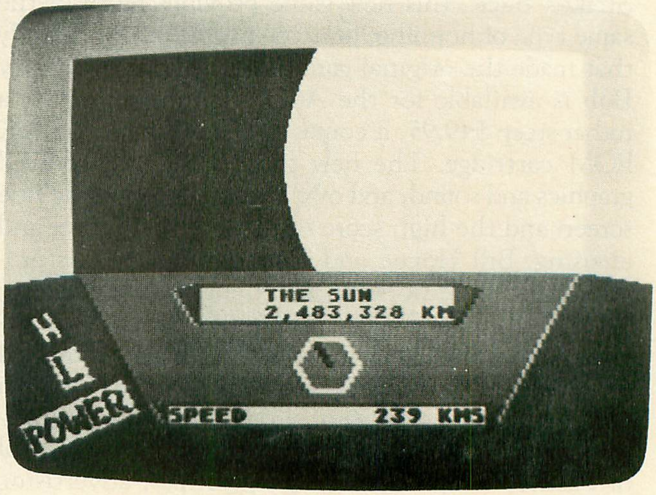
Epyx also announced three more games for the Atari and other computers: **G.I. Joe**, **Hot Wheels** and **Barbie**. All will be priced under $30 each and should be available soon.

Imagic displayed their latest release, **Chopper Hunt**, in which you play a soldier of fortune, piloting a helicopter in search of buried treasures. As you blast for objects, the holes you form are rapidly filled in by an enemy plane flying overhead. It's available on a flip-disk for Atari or Commodore at $19.95.

MicroLab announced a couple of new programs for the Atari. **Personal Banker** allows you to keep track of your checkbook (up to 100 transactions per disk, with a running balance). Expenses can be sorted according to budget categories, and the program can reconcile your bank statement. **HomeWriter** is MicroLab's new word processor. It seems to have most of the usual word processing features and is easy to use. Both will sell for about $50.

Several educational games were also introduced. **Barnaby Builder** and **Barnaby Math** are both arcade-style games to develop early math, logic, planning and perceptual skills. **Mind Bind** is a type of development game, presumably for young children. It was developed by educational psychologist Dr. Dorothy Rubin, but what it teaches is unclear to me. (Boy, I wish some of these companies would just come out and *tell* you what their product does.)

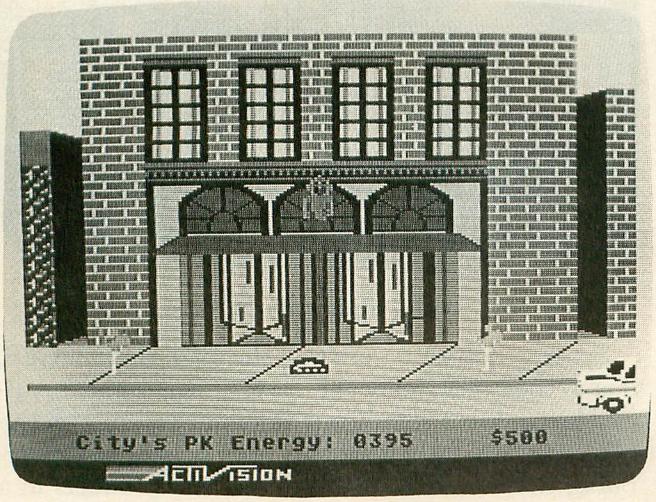
Mindscape publishes educational software of generally high caliber, representing valid learning principles. The company—new to the Atari market—was exhibiting only a few products for Atari.



The Halley Project.

Their most impressive Atari software title is **The Halley Project: A Mission in Our Solar System**. This is a real-time space adventure simulation written by Tom Snyder. Every planet, star or moon depicted here moves at the same rate of speed and in the same orbit as it does in our solar system.

The Halley Project uses high resolution graphics and attention to detail in what looks like a very good simulation of outer space. Players must qualify for the top secret "Halley Project" by completing a series of navigational tests. Through the tests and obstacles, the program helps players master facts about our solar system, including Halley's Comet and its orbit. Understanding of gravity, atmospheric conditions, orbital motion, relative size, position and orbits of planets and moons, plus locations of constellations and how eclipses work are all provided. **The Halley Project** will be available in March 1985 for $45.



Ghostbusters.

Activision was very visible at the show with several new products, all flashing a new, classy logo. You may never get the chance to be in the *Ghostbusters* sequel, but with your Atari, you can get a crack at bustin' some ghosts—as a G.B. franchise owner. As you travel through the city, you use a nuke accelerator pack and G.B. squad car to nab the ghosts—constantly on the watch for the marshmallow man. If it's a job well done, you'll have the chance to enter the Temple of Zuul for the grand finale.

The **Designers Pencil** offers a menu of over eighty English commands used to build "programs," to create pictures, colors and sounds. **Space Shuttle**, the popular VCS simulation, is now available for the home computer line. Other new titles eventually to be on the Atari are: **Rock N' Bolt, Web Dimension, Master of Lamps, Countdown to Shutdown, Alcazar: The Forgotten Fortress, Fireworks** and **The Great American Cross Country Road Race** (whew). Several of these are graphic adventures which Activision feels "are in tune with the consumer," as is their lowered pricing.

Microprose, *the* experts on flight "anything" for Atari computers, have several new war simulations on the way. **Crusades of Europe** pits the Americans/British against the Nazis, from D-Day through the Bat-

(continued on next page)

tle of the Bulge. **Decisions in the Desert** relives the fierce battle between Rommel's Afrika Korps and the British 8th Army. These games are a quantum leap beyond their previous **NATO Commander**—certainly state-of-the-art computer wargaming.

Later, you can be on the lookout for **Blitzkrieg 1940**, the **Sword of Zion** and the **Drive on Moscow**. The last three will retail for $39.95 each. Also to be released shortly is **Kennedy Approach**, a real-life air traffic control simulator, complete with speech synthesis "broadcast" messages from the computer. You control small civilian aircraft all the way up to the Concorde SST, for $34.95.

Parker Brothers, the folks who brought us **Q*Bert**, **Gyruss**, **Mister Do**, **Montezuma's Revenge** and **Chess**, announced only one new title for the Atari computer. **Q*Bert's Qubes**, a sequel to **Q*Bert**, will allow you, once again, to have that lovable **Q*Bert** hopping around your video screen. In this "upbeat sequel," **Q*Bert** must hop from qube to qube—rotating them with his feet while dodging menacing Meltniks, Shoobops and the Infamous Rat-A-Tat-Tat. (Hey, *I* didn't make these names up!) The game will come on a disk, sell for $25, and be available in the first quarter of 1985.

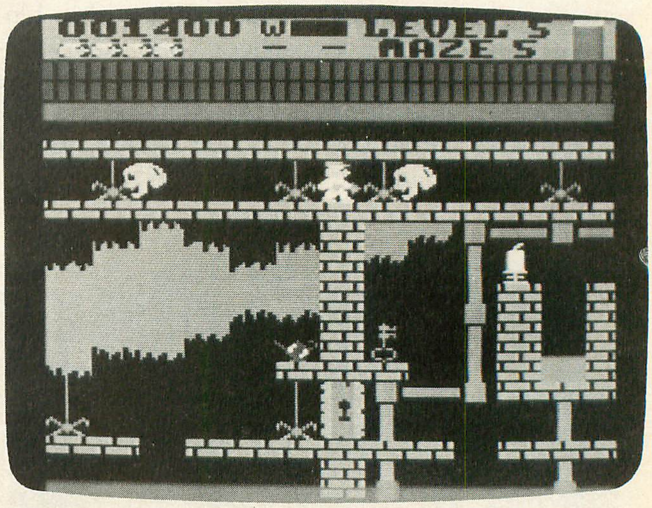
Spinnaker announced some new educational titles. **Math Busters** develops and improves the player's ability to use and manipulate the four arithmetic processes—addition, subtraction, multiplication and division. The program takes place within the overall context of a fast-moving adventure game. It's recommended for ages 8 through 14 and will retail for $20.

Rock 'N Rhythm (for ages 10 through adult) encourages the player to experiment with and develop his or her sense of rhythm, tempo and melody. All this takes place as the player's own band and recording studio composes, plays and records songs. The price will be $20.

Suncom announced a new version of **PQ—The Party Quiz Game**. With over 2700 new questions, the Bible Edition of **PQ** allows trivia buffs to test their knowledge of the Old and New Testaments. The questions were authored by Father John Massion of the St. Isaac Jogues Catholic Church in Niles, Illinois. Questions cover the history, geography, psalms, proverbs, parables, kings, people, places and events of the Bible. The Bible add-on questions retail for $24.95. The basic **PQ** game has been reduced to $49.95.

After a long lapse, Creative Software has a new Atari product, **Trolls and Tribulations**. The player leads trolls into underground caverns to find ancient treasures. The dangers are many, and there are thirty-two chambers to explore, at seven levels. Keep an eye on this company. This one retails at $24.95.

Epson introduced their latest printer to the world, the **HomeWriter 10**. The sleekest printer we've seen from anyone yet, the unit is color-coordinated with whatever computer it's interfaced to (using plug-in interface cartridges). Print modes include "draft" and "near letter-quality," at speeds of 1000 wpm (100 cps) in draft, or 16 cps in near letter-quality. Other features include friction paper feed, bi-directional printing and a 1K buffer, for $269 retail.



Trolls and Tribulations.

Synapse Software wasn't on the main floor of CES, due to their new financially austere management policy. Having lost a significant amount of money on the old Atari, and having recently been bought by Broderbund, they chose to exhibit their new products in a more private setting.

In the comfort of a hotel suite, they were proudly showing some excellent new text adventure games. **Mindwheel** is the first of these so-called "electronic novels." Packaged in hardbound book form, its early chapters set the scenes and story lines, and introduce the characters. The game really begins with the enclosed floppy disk.

Like other interactive fiction, **Mindwheel** makes the user the center of action. Decisions he or she makes will determine the plot of the story and, ultimately, the outcome. Interestingly, the manner in which the user talks with the various personae will elicit an in-character reply.

In **Mindwheel**, you must journey into the minds of four deceased people of extraordinary power. You take a mind-bending telepathic trip back to the beginning of human civilization, in order to retrieve the "Wheel of Wisdom." Robert Pinsky, published poet and Shakespeare lecturer, is the author of this first adventure.

In other Synapse news, the list prices on **SynFile+**, **SynCalc** and **SynTrend** have all been lowered to $50. This makes these already-excellent programs even more of a value. Also, most of their computer games now come two to a box and are attractively priced at about $25.

That's it for this report. . .the CES was, as you can see, a busy one. My socks are still with me, but Atari *did* capture its share of attention. □

SOFTWARE MOVIES: VISUALIZER
by Tim Kilby
MAXIMUS
6723 Whittier Avenue
McLean, VA 22101
400/800 Cassette or Disk $49.95

by Arthur Leyenberger

There are several programs currently available for the Atari computer that allow the creation of graphics screens. Most of these programs allow you not only to create your screens, but also to save them on disk or tape, and to print them on a printer.

However, **Software Movies: Visualizer** is the only program I know of that, in addition to the above features, also simulates a slide projector. Screens can be sequenced in any order (up to twenty per disk) and shown either automatically or manually. A cassette recorder may also be used to provide a voice narration or music, and to synchronize the slide show.

Written by Tim Kilby, **Visualizer** comes packaged in a plastic, book-style binder with a two-sided disk and a cassette tape for running the synchronized demo program.

Side one of the disk contains the program, and side two contains the sample slides. **Visualizer** is really three programs in one. If offers a graphics creation program, an electronic "slide" creator/projector program and a screen dump program. A jigsaw game is also provided. The program is easy to use, and the manual is brief but well written.

Slides are created using the graphics editor program. The graphics 7 screen is divided into two windows—the graphics window and the caption window. The graphics window is used for drawing and marking position, with a cross hair cursor indicating location.

The caption window contains prompts, menus and questions during slide creation. It also may contain a caption for the finished slide.

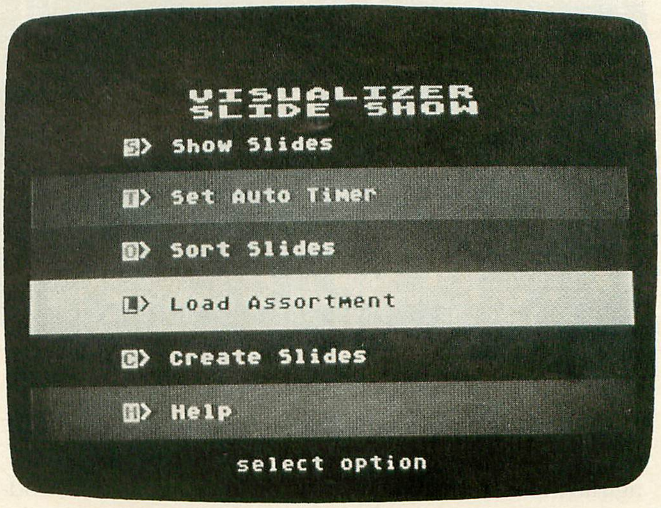
Visualizer uses the computer keyboard and a joystick. The speed of the on-screen cursor may be adjusted, and the joystick is used for drawing. The cursor control keys are used for fine cursor movement. Although the program is straightforward and easy to use, on-screen help is provided for any of the functions.

Four colors may be selected from the "paint pots" displayed at the bottom of the screen. These color registers may be changed at any time. For drawing, any of four line widths may be selected. In addition, built-in functions for circles, ovals, rectangles and diagonals are provided.

Any area of the screen can be filled, either with a solid color or a textured combination, with the fill option. A border may be drawn around the entire screen, and new fonts, created with other character set generators, may be used. The text function is especially useful.

Text can be created in either the graphics window or the caption window. Text may include letters, numbers, punctuation, mathematic operators and block graphics characters. In fact, any of the 128 keyboard characters can be drawn on the screen in any of four colors, in either normal or inverse video modes.

The text can be displayed in any of twenty-four varieties or styles. Tall, wide, italicized, shadowed or striped letters are available. Depending upon the options chosen, from eight to nineteen characters will fit on one line. I have created dozens of title slides for presentations using the text function of **Visualizer**. The shadow lettering is especially attractive and adds some class to the title slide.



Visualizer.

Once your slides are created, you may store them on disk. You can even merge slides. There is an animation function of **Visualizer** that can give your slides the illusion of movement.

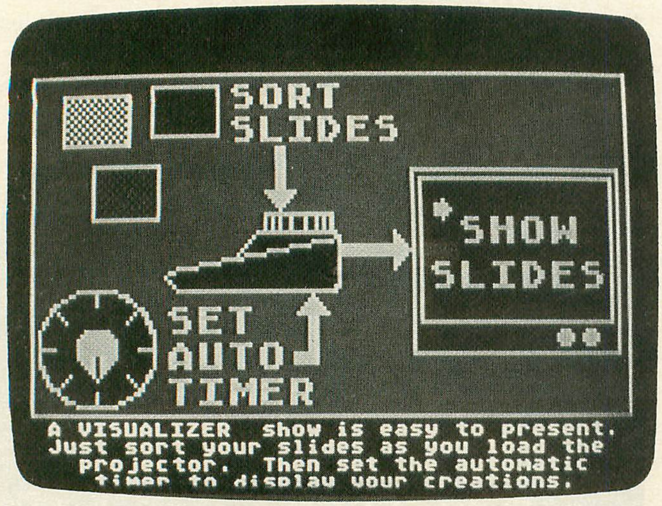
One-color switching alternates color #1 with the background color, to give an effect much like a flashing neon sign. Two-color switching switches colors #1 and #2 back and forth. A carefully-designed slide may appear to rotate by using this feature. Three-color switching provides the greatest amount of simulated movement. Here, the three line colors continuously alternate. The background color is the only one that remains static.

Three other animation options are available. Marquee operates on color #1. Moving colors appear, and the effect works well with large shapes. Sparkle is similar to marquee, in that it gives a sequence of color to any lines drawn in color #1. Its effect is somewhat more subtle than that of marquee.

Finally, rainbow animation varies the hues of anything drawn in color #1. Colors appear to be pastel, reducing the dramatic effect while still attracting attention to the shapes. The moving colors appear as constant luminance on a monochromatic screen, and the effect is best used with striped or textured shapes.

The animation option chosen for the slide will be saved, along with the picture, on the disk file.

The slide show portion of **Visualizer** simulates a rotary tray slide projector's operation. With a slide projector, a round tray is loaded with pictures which are then projected onto a screen, in sequence. The projectionist may advance the slides forward or backward and may set the projector's automatic timer to advance the pictures automatically.



Visualizer's projector works the same way. Slides that you create are selected and loaded onto an imaginary slide tray. The "electronic projector" has advance, reverse and automatic timer features, just like a real slide projector.

Once a disk of slides is loaded into the "projector," you may choose all or some slides and sort them in any order. This sequenced assortment may be saved for multiple screenings—and to avoid having to perform the sorting process in subsequent showings.

The built-in timer allows four preset time intervals, or you can enter your own interval. Also, the timer can be set to function with an Atari 410 or 1010 Program Recorder. The manual provides instructions for synchronizing the slide show with a prerecorded tape. Directions are also given for creating synchronization tapes with a stereo cassette tape deck or recorder.

My only criticism of **Visualizer** concerns the way in which slides are shown. A slide is displayed on the screen, then the screen blanks out while the next slide loads. Although the screen is dark for only a few seconds, it would be better if the next slide could load while the current slide is being displayed.

A utility menu is available at any time during the slide creation program. Slides may be renamed or deleted, and an index of slides currently on the disk may be viewed. Also, a disk can be formatted from within the slide creation program, to avoid having to load DOS and then reboot **Visualizer**.

In addition to providing instructions for creating, showing and printing slides, the 40-page manual contains some other useful information. Suggestions for parents on children's activities, as well as suggestions for making creative slides are included. There is also a section for programmers that has a program for loading **Visualizer** slides into your own BASIC programs.

Unlike many software companies, Maximus provides user support for their programs. A toll-free "hotline" phone number is given, for help with the program or to find out about their other products.

Visualizer is an excellent graphics package. It can be used for creative experiments with graphics or to produce good-looking title slides. My major use of the program has been to create title slides. Then I photograph the screens and use the real slides in my presentations. Many people think I have used an expensive stand-alone graphics system to produce these slides. When I tell them that I used an Atari computer to make the slides, they are amazed.

Thanks to **Visualizer**, my presentations are more professional and interesting. For those of you who fancy yourselves as budding video producers and directors, **Visualizer** could come in handy for making up very professional-looking titles. And, if you happen to have a video mixer, some really neat things can be done with this program and your video cassette recorder. □

Number Conversion Chart

Here's a handy little program, written by Scott Sheck, of Gaithersburg, Maryland. It will generate a chart of decimal numbers from 0 to 255, along with their hexadecimal and binary equivalents. The chart will fit nicely on one sheet of 8½×11 printer paper, and provides all those numbers at a glance.

```
1 REM ******************************
2 REM *                            *
3 REM * HEX/DECIMAL/BINARY CHART *
4 REM *       BY SCOTT SHECK       *
5 REM *  A.N.A.L.O.G. COMPUTING   *
6 REM *                            *
7 REM ******************************
10 DIM HEX$(2),H$(16),BINARY$(8):H$="0
123456789ABCDEF":POKE 201,5:OPEN #1,8,
0,"P:"
20 FOR X=1 TO 4:? #1;"DEC. HEX BINARY
 |   ";:NEXT X:? #1
30 FOR X=0 TO 63
40 FOR DEC=X TO 255 STEP 64:V=DEC:GOSU
B 60:V=DEC:GOSUB 80:? #1;DEC,HEX$;" ";
BINARY$;" |   ";:NEXT DEC
50 ? #1:NEXT X:END
60 REM DECIMAL TO HEX
70 FOR I=2 TO 1 STEP -1:T=INT(V/16):R=
V-16*T:HEX$(I,I)=H$(R+1,R+1):V=T:NEXT
I:RETURN
80 REM DECIMAL TO BINARY
90 FOR I=8 TO 1 STEP -1:T=INT(V/2):R=V
-2*T:BINARY$(I,I)=STR$(R):V=T:NEXT I:R
ETURN
```

•

INDEX TO ADVERTISERS

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

Put a $20,000,000 Thrill in your Computer!!!

The F-15 STRIKE EAGLE, one of the world's most sophisticated fighters, costs the Air Force more than $20,000,000.00 each. Now you too can strap into your ejection seat and prove how good you really are in exciting modern jet fighter combat. Fly combat missions, engage enemy aircraft, and destroy enemy ground targets from historic missions over Southeast Asia to today's defense of the strategic oil routes through the Straits of Hormuz.

F-15 STRIKE EAGLE (simulator) has all the features that makes the real EAGLE the great fighter aircraft it is — afterburners, multiple radars, air to air missiles, high explosive bombs, cannon, drop tanks, Electronic Counter Measures (ECM) flares, electronic ground tracking maps, Heads Up Display (HUD), outstanding 3-dimensional cockpit visibility, and realistic F-15 maneuverability. Your mission is to take off from your base, fly to and destroy your primary target through all the dangers of enemy territory including Surface to Air Missiles and enemy aircraft. Of course, you have to get back home again, too!!

F-15 STRIKE EAGLE is a very real simulation, accomplished with the guidance of real fighter pilots, and includes seven different combat missions, four skill levels, and an infinite number of exciting scenarios. F-15 will thrill and challenge you and give you the chance to prove you have the "Right Stuff" of an EAGLE fighter pilot!!

F-15 STRIKE EAGLE is available for Commodore 64, ATARI (48K), and Apple II (64K) computers. Suggested retail price is only $34.95. Find STRIKE EAGLE at your local retailer, or call or write for MC/VISA, or COD orders. Add $2.50 for Postage and Handling (Int'l add $4.00USD). MD residents add 5% sales tax.

Experience the reality of these other great simulations from MicroProse

As close to the Real Thrill of Flying as You Can Handle!!!

Heart Pounding Accelerated Real-Time Defense of Europe Against Soviet Invasion!!!

Sensational Assault Chopper Rescue Raids for Daring Pilots!!!

MicroProse Software

The Action is Simulated — the Excitement is REAL!!

10616 Beaver Dam Road
Hunt Valley, MD 21030
(301) 667-1151

Commodore 64, ATARI, APPLE, and IBM are registered trademarks of Commodore Business Machines Inc., ATARI Inc., APPLE Computer Inc., and International Business Machines Inc., respectively.

CIRCLE #145 ON READER SERVICE CARD