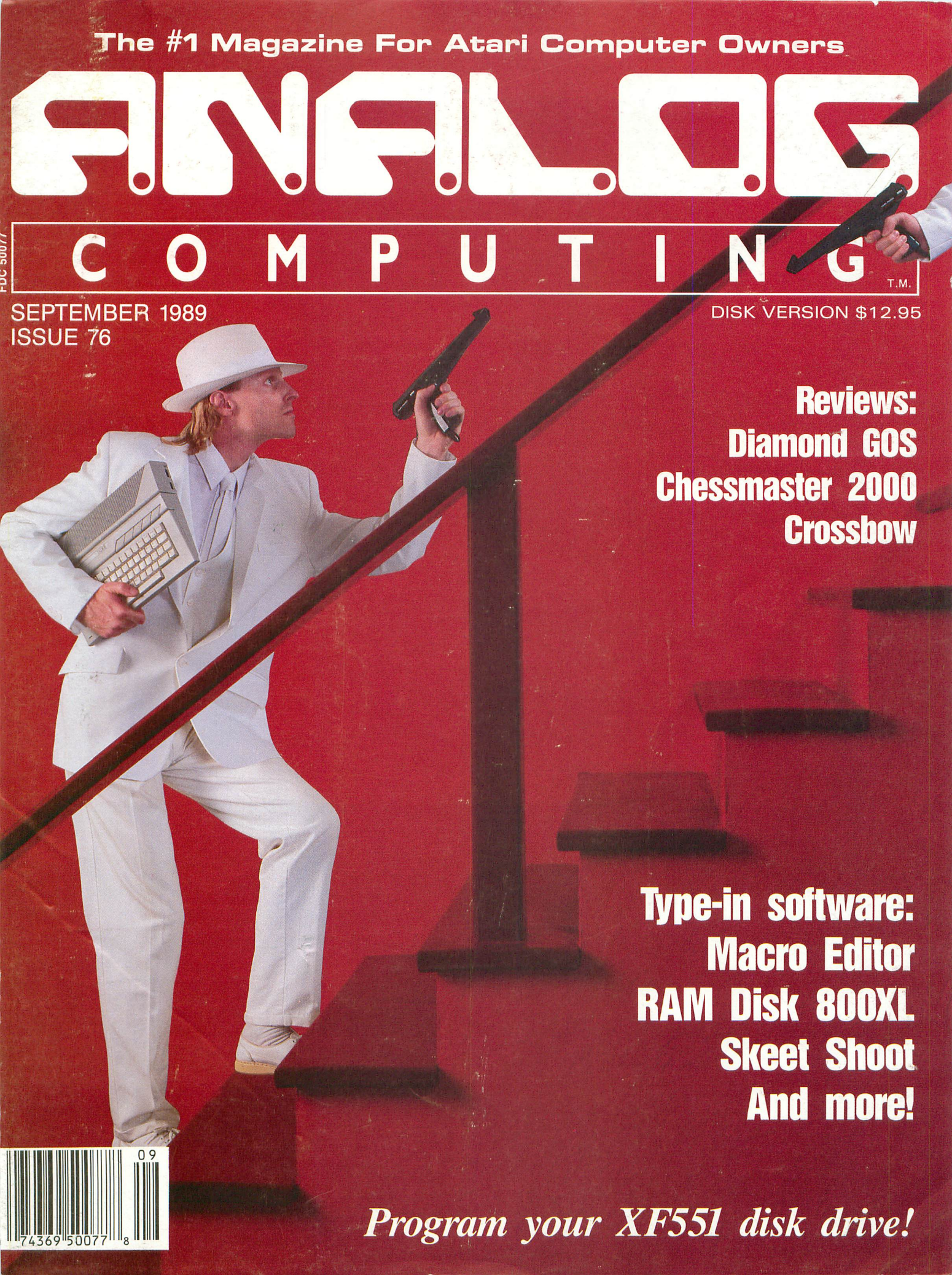
The #1 Magazine For Atari Computer Owners

ANALOG COMPUTING

T.M.

SEPTEMBER 1989
ISSUE 76

DISK VERSION $12.95



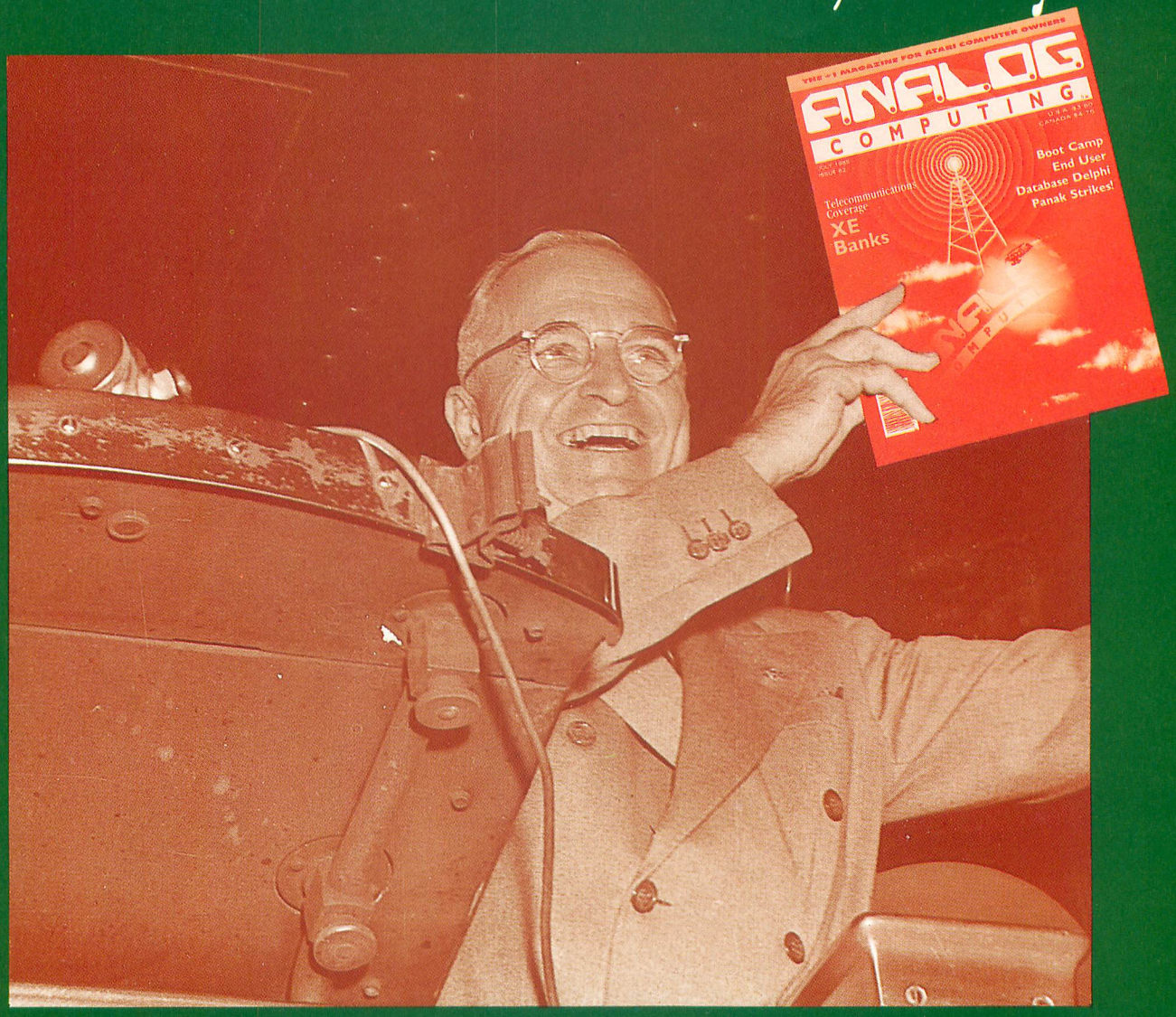
Reviews:
Diamond GOS
Chessmaster 2000
Crossbow

Type-in software:
Macro Editor
RAM Disk 800XL
Skeet Shoot
And more!

Program your XF551 disk drive!

09
74369 50077 8

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



Two Historic Facts:

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

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

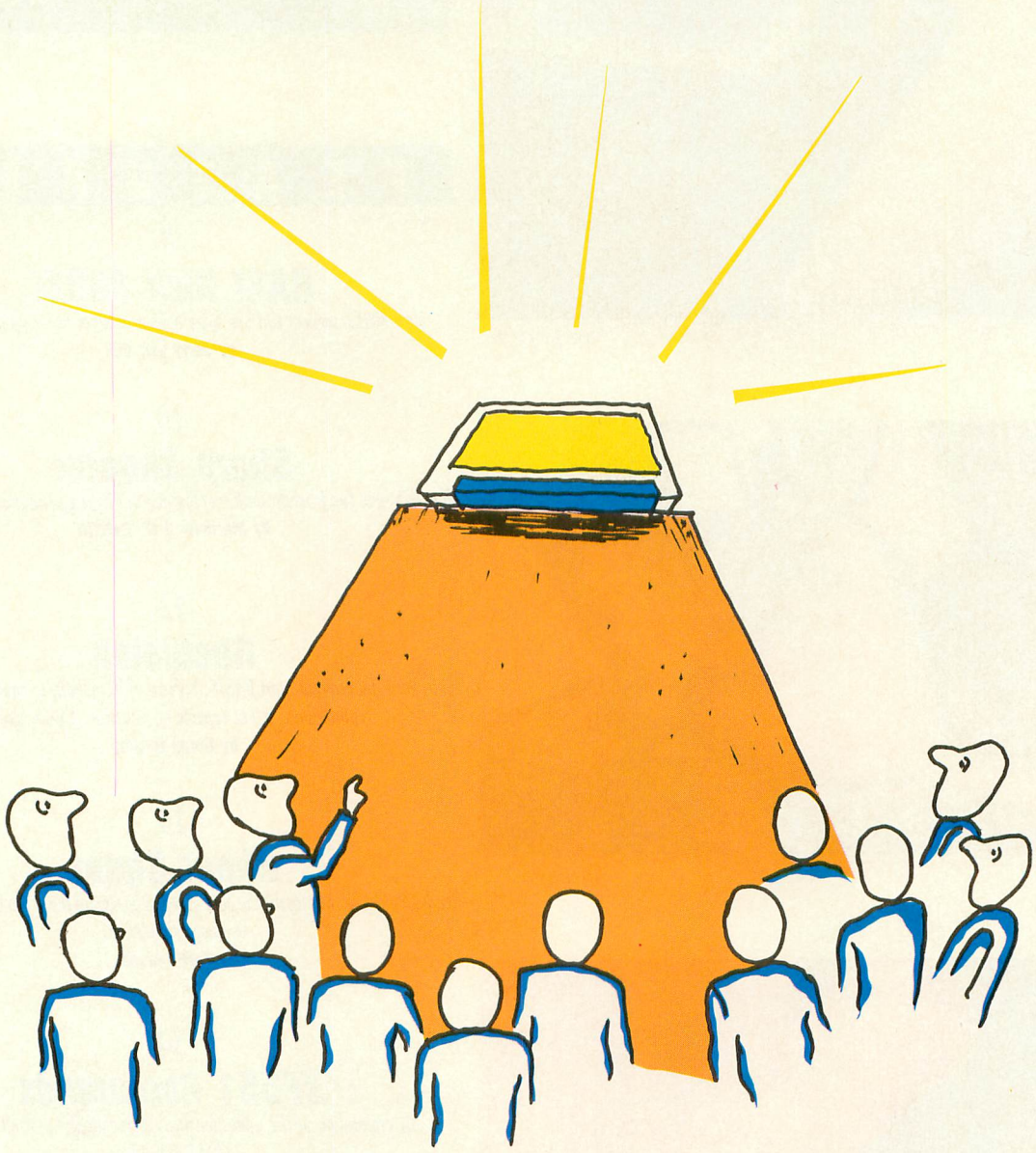
1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE!

Editorial



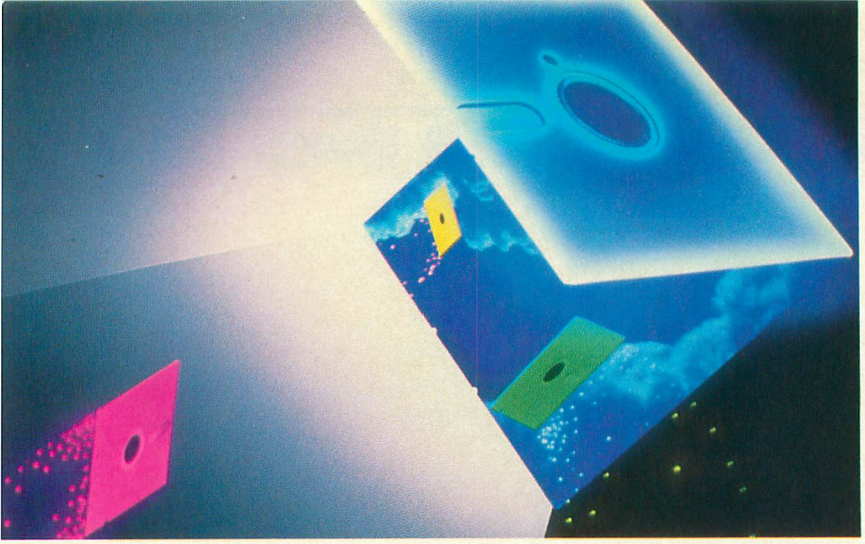
BY CLAYTON WALNUM

The latest Consumer Electronics Show was recently held in Chicago, and Atari amazed everyone by unveiling a new product that virtually no one had heard of before the show. Although the rumored 16-bit game system has yet to be released, Atari displayed a new hand-held color game machine. The unit, which is about the size of a videotape, has a 3½-inch screen capable of displaying 16 colors simultaneously. An eight-position controller pad takes the place of a joystick, while several other buttons take on various control and firing duties.

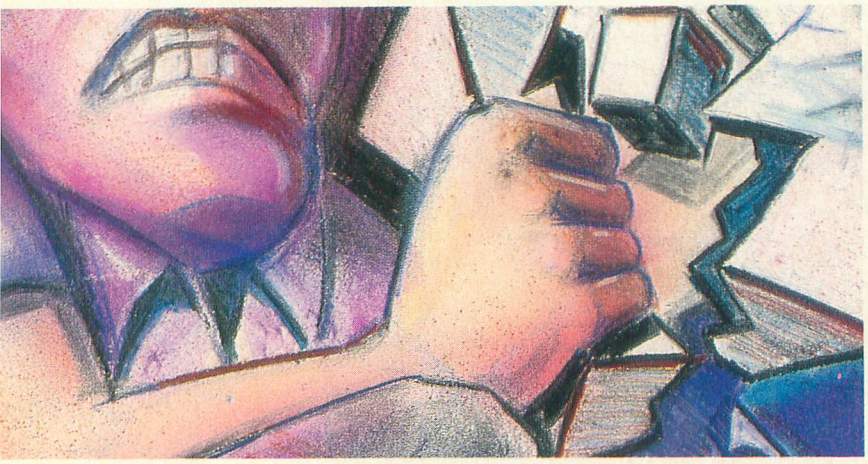
One of the machine's unique features is its ability to flip the screen image so that it can be held with the controller pad on either the left or right side. That ought to make all you lefties happy. But what really makes this unit special is that, unlike the hand-held game machines currently available, the new Atari model is not limited to a single game, but rather incorporates cartridges, just like its larger cousins. Six games from Epyx have already been announced. The "cartridges" are actually small cards about the size of a credit card. Each game card can hold as much as two megabytes of data, although the current cards hold only 128K.

The $149 machine, which is projected for release in September, runs for up to eight hours on six "AA" batteries and includes a headphone jack. How did Atari manage to slip such a surprise into the show? The fact is that the new game machine was originally developed by Epyx, which intended to release it themselves but for some reason (rumor has it that the announcement of the Nintendo "Game Boy," a similar machine, scared the powers-that-be at Epyx) decided to drop the project. Atari apparently decided the machine was a much better unit than the one planned by Nintendo (and, if all the specifications are accurate, it is), and decided to take a chance with it, making the necessary agreements with Epyx.

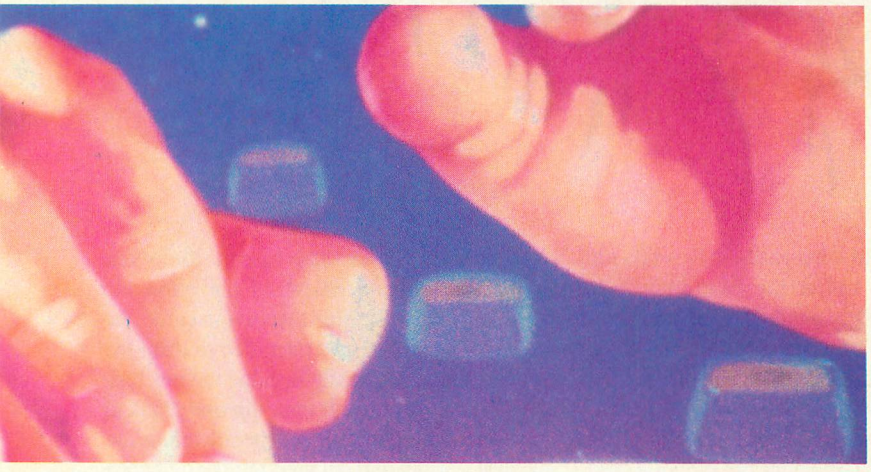
So although the new game machine wasn't developed by Atari's research department, it is a perfect addition to their videogame line. It will be interesting to see how Atari handles the marketing of this unique product—one that could prove to be immensely popular. Let's hope they take the aggressive approach they've been promising.



on page 8



on page 12



on page 58

Cont

FEATURES

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

e n t s

SEPTEMBER 1989
ISSUE 76

REVIEWS

COLUMNS

DEPARTMENTS

Where to Write

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

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

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

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

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

Advertising Sales

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

Permissions

Subscriptions

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

Authors

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

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

JE Publishers Representative

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

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

8 BIT NEWS

Disk utility package

A new package just released by Creative Software Systems provides disk-drive owners with a set of handy utilities, including a sector editor, a file copier and two sector copiers. The system supports most DOS functions—lock, unlock, rename, delete and format—and adds some new ones: verify, close and undelete. Also, directories may be sorted and printed. The utilities are fully menu-driven and run on any 8-bit Atari with at least 48K. The price is $15.95.

Creative Software Systems has also released a self-documenting disassembler, which can disassemble from a disk file, from memory or from a sector, inserting comments on key memory locations. The listing may be sent to a disk file or a printer. The disassembler sells for $5.95.

Creative Software Systems
8715 Valley View, #3
Barrien Springs, MI 49103
(616) 471-3745

CIRCLE #109 ON READER SERVICE CARD.

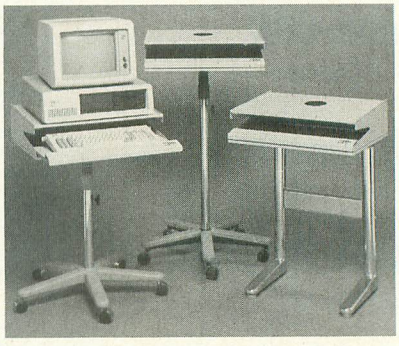


Image Scanner

Innovative Concepts has announced several new products for the 8-bit Atari computers, including Easy Scan II, a graphics image scanner that replaces the original Easy Scan and now supports graphics modes 8, 9, 10, 11 and 15. Scanned graphics can be printed, displayed on the monitor or saved to disk as standard 62-Sector picture files. The scanner, which sells for $99.95, requires an XL or XE with 128K and an Epson-compatible printer. Original Easy Scan owners can upgrade their software for $20.00.

Also available from Innovative Concepts is "Ramdrive + XL to XE," a 128K upgrade for the Atari 800XL. Innovative Concepts claims that this upgrade makes the 800XL fully compatible with the 130XE, including the extended ANTIC modes. The kit includes the upgrade board, RAM chips, and instructions for use with DOS 2.5, MyDOS, SpartaDOS and the SpartaDOS X cartridge. It sells for $59.95.

Finally, Innovative offers "The Happy Doubler," a utility that allows Happy 1050 owners to program up to eight drives for complete compatibility with ICD's U.S. Doubler. This $19.95 package also includes an extra disk full of additional utilities.

Innovative Concepts
31172 Shawn Drive
Warren, MI 48093
(313) 293-0730

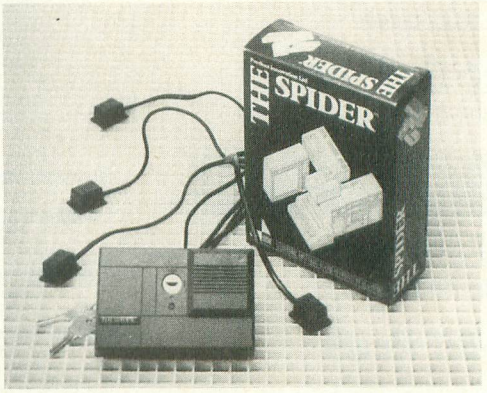
CIRCLE #108 ON READER SERVICE CARD.

New computer stands

CompuStac Company has announced their new CompuStac height-adjustable pedestal-type computer stands. The CompuStac is unique in that it can be adjusted to the exact height required by the user, and the pedestal design makes it convenient to get your computer off your desk while keeping it within easy reach. Prices vary depending on the model chosen.

CompuStac Company
819 West 4th Street, Suite 7
San Pedro, CA 90731
(213) 831-8017

CIRCLE #110 ON READER SERVICE CARD.



A helpful arachnid

If you're worried about your computer equipment getting stolen, there's something you can do about it. The Spider, a new product from AlteCon Data Communications, is a battery-operated alarm that will warn you when someone is tampering with your equipment. The Spider has five "tentacles," each of which may be attached to a piece of your equipment. If the tentacles are cut or ripped off, or the alarm itself is tampered with, a 98-decibel alarm will sound for up to two and a half hours. The Spider is priced at $107.50.

AlteCon Data Communications, Inc.
1333 Strad Avenue
North Tonawanda, NY 14120
(716) 693-2121

CIRCLE #111 ON READER SERVICE CARD.

A LETTER FROM THE PUBLISHER

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

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

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

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

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

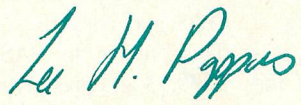
Don't panic! When you think about it, the merging of the magazines will allow us to produce a much nicer publication. And since the single magazine will be larger than either of the individual ones, we won't have to cut much from our content. In fact, after doing some analysis, we've discovered that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

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

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

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

Here's to the future!



Lee H. Pappas
Publisher

RAM Disk 800XL

by Jerry van Dijk

A RAM disk (a part of memory that DOS thinks is a fast disk drive) is a nice thing to have. Ask any 130XE owner. Unfortunately, the RAM disk driver that comes with DOS 2.5 only works with the extended memory of the 130XE. And the poor and helpless 800XL owner is left with nothing but dreams of all the things he could do if only. . . .

Building the RAM disk without glue

Before taking the program apart to see how it works, let's see it *at* work. Type in the data from listing one as MDRIVE.OBJ using the M/L EDITOR. Next format a fresh disk and write the system files to it (using option H from the DOS menu; make sure you're using DOS 2.5). Now copy MDRIVE.OBJ to the new system disk, and finally rename it to RAMDISK.COM.

Don't forget to mark the disk as containing the 800XL RAM disk to prevent future confusion with the 130XE version.

Putting it to the test

To use the RAM disk, simply boot the new system disk. After loading DOS.SYS, the RAM disk driver is automatically loaded. After a message, the boot continues normally with BASIC, DUP or AUTORUN.SYS. Now you have a RAM disk called D5: at your disposal. Try a directory of D5: (Press "A" from the DOS menu and type "D5:" at the "filspec" prompt). If everything went according to plan, you'll see a short flash (which cannot be helped since every time D5: is accessed the operating system is temporarily disabled) and find yourself with 108 free Sectors left on D5: for your use.

The D5: device can be treated like any other drive on your system, with four minor exceptions:

1. Part of the runtime code of the RAM disk driver uses page 6. From $6BC (1724) up to and including $6FF (1791) to be precise. So you'll have to be careful with programs that store ML routines here. Most will work without a hitch, however, and for those that won't. . .well, you can't have everything.

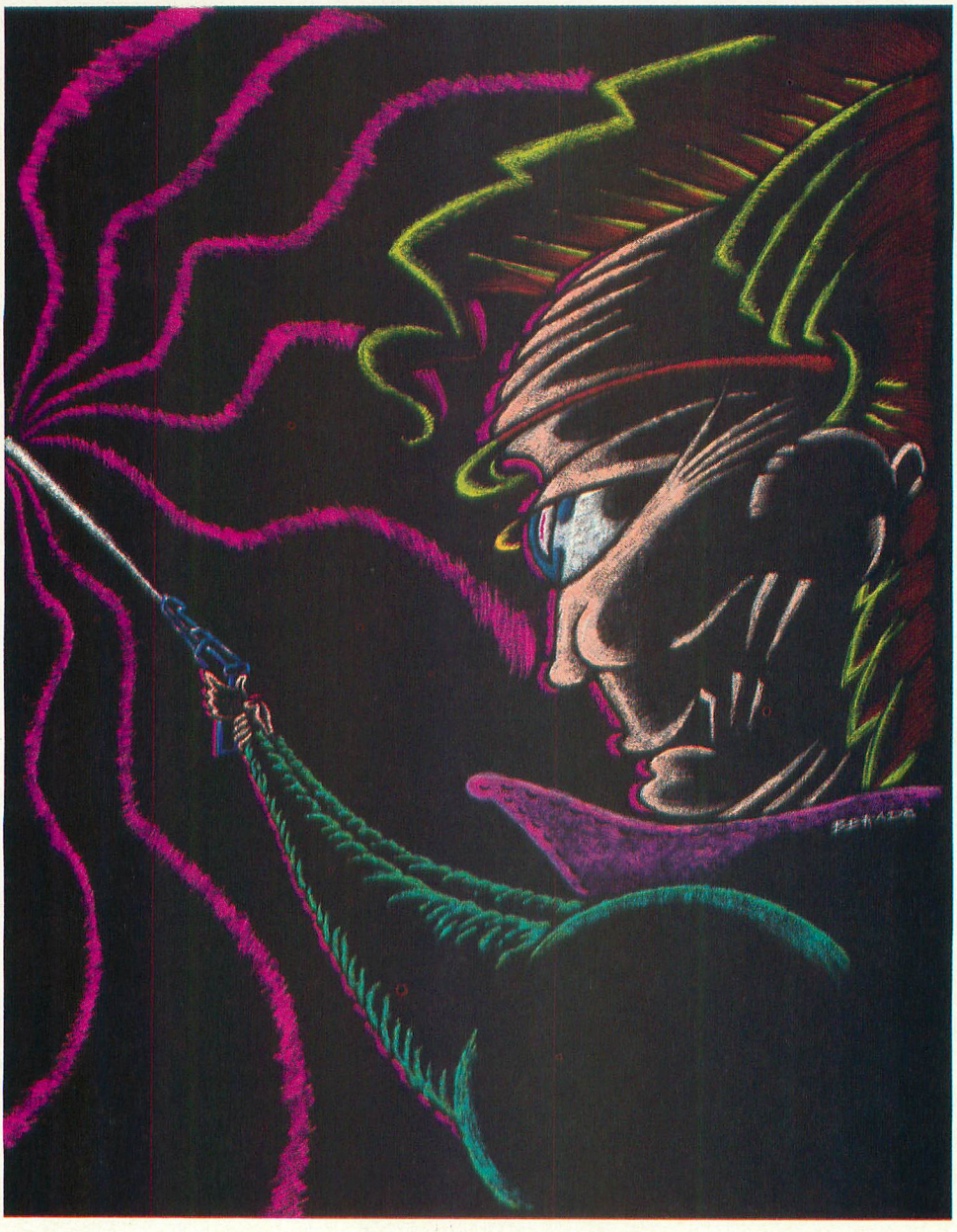
2. When power is switched off or the system crashes, you lose the contents of the RAM disk. Such is the nature of the beast, so beware. (It is, of course, Reset-resistant.)

3. Contrary to normal disks, D5: has only one directory Sector, which means the maximum number of files possible is eight. This is reasonable. The small number of available Sectors and the need to save them back to a normal disk when shutting down makes it unlikely that more files will ever be needed. It also means there are seven more Sectors for you to play with. If you do try to access one of the nonexistent directory Sectors, DOS returns with error code 144 (disk error).

4. Last, it's impossible to reformat the D5: device from DOS. The only way to reformat D5: is to run the RAMDISK.COM program again using the L option from the DOS menu. If you accidentally do try to format D5:, DOS will show you the errors of your ways with error code 168 (invalid command).

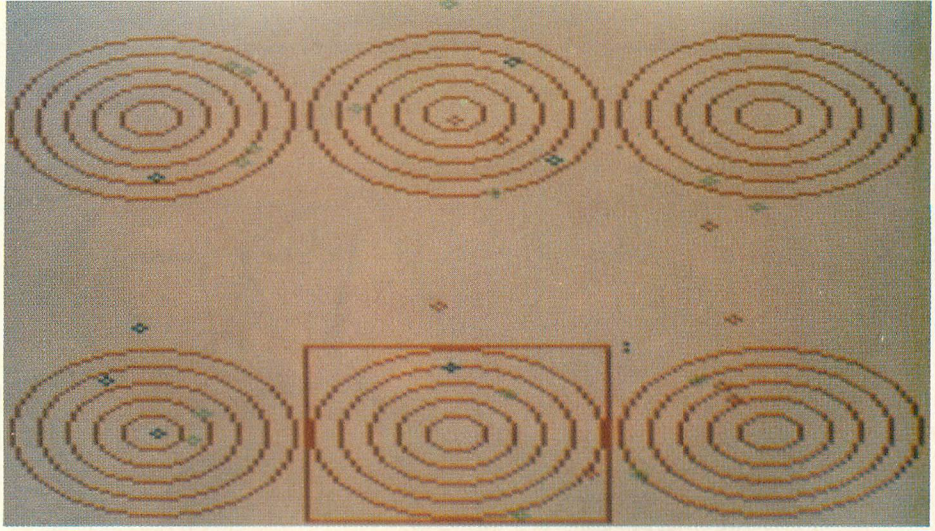
If you're working in BASIC, you can also use the RAM disk to store DUP.SYS (and, if activated, MEM.SAV) and access it instantly. To do this, boot a disk containing the 800XL RAM disk file. From BASIC, go to DOS and copy the DUP.SYS file to D5:. Then return to BASIC and type POKE 5439,53 and press Return. Type DOS again and the menu should appear immediately. Any BASIC program in memory will be overwritten, so you should have MEM.SAV activated.

continued on page 26



by Matthew J.W. Ratcliff

Sharp Shooter



Sharp Shooter is a light-gun game for any Atari XL, XE or XEGS computer equipped with an Atari light gun. A challenging target-practice game, it requires quick reflexes and a good eye, and will improve your shooting skills for *Bug Hunt*, *Barnyard Blaster*, *Cross Bow* and other Atari shoot-'em-ups. *Sharp Shooter* is best when played with several friends for some neighborly competition.

Typing it in

To create your copy of *Sharp Shooter*, you may either type in Listing 1 using *M/L Editor* (found elsewhere in this issue) or type in the Action! listings (Listings 1 and 2) and compile them yourself. The compiled game, as created by Listing 1 or as supplied on this month's disk, may be loaded from DOS as a binary file, without BASIC or any external cartridge installed. If you wish to compile the Action! listings, you will need the Action! cartridge.

The game

When the program is run, the title screen will display a reminder to attach the light gun. Press the gun's trigger to start the game, or the escape key to return to DOS.

The game screen is presented in the form of six pistol targets, three across the top and three across the bottom. Each is made up five concentric circles, the center, of course, being the bull's-eye.

Sharp Shooter selects a target at random and draws a rectangle around it. Take aim and squeeze off a shot. If the bull's-eye is hit, a pleasant "ding" sound will be heard. A complete miss of the target results in a dull thud. A hit anywhere else on the target is acknowledged by a brief "splat" sound, and the game continues. Each target is selected ten times throughout the game, for a total of 60 shots per game. Since each target can be randomly selected at any time, you cannot anticipate where to aim next. This hones reflexes and hand/eye coordination used to aim the gun. Press the escape key during game play to quit early.

At the end of a game, firing statistics are tallied and displayed. At the top of the screen, the average bullets-fired-per-minute statistic is presented. A good shooter will average about 40 to 45. Total successful target hits are displayed, out of the 60 shots fired. Below this is the missed-shot count—the sum of all shots that hit beyond the outermost ring of the selected target. A count of bull's-eyes is displayed, and score and accuracy round out the statistics. During game play each bullet's distance from dead center is calculated. Of course, the closer to the center, the higher the score. Accuracy is a percentage based on total bull's-eyes accrued versus total shots fired (60). A running high score and best accuracy are also presented.

Programming notes

Action! programmers may wish to take a look at GUNREAD.ACT. This procedure, GUNREAD, returns the light gun's coordinates. The caller must pass a pointer to the x (card) and y (byte) variables to receive the readings. This routine maps the gun position to coordinates for the present graphics mode. The comments for this routine explain the algorithm fully. (This is an Action! version of the assembly language routine employed in *Gun Assist* from last month's issue of ANALOG.)

You may wish to take a look at the function ISqrt in the main program as well. This routine will return the square root of an integer as a byte value. The algorithm has been around for a long time; I found it in a 6502 assembly language programming manual written by Leo J. Scanlon.

It works like this: Count the number of times successive odd numbers (1, 3, 5, 7, etc.) can be subtracted from the number of interest until it goes to zero or negative. This count is the integer square root.

The integer square root comes in handy in many applications. It is used in *Sharp Shooter* to solve for the radius, r, of each shot from the center of the bull's-eye. The formula is r = ISqrt(x*x + y*y), where x and y are the differences between the bullet's impact point and the center of the current target.

The code in the GAMESCREEN procedure generates all the pistol targets, from a table of circle centers, XCS and YCS. The circles are drawn entirely with integer math computations. A circle is eight-way symmetrical, so only an eighth of the points need to be calculated. If you ever draw a circle using floating point calculations and sine or cosine functions, then you are wasting a lot of computing time!

Conclusion

Sharp Shooter is certainly not the most sophisticated Action! game to appear in the pages of ANALOG, but it does present the basics of implementing the Atari light gun in a game. The GUNREAD procedure may prove useful in your Action! programming efforts. Even if you don't program in Action!, it should be fairly simple for you to translate it to Atari BASIC.

Sharp Shooter is my first Action! program. Now that I have finally taken the plunge into this high-level structured language for the Atari, I am seriously addicted. Its similarities to the C programming language—what I spend most of my workday using—are very strong, and Action!'s editor is more sophisticated than some word processors. If you hope to move up to C, Pascal, Ada or some other "high-level structured language" in the future, I think that you will find the Action! language a superb stepping stone.

Matthew Ratcliff, a frequent contributor to ANALOG Computing, *lives in St. Louis, Missouri, with his wife and two children.*

continued on page 30

Illustration by Steve Sterling

Recursion

by Gregg Hesling

If you've spent any time away from BASIC (gasp!), you've probably run into something called "recursion." Simply put, recursion is a subroutine's ability to call itself. BASIC, as many people will tell you, doesn't support recursion. But, as I'm going to explain beyond all levels of reason, not only does BASIC support recursion, but it's very easy to implement!

Solving a problem usually requires breaking it down into smaller problems. Sometimes, however, you break down a problem and find you've got the original problem, only in a smaller form. This type of solution is called "recursive." There are three main requirements for recursion: 1) the subroutine must call itself; 2) on each successive call, the problem must be the same but smaller in size; and 3) a "degenerative case" is always reached and handled directly, without calling the subroutine. The degenerative case is usually when the problem is so small it can be accomplished in one step.

An effective example of this is an exponential program that multiplies a number by itself a certain number of times. If we wanted to know what 2 to the power of 3 (2 ^3) was, we'd break it down into this:

A) Set result to equal 1.
B) Multiply the result by 2.
C) Multiply the result by 2.
D) Multiply the result by 2.
E) Print result.

If we set up a subroutine to multiply the result by the root, we could call it from the main program three times. Or we could have the subroutine call itself three times and then return to the main program to print the result. The latter would be a recursive solution.

To be flexible, we'd need the subroutine to multiply the result by the root a number of times equal to the exponent. If we decremented the exponent every time we called the subroutine, we'd be making the problem smaller and smaller until exponent was 0, and we'd stop multiplying. That is the degenerative case. Then we go back to the main program and print the results.

Oops, one small problem. When we print the exponent, it will always equal 0! We could print the exponent first, but that's cheating. We could set up another variable to hold the original value of the exponent, but that has nothing to do with the article. Here is a recursive program with an interesting solution:

```
10 ? "Root ";:INPUT ROOT
20 ? "Exponent ";:INPUT EXPON
ENT
30 NUMBER=1:GOSUB 60
40 ? ROOT;"^";EXPONENT;"=";NU
MBER
50 END
60 EXPONENT=EXPONENT-1:IF EXP
ONENT>0 THEN GOSUB 60
70 NUMBER=NUMBER*ROOT:EXPONEN
T=EXPONENT+1
80 RETURN
```

If EXPONENT is 3, line 60 calls itself three times. When EXPONENT is 0, RESULT is multiplied by the ROOT and EXPONENT is incremented. When RETURN is reached, the program pops back up to the end of line 60 and immediately falls through to line 70, repeating the process until EXPONENT is back to its original value and RESULT is the answer. Then, when it hits RETURN again, it returns control to line 30.

"Now, wait," you're probably saying. "Aren't there other, easier ways to do this?" Well, sure. How about changing line 60 to this:

```
60 FOR LOOP=1 TO EXPONENT:NUM
BER=NUMBER*ROOT:NEXT LOOP:RET
URN
```

Now, wasn't that easy? This solution is, in most respects, better than the recursive solution! So why recurse? To answer that, let's look at a program that requires recursion. Then we'll see how to make recursion work in BASIC.

The Towers of Hanoi

The first program, HANOI.BAS, is based on a supposedly ancient idea, but I can't imagine anyone thinking of it without having a computer to solve it. The problem is this: If you have three poles, with a number of disks stacked (in descending sizes, like a pyramid) on the first pole, how do you move them to the second pole without placing a larger disk on top of a smaller one? You may, of course, place them on the third pole temporarily.

The solution, ironically, is very simple; it's the *execution* that's difficult. To move the bottom disk from the first pole to the second without putting it on top of one of the other disks (all of which are smaller), you're going to have to move the rest of the disks to the third pole. To get all of those disks to the third pole, you're going to have to move all the disks but the bottom two to the second pole. (It took me weeks to reason this out, so please make sure you understand it before continuing.)

Let's rewrite that paragraph substituting pole #1 with "Source," pole #2 with "Dest," pole #3 with "Spare," and number of disks with N.

To move N disks from the Source pole to the Dest pole, you're going to have to move N-1 disks to the Spare pole. Since the spare is now the destination, let's swap the poles' labels and decrement N. To move N disks (really N-1 disks) to the new Dest pole, you're going to have to move N-1 (N-2) disks to the new Spare pole. Since the spare is now the destination. . . .

If we continue this until N equals 1 (the "degenerative case"), we'll just move the top disk from the Source pole to the Dest pole, whichever that happens to be.

That was the easy part. Next we have to back up and "de-switch" the Dest and Spare poles in order to move the next disk (N+1) to the other pole. If you're completely baffled, it's because recursion usually does this. But don't get diskouraged! I suggest you revert to those old standbys, the pencil and paper. And be sure the pencil has a big eraser on it!

Once we have N-1 disks on the spare pole and move the Nth disk to the Dest pole, we still aren't finished. Now we have to move N-1 disks from the spare pole to the dest pole. The simplest solution would be to, of course, exchange the Spare pole with the Source pole. Now we have to move N-1 disks from the Spare pole to the Dest pole.

It would take no less than 1,023 moves to solve for ten disks by hand. But by substituting labels and adding recursion, we should be able to do it in under twenty lines of BASIC. *That* is why one should recurse!

How recursion works

In standard Pascal, the algorithm looks like Figure 1. You'll note that at the end of the subroutine there are three successive calls to the subroutine. In Pascal (and most languages), whenever you call a subroutine, a whole new set of variables is created and the values are passed. When the subroutine returns, the original values are restored. Because of this, variables with the same name can exist. Also because of this, BASIC "shouldn't" support recursion. If you used the same variable name in BASIC, you'd lose the original value without any chance of recovery. Since recursion needs that recovery (you're going to hate me for saying this), we have to save the values ourselves.

Fortunately, it's easy. To save the variables, all you need to do is save them in an array. This way you only need a variable, such as LEVEL(X), to tell you which set of variables to restore. Just replace X with a reference number and you're all set.

Alas, there is a price, and it can be a dear one. In Pascal, when you return from a subroutine and the new set of variables is tossed, you get all the RAM back. In BASIC, however, you have to declare your arrays (and their limits) before you use them. And when I say limits, I mean the maximum possible amount needed. And none of this is returned to you until the program is finished. This is a serious drawback. BASIC, though, isn't often used for professional purposes, and even with the reduced RAM, it should handle all of your needs.

In the next two programs, I managed to cut

```
PROCEDURE Towers (Count, Source, Dest, Spare)

BEGIN
 IF Count=1 (* Degenerative case *)
  THEN
   (* Move the disk from Source to Dest *)
  ELSE
   BEGIN
    Towers (Count-1, Source, Spare, Dest);
     (* Decrement Count, swap Dest and Spare *)
    Towers (1, Source, Dest, Spare);
 (* Count is set to 1, so an immediate move is made *)
  Towers (Count-1, Spare, Dest, Source);
     (* Decrement Count, swap Source and Spare *)
   END;
END;
```

FIGURE 1

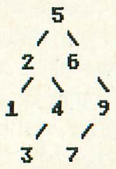
down on space by using strings instead of arrays. But this also increases confusion, and I'm not sure where the trade-off should begin.

Heap Sort

Heap Sort is a sorting algorithm that sorts entries as you enter them. Since BASIC also doesn't support dynamic allocation (I'm not even going to get into that here), it isn't really in order. It stores the entries in a pseudo-binary tree, with left and right pointers determining the order. To sort it out upon request, we use—you guessed it—recursion.

We'll have to use pointers from a parent to its children (hi-tech computer jargon) because setting it up as a perfect binary tree in BASIC would mean reorganizing it every time something was added, and that would take far too much time and waste far too much space. To store, we just compare our number with the top number and go left or right appropriately. We keep comparing until we find a hole to store that number. If this sounds a little bit like recursion, you're learning quickly. It *is* recursion, but it's "easy" recursion because it doesn't store previous variables. We'll concentrate on the "fetching" part of the program, which does.

The array is stored something like this:



The lowest number is all the way down the left side. The next-smallest number is its parent, and then the larger numbers are off to the right. We could go down the left side easily to retrieve the number, but then how would we get back up? We'll need a pointer to point back to the previous number, or we'll quickly get lost. We can use an array or a string, depending on how much RAM you can waste. This way, with just a little help from recursion, we can figure out just where we're heading (and where we just came from.)

Recursion works perfectly here because the problem is thus: We need to go all the way down the left side until we find the bottom number. We can rewrite this as "Go left until there is no more left to go." This is a classic recursive action. The degenerative case is the bottom of the array, while the program is broken down into just one action: Go left (young man, go left). After we reach the bottom, we'll need to back up one. Thus, we'll need to store the previous number's position in an array variable. Then we'll go to the right and check for a left child. We don't store the position of a parent with a right child because we've already printed that number and don't need to return there. When we're completely through with the right leg, we just jump back to the point before the left leg, as everything beneath that has been "extracted."

Gee, that was short, but that's pretty much all there is to it. Whether you dissect the program to figure out how it ticks, or quietly shelve it in your library (or wonder why you bought a disk subscription), it's an. . .um. . . interesting program.

QuikSort

QuikSort is, in every instance I've seen, faster than any other sorting algorithm. Why this is, I'm not entirely sure, but for quite some time I've been denied the ability to use QuikSort in BASIC because it uses recursion. Well, it's finally here.

QuikSort works on the premise of breaking an array down to smaller arrays and sorting them. For instance, to sort "CEBDA" alphabetically, QuikSort would first sort "C" by moving it over and putting the "lesser" letters behind it, thus ending up with "BACED." The next step is to sort "BA," and then "ED." The final result, "ABCDE," is correct, took only three passes, was quite quick and was terribly confusing. Try it on a few other words and you'll discover how good QuikSort is.

The problem comes when you need to sort "BA" and "ED." Unless your computer supports multi-tasking, the computer needs to do one at a time and you need to remember the position of "ED." As in Heap Sort, all you need to do is store the location of the right array before sorting the left. If the left array needs to be broken down more, then this needs to be stored also, etc. So we label the strings (or arrays) FIRST$ and LAST$, and store that information. It eats up RAM, but the return in speed is definitely worth it.

A few final words: As I tried to explain in the opening paragraphs, recursion is not a panacea. In actuality, there aren't many programs that require recursion, or can even use it. And those that do should be thought over carefully to see whether or not they could be done in a simpler fashion. But some problems just can't be beat when using recursion, and I hope you understand when to use this powerful tool (and when not to!) now.

Gregg Hesling lives in sunny Southern California, but might as well live in Alaska for all the time he spends outdoors. He was disappointed on his nineteenth birthday when he realized that he knew everything there was to know about Atari BASIC, but had never done anything with the information. He feels better now.

LISTING 1: BASIC

```
RP 1 REM ***** THE TOWERS  OF HANOI *****
AZ 2 REM ******* by GREGG HESLING *******
LX 5 GRAPHICS 0:POKE 710,0:POKE 709,14:PO
   KE 82,0:? "COUNT ";:INPUT COUNT:IF COU
   NT<1 OR COUNT>12 THEN 5
AT 9 REM RAM is not a problem in this pro
   gram, so we indiscriminantly waste it
GY 10 DIM POLE(2,12),COUNT(COUNT),SOURCE(
   COUNT),DEST(COUNT),SPARE(COUNT):POKE 7
   52,1:? "K"
UV 14 REM Assign the poles temporary labe
   ls and tell how many discs are on each
    pole
YW 15 SOURCE=0:DEST=1:SPARE=2:POLE(SOURCE
   ,0)=COUNT:POLE(DEST,0)=0:POLE(SPARE,0)
   =0
BO 19 REM Make the poles and put the disc
   s on pole #1
JD 20 FOR A=10 TO 21:POSITION 6,A:? "[]",
   "   []","      []":NEXT A:? "[]
   []      []"
SL 25 FOR A=1 TO COUNT:POLE(SOURCE,A)=COU
   NT-A+1:B=A-INT(A/2):C=(A/2=INT(A/2)):P
   OSITION 6-B,A-COUNT+21:GOSUB 80:NEXT A
TF 30 GOSUB 90:IF COUNT=1 THEN GOSUB 50:G
   OTO 40:REM Move the top disc and don't
    save the labels
ZD 33 POSITION 9,0:? "SAVING":COUNT(LEVEL
   )=COUNT:SOURCE(LEVEL)=SOURCE:DEST(LEVE
   L)=DEST:SPARE(LEVEL)=SPARE
GP 35 REM Save the current labels, then s
   witch the DEST and SPARE poles
ZZ 37 A=DEST:DEST=SPARE:SPARE=A:COUNT=COU
   NT-1:LEVEL=LEVEL+1:GOTO 30
UF 40 LEVEL=LEVEL-1:IF LEVEL<0 THEN POSIT
   ION 9,0:? "COMPLETE":END
JQ 43 POSITION 9,0:? "RESTORING":COUNT=CO
   UNT(LEVEL):SOURCE=SOURCE(LEVEL):DEST=D
```

```
   EST(LEVEL):SPARE=SPARE(LEVEL):GOSUB 90
RR 45 REM Restore the previous labels, mo
   ve the top disc to the DEST pole, and
   swap SOURCE with SPARE
VQ 47 GOSUB 50:POSITION 9,0:? "SWITCHING"
   :COUNT=COUNT-1:A=SOURCE:SOURCE=SPARE:S
   PARE=A:GOTO 30
AM 49 REM Move disc to top of screen
UW 50 A=POLE(SOURCE,0):POLE(SOURCE,0)=A-1
   :B=POLE(SOURCE,A):C=(B/2=INT(B/2)):B=B
   -INT(B/2)
QP 55 FOR D=A TO 20:POSITION SOURCE*13+6-
   B,21-D:GOSUB 80:POSITION SOURCE*13+1,2
   2-D:GOSUB 85:NEXT D
IJ 59 REM Move disc from SOURCE to DEST
UG 60 POSITION 0,1:FOR A=1 TO ABS(DEST-SO
   URCE)*13:? CHR$(254+((DEST-SOURCE)>0))
   ;:NEXT A
KK 70 A=POLE(DEST,0)+1:POLE(DEST,0)=A:POL
   E(DEST,A)=B*2+C-1:FOR D=21 TO A+1 STEP
    -1:POSITION DEST*13+6-B,23-D:GOSUB 80
BH 74 REM Lower disc down to DEST
XF 75 POSITION DEST*13+1,22-D:GOSUB 85:NE
   XT D:RETURN
MY 79 REM Print disc
YG 80 ? CHR$(32+121*C);:FOR E=1 TO B*2:?
   "";:NEXT E:? CHR$(32-7*C);:RETURN
BI 84 REM Erase disc
UZ 85 ? "     ";CHR$(32+121*(D<13));CHR$(
   32-7*(D<13));"     ":RETURN
YU 90 POSITION 0,0:? "COUNT=";COUNT;" ":P
   OSITION 19,0:? "LEVEL=":IF LEVEL THEN
   POSITION 24+LEVEL,0:? LEVEL-1;"  ";
SG 93 POSITION 13*SOURCE+4,23:? "SOURCE";
   :POSITION 13*DEST+4,23:? " DEST  ";:PO
   SITION 13*SPARE+4,23:? "SPARE ";
TI 97 FOR LOOP=1 TO 100:NEXT LOOP:POSITIO
   N 9,0:? "          ":RETURN :REM To spe
   ed things up, change 100 to 1
```

LISTING 2: BASIC

```
GU 3 REM ********** HEAPSORT **********
WC 5 REM ******  by GREGG HESLING *******
CA 10 TRAP 10:CLR :? "Max. size/entry";:I
   NPUT SIZE:RAM=INT((FRE(0)-SIZE-500)/(S
   IZE+6)):DIM ARRAY$(RAM*SIZE),A$(SIZE)
EU 11 REM Recursion devours RAM, up to 28
   % in this program depending on the "ma
   ximum size per entry"
LE 15 DIM LEFT$(RAM*2),RIGHT$(RAM*2),LEVE
   L$(RAM*2):LEFT$="♥":LEFT$(RAM*2)="♥":L
   EFT$(2)=LEFT$:RIGHT$=LEFT$:GOTO 50
WM 16 REM LEFT$ and RIGHT$ will store the
    pointers, while LEVEL$ will be the "r
   ecursive variable saver"
TL 20 Y=ASC(LEFT$(X*2-1))*256+ASC(LEFT$(X
   *2)):IF Y=0 THEN 30:REM There is no le
   ft child
PK 24 REM The address of the current node
    is stored, then we make the left chil
   d the current node and go again
KY 25 Z=INT(X/256):LEVEL$(LEVEL+1,LEVEL+1
   )=CHR$(Z):LEVEL$(LEVEL+2,LEVEL+2)=CHR$
   (X-Z*256):LEVEL=LEVEL+2:X=Y:GOTO 20
QW 30 ? ARRAY$((X-1)*SIZE+1,X*SIZE):REM A
   RRAY$ is printed in sorted order -- le
   ft child, parent, then right child
YQ 35 Y=ASC(RIGHT$(X*2-1))*256+ASC(RIGHT$
   (X*2)):IF Y THEN X=Y:GOTO 20:REM Follo
   w right leg without saving positions
GZ 39 REM Restore the last saved position
    -- the parent with a left child -- an
   d go directly to PRINT
MC 40 IF LEVEL THEN LEVEL=LEVEL-2:X=ASC(L
   EVEL$(LEVEL+1))*256+ASC(LEVEL$(LEVEL+2
   )):GOTO 30
CT 50 ? :? COUNT;" records used",RAM-COUN
   T;" records left"
WH 55 ? "Entry: ";:INPUT A$:X=1:IF A$=""
   THEN SORT=1:LEVEL=0:GOTO 20:REM Go to
   recursive printing routine
BJ 59 REM Store new string, then use a bi
   nary tree search to determine the new
   string's position and set pointers
QC 60 A=SIZE*COUNT:FOR B=1 TO SIZE:ARRAY$
   (A+B)=" ":NEXT B:ARRAY$(A+1,A+LEN(A$))
   =A$:COUNT=COUNT+1:IF COUNT=1 THEN 55
SF 65 IF ARRAY$((COUNT-1)*SIZE+1,COUNT*SI
   ZE)>ARRAY$((X-1)*SIZE+1,X*SIZE) THEN 7
   5:REM Follow left or right branch?
GT 69 REM Left branch.  If there is a chi
   ld, go to it.  Else, save new entry at
    this point
JN 70 Y=ASC(LEFT$(X*2-1))*256+ASC(LEFT$(X
   *2)):GOSUB 80:LEFT$(A,A)=CHR$(C):LEFT$
   (A+1,A+1)=CHR$(B-C*256):GOTO 55
WT 74 REM Right branch.  If there is a ch
   ild, go to it.  Else, save new entry a
   t this point
JS 75 Y=ASC(RIGHT$(X*2-1))*256+ASC(RIGHT$
   (X*2)):GOSUB 80:RIGHT$(A,A)=CHR$(C):RI
   GHT$(A+1,A+1)=CHR$(B-C*256):GOTO 55
SG 80 IF Y THEN X=Y:POP :GOTO 65
ZU 85 A=X*2-1:B=COUNT:C=INT(B/256):RETURN
LF 89 REM Change line 55 to line 56 and t
   ype "55 IF COUNT THEN GOSUB 95" to see
    previous entries and their pointers
RM 90 FOR X=1 TO COUNT:? ASC(LEFT$(X*2-1)
   )*256+ASC(LEFT$(X*2)),ARRAY$((X-1)*SIZ
   E+1,X*SIZE),
UY 95 ? ASC(RIGHT$(X*2-1))*256+ASC(RIGHT$
   (X*2)):NEXT X:RETURN
```

LISTING 3: BASIC

```
BX 3 REM ********** QUIKSORT **********
WC 5 REM ******  by GREGG HESLING *******
AC 10 CLR :? "Max. size/entry";:INPUT SIZ
   E:RAM=INT((FRE(0)-SIZE-500)/(SIZE+4/3)
   ):DIM ARRAY$(RAM*SIZE),A$(SIZE)
DJ 11 REM 57% of RAM is lost when SIZE eq
   uals 1, but only 1% is lost when SIZE
   is 65 or more
GY 15 DIM FIRST$(RAM/3*2),LAST$(RAM/3*2):
   LAST$="♥":LAST$(RAM/3*2)="♥":LAST$(2)=
   LAST$:FIRST$=LAST$:GOTO 85
PK 16 REM FIRST$ and LAST$ will hold poin
   ters to the beginning and end of array
   s that need to be sorted
WB 18 REM Lines 20-40 take the first entr
   y in the array and move it over until
   everything less than the pivot
QW 19 REM is to the left, while everythin
   g greater is to the right.
AV 20 PIVOT=(FIRST-1)*SIZE+1:A$=ARRAY$(PI
   VOT,PIVOT+SIZE-1)
YR 30 FOR A=FIRST*SIZE+1 TO (LAST-1)*SIZE
   +1 STEP SIZE:IF ARRAY$(A,A+SIZE-1)>A$
   THEN 40
BY 35 ARRAY$(PIVOT,PIVOT+SIZE-1)=ARRAY$(A
   ,A+SIZE-1):PIVOT=PIVOT+SIZE:ARRAY$(A,A
   +SIZE-1)=ARRAY$(PIVOT,PIVOT+SIZE-1)
QB 40 NEXT A:ARRAY$(PIVOT,PIVOT+SIZE-1)=A
   $:PIVOT=(PIVOT-1)/SIZE+1:IF PIVOT+1>=L
   AST THEN 60
MG 49 REM If there are entries to the rig
   ht of PIVOT, the first and last positi
   ons are saved
UT 50 A=PIVOT+1:B=INT(A/256):FIRST$(LEVEL
   +1)=CHR$(B):FIRST$(LEVEL+2)=CHR$(A-B*2
   56):A=INT(LAST/256)
EY 55 LAST$(LEVEL+1)=CHR$(A):LAST$(LEVEL+
   2)=CHR$(LAST-A*256):LEVEL=LEVEL+2:? CH
```

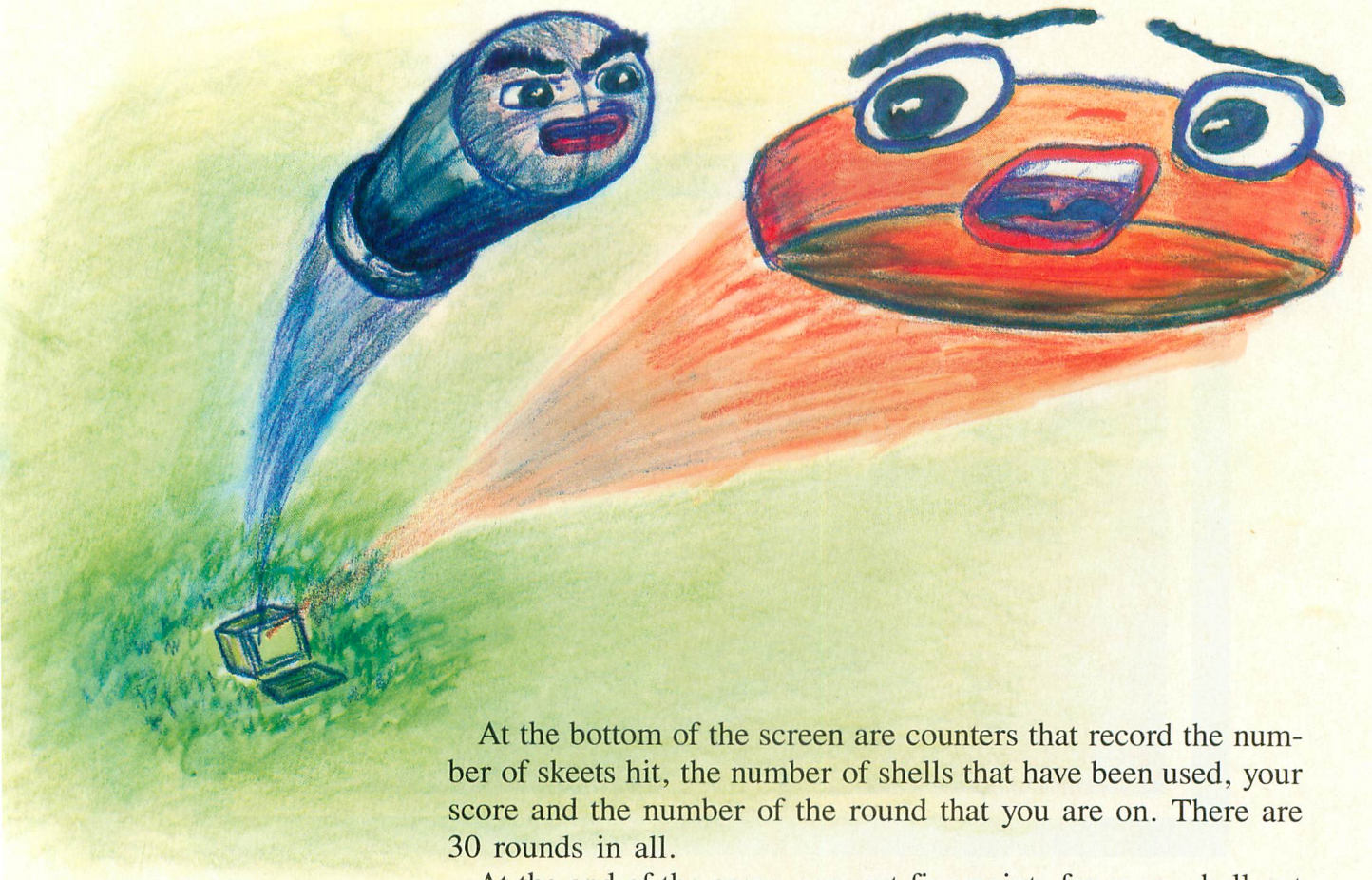
by Tracy Jacobs

Skeet Shoot

Skeet Shoot is a one-player action game written in 100% machine language that will run on all 8-bit Atari computers. Type in Listing 1 using M/L Editor, then load *Skeet Shoot* using Atari's DOS binary load.

Once the game is booted up, the title screen will appear. Press START to begin. The gunsight will appear in the middle of the screen. Be aware that gravity pulls the gunsight down. Press the joystick up to release the clay skeets.

The object of the game is to get the highest possible score. The skeets are slung out at random speeds—slow, medium and fast. The score for hitting the slow skeets is ten points, 25 for the medium and 50 for the fast.



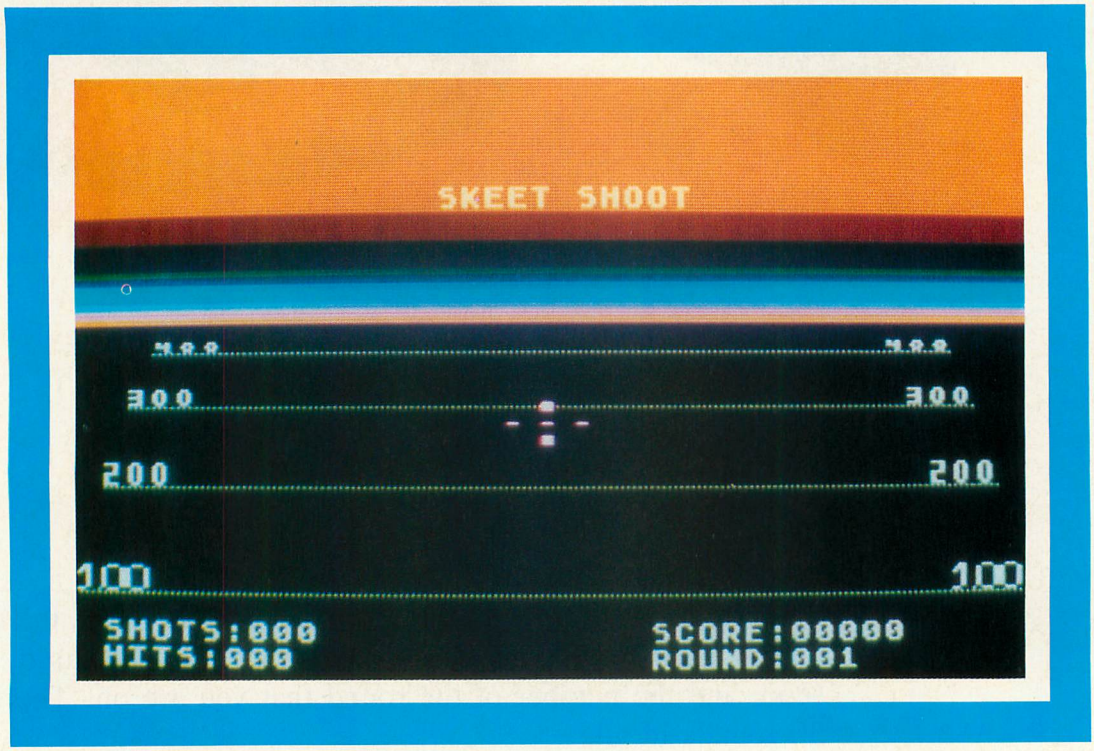
At the bottom of the screen are counters that record the number of skeets hit, the number of shells that have been used, your score and the number of the round that you are on. There are 30 rounds in all.

At the end of the game, you get five points for every shell not used. There is a total of 60 shells, but you are allowed only two shells per round.

Press START to play again.

Tracy Jacobs, a high school student, has been programming his 800 XL for about five years, and programming in assembly language with his older brother, Michael, an electronics technician, for a little over a year.

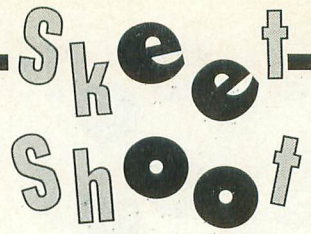
Skeet Shoot



LISTING 1: M/L EDITOR DATA

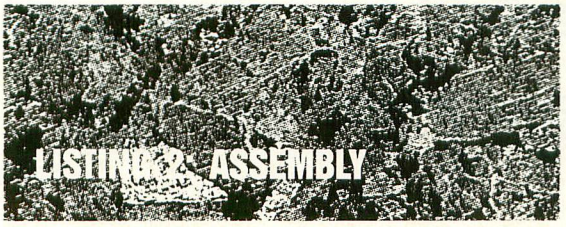
```
1000 DATA 255,255,30,72,180,80,169,69,
133,12,169,72,133,13,169,0,3744
1010 DATA 162,0,157,0,64,157,0,65,157,
0,66,157,0,67,157,0,741
1020 DATA 68,157,0,69,157,0,70,157,0,7
1,232,208,229,160,0,185,8144
1030 DATA 20,80,170,200,185,20,80,141,
0,96,72,24,173,80,72,105,3422
1040 DATA 1,141,80,72,173,81,72,105,0,
141,81,72,104,202,208,231,8707
1050 DATA 200,192,124,208,218,169,144,
141,48,2,169,80,141,49,2,169,5602
1060 DATA 192,141,14,212,169,150,141,0
,2,169,79,141,1,2,169,0,2001
1070 DATA 141,22,72,169,242,141,26,208
,169,64,141,7,212,169,88,141,8543
1080 DATA 192,2,141,194,2,169,134,141,
195,2,169,10,141,193,2,169,6588
1090 DATA 120,141,0,208,141,1,72,141,1
,208,169,0,141,2,208,141,5852
1100 DATA 3,72,141,3,208,141,4,72,141,
8,72,141,7,72,141,24,1603
1110 DATA 72,141,25,72,141,28,72,141,2
6,72,169,16,141,46,100,141,3570
1120 DATA 47,100,141,48,100,141,239,99
,141,240,99,141,241,99,141,22,9229
1130 DATA 100,141,23,100,141,24,100,16
9,62,141,47,2,169,3,141,29,2209
1140 DATA 208,169,120,141,0,72,169,208
,141,12,72,141,13,72,169,1,3490
1150 DATA 141,8,208,141,9,208,32,13,78
,169,1,141,5,72,141,6,1495
1160 DATA 72,162,0,32,152,76,32,85,76,
```

```
32,77,78,162,0,189,52,2364
1170 DATA 73,157,135,96,232,224,11,240
,14,76,38,73,51,43,37,37,1439
1180 DATA 52,0,51,40,47,47,52,189,0,22
4,157,0,64,189,255,224,8792
1190 DATA 157,255,64,232,208,241,169,6
4,141,244,2,162,80,189,221,78,1082
1200 DATA 157,8,64,202,16,247,173,31,2
08,201,6,208,249,141,30,208,827
1210 DATA 162,0,189,120,73,157,135,96,
232,224,11,240,45,76,106,73,7175
1220 DATA 0,0,0,0,0,0,0,0,0,0,0,173,11
,212,201,123,1390
1230 DATA 208,249,173,42,2,208,3,32,10
1,77,173,11,212,201,123,240,9127
1240 DATA 249,173,17,72,201,2,48,64,24
0,0,32,215,77,32,242,77,5442
1250 DATA 169,114,141,0,208,141,1,72,1
69,130,141,0,72,32,13,78,1738
1260 DATA 169,0,141,1,210,173,120,2,20
1,14,208,249,169,0,141,17,6609
1270 DATA 72,173,10,210,141,16,72,169,
0,141,42,2,169,18,141,29,2139
1280 DATA 72,32,101,77,169,0,133,77,76
,134,77,173,11,72,201,1,3548
1290 DATA 240,25,16,20,174,28,72,224,2
,16,19,173,16,208,201,0,3640
1300 DATA 240,3,76,6,74,76,211,74,76,2
44,74,76,28,75,172,120,5457
1310 DATA 2,174,1,72,185,45,79,201,0,2
40,19,16,10,56,224,40,3022
1320 DATA 144,2,202,202,76,38,74,56,22
4,204,176,2,232,232,142,1,8882
1330 DATA 72,142,0,208,76,178,74,172,1
20,2,185,61,79,240,21,16,4685
1340 DATA 2,48,20,56,173,0,72,201,212,
176,9,32,140,74,32,140,4426
1350 DATA 74,32,140,74,76,131,73,56,17
3,0,72,201,16,144,245,32,5501
1360 DATA 102,74,32,102,74,32,102,74,3
2,102,74,76,131,73,24,173,2869
1370 DATA 10,72,105,14,141,9,72,174,0,
72,172,10,72,202,185,77,5053
1380 DATA 79,157,0,68,232,200,204,9,72
,208,243,169,0,157,0,68,6620
1390 DATA 206,0,72,96,24,173,10,72,105
,14,141,9,72,174,0,72,1268
1400 DATA 172,10,72,169,0,157,0,68,232
,185,77,79,157,0,68,200,5964
1410 DATA 232,204,9,72,208,243,238,0,7
2,96,24,173,10,72,105,14,3414
1420 DATA 141,9,72,174,0,72,172,10,72,
185,77,79,157,0,68,232,5273
1430 DATA 200,204,9,72,208,243,240,0,7
6,47,74,169,0,141,42,2,3163
1440 DATA 169,0,141,29,72,173,42,2,208
,3,32,101,77,174,28,72,2331
1450 DATA 232,142,28,72,169,15,141,11,
72,238,8,72,173,1,72,141,3927
1460 DATA 1,208,24,173,10,72,105,9,141
,9,72,174,0,72,172,10,1917
1470 DATA 72,185,137,79,157,0,69,232,2
00,204,9,72,208,243,206,11,9938
1480 DATA 72,76,83,75,173,10,72,105,9,
141,9,72,174,0,72,172,3070
1490 DATA 10,72,169,0,141,11,72,169,0,
157,0,69,232,200,204,9,6196
1500 DATA 72,208,244,174,8,72,32,32,78
,32,152,78,76,131,73,141,4171
1510 DATA 29,72,169,0,141,42,2,32,101,
77,96,173,13,208,41,4,1988
1520 DATA 208,13,173,13,208,141,30,208
,41,8,208,49,76,131,73,173,6095
1530 DATA 5,72,201,15,16,236,238,7,72,
173,5,72,105,22,141,5,2725
1540 DATA 72,169,42,32,71,75,174,7,72,
32,32,78,32,133,78,169,2691
1550 DATA 128,32,190,78,173,27,72,32,1
84,75,76,90,75,173,6,72,3372
1560 DATA 201,15,16,25,238,7,72,173,6,
72,105,22,141,6,72,169,2953
1570 DATA 42,32,71,75,32,190,78,173,27
,72,32,184,75,76,131,73,4114
1580 DATA 24,109,25,72,141,25,72,169,0
,109,24,72,141,24,72,32,875
1590 DATA 77,78,96,174,23,72,232,142,2
3,72,224,49,208,13,173,6,5670
1600 DATA 72,105,3,141,6,72,169,0,141,
23,72,24,173,6,72,105,1772
1610 DATA 4,141,9,72,174,13,72,172,6,7
```



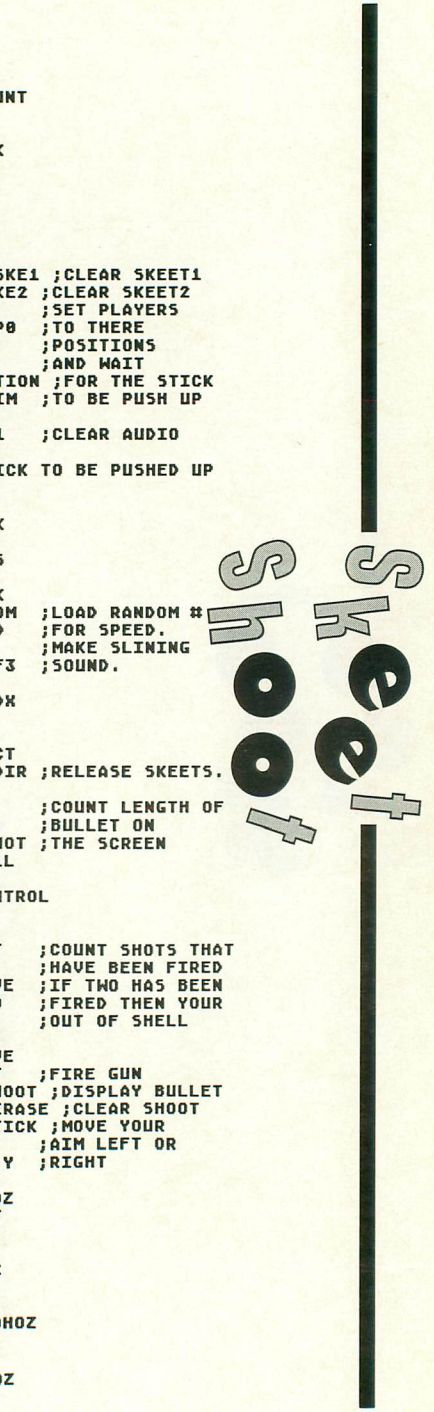

```
2,202,185,90,79,157,0,4886
1620 DATA 71,232,200,204,9,72,208,243,
169,0,157,0,71,206,13,72,5850
1630 DATA 173,13,72,201,24,240,70,96,1
74,23,72,232,142,23,72,224,7871
1640 DATA 49,208,13,173,5,72,105,3,141
,5,72,169,0,141,23,72,1662
1650 DATA 24,173,5,72,105,4,141,9,72,1
74,12,72,172,5,72,202,3933
1660 DATA 185,90,79,157,0,70,232,200,2
04,9,72,208,243,169,0,157,9785
1670 DATA 0,70,206,12,72,173,12,72,201
,24,240,68,96,32,242,77,6597
1680 DATA 238,26,72,238,17,72,173,10,2
10,141,15,72,32,5,77,105,2596
1690 DATA 8,141,3,208,141,4,72,169,208
,141,13,72,169,1,141,6,4117
1700 DATA 72,169,0,141,28,72,141,0,210
,141,1,210,141,28,2,174,5103
1710 DATA 26,72,224,31,240,45,32,32,78
,32,171,78,141,30,208,96,5374
1720 DATA 32,215,77,238,17,72,173,10,2
10,141,14,72,32,255,76,141,6873
1730 DATA 2,208,141,3,72,169,208,141,1
2,72,169,1,141,5,72,141,4479
1740 DATA 30,208,96,169,5,32,184,75,17
4,8,72,232,142,8,72,224,7099
1750 DATA 60,240,3,76,187,76,162,0,189
,246,76,157,135,96,232,224,2172
1760 DATA 9,240,3,76,208,76,173,31,208
,201,6,240,11,169,0,141,7110
1770 DATA 2,208,141,3,208,76,222,76,32
,13,78,76,144,72,39,33,2462
1780 DATA 45,37,0,47,54,37,50,173,14,7
2,76,8,77,173,15,72,891
1790 DATA 201,85,48,17,201,170,48,8,16
,1,96,169,50,76,18,77,1252
1800 DATA 169,118,76,18,77,169,192,76,
18,77,32,16,76,173,16,72,2134
1810 DATA 56,201,50,144,102,32,16,76,1
73,16,72,56,201,160,144,91,6066
1820 DATA 32,16,76,76,147,77,32,203,75
,56,173,0,72,201,212,176,8345
1830 DATA 3,32,140,74,173,16,72,56,201
,50,144,14,32,203,75,173,5738
1840 DATA 16,72,56,201,160,144,3,32,20
3,75,76,227,73,174,29,72,6022
1850 DATA 189,219,79,141,0,210,232,189
,219,79,141,1,210,232,189,219,4315
1860 DATA 79,141,28,2,232,142,29,72,16
9,255,141,42,2,96,173,14,5419
1870 DATA 72,201,85,48,22,201,170,48,1
45,16,29,173,15,72,201,85,5119
1880 DATA 48,35,201,170,48,160,16,42,7
6,227,73,174,3,72,202,142,7123
1890 DATA 3,72,142,2,208,76,34,77,174,
3,72,232,142,3,72,142,5233
1900 DATA 2,208,76,34,77,174,4,72,202,
142,3,208,142,4,72,76,4680
1910 DATA 62,77,174,4,72,232,142,3,208
,142,4,72,76,62,77,173,5413
1920 DATA 5,72,105,4,141,9,72,174,12,7
2,172,5,72,169,0,157,3649
1930 DATA 0,70,232,200,204,9,72,208,24
4,96,173,6,72,105,4,141,6661
1940 DATA 9,72,174,13,72,172,6,72,169,
0,157,0,71,232,200,204,8360
1950 DATA 9,72,208,244,96,162,0,169,0,
232,157,0,68,157,0,69,4740
1960 DATA 224,255,208,245,32,102,74,96
,160,0,138,56,233,100,144,4,6639
1970 DATA 200,170,176,248,24,152,105,1
6,141,19,72,160,0,138,56,233,6596
1980 DATA 10,144,4,200,170,176,248,24,
152,105,16,141,20,72,138,105,6228
1990 DATA 16,141,21,72,96,173,25,72,13
3,212,173,24,72,133,213,32,6921
2000 DATA 170,217,32,230,216,160,0,177
,243,48,3,200,208,249,41,127,1013
2010 DATA 162,4,56,233,32,157,6,100,22
4,0,240,16,202,136,192,255,1562
2020 DATA 240,5,177,243,24,144,235,200
,169,48,208,230,96,173,19,72,158
2030 DATA 141,22,100,173,20,72,141,23,
100,173,21,72,141,24,100,96,3840
2040 DATA 173,19,72,141,239,99,173,20,
72,141,240,99,173,21,72,141,7956
2050 DATA 241,99,96,173,19,72,141,46,1
00,173,20,72,141,47,100,173,5824
2060 DATA 21,72,141,48,100,96,173,16,7
2,56,201,160,176,17,56,201,7176
2070 DATA 50,176,6,169,10,141,27,72,96
,169,25,141,27,72,96,169,4851
2080 DATA 50,141,27,72,96,56,120,216,2
4,24,24,24,225,0,0,0,98
2090 DATA 0,0,0,0,0,0,124,12,12,124,96
,96,124,0,0,0,8222
2100 DATA 124,12,124,12,124,0,0,0,0,0,
54,62,6,60,195,195,1589
2110 DATA 195,195,195,195,60,0,56,108,
108,108,108,108,56,0,0,0,880
2120 DATA 56,108,108,108,56,0,0,0,0,0,
28,54,28,85,0,0,5938
2130 DATA 0,0,0,0,0,0,0,0,0,0,1,1,1,0,
255,255,71
2140 DATA 255,0,0,0,0,0,0,0,0,0,1,255,
0,0,1,255,9561
2150 DATA 0,0,1,255,0,16,16,16,0,0,0,1
46,0,0,0,16,5517
2160 DATA 16,16,0,56,124,254,0,48,120,
252,0,32,112,248,0,32,4384
2170 DATA 112,0,0,0,0,0,0,0,0,0,0,0,0,
20,65,8,3665
2180 DATA 34,8,32,132,32,80,128,16,128
,64,128,32,0,0,0,0,8102
2190 DATA 0,0,0,0,124,124,124,124,124,
124,124,0,0,0,72,138,2422
2200 DATA 72,174,22,72,189,188,79,232,
142,22,72,141,10,212,141,26,7067
2210 DATA 208,141,24,208,169,30,205,22
,72,208,5,169,0,141,22,72,4507
2220 DATA 104,170,104,64,242,242,242,2
42,242,242,226,210,194,178,162,146,890
8
2230 DATA 130,114,98,82,66,2,2,2,4,4,4
,4,4,4,4,4,3982
2240 DATA 4,4,4,20,143,6,18,138,4,16,1
33,2,14,130,10,10,8320
```

```
2250 DATA 130,12,0,0,0,20,132,4,18,134
,4,16,136,4,14,138,9460
2260 DATA 4,40,45,1,40,42,2,40,36,2,0,
0,0,10,143,10,6058
2270 DATA 9,138,8,8,135,7,7,133,6,6,13
1,5,200,0,200,0,1656
2280 DATA 160,0,3,0,1,5,2,9,28,0,1,5,2
,9,6,0,3135
2290 DATA 34,10,5,0,1,4,2,8,30,0,1,4,2
,8,4,0,2993
2300 DATA 36,10,43,0,1,3,2,7,32,0,1,3,
2,7,2,0,3067
2310 DATA 38,10,81,0,1,1,2,6,34,0,1,1,
2,6,40,10,3883
2320 DATA 1,0,1,51,1,40,1,47,1,52,1,51
,1,26,17,0,4940
2330 DATA 1,51,1,35,1,47,1,50,1,37,1,2
6,11,0,1,40,4770
2340 DATA 1,41,1,52,1,51,1,26,18,0,1,5
0,1,47,1,53,5467
2350 DATA 1,46,1,36,1,26,11,0,240,240,
240,194,0,96,130,130,7730
2360 DATA 130,130,130,140,140,141,141,
141,141,141,141,130,130,130,130,82
3
2370 DATA 130,130,130,130,130,130,130,
130,2,65,144,80,0,226,2,227,7088
2380 DATA 2,30,72,0,0,0,0,0,0,0,0,0,0,
0,0,0,2658
```




```
10       .OPT NO LIST
20       .OPT OBJ
30 ;   *** SKEET SHOOT ***
40 ;programed by Tracy and Mike Jacobs
50 ;programed in Mac/65 by OSS inc.
60       *=  $4000
70 PLAYERS .DS $0400 ;RESERVED
80 PLAYER0 .DS $0100 ;MEMORY FOR
90 PLAYER1 .DS $0100 ;PLAYERS
0100 PLAYER2 .DS $0100
0110 PLAYER3 .DS $0100
0120 DOSVIN = $0C    ;RESET POINTER
0130 SDLSTL = $0230  ;DL POINTER (LB)
0140 SDLSTH = $0231  ;DL POINTER (HB)
0150 INTL =  $0200   ;DLI POINTER (LB)
0160 INTH =  $0201   ;DLI POINTER (HB)
0170 NMIEN = $D40E   ;INTERRUPT ENABLE
0180 WSYNC = $D40A   ;WAIT HOR. SYNC
0190 CHSET = PLAYERS ;NEW CHAR SET
0200 HPOSP0 = $D000  ;HOR. PL0
0210 HPOSP1 = $D001  ;HOR. PL1
0220 HPOSP2 = $D002  ;HOR. PL2
0230 HPOSP3 = $D003  ;HOR. PL3
0240 SIZEP0 = $D008  ;SIZE OF PL0
0250 SIZEP1 = $D009  ;SIZE OF PL1
0260 SDMCTL = $022F  ;(DMA) CONTROL
0270 GRACTL = $D01D  ;GRAPHIC CONTROL
0280 PCOLP0 = $02C0  ;COLOR OF
0290 PCOLP1 = $02C1  ;PLAYERS
0300 PCOLP2 = $02C2
0310 PCOLP3 = $02C3
0320 PMBASE = $D407  ;P/M BASE
0330 STICK = $0278   ;JOYSTICK (A)
0340 TRIG0 = $D010   ;JOYSTICK TRIGGER
0350 VCOUNT = $D40B  ;VER. LINE COUNT
0360 P1PL =  $D00D   ;PL1 TO PLAYERS
0370 HITCRL = $D01E  ;COLLISSION CLR
0380 RANDOM = $D20A  ;RANDOM #
0390 CHBAS = $02F4   ;CHARACTER BASE
0400 COLBK = $D01A   ;BACKGROUND COLOR
0410 COLPF2 = $D018  ;COLOR PLAYFIELD2
0420 FR0 =   $D4     ;FLOATING POINT #
0430 IFP =   $D9AA   ;(FP) CONVERSION
0440 FASC =  $D8E6   ;CONVERSION (SUB)
0450 INBUFF = $F3    ;POINTER TO
0460 ; BUFFER ASCII
0470 CDTMV3 = $021C  ;TIMER 3
0480 CDTMF3 = $022A  ;(3) FLAG/VECTOR
0490 CONSOL = $D01F  ;CONSOL PORT KEYS
0500 AUDC1 = $D201   ;AUDIO(1) CONTROL
0510 AUDF1 = $D200   ;AUDIO(1) FREQ.
0520 AUDCTL = $D208  ;AUDIO CONTROL
0530 ATRACT = $4D    ;MODE TIMER
0540 SCR =   PLAYERS+$2000 ;DISPLAY
0550 DISP =  SCR+$03E8 ;COUNTERS
0560 ;
0570 ;Reserved Bytes for Variables
0580 ;
0590 LOCATION .DS 1  ;PLAYER 1,2 Y POS
0600 PLX0 .DS 1      ;AIM X POS.
0610 PLX1 .DS 1      ;BULLET X POS.
0620 PLX2 .DS 1      ;SKEET1 X POS.
0630 PLX3 .DS 1      ;SKEET2 X POS.
0640 SKEE1 .DS 1     ;CHAR FOR SKEET1
0650 SKEE2 .DS 1     ;CHAR FOR SKEET2
0660 HIT .DS 1       ;HIT COUNTER
0670 SHOTS .DS 1     ;SHOT COUNTER
0680 TEMPO .DS 1     ;TEMPORARY REG.
0690 DRAW .DS 1      ;PLAYERS POINTER
0700 DIS2 .DS 1      ;LENGHT OF BULLET
0710 ;ON THE SCREEN
0720 LOSKEE1 .DS 1   ;Y POS. OF SKEE1
0730 LOSKEE2 .DS 1   ;Y POS. OF SKEE2
0740 DIRECT .DS 1    ;SKEET1 DIRECTION
0750 DIRECT2 .DS 1   ;SKEET2 DIRECTION
0760 SPEED .DS 1     ;WHICH SPEED
0770 CHECK .DS 1     ;# FINISH SKEES
0780 NUMBER .DS 1    ;MATH REGISTERS
0790 HUNDRED .DS 1
0800 TEN .DS 1
0810 ONE .DS 1
0820 DLIREG .DS 1    ;DLI REGISTER
0830 CSIZE .DS 1     ;SKEETS DISTANCE
0840 SCOREH .DS 1    ;HI BYTE OF SCORE
0850 SCOREL .DS 1    ;LO BYTE OF SCORE
0860 ROUND .DS 1     ;ROUND COUNTER
0870 POINT .DS 1     ;VALUE OF SKEETS
0880 TSHOT .DS 1     ;STAY OF BULLET
0890 AUINDX .DS 1    ;AUDIO REG.
0900 ;
0910 ;START SET UP
0920 ;
0930     LDA # <BEGIN ;WHEN RESET IS
0940     STA DOSVIN   ;PRESS
0950     LDA # >BEGIN ;GAME WILL
0960     STA DOSVIN+1 ;START OVER.
0970 ;
0980 ; CLEAR MEMORY FOR PLAYER,
0990 ; CHARACTER SET, AND SCREEN
1000 ;
1010     LDA #0
1020     LDX #0
1030 CLEAR STA PLAYERS,X
1040     STA PLAYERS+$0100,X
1050     STA PLAYERS+$0200,X
1060     STA PLAYERS+$0300,X
1070     STA PLAYERS+$0400,X
1080     STA PLAYERS+$0500,X
1090     STA PLAYERS+$0600,X
1100     STA PLAYERS+$0700,X
1110     INX
1120     BNE CLEAR
1130 ;
```

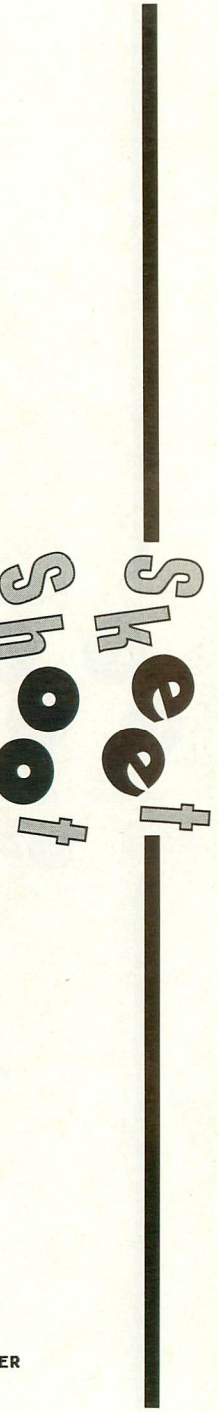
```
1140 BEGIN LDY #0    ;DRAW
1150 CONTST LDA CHARDT,Y ;CHARACTERS
1160     TAX         ;ON THE
1170     INY         ;SCREEN
1180     LDA CHARDT,Y
1190 STORDT STA SCR
1200     PHA
1210     CLC
1220     LDA STORDT+1
1230     ADC #1
1240     STA STORDT+1
1250     LDA STORDT+2
1260     ADC #0
1270     STA STORDT+2
1280     PLA
1290     DEX
1300     BNE STORDT
1310     INY
1320     CPY #124
1330     BNE CONTST
1340 ;
1350 ;SET UP SCREEN
1360 ;
1370 SUS LDA #LST&255
1380     STA SDLSTL
1390     LDA #LST/256
1400     STA SDLSTH
1410     LDA #192
1420     STA NMIEN
1430     LDA #DLI&255 ;SET DLI
1440     STA INTL
1450     LDA #DLI/256
1460     STA INTH
1470     LDA #0
1480     STA DLIREG
1490     LDA #242
1500     STA COLBK
1510 ;
1520 ; SET UP P/M GRAPHICS
1530 ;
1540 START LDA # >PLAYERS
1550     STA PMBASE
1560     LDA #88
1570     STA PCOLP0
1580     STA PCOLP2
1590     LDA #134
1600     STA PCOLP3
1610     LDA #$0A
1620     STA PCOLP1
1630     LDA #120
1640     STA HPOSP0
1650     STA PLX0
1660     STA HPOSP1
1670 ;
1680 ;CLEAR REGISTERS
1690     LDA #0
1700     STA HPOSP2
1710     STA PLX2
1720     STA HPOSP3
1730     STA PLX3
1740     STA SHOTS
1750     STA HIT
1760     STA SCOREH
1770     STA SCOREL
1780     STA TSHOT
1790     STA ROUND
1800 ; CLEAR NUMBERS ON SCREEN
1810     LDA #16
1820     STA DISP+70
1830     STA DISP+71
1840     STA DISP+72
1850     STA DISP+7
1860     STA DISP+8
1870     STA DISP+9
1880     STA DISP+46
1890     STA DISP+47
1900     STA DISP+48
1910 ;
1920 ; SET UP SCREEN POINTERS
1930 ;
1940     LDA #62
1950     STA SDMCTL
1960     LDA #3
1970     STA GRACTL
1980     LDA #120
1990     STA LOCATION
2000     LDA #208    ;SET SKEETS
2010     STA LOSKEE1 ;IN THERE
2020     STA LOSKEE2 ;STARTING
2030     LDA #1      ;POSITION
2040     STA SIZEP0
2050     STA SIZEP1
2060     JSR CLRAIM  ;CLEAR GUNSIGHT
2070     LDA #1
2080     STA SKEE1
2090     STA SKEE2
2100     LDX #0
2110     JSR RESET
2120     JSR RESET2
2130     JSR TSCORE
2140 ;
2150 ; PRINT TITLE IN SKY
2160 ;
2170     LDX #0
2180 PRINT LDA SKESHO,X
2190     STA SCR+135,X
2200     INX
2210     CPX #11
2220     BEQ LCHAR
2230     JMP PRINT
2240 SKESHO .SBYTE "SKEET SHOOT"
2250 ;
2260 ; REDEFINE CHARACTER SET
2270 ;
2280 LCHAR LDA $E000,X
2290     STA CHSET,X
2300     LDA $E0FF,X
2310     STA CHSET+$FF,X
2320     INX
2330     BNE LCHAR
2340     LDA # >CHSET
2350     STA CHBAS
2360     LDX #80
2370 CHANCH LDA CHDATA,X
2380     STA CHSET+8,X
2390     DEX
2400     BPL CHANCH
2410 ;
2420 ; WAIT FOR (START) KEY
2430 ;
2440 CKEY LDA CONSOL
2450     CMP #6
2460     BNE CKEY
2470     STA HITCRL
2480 ;
2490 ;CLEAR TITLE
2500 ;
2510     LDX #0
2520 CSCR LDA CLRSCR,X
2530     STA SCR+135,X
2540     INX
2550     CPX #11
2560     BEQ PULL
2570     JMP CSCR
2580 CLRSCR .SBYTE "           "
2590 ;
2600 ; VCOUNT DELAY ROUTINE
2610 ;
2620 CHK LDA VCOUNT
2630     CMP #123
2640     BNE CHK
2650     LDA CDTMF3
2660     BNE CHK2
2670     JSR AU0
2680 CHK2 LDA VCOUNT
2690     CMP #123
2700     BEQ CHK2
2710     LDA CHECK
2720     CMP #2
2730     BMI DIR
2740     BEQ PULL
2750 ;
2760 ; BEGIN PLAY
2770 ;
2780 PULL JSR CLRSKE1 ;CLEAR SKEET1
2790     JSR CLRSKE2 ;CLEAR SKEET2
2800     LDA #114    ;SET PLAYERS
2810     STA HPOSP0  ;TO THERE
2820     STA PLX0    ;POSITIONS
2830     LDA #130    ;AND WAIT
2840     STA LOCATION ;FOR THE STICK
2850     JSR CLRAIM  ;TO BE PUSH UP
2860     LDA #0
2870     STA AUDC1   ;CLEAR AUDIO
2880 ;
2890 ;WAIT FOR STICK TO BE PUSHED UP
2900 ;
2910 PULLS
2920     LDA STICK
2930     CMP #14
2940     BNE PULLS
2950     LDA #0
2960     STA CHECK
2970     LDA RANDOM  ;LOAD RANDOM #
2980     STA SPEED   ;FOR SPEED.
2990     LDA #0      ;MAKE SLINING
3000     STA CDTMF3  ;SOUND.
3010     LDA #18
3020     STA AUINDX
3030     JSR AU0
3040     LDA #0
3050     STA ATRACT
3060 DIR JMP PICKDIR ;RELEASE SKEETS.
3070 ;
3080 COU LDA DIS2    ;COUNT LENGTH OF
3090     CMP #1      ;BULLET ON
3100     BEQ CLRSHOT ;THE SCREEN
3110     BPL GOBULL
3120 ;
3130 ;JOYSTICK CONTROL
3140 ;
3150 TRIG
3160     LDX TSHOT   ;COUNT SHOTS THAT
3170     CPX #2      ;HAVE BEEN FIRED
3180     BPL LRMOVE  ;IF TWO HAS BEEN
3190     LDA TRIG0   ;FIRED THEN YOUR
3200     CMP #0      ;OUT OF SHELL
3210     BEQ FIRE
3220     JMP LRMOVE
3230 FIRE JMP SHOT   ;FIRE GUN
3240 GOBULL JMP SHOOT ;DISPLAY BULLET
3250 CLRSHOT JMP ERASE ;CLEAR SHOOT
3260 LRMOVE LDY STICK ;MOVE YOUR
3270     LDX PLX0    ;AIM LEFT OR
3280     LDA STRX,Y  ;RIGHT
3290     CMP #0
3300     BEQ STOHOZ
3310     BPL RIGHT
3320 LEFT SEC
3330     CPX #40
3340     BCC LEFT2
3350     DEX
3360     DEX
3370 LEFT2 JMP STOHOZ
3380 RIGHT SEC
3390     CPX #204
3400     BCS STOHOZ
3410     INX
```



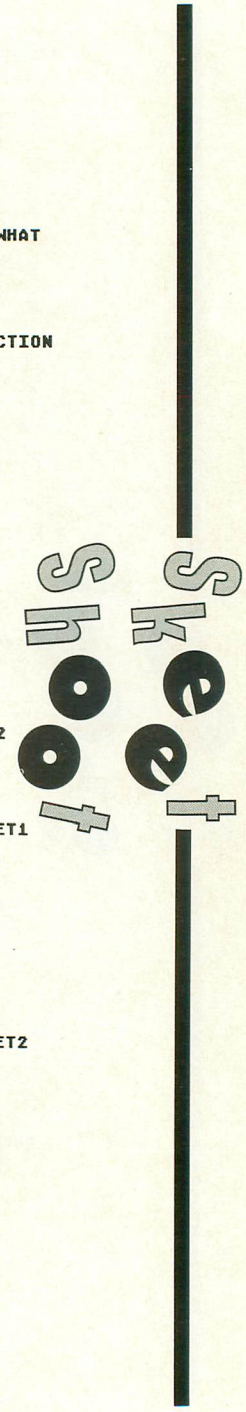
```
3420      INX
3430 STOHOZ STX PLX0
3440      STX HPOSP0
3450      JMP HMOVE
3460 UDMOVE LDY STICK ;MOVE YOUR AIM
3470      LDA STRY,Y  ;UP OR DOWN
3480      BEQ MAC
3490      BPL MDN
3500      BMI MUP
3510 MDN SEC
3520      LDA LOCATION
3530      CMP #212
3540      BCS MAC
3550      JSR MOVEDN
3560      JSR MOVEDN
3570      JSR MOVEDN
3580 MAC JMP CHK
3590 MUP SEC
3600      LDA LOCATION
3610      CMP #16
3620      BCC MAC
3630      JSR MOVEUP
3640      JSR MOVEUP
3650      JSR MOVEUP
3660      JSR MOVEUP
3670      JMP CHK
3680 MOVEUP CLC       ;MOVE PL0 UP
3690      LDA DRAW
3700      ADC #14
3710      STA TEMPO
3720      LDX LOCATION
3730      LDY DRAW
3740      DEX
3750 LOOPUP LDA PIC,Y
3760      STA PLAYER0,X
3770      INX
3780      INY
3790      CPY TEMPO
3800      BNE LOOPUP
3810      LDA #0
3820      STA PLAYER0,X
3830      DEC LOCATION
3840      RTS
3850 ;
3860 MOVEDN CLC       ;MOVE PL0 DOWN
3870      LDA DRAW
3880      ADC #14
3890      STA TEMPO
3900      LDX LOCATION
3910      LDY DRAW
3920      LDA #0
3930      STA PLAYER0,X
3940      INX
3950 LOOPDN LDA PIC,Y
3960      STA PLAYER0,X
3970      INY
3980      INX
3990      CPY TEMPO
4000      BNE LOOPDN
4010      INC LOCATION
4020      RTS
4030 ;
4040 HMOVE CLC        ;HOZ. MOVE
4050      LDA DRAW
4060      ADC #14
4070      STA TEMPO
4080      LDX LOCATION
4090      LDY DRAW
4100 LOOPH LDA PIC,Y
4110      STA PLAYER0,X
4120      INX
4130      INY
4140      CPY TEMPO
4150      BNE LOOPH
4160      BEQ JUMPUP
4170 JUMPUP JMP UDMOVE
```

```
4180 ;
4190 ; SHOOT AT THE SKEET
4200 ;
4210 SHOT LDA #0      ;MAKE AUDIO
4220      STA CDTMF3  ;GUN SHOT
4230      LDA #0
4240      STA AUINDX
4250      LDA CDTMF3
4260      BNE CONTS
4270      JSR AU0
4280 CONTS LDX TSHOT ;DISPLAY BULLET
4290      INX         ;15 LOOPS ON THE
4300      STX TSHOT   ;SCREEN
4310      LDA #15
4320      STA DIS2
4330      INC SHOTS
4340 SHOOT LDA PLX0
4350      STA HPOSP1
4360      CLC
4370      LDA DRAW
4380      ADC #9
4390      STA TEMPO
4400      LDX LOCATION
4410      LDY DRAW
4420 BULLET LDA PIC2,Y
4430      STA PLAYER1,X
4440      INX
4450      INY
4460      CPY TEMPO
4470      BNE BULLET
4480      DEC DIS2
4490      JMP COLL
4500 ;
4510 ERASE LDA DRAW  ;ERASE THE BULLET
4520      ADC #9
4530      STA TEMPO
4540      LDX LOCATION
4550      LDY DRAW
4560      LDA #0
4570      STA DIS2
4580 CLRLOP LDA #0
4590      STA PLAYER1,X
4600      INX
4610      INY
4620      CPY TEMPO
4630      BNE CLRLOP
4640      LDX SHOTS   ;CHANGE SHOT REG.
4650      JSR DISPNUM
4660      JSR PUTSHOT
4670      JMP CHK
4680 ;
4690 SOUMAK STA AUINDX
4700      LDA #8
4710      STA CDTMF3
4720      JSR AU0
4730      RTS
4740 ;
4750 ;COLLISION
4760 ;
4770 COLL LDA P1PL
4780      AND #$04
4790      BNE CCOLOR
4800 ;
4810 COLL2 LDA P1PL
4820      STA HITCRL
4830      AND #$08
4840      BNE HIT2
4850      JMP CHK
4860 ;
4870 CCOLOR LDA SKEE1 ;SHOT SKEET1
4880      CMP #15     ;CHECK IF IT
4890      BPL COLL2   ;HAS BEEN HIT
4900      INC HIT     ;BEFORE.
4910      LDA SKEE1   ;NO!
4920      ADC #22     ;CHANGE CHARACTER
4930      STA SKEE1   ;MAKE SOME
```

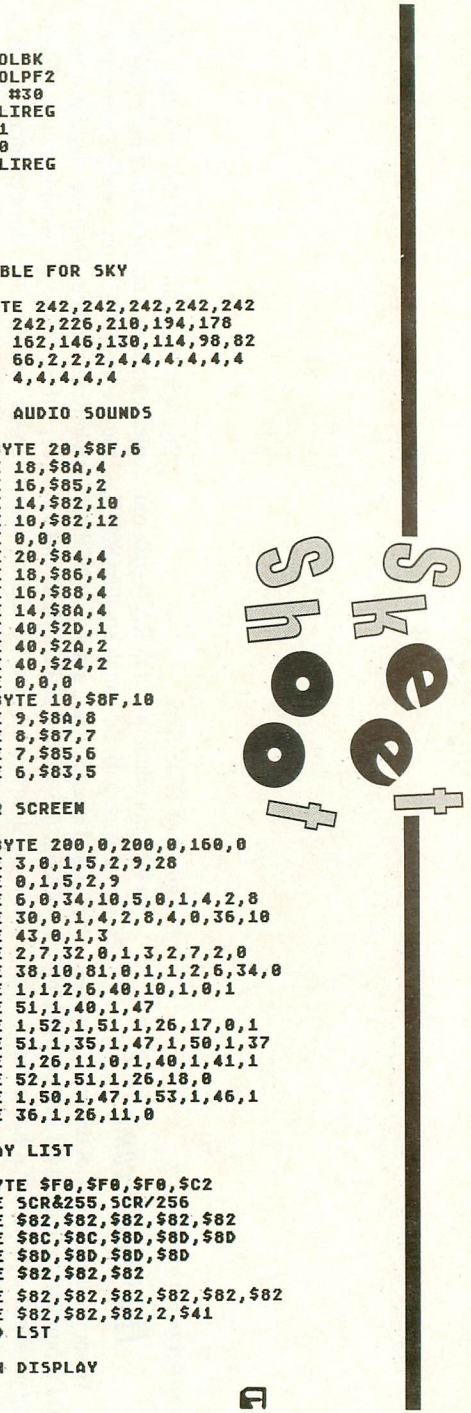
```
4940      LDA #42     ;NOISE
4950      JSR SOUMAK  ;AND GIVE ME
4960      LDX HIT     ;SOME POINTS.
4970      JSR DISPNUM
4980      JSR PUTHIT
4990      LDA #128
5000      JSR SCSP
5010      LDA POINT
5020      JSR B16
5030      JMP COLL2
5040 ;
5050 HIT2 LDA SKEE2  ;SHOT SKEET2
5060      CMP #15     ;SAME AS BEFORE
5070      BPL MAC2
5080      INC HIT
5090      LDA SKEE2
5100      ADC #22
5110      STA SKEE2
5120      LDA #42
5130      JSR SOUMAK
5140      JSR SCSP
5150      LDA POINT
5160      JSR B16
5170 MAC2 JMP CHK
5180 ;
5190 B16 CLC          ;16-BIT MATH
5200      ADC SCOREL  ;ADDITION
5210      STA SCOREL  ;ROUTINE
5220      LDA #0
5230      ADC SCOREH
5240      STA SCOREH
5250      JSR TSCORE
5260      RTS
5270 ;
5280 SMALL2 LDX CSIZE ;KEEP TRACK
5290      INX         ;OF SKEET
5300      STX CSIZE   ;DISTANCE
5310      CPX #49
5320      BNE SKEE2UP
5330 MAKE2 LDA SKEE2 ;CHANGE
5340      ADC #3      ;CHARACTER OF
5350      STA SKEE2   ;THE SKEET
5360      LDA #0
5370      STA CSIZE
5380 ;
5390 ; P/M OF SKEET (2)
5400 ;
5410 SKEE2UP CLC
5420      LDA SKEE2
5430      ADC #4
5440      STA TEMPO
5450      LDX LOSKEE2
5460      LDY SKEE2
5470      DEX
5480 LOOPSK2 LDA MG,Y
5490      STA PLAYER3,X
5500      INX
5510      INY
5520      CPY TEMPO
5530      BNE LOOPSK2
5540      LDA #0
5550      STA PLAYER3,X
5560      DEC LOSKEE2
5570      LDA LOSKEE2
5580      CMP #24
5590      BEQ RESET2
5600      RTS
5610 ;
5620 SMALL1 LDX CSIZE ;KEEP TRACK
5630      INX         ;OF DISTANCE
5640      STX CSIZE   ;SKEET1
5650      CPX #49
5660      BNE SKEE1UP
5670 MAKE1 LDA SKEE1 ;CHANGE CHARACTER
5680      ADC #3
5690      STA SKEE1
```



```
5700      LDA #0
5710      STA CSIZE
5720 ;
5730 ; P/M OF SKEET (1)
5740 ;
5750 SKEE1UP CLC
5760      LDA SKEE1
5770      ADC #4
5780      STA TEMPO
5790      LDX LOSKEE1
5800      LDY SKEE1
5810      DEX
5820 LOOPSK1 LDA MG,Y
5830      STA PLAYER2,X
5840      INX
5850      INY
5860      CPY TEMPO
5870      BNE LOOPSK1
5880      LDA #0
5890      STA PLAYER2,X
5900      DEC LOSKEE1
5910      LDA LOSKEE1
5920      CMP #24
5930      BEQ RESET
5940      RTS
5950 ;
5960 ; RESET REGISTERS & COUNT ROUNDS
5970 ;
5980 RESET2 JSR CLRSKE2
5990      INC ROUND
6000      INC CHECK
6010      LDA RANDOM
6020      STA DIRECT2
6030      JSR SIDE2
6040      ADC #8
6050      STA HPOSP3
6060      STA PLX3
6070      LDA #208
6080      STA LOSKEE2
6090      LDA #1
6100      STA SKEE2
6110      LDA #0
6120      STA TSHOT
6130      STA AUDF1
6140      STA AUDC1
6150      STA CDTMV3
6160      LDX ROUND
6170      CPX #31
6180      BEQ ADDUP
6190      JSR DISPNUM
6200      JSR PUTROD
6210      STA HITCRL
6220      RTS
6230 ;
6240 ; RESET SKEET1 REGISTERS
6250 ;
6260 RESET JSR CLRSKE1
6270      INC CHECK
6280      LDA RANDOM
6290      STA DIRECT
6300      JSR SIDE1
6310      STA HPOSP2
6320      STA PLX2
6330      LDA #208
6340      STA LOSKEE1
6350      LDA #1
6360      STA SKEE1
6370      STA HITCRL
6380      RTS
6390 ;
6400 ADDUP LDA #5     ;AT END OF GAME
6410      JSR B16     ;GIVE 5 POINTS
6420      LDX SHOTS   ;FOR EVER BULLET
6430      INX         ;THAT IS LEFT.
6440      STX SHOTS
6450      CPX #60
6460      BEQ GAME
6470      JMP ADDUP
6480 ;
6490 GAME LDX #0      ;DISPLAY ON
6500 OVER LDA END,X   ;THE SCREEN
6510      STA SCR+135,X ;GAME OVER
6520      INX
6530      CPX #9
6540      BEQ JAM
6550      JMP OVER
6560 ;
6570 JAM LDA CONSOL   ;WAIT FOR
6580      CMP #6      ;START KEY TO
6590      BEQ AGAIN   ;BEGIN.
6600      LDA #0      ;
6610      STA HPOSP2  ;GET PLAYERS 1,2
6620      STA HPOSP3  ;OUT OF THE
6630      JMP JAM     ;WAY
6640 ;
6650 AGAIN JSR CLRAIM
6660      JMP START
6670 ;
6680 END .SBYTE "GAME OVER"
6690 ;
6700 SIDE1 LDA DIRECT ;SKEET ONE
6710      JMP SIDE    ;DIRECTION
6720 SIDE2 LDA DIRECT2 ;SKEET TWO
6730 SIDE CMP #85     ;DIRECTION
6740      BMI LSIDE
6750      CMP #170
6760      BMI MIDDLE
6770      BPL RSIDE
6780 RE   RTS
6790 RSIDE LDA #50    ;LEFT SIDE
6800      JMP RE
6810 MIDDLE LDA #118 ;MIDDLE
6820      JMP RE
6830 LSIDE LDA #192   ;RIGHT SIDE
6840      JMP RE
6850 SKEEU2 JSR SMALL1 ;MOVE SKEET1
6860      LDA SPEED   ;UP
6870      SEC
6880      CMP #50
6890      BCC PICK2
6900      JSR SMALL1
6910      LDA SPEED
6920      SEC
6930      CMP #160
6940      BCC PICK2
6950      JSR SMALL1
6960      JMP PICK2
6970 SKEE2U2 JSR SMALL2 ;MOVE SKEET2
6980      SEC         ;UP
6990      LDA LOCATION
7000      CMP #212
7010      BCS PAST
7020      JSR MOVEDN  ;GRAVITY
7030 PAST LDA SPEED
7040      SEC
7050      CMP #50
7060      BCC GOUP
7070      JSR SMALL2
7080      LDA SPEED
7090      SEC
7100      CMP #160
7110      BCC GOUP
7120      JSR SMALL2
7130 ;
7140 GOUP JMP COU
7150 ;
7160 ;MAKE 16-BIT AUDIO WITH SUSTAIN
7170 ;
7180 AU0 LDX AUINDX
7190      LDA SOUND1,X
7200      STA AUDF1
7210      INX
7220      LDA SOUND1,X
7230      STA AUDC1
7240      INX
7250      LDA SOUND1,X
7260      STA CDTMV3
7270      INX
7280      STX AUINDX
7290      LDA #$FF
7300      STA CDTMF3
7310      RTS
7320 ;
7330 PICKDIR LDA DIRECT ;MOVE IN WHAT
7340      CMP #85     ;DIRECTION
7350      BMI SLEFT   ;SKEET1
7360      CMP #170
7370      BMI SKEEU2
7380      BPL SRIGHT
7390 PICK2 LDA DIRECT2 ;MOVE DIRECTION
7400      CMP #85     ;SKEET2
7410      BMI S2LEFT
7420      CMP #170
7430      BMI SKEE2U2
7440      BPL S2RIGHT
7450      JMP COU
7460 SLEFT LDX PLX2   ;MOVE SKEET1
7470      DEX         ;LEFT
7480      STX PLX2
7490      STX HPOSP2
7500      JMP SKEEU2
7510 SRIGHT LDX PLX2 ;MOVE SKEET1
7520      INX         ;RIGHT
7530      STX PLX2
7540      STX HPOSP2
7550      JMP SKEEU2
7560 S2LEFT LDX PLX3 ;MOVE SKEET2
7570      DEX         ;LEFT
7580      STX HPOSP3
7590      STX PLX3
7600      JMP SKEE2U2
7610 S2RIGHT LDX PLX3 ;MOVE SKEET2
7620      INX         ;RIGHT
7630      STX HPOSP3
7640      STX PLX3
7650      JMP SKEE2U2
7660 CLRSKE1 LDA SKEE1 ;CLEAR SKEET1
7670      ADC #4
7680      STA TEMPO
7690      LDX LOSKEE1
7700      LDY SKEE1
7710 BLANK1 LDA #0
7720      STA PLAYER2,X
7730      INX
7740      INY
7750      CPY TEMPO
7760      BNE BLANK1
7770      RTS
7780 CLRSKE2 LDA SKEE2 ;CLEAR SKEET2
7790      ADC #4
7800      STA TEMPO
7810      LDX LOSKEE2
7820      LDY SKEE2
7830 BLANK2 LDA #0
7840      STA PLAYER3,X
7850      INX
7860      INY
7870      CPY TEMPO
7880      BNE BLANK2
7890      RTS
7900 CLRAIM LDX #0    ;CLEAR AIM
7910      LDA #0
7920 CLRLOOP INX
7930      STA PLAYER0,X
7940      STA PLAYER1,X
7950      CPX #255
7960      BNE CLRLOOP
7970      JSR MOVEUP
7980      RTS
```



```
7990 DISPNUM LDY #0  ;DISPLAY NUMBER
8000     TXA         ;ON THE SCREEN
8010     SEC
8020 L01 SBC #$64
8030     BCC L02
8040     INY
8050     TAX
8060     BCS L01
8070 L02 CLC
8080     TYA
8090     ADC #$10
8100     STA HUNDRED
8110     LDY #0
8120     TXA
8130     SEC
8140 L03 SBC #$0A
8150     BCC L04
8160     INY
8170     TAX
8180     BCS L03
8190 L04 CLC
8200     TYA
8210     ADC #$10
8220     STA TEN
8230     TXA
8240     ADC #$10
8250     STA ONE
8260     RTS
8270 ;
8280 TSCORE LDA SCOREL ;PRINT SCORE
8290     STA FR0     ;ON SCREEN
8300     LDA SCOREH
8310     STA FR0+1
8320     JSR IFP
8330     JSR FASC
8340     LDY #0
8350 S01 LDA (INBUFF),Y
8360     BMI S02
8370     INY
8380     BNE S01
8390 S02 AND #$7F
8400     LDX #4
8410 S03 SEC
8420     SBC #$20
8430     STA DISP+30,X
8440     CPX #0
8450     BEQ S05
8460     DEX
8470     DEY
8480     CPY #$FF
8490     BEQ S04
8500     LDA (INBUFF),Y
8510     CLC
8520     BCC S03
8530 S04 INY
8540     LDA #$30
8550     BNE S03
8560 S05 RTS
8570 PUTHIT LDA HUNDRED ;PUT # OF
8580     STA DISP+46 ;HITS ON
8590     LDA TEN     ;SCREEN
8600     STA DISP+47
8610     LDA ONE
8620     STA DISP+48
8630     RTS
8640 PUTSHOT LDA HUNDRED ;PUT # OF
8650     STA DISP+7  ;SHOTS ON
8660     LDA TEN     ;SCREEN
8670     STA DISP+8
8680     LDA ONE
8690     STA DISP+9
8700     RTS
8710 PUTROD LDA HUNDRED ;PUT # OF
8720     STA DISP+70 ;ROUNDS
8730     LDA TEN     ;ON SCREEN
8740     STA DISP+71
8750     LDA ONE
8760     STA DISP+72
8770     RTS
8780 SCSP LDA SPEED  ;SET VALUE OF
8790     SEC         ;SKEETS
8800     CMP #160    ;DEPENDING ON
8810     BCS SC50    ;SPEED
8820     SEC
8830     CMP #50
8840     BCS SC25
8850 SC10 LDA #10    ;GIVE 10 FOR SLOW
8860     STA POINT
8870     RTS
8880 SC25 LDA #25    ;GIVE 25 FOR MED.
8890     STA POINT
8900     RTS
8910 SC50 LDA #50    ;GIVE 50 FOR FAST
8920     STA POINT
8930     RTS
8940 ;
8950 ; DATA FOR NEW CHARACTER SET
8960 ;
8970 CHDATA .BYTE 56,120,216,24,24
8980     .BYTE 24,24,225
8990     .BYTE 0,0,0,0,0,0,0,0
9000     .BYTE 0,124,12,12,124
9010     .BYTE 96,96,124
9020     .BYTE 0,0,0,124,12,124,12,124
9030     .BYTE 0,0,0,0,0,0,54,62,6
9040     .BYTE 60,195,195,195,195
9050     .BYTE 195,195,60
9060     .BYTE 0,56,108,108,108
9070     .BYTE 108,108,56
9080     .BYTE 0,0,0,56,108,108,108,56
9090     .BYTE 0,0,0,0,0,0,28,54,28
9100     .BYTE 85,0,0,0,0,0,0,0
9110 ;
9120 ;LEFT OR RIGHT DATA FOR STICK
9130 ;
9140 STRX .BYTE 0,0,0,0,0,1,1,1
9150     .BYTE 0,-1,-1,-1,0,0,0,0
9160 ;
9170 ;UP OR DOWN DATA FOR STICK
9180 ;
9190 STRY .BYTE 0,0,0,0,0,1,-1,0
9200     .BYTE 0,1,-1,0,0,1,-1,0
9210 ;
9220 ;GRAPHICS FOR AIM(GUNSIGHT)
9230 ;
9240 PIC .BYTE 16,16,16,0,0,0
9250     .BYTE 146,0,0,0,16,16,16
9260 ;
9270 ;GRAPHICS FOR SKEETS
9280 ;
9290 MG  .BYTE 0,56,124,254,0,48,120
9300     .BYTE 252,0,32,112,248,0,32
9310     .BYTE 112,0,0,0,0,0,0,0,0
9320     .BYTE 0,0,0,0,20,65,8,34,8
9330     .BYTE 32,132,32,80,128,16
9340     .BYTE 128,64,128,32,0,0,0,0,0
9350 ;
9360 ;GRAPHICS FOR BULLET
9370 ;
9380 PIC2 .BYTE 0,0,0,124,124,124
9390     .BYTE 124,124,124,124,0,0,0
9400 ;
9410 ;DLI ROUTINE
9420 ;
9430 DLI PHA
9440     TXA
9450     PHA
9460     LDX DLIREG
9470     LDA TABLE,X
9480     INX
9490     STX DLIREG
9500     STA WSYNC
9510     STA COLBK
9520     STA COLPF2
9530 DLOOP LDA #30
9540     CMP DLIREG
9550     BNE A1
9560     LDA #0
9570     STA DLIREG
9580 A1  PLA
9590     TAX
9600     PLA
9610     RTI
9620 ;
9630 ;COLOR TABLE FOR SKY
9640 ;
9650 TABLE .BYTE 242,242,242,242,242
9660     .BYTE 242,226,210,194,178
9670     .BYTE 162,146,130,114,98,82
9680     .BYTE 66,2,2,2,4,4,4,4,4,4
9690     .BYTE 4,4,4,4,4
9700 ;
9710 ;DATA FOR AUDIO SOUNDS
9720 ;
9730 SOUND1 .BYTE 20,$8F,6
9740     .BYTE 18,$8A,4
9750     .BYTE 16,$85,2
9760     .BYTE 14,$82,10
9770     .BYTE 10,$82,12
9780     .BYTE 0,0,0
9790     .BYTE 20,$84,4
9800     .BYTE 18,$86,4
9810     .BYTE 16,$88,4
9820     .BYTE 14,$8A,4
9830     .BYTE 40,$2D,1
9840     .BYTE 40,$2A,2
9850     .BYTE 40,$24,2
9860     .BYTE 0,0,0
9870 SOUND2 .BYTE 10,$8F,10
9880     .BYTE 9,$8A,8
9890     .BYTE 8,$87,7
9900     .BYTE 7,$85,6
9910     .BYTE 6,$83,5
9920 ;
9930 ;DATA FOR SCREEN
9940 ;
9950 CHARDT .BYTE 200,0,200,0,160,0
9960     .BYTE 3,0,1,5,2,9,28
9970     .BYTE 0,1,5,2,9
9980     .BYTE 6,0,34,10,5,0,1,4,2,8
9990     .BYTE 30,0,1,4,2,8,4,0,36,10
010000   .BYTE 43,0,1,3
010010   .BYTE 2,7,32,0,1,3,2,7,2,0
010020   .BYTE 38,10,81,0,1,1,2,6,34,0
010030   .BYTE 1,1,2,6,40,10,1,0,1
010040   .BYTE 51,1,40,1,47
010050   .BYTE 1,52,1,51,1,26,17,0,1
010060   .BYTE 51,1,35,1,47,1,50,1,37
010070   .BYTE 1,26,11,0,1,40,1,41,1
010080   .BYTE 52,1,51,1,26,18,0
010090   .BYTE 1,50,1,47,1,53,1,46,1
010100   .BYTE 36,1,26,11,0
010110 ;
010120 ;DISPLAY LIST
010130 ;
010140 LST .BYTE $F0,$F0,$F0,$C2
010150   .BYTE SCR&255,SCR/256
010160   .BYTE $82,$82,$82,$82,$82
010170   .BYTE $8C,$8C,$8D,$8D,$8D
010180   .BYTE $8D,$8D,$8D,$8D
010190   .BYTE $82,$82,$82
010200   .BYTE $82,$82,$82,$82,$82,$82
010210   .BYTE $82,$82,$82,2,$41
010220   .WORD LST
010230 ;
010240 ;SCREEN DISPLAY
010250 ;
010260   .END
```



continued from page 9

How things came to be

When my usual attack of the pre-Christmas flu grounded me recently, I started thinking again about a project that had already been on my mind for quite some time, but for which I had never found enough time to implement: a RAM disk for the 800XL. There is, after all, in every 64K XL, a 14K bank of RAM beneath the operating system ROM's where it does, more or less, nothing.

The question then was how to use this RAM as a drive. The first possible solution that occurred to me was to write a RAM disk device driver. I soon discarded that suggestion, however, because it would involve writing a complete File Management system. And that particular wheel is already among us in the guise of DOS.SYS.

So why not adapt the DOS RAM-disk routines to work with the much smaller 800XL free RAM space? Alas, after a lot of disassembling, this proved to be a dead end too. I have no doubt at all that it could be done, but not without a commented source listing, like the one available for DOS 2.0 in *Inside Atari DOS*.

At this point I was ready to abandon the project had my fiancée not challenged me to go on. Wasn't I the one who kept telling people what a wonderful and flexible machine the 800XL really was? (See, Caroline, I told you you'd get the credit you're due.) So grumbling (quietly) I started again by studying the DOS 2.0 listing.

And then the golden idea hit home.

All DOS functions eventually vector through SIOV and DSKINV for their implementation. And for each of these routines there is in DOS 2.5 one, and only one, place where it is called. So if we could just intercept these calls, we would then be able to check whether the device addressed is a RAM disk and, if so, take appropriate measures. And that is precisely what *RAM Disk 800XL* does.

For the hard-core addict

Listing 2 is the assembly language source code for *RAM Disk 800XL*. You don't need to type it in, it is there only for those people interested in assembly language programming. Figure 1 shows how the extra RAM is used by the D5: driver. Figure 2 gives you the MDRIVE memory usage.


```
Address:  Use:
-------   ----
$C000     +------------------------+
          |   MDRIVE SIO routine   |
$C100     +------------------------+
          |   VTOC (sector $168)   |
$C180     +------------------------+
          |  Directory (sector $169)|
$C200     +------------------------+
          |  data (sector $171-$18C |
$CFFF     +-------------------------!
          .                         .
          .                         .
$D800     +------------------------+
          |  data (sector $18D-$1DC |
$FFFF     +------------------------+
```

Figure 1.
RAM disk memory usage

```
MDRIVE:  Run     Use:
-------  ---     ----
$3800    $3800   +-----------------+
                 |  install MDRIVE |
$38EC    $6BC    +-----------------+
                 |  runtime routines|
$3930    $C000   +-----------------+
                 |  SIO commands   |
$39CF            +-----------------+
```

Figure 2
MDRIVE memory usage

Jerry van Dijk uses his Atari both as a study tool and for recreation. His main interests are system-level programming and the use of computers in law practice.

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,56,208,57,32,209,5
6,169,234,141,35,241,173,35,8711
1010 DATA 241,170,32,226,56,224,234,24
0,1,96,169,11,162,0,141,66,6010
1020 DATA 3,169,163,141,68,3,169,56,14
1,69,3,173,205,56,141,72,5187
1030 DATA 3,173,206,56,141,73,3,32,86,
228,32,209,56,169,0,168,5337
1040 DATA 153,0,193,153,128,193,200,20
8,247,169,2,141,0,193,169,110,9870
1050 DATA 141,1,193,169,108,141,3,193,
169,127,141,56,193,169,255,160,1673
1060 DATA 57,153,0,193,200,192,69,208,
248,169,248,153,0,193,172,208,3590
1070 DATA 56,185,49,57,153,255,191,136
,208,247,32,226,56,172,207,56,1134
1080 DATA 185,237,56,153,187,6,136,208
,247,173,10,7,9,80,141,10,3765
1090 DATA 7,32,224,7,169,188,141,107,1
6,169,6,141,108,16,169,202,6664
1100 DATA 141,176,7,169,6,141,177,7,96
,125,29,29,29,29,29,29,8924
1110 DATA 32,32,83,101,116,116,105,110
,103,32,85,112,32,65,84,65,1902
1120 DATA 82,73,32,56,48,48,88,76,32,8
2,97,109,32,68,105,115,1686
1130 DATA 107,155,155,42,0,69,160,120,
169,0,141,14,212,173,1,211,6483
1140 DATA 133,203,169,254,141,1,211,96
,165,203,141,1,211,88,169,64,8770
1150 DATA 141,14,212,96,32,243,6,240,3
,76,83,228,160,1,140,3,4597
1160 DATA 3,96,32,243,6,240,3,76,89,22
8,120,169,0,141,14,212,6527
1170 DATA 173,1,211,133,203,169,254,14
1,1,211,32,0,192,165,203,141,23
1180 DATA 1,211,88,169,64,141,14,212,1
92,0,96,173,0,3,201,49,4204
1190 DATA 208,5,173,1,3,201,5,96,173,2
,3,201,82,240,37,201,6174
1200 DATA 80,240,18,201,87,240,14,197,
83,240,4,160,168,208,2,160,8964
1210 DATA 1,140,3,3,96,32,72,192,173,4
,3,133,204,173,5,3,2647
1220 DATA 133,205,208,13,32,72,192,134
,204,132,205,174,4,3,172,5,5700
1230 DATA 3,134,206,132,207,160,127,17
7,204,145,206,136,16,249,48,207,1857
1240 DATA 173,11,3,201,1,240,6,104,104
,160,144,208,196,173,10,3,6351
1250 DATA 201,104,208,5,162,0,160,193,
96,201,105,208,5,162,128,160,9115
1260 DATA 193,96,201,113,144,225,201,2
21,176,221,201,141,176,10,56,233,2638
1270 DATA 113,32,143,192,169,194,208,8
,56,233,141,32,143,192,169,216,1480
1280 DATA 24,101,205,168,166,204,96,13
3,204,169,0,133,205,162,7,6,6839
1290 DATA 204,38,205,202,208,249,96,22
6,2,227,2,0,56,0,0,0,1045
```

LISTING 2: ASSEMBLY

```
00010          .LI OFF
00020 ********************************
00030 *                              *
00040 *        MDRIVE 2.5            *
00050 *        ----------            *
00060 *                              *
00070 *  Ram disk driver for 800XL   *
00080 *  ------ With DOS 2.5 -----   *
00090 *                              *
00100 *   Use as RAMDISK.COM (D5:)   *
00110 *                              *
00120 *  Author: Jerry van Dijk      *
00130 *          Pelikaanhof 15      *
00140 *          2312 EA Leiden      *
00150 *          The Netherlands     *
00160 *                              *
00170 * Last revision: 12-dec-1988   *
00180 *                              *
00190 *   Written in SynAssembler    *
00200 *                              *
00210 * >>> USES PART OF PAGE 6 <<< *
00220 *                              *
00230 ********************************
00240 ;
00250 ;------------------------------
00260 ;         MDRIVE CONSTANTS
00270 ;------------------------------
00280 ;
00290 MAXSEC    .EQ $01 ;Max sector hi
00300 OFFSET    .EQ $38 ;VTOC off-set
00310 SECNUM    .EQ $45 ;VTOC data sec
00320 VSEC      .EQ $68 ;VTOC sec lo
00330 DSEC      .EQ $69 ;DIR sec lo
00340 SECLOW    .EQ $71 ;DATA sec low
00350 SECMID    .EQ $8D ;DATA sec mid
00360 SECHIGH   .EQ $DD ;DATA sec high
00370 TESTBYT   .EQ $EA ;RAM test byte
00380 ;
00390 ;------------------------------
00400 ;         MDRIVE EQUATES
00410 ;------------------------------
00420 ;
00430 TEMP      .EQ $CB   ;Memory stat
00440 ZPAGE     .EQ $CC   ;Temp. adr
00450 MPROG     .EQ $3800 ;Prog begin
00460 RTSTART   .EQ $38EE ;RT origin
00470 EXSTART   .EQ $3932 ;EX origin
00480 RAMLOW    .EQ $C000 ;Low RAM blk
00490 VTOC      .EQ $C100 ;VTOC adr
00500 DIR       .EQ $C180 ;DIR adr
00510 LOWBANK   .EQ $C200 ;DATA lo adr
00520 HIGHBANK  .EQ $D800 ;DATA hi adr
00530 SOMERAM   .EQ $F123 ;Test RAM
00540 ;
00550 ;------------------------------
00560 ;         DOS 2.5 CONSTANTS
00570 ;------------------------------
00580 ;
00590 D5        .EQ $05 ;D5: device
00600 DRV5      .EQ $50 ;Drive 5
00610 SECLEN    .EQ $7F ;Sector length
00620 ;
00630 ;------------------------------
00640 ;         DOS 2.5 EQUATES
00650 ;------------------------------
00660 ;
00670 DRVBYT    .EQ $70A  ;Act. drives
00680 DSIO      .EQ $7B0  ;SIOV call
00690 DINIT     .EQ $7E0  ;Init DOS
00700 DDSK      .EQ $106B ;DSKINV call
00710 ;
00720 ;------------------------------
00730 ;         ATASCII CODE'S
00740 ;------------------------------
00750 ;
00760 CD        .EQ $1D ;Cursor down
00770 CLS       .EQ $7D ;Clear screen
00780 EOL       .EQ $9B ;Clear screen
00790 ;
00800 ;------------------------------
00810 ;         XL SYSTEM CONSTANTS
00820 ;------------------------------
00830 ;
00840 IOCB0     .EQ $00 ;IOCB 0 offset
00850 NMIOFF    .EQ $00 ;NMI off value
00860 OK        .EQ $01 ;No error code
00870 PUTBUF    .EQ $0B ;CIO put buf
00880 DISK      .EQ $31 ;SIO dis code
00890 NMION     .EQ $40 ;NMI on value
00900 WRITE     .EQ $50 ;SIO put cmd
00910 READ      .EQ $52 ;SIO get cmd
00920 STATUS    .EQ $53 ;SIO stat cmd
00930 VERIFY    .EQ $57 ;SIO put cmd
00940 NOSEC     .EQ $90 ;Sector error
00950 NOCMD     .EQ $A8 ;Cmd error
00960 RAMON     .EQ $FE ;ROM disable
00970 ;
00980 ;------------------------------
00990 ;         XL SYSTEM EQUATES
01000 ;------------------------------
01010 ;
01020 DDEVIC    .EQ $300  ;DCB device
01030 DUNIT     .EQ $301  ;DCB unit
01040 DCOMND    .EQ $302  ;DCB cmd
01050 DSTATS    .EQ $303  ;DCB status
01060 DBUFLO    .EQ $304  ;DCB buf lo
01070 DBUFHI    .EQ $305  ;DCB buf hi
01080 DAUX1     .EQ $30A  ;DCB sec lo
01090 DAUX2     .EQ $30B  ;DCB sec hi
01100 ICCOM     .EQ $342  ;CIO cmd
01110 ICBAL     .EQ $344  ;CIO buf lo
01120 ICBAH     .EQ $345  ;CIO buf hi
01130 ICBLL     .EQ $348  ;CIO len lo
01140 ICBLH     .EQ $349  ;CIO len hi
01150 PAGE6     .EQ $6BC  ;Free space
01160 PORTB     .EQ $D301 ;Memory ctrl
01170 NMIEN     .EQ $D40E ;NMI control
01180 DSKINV    .EQ $E453 ;SIO status
01190 CIOV      .EQ $E456 ;CIO vector
01200 SIOV      .EQ $E459 ;SIO vector
01210 ;
01220 ********************************
01230 *     Check for RAM module     *
01240 ********************************
01250 ;
01260 ; This module is the first run
01270 ; when MDRIVE is executed. It
01280 ; checks whether there is RAM
01290 ; beneath the OS. If there is
01300 ; it runs the install module
```

```
01310 ; otherwise it simply exits.
01320 ;
01330          .OR MPROG
01340 ;
01350 MDRIVE
01360 ;
01370 ; Disable OS ROM's
01380 ;
01390          JSR ROMOFF
01400 ;
01410 ; Store & retrieve a byte
01420 ;
01430          LDA #TESTBYT
01440          STA SOMERAM
01450          LDA SOMERAM
01460          TAX
01470 ;
01480 ; Restore OS ROM's
01490 ;
01500          JSR ROMON
01510 ;
01520 ; Check if there is RAM
01530 ;
01540          CPX #TESTBYT
01550          BEQ INSTALL
01560          RTS
01570 ;
01580 ********************************
01590 *   INSTALL RAM DISK MODULE    *
01600 ********************************
01610 ;
01620 ; Module to install the RAM
01630 ; disk. It formats the disk,
01640 ; makes the DOS patches and
01650 ; copies the runtime and
01660 ; modules in place.
01670 ;
01680 INSTALL
01690 ;
01700 ; First print a message
01710 ;
01720          LDA #PUTBUF
01730          LDX #IOCB0
01740          STA ICCOM
01750          LDA #MSG
01760          STA ICBAL
01770          LDA /MSG
01780          STA ICBAH
01790          LDA MSGLEN
01800          STA ICBLL
01810          LDA MSGLEN+1
01820          STA ICBLH
01830          JSR CIOV
01840 ;
01850 ; Disable OS ROM's
01860 ;
01870          JSR ROMOFF
01880 ;
01890 ; Clear vtoc & directory
01900 ;
01910          LDA #0
01920          TAY
01930 .0       STA VTOC,Y
01940          STA DIR,Y
01950          INY
01960          BNE .0
01970 ;
01980 ; Write VTOC sector
01990 ;
02000          LDA #2
02010          STA VTOC
02020          LDA #110
02030          STA VTOC+1
02040          LDA #108
02050          STA VTOC+3
02060          LDA #%01111111
02070          STA VTOC+OFFSET
02080          LDA #%11111111
02090          LDY #OFFSET+1
02100 .1       STA VTOC,Y
02110          INY
02120          CPY #SECNUM
02130          BNE .1
02140          LDA #%11111000
02150          STA VTOC,Y
02160 ;
02170 ; Copy execute module in place
02180 ;
02190          LDY EXLEN
02200 .2       LDA EXSTART-1,Y
02210          STA RAMLOW-1,Y
02220          DEY
02230          BNE .2
02240 ;
02250 ; Enable OS ROM's
02260 ;
02270          JSR ROMON
02280 ;
02290 ; Copy runtime module in place
02300 ;
02310          LDY RTLEN
02320 .3       LDA RTSTART-1,Y
02330          STA PAGE6-1,Y
02340          DEY
02350          BNE .3
02360 ;
02370 ; Add D5: to DOS
02380 ;
02390          LDA DRVBYT
02400          ORA #DRV5
02410          STA DRVBYT
02420          JSR DINIT
02430 ;
02440 ; Patch DOS DSKINV call
02450 ;
02460          LDA #MDSK
02470          STA DDSK
02480          LDA /MDSK
02490          STA DDSK+1
02500 ;
02510 ; Patch DOS SIO call
02520 ;
02530          LDA #MSIO
02540          STA DSIO
02550          LDA /MSIO
02560          STA DSIO+1
02570 ;
02580 ; Imstallation done
02590 ;
02600          RTS
02610 ;
02620 ; The message
02630 ;
02640 MSG      .DA #CLS,#CD,#CD,#CD
02650          .DA #CD,#CD,#CD
02660          .AS '  Setting Up ATA'
02670          .AS 'RI 800XL Ram Dis'
02680          .AS 'k'
02690          .DA #EOL,#EOL
02700 MSGLEN   .DA MSGLEN-MSG
02710 ;
02720 ; Runtime module length
02730 ;
02740 RTLEN    .DA #RTEND-PAGE6+1
02750 ;
02760 ; Execute module length
02770 ;
02780 EXLEN    .DA #EXEND-RAMLOW+1
02790 ;
02800 ;-----------------------------
02810 ;     MDRIVE SUBROUTINES
02820 ;-----------------------------
02830 ;
02840 ROMOFF
02850 ;
02860 ; Disable OS ROM's
02870 ;
02880          SEI
02890          LDA #NMIOFF
02900          STA NMIEN
02910          LDA PORTB
02920          STA TEMP
02930          LDA #RAMON
02940          STA PORTB
02950          RTS
02960 ;
02970 ROMON
02980 ;
02990 ; Enable OS ROM's
03000 ;
03010          LDA TEMP
03020          STA PORTB
03030          CLI
03040          LDA #NMION
03050          STA NMIEN
03060          RTS
03070 ;
03080 ********************************
03090 *      RUNTIME MODULE          *
03100 ********************************
03110 ;
03120 ; This code is called by DOS if
03130 ; it executes a DSKINV (SIO
03140 ; status) or SIO call.
03150 ; If device is D5: and it is a
03160 ; status call the status is set
03170 ; to OK and the routine exits.
03180 ; If device is D5: and it is a
03190 ; SIO call then the OS ROM's
03200 ; are disabled and a jump is
03210 ; made to the execute module to
03220 ; execute the command.
03230 ; If the device isn't D5: then
03240 ; the routine continues with
03250 ; DSKINV or SIOV.
03260 ;
03270 ; Runtime code origin:
03280 ;
03290          .OR PAGE6
03300          .TA RTSTART
03310 ;
03320 ;-----------------------------
03330 ;         DSKINV PATCH
03340 ;-----------------------------
03350 ;
03360 MDSK
03370 ;
03380 ; Check if device is D5:
03390 ;
03400          JSR CHKDEV
03410          BEQ DODSK
03420 ;
03430 ; If not continue with DSKINV
03440 ;
03450          JMP DSKINV
03460 ;
03470 ;Otherwise set status & return
03480 ;
03490 DODSK    LDY #OK
03500          STY DSTATS
03510          RTS
03520 ;
03530 ;-----------------------------
03540 ;         SIOV PATCH
03550 ;-----------------------------
03560 ;
03570 MSIO
03580 ;
03590 ; Check if device is D5:
03600 ;
03610          JSR CHKDEV
03620          BEQ DOSIO
03630 ;
03640 ; If not continue with SIOV
```

```
03650 ;
03660           JMP SIOV
03670 ;
03680 ; Otherwise run execute module
03690 ;
03700 DOSIO     SEI
03710           LDA #NMIOFF
03720           STA NMIEN
03730           LDA PORTB
03740           STA TEMP
03750           LDA #RAMON
03760           STA PORTB
03770           JSR EXEC
03780           LDA TEMP
03790           STA PORTB
03800           CLI
03810           LDA #NMION
03820           STA NMIEN
03830           CPY #0
03840           RTS
03850 ;
03860 ;-------------------------------
03870 ;         RUNTIME SUBROUTINES
03880 ;-------------------------------
03890 ;
03900 CHKDEV
03910 ;
03920 ; Check if SIO device is D5:
03930 ; Returns with zero flag set is
03940 ; it is.
03950 ;
03960           LDA DDEVIC
03970           CMP #DISK
03980           BNE CHKDN
03990           LDA DUNIT
04000           CMP #D5
04010 CHKDN     RTS
04020 ;
04030 ;-------------------------------
04040 ;         END OF RUNTIME
04050 ;-------------------------------
04060 ;
04070 RTEND
04080 ;
04090 ********************************
04100 *    EXECUTE COMMAND MODULE    *
04110 ********************************
04120 ;
04130 ; This code, which is hidden
04140 ; beneath the OS, executes the
04150 ; RAM Disk SIO commands.
04160 ;
04170           .OR RAMLOW
04180           .TA EXSTART
04190 ;
04200 ; Decode command
04210 ;
04220 EXEC      LDA DCOMND
04230           CMP #READ
04240           BEQ GETSEC
04250           CMP #WRITE
04260           BEQ PUTSEC
04270           CMP #VERIFY
04280           BEQ PUTSEC
04290           CMP STATUS
04300           BEQ SETOK
04310 ;
04320 ; No command then return error
04330 ;
04340           LDY #NOCMD
04350           BNE ERRXIT
04360 ;
04370 ;-------------------------------
04380 ;         SET STATUS
04390 ;-------------------------------
04400 ;
04410 SETOK     LDY #OK
04420 ERRXIT    STY DSTATS
04430           RTS
04440 ;
04450 ;-------------------------------
04460 ;         PUT SECTOR
04470 ;-------------------------------
04480 ;
04490 PUTSEC
04500 ;
04510 ; Calculate sector address
04520 ;
04530           JSR CALC
04540 ;
04550 ; Move data set-up
04560 ;
04570           LDA DBUFLO
04580           STA ZPAGE
04590           LDA DBUFHI
04600           STA ZPAGE+1
04610           BNE MOVE
04620 ;
04630 ;-------------------------------
04640 ;         GET SECTOR
04650 ;-------------------------------
04660 ;
04670 GETSEC
04680 ;
04690 ; Calculate sector address
04700 ;
04710           JSR CALC
04720 ;
04730 ; Move data set-up
04740 ;
04750           STX ZPAGE
04760           STY ZPAGE+1
04770           LDX DBUFLO
04780           LDY DBUFHI
04790 ;
04800 ; Move the data: FROM address
04810 ; in (ZPAGE), TO address in X,Y
04820 ;
04830 MOVE      STX ZPAGE+2
04840           STY ZPAGE+3
04850           LDY #SECLEN
04860 .0        LDA (ZPAGE),Y
04870           STA (ZPAGE+2),Y
04880           DEY
04890           BPL .0
04900           BMI SETOK
04910 ;
04920 ;-------------------------------
04930 ;    CALCULATE SECTOR ADDRESS
04940 ;-------------------------------
04950 ;
04960 CALC
04970 ;
04980 ; First check the high byte
04990 ;
05000           LDA DAUX2
05010           CMP #MAXSEC
05020           BEQ DOCALC
05030 ;
05040 ; If not then a illegal sector
05050 ;
05060 SECERR    PLA
05070           PLA
05080           LDY #NOSEC
05090           BNE ERRXIT
05100 ;
05110 ; Check if vtoc sector
05120 ;
05130 DOCALC    LDA DAUX1
05140           CMP #VSEC
05150           BNE CHKDIR
05160 ;
05170 ; If it is set address $ return
05180 ;
05190           LDX #VTOC
05200           LDY /VTOC
05210           RTS
05220 ;
05230 ; Check if directory sector
05240 ;
05250 CHKDIR    CMP #DSEC
05260           BNE CHKDAT
05270 ;
05280 ; If so, set address & return
05290 ;
05300           LDX #DIR
05310           LDY /DIR
05320           RTS
05330 ;
05340 ; Check if sector not too low
05350 ;
05360 CHKDAT    CMP #SECLOW
05370           BCC SECERR
05380 ;
05390 ; Check if not too high
05400 ;
05410           CMP #SECHIGH
05420           BCS SECERR
05430 ;
05440 ; Find RAM bank
05450 ;
05460           CMP #SECMID
05470           BCS .0
05480 ;
05490 ; Calculate low bank
05500 ;
05510           SEC
05520           SBC #SECLOW
05530           JSR MULT
05540           LDA /LOWBANK
05550           BNE .1
05560 ;
05570 ; Calculate high bank
05580 ;
05590 .0        SEC
05600           SBC #SECMID
05610           JSR MULT
05620           LDA /HIGHBANK
05630 ;
05640 ; Add all up & return
05650 ;
05660 .1        CLC
05670           ADC ZPAGE+1
05680           TAY
05690           LDX ZPAGE
05700           RTS
05710 ;
05720 ;-------------------------------
05730 ;         MULTIPLY $80
05740 ;-------------------------------
05750 ;
05760 ; Multiply Accu with $80.
05770 ; Result in (ZPAGE).
05780 ;
05790 MULT      STA ZPAGE
05800           LDA #0
05810           STA ZPAGE+1
05820           LDX #7
05830 .0        ASL ZPAGE
05840           ROL ZPAGE+1
05850           DEX
05860           BNE .0
05870           RTS
05880 ;
05890 ;-------------------------------
05900 ;         END OF EXECUTE
05910 ;-------------------------------
05920 ;
05930 EXEND
05940 ;
05950 ********************************
05960 *     END OF MDRIVE 2.5        *
05970 ********************************
05980 ;
```

continued from page 11

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,214,63,136,88,96,26,
79,132,26,79,132,215,63,48,4451
1010 DATA 48,48,144,144,144,223,63,0,0
,6,0,1,5,32,40,99,7430
1020 DATA 41,49,57,56,51,32,65,99,116,
105,111,110,32,67,111,109,2646
1030 DATA 112,117,116,101,114,32,83,10
1,114,118,105,99,101,115,162,255,8261
1040 DATA 134,166,160,12,208,10,132,16
6,160,11,208,4,132,166,160,5,5792
1050 DATA 134,165,162,0,134,163,10,10,
10,10,170,152,157,66,3,165,3332
1060 DATA 163,240,10,157,74,3,165,164,
157,75,3,169,0,168,157,73,5315
1070 DATA 3,177,165,157,72,3,240,18,24
,165,165,105,1,157,68,3,2972
1080 DATA 165,166,105,0,157,69,3,76,86
,228,96,134,165,132,166,160,8481
1090 DATA 3,76,38,64,134,165,132,166,1
62,0,134,163,160,9,32,38,3709
1100 DATA 64,208,10,169,11,157,66,3,16
9,155,76,86,228,96,133,76,6227
1110 DATA 130,64,141,126,64,108,10,0,1
9,17,1,131,186,142,193,4,2622
1120 DATA 160,128,152,76,127,64,164,13
2,240,10,134,133,10,38,133,136,5682
1130 DATA 208,250,166,133,96,164,132,2
40,10,134,133,70,133,106,136,208,9490
1140 DATA 250,166,133,96,164,211,16,16
,133,134,134,135,56,169,0,229,7220
1150 DATA 134,168,169,0,229,135,170,15
2,96,134,211,224,0,16,3,32,4482
1160 DATA 184,64,133,130,134,131,165,1
33,16,14,170,69,211,133,211,165,9458
1170 DATA 132,32,184,64,133,132,134,13
3,169,0,133,135,96,240,27,202,8482
1180 DATA 134,199,170,240,21,134,198,1
69,0,162,8,10,6,198,144,2,3699
1190 DATA 101,199,202,208,246,24,101,1
35,133,135,165,134,166,135,96,32,8258
1200 DATA 201,64,166,130,240,27,134,19
8,166,132,240,21,202,134,199,162,2216
1210 DATA 8,10,38,135,6,198,144,6,101,
199,144,2,230,135,202,208,9911
1220 DATA 240,133,134,165,130,166,133,
32,237,64,165,131,166,132,32,237,59
1230 DATA 64,76,180,64,32,201,64,165,1
33,240,39,162,8,38,130,38,4540
1240 DATA 131,38,135,56,165,131,229,13
2,168,165,135,229,133,144,4,133,9674
1250 DATA 135,132,131,202,208,231,165,
130,42,162,0,164,131,132,134,76,8214
1260 DATA 180,64,162,16,38,130,38,131,
42,176,4,197,132,144,3,229,6389
1270 DATA 132,56,202,208,239,38,130,38
,131,133,134,165,130,166,131,76,8747
1280 DATA 180,64,32,68,65,165,134,166,
135,96,133,160,134,161,132,162,9663
1290 DATA 24,104,133,132,105,3,168,104
,133,133,105,0,72,152,72,160,5386
1300 DATA 1,177,132,133,130,200,177,13
2,133,131,200,177,132,168,185,160,2962
1310 DATA 0,145,130,136,16,248,165,17,
208,15,230,17,76,140,64,8,4185
1320 DATA 99,9,17,25,24,19,33,35,51,96
,16,22,192,136,240,8,2320
1330 DATA 152,192,128,240,18,76,127,64
,138,74,74,74,74,170,152,157,6975
1340 DATA 192,5,96,162,1,134,17,72,32,
140,64,104,168,96,72,134,4374
1350 DATA 161,132,162,168,169,0,153,19
2,5,168,177,161,141,0,5,168,6585
1360 DATA 200,169,155,208,2,177,161,15
3,0,5,136,208,248,104,162,0,7770
1370 DATA 160,5,32,91,64,76,218,65,134
,161,170,164,161,165,183,32,9136
1380 DATA 100,64,76,218,65,32,14,64,76
,218,65,134,161,170,164,161,8531
1390 DATA 165,183,32,22,64,76,218,65,3
2,28,64,132,160,189,72,3,3637
1400 DATA 240,3,56,233,1,160,0,145,165
,164,160,96,134,162,170,164,92
1410 DATA 162,165,183,72,169,255,133,1
63,104,72,134,161,132,162,160,0,8795
1420 DATA 165,163,145,161,104,164,162,
32,72,66,76,218,65,162,7,134,6006
1430 DATA 163,10,10,10,10,170,165,163,
157,66,3,169,0,157,72,3,2672
1440 DATA 157,73,3,152,32,86,228,133,1
60,76,218,65,169,155,170,165,631
1450 DATA 183,134,161,164,161,162,11,7
6,127,66,160,155,208,247,32,38,8175
1460 DATA 64,76,218,65,133,212,134,213
,32,170,217,32,230,216,160,255,4422
1470 DATA 162,0,200,232,177,243,157,80
,5,16,247,73,128,157,80,5,6182
1480 DATA 142,80,5,96,162,0,32,180,66,
165,183,162,80,160,5,32,4723
1490 DATA 22,64,76,218,65,162,0,32,214
,66,76,156,66,160,0,133,4813
1500 DATA 160,138,132,162,166,162,32,1
80,66,165,160,76,219,66,160,0,7533
1510 DATA 32,239,66,165,160,76,170,66,
134,162,170,164,162,165,183,192,2749
1520 DATA 0,16,22,72,134,161,132,162,1
60,45,32,165,66,56,169,0,4161
1530 DATA 229,161,170,169,0,229,162,16
8,104,76,239,66,32,8,67,76,4985
1540 DATA 156,66,32,15,67,165,160,76,1
70,66,134,162,132,163,162,0,7073
1550 DATA 164,162,132,162,32,180,66,20
0,185,80,5,145,162,136,16,248,8862
1560 DATA 96,224,0,16,237,133,160,134,
161,132,162,56,169,0,229,160,9758
1570 DATA 168,169,0,229,161,170,152,32
,180,66,232,138,168,185,79,5,8664
1580 DATA 145,162,136,208,248,138,145,
162,200,169,45,145,162,96,165,183,2246
1590 DATA 162,19,142,80,5,162,80,160,5
,32,119,66,169,80,162,5,3666
1600 DATA 133,164,134,165,160,0,132,16
0,132,161,132,162,177,164,133,163,1521
1610 DATA 230,163,169,32,200,209,164,2
08,5,200,196,163,48,247,177,164,3385
1620 DATA 201,45,208,3,133,162,200,196
,163,16,54,177,164,201,48,48,7931
1630 DATA 48,201,58,16,44,56,233,48,17
0,165,161,72,165,160,10,38,5847
1640 DATA 161,10,38,161,24,101,160,133
,160,104,101,161,133,161,6,160,7645
1650 DATA 38,161,24,138,101,160,133,16
0,144,2,230,161,200,196,163,48,645
1660 DATA 202,165,162,240,13,56,169,0,
229,160,133,160,169,0,229,161,474
1670 DATA 133,161,96,133,164,134,165,1
69,4,133,166,169,36,32,158,66,6638
1680 DATA 169,0,162,4,6,164,38,165,42,
202,208,248,105,48,201,58,8593
1690 DATA 48,2,105,6,32,158,66,198,166
,208,229,96,133,192,134,193,1995
1700 DATA 140,240,5,160,0,177,192,133,
194,230,194,162,13,181,162,157,2214
1710 DATA 240,5,202,208,248,134,139,13
4,138,230,138,164,138,196,194,176,4779
1720 DATA 218,177,192,201,37,208,15,23
0,138,200,177,192,201,37,240,6,1370
1730 DATA 201,69,208,8,169,155,32,158,
66,76,73,68,164,139,230,139,8713
1740 DATA 230,139,133,160,185,240,5,19
0,241,5,164,160,192,67,240,230,3864
1750 DATA 192,83,208,6,32,59,66,76,73,
```

```
68,192,73,208,6,32,8,2061
1760 DATA 67,76,73,68,192,72,208,6,32,
3,68,76,73,68,32,214,3149
1770 DATA 66,76,73,68,134,161,132,162,
10,10,10,10,170,169,38,157,4413
1780 DATA 66,3,32,86,228,32,218,65,160
,0,189,78,3,145,163,189,7663
1790 DATA 76,3,145,161,189,77,3,200,14
5,161,96,134,161,10,10,10,4101
1800 DATA 10,170,152,157,77,3,165,161,
157,76,3,165,163,157,78,3,5801
1810 DATA 169,37,157,66,3,32,86,228,76
,218,65,2,83,58,235,68,5528
1820 DATA 2,69,58,240,68,72,169,0,32,5
3,66,169,12,133,163,169,5788
1830 DATA 0,174,243,68,172,244,68,32,2
54,65,169,6,32,53,66,104,4914
1840 DATA 133,164,41,48,73,28,133,163,
169,6,174,238,68,172,239,68,9700
1850 DATA 76,254,65,133,91,134,92,132,
90,133,85,134,86,132,84,96,6565
1860 DATA 32,41,69,173,253,2,141,251,2
,173,238,68,133,165,173,239,2785
1870 DATA 68,133,166,169,0,133,163,133
,164,169,6,96,32,48,69,160,5448
1880 DATA 17,76,174,66,32,35,69,169,6,
76,125,66,32,41,69,169,2750
1890 DATA 6,174,253,2,76,161,66,201,5,
16,22,133,160,152,41,15,3533
1900 DATA 133,162,138,10,10,10,10,5,16
2,166,160,157,196,2,157,22,5076
1910 DATA 208,96,32,48,69,160,18,76,17
4,66,174,10,210,201,0,240,8281
1920 DATA 9,134,132,162,0,134,133,32,1
5,65,134,160,96,10,132,162,5371
1930 DATA 168,201,7,48,5,160,100,32,12
7,64,138,153,0,210,165,162,7798
1940 DATA 10,10,10,10,5,163,153,1,210,
96,173,50,2,41,239,141,5916
1950 DATA 50,2,141,15,210,169,0,162,8,
157,0,210,202,16,250,96,8145
1960 DATA 170,189,112,2,133,160,96,162
,0,201,4,48,3,232,41,3,3025
1970 DATA 168,189,0,211,57,234,69,133,
160,96,4,8,64,128,162,0,4190
1980 DATA 201,2,48,3,232,41,1,168,189,
0,211,136,208,4,74,74,5806
1990 DATA 74,74,41,15,133,160,96,170,1
89,16,208,133,160,96,133,162,9808
2000 DATA 134,163,160,0,177,162,133,16
0,200,177,162,133,161,96,133,160,1948
2010 DATA 134,161,152,160,0,145,160,96
,32,30,70,200,165,163,145,160,9240
2020 DATA 96,72,169,0,133,164,104,133,
160,134,161,132,162,160,0,165,9329
2030 DATA 164,166,163,240,16,145,160,2
00,208,251,230,161,198,163,208,245,838
5
2040 DATA 240,3,145,160,200,196,162,20
8,249,96,133,160,134,161,132,162,3487
2050 DATA 160,0,165,165,240,22,177,162
,145,160,200,208,249,230,161,230,7385
2060 DATA 163,198,165,208,241,240,5,17
7,162,145,160,200,196,164,208,247,7026
2070 DATA 96,133,164,134,165,132,162,1
60,0,132,160,132,161,177,164,209,2530
2080 DATA 162,240,3,32,169,70,201,0,20
8,1,96,133,166,200,177,164,302
2090 DATA 209,162,208,5,196,166,144,24
5,96,162,255,134,160,144,3,177,2081
2100 DATA 162,232,134,161,96,133,160,1
34,161,132,162,160,0,177,162,145,941
2110 DATA 160,240,8,168,177,162,145,16
0,136,208,249,96,133,160,134,161,3348
2120 DATA 132,162,160,0,177,162,197,16
5,176,2,133,165,198,164,24,165,529
2130 DATA 162,101,164,133,162,144,2,23
0,163,56,165,165,229,164,176,2,813
2140 DATA 169,0,76,191,70,133,160,134,
161,132,162,160,0,177,162,240,1860
2150 DATA 13,133,166,198,164,56,165,16
5,229,164,240,2,176,1,96,170,177
2160 DATA 197,166,144,8,24,165,166,170
,101,164,133,165,165,165,209,160,2927
2170 DATA 144,3,145,160,24,165,160,101
,164,133,160,144,2,230,161,138,596
2180 DATA 76,195,70,110,83,92,83,2,3,3
,1,1,1,0,0,128,7001
2190 DATA 1,1,1,2,2,0,0,55,71,2,2,3,2,
1,1,0,3426
2200 DATA 0,128,128,128,128,2,3,128,12
8,73,71,80,0,58,128,15,1644
2210 DATA 76,99,71,32,154,65,51,71,3,1
69,0,141,92,71,173,52,3936
2220 DATA 2,141,91,71,173,91,71,41,255
,141,91,71,173,92,71,41,6113
2230 DATA 0,141,92,71,173,53,2,141,93,
71,165,87,141,95,71,169,6735
2240 DATA 40,205,91,71,169,0,237,92,71
,176,3,76,200,71,24,173,6553
2250 DATA 91,71,105,227,141,91,71,173,
92,71,105,0,141,92,71,169,6421
2260 DATA 255,205,91,71,169,0,237,92,7
1,144,3,76,200,71,160,0,5740
2270 DATA 140,92,71,169,255,141,91,71,
169,90,205,91,71,169,0,237,9658
2280 DATA 92,71,176,3,76,225,71,160,0,
140,92,71,169,90,141,91,6853
2290 DATA 71,56,173,91,71,233,90,141,9
1,71,173,92,71,233,0,141,7844
2300 DATA 92,71,169,159,205,91,71,169,
0,237,92,71,144,3,76,11,4561
2310 DATA 72,160,0,140,92,71,169,159,1
41,91,71,24,173,71,71,109,5903
2320 DATA 95,71,133,174,173,72,71,105,
0,133,175,160,0,177,174,141,8805
2330 DATA 94,71,173,94,71,73,128,240,3
,76,53,72,14,91,71,46,2561
2340 DATA 92,71,76,75,72,173,94,71,133
,132,173,92,71,170,173,91,8604
2350 DATA 71,32,165,64,141,91,71,138,1
41,92,71,56,173,93,71,233,8074
2360 DATA 17,141,93,71,169,127,205,93,
71,144,3,76,99,72,160,0,4727
2370 DATA 140,93,71,169,95,205,93,71,1
44,3,76,114,72,169,95,141,7022
2380 DATA 93,71,24,173,89,71,109,95,71
,133,174,173,90,71,105,0,5471
2390 DATA 133,175,160,0,177,174,141,94
,71,173,94,71,73,128,240,3,7665
2400 DATA 76,153,72,14,93,71,76,169,72
,173,94,71,133,132,173,93,7753
2410 DATA 71,162,0,32,165,64,141,93,71
,173,51,71,133,174,173,52,7247
2420 DATA 71,133,175,173,92,71,160,1,1
45,174,173,91,71,136,145,174,9814
2430 DATA 173,53,71,133,174,173,54,71,
133,175,173,93,71,145,174,96,9373
2440 DATA 76,211,72,160,0,132,20,132,1
9,96,53,12,76,223,72,169,5534
2450 DATA 1,133,133,169,0,133,132,165,
19,162,0,32,15,65,133,174,4893
2460 DATA 138,133,175,24,165,174,101,2
0,141,218,72,165,175,105,0,141,8443
2470 DATA 219,72,173,219,72,133,161,17
3,218,72,133,160,96,80,0,58,7258
2480 DATA 0,101,0,76,22,73,32,154,65,1
3,73,2,160,0,132,77,1824
2490 DATA 173,13,73,201,1,173,14,73,23
3,0,144,3,76,54,73,140,4233
2500 DATA 14,73,200,140,13,73,169,158,
205,13,73,169,0,237,14,73,6272
2510 DATA 144,3,76,79,73,160,0,140,14,
73,169,158,141,13,73,173,6138
2520 DATA 15,73,201,1,144,3,76,94,73,1
60,1,140,15,73,169,190,6050
```

```
2530 DATA 205,15,73,144,3,76,109,73,16
9,190,141,15,73,56,173,13,5066
2540 DATA 73,233,1,133,160,173,14,73,2
33,0,133,161,172,15,73,166,7823
2550 DATA 161,165,160,32,92,69,24,173,
13,73,105,1,133,160,173,14,4877
2560 DATA 73,105,0,133,161,172,15,73,1
66,161,165,160,32,92,69,56,6375
2570 DATA 173,15,73,233,1,133,162,164,
162,174,14,73,173,13,73,32,5439
2580 DATA 92,69,24,173,15,73,105,1,133
,162,164,162,174,14,73,173,7716
2590 DATA 13,73,32,92,69,169,16,141,16
,73,169,15,133,163,160,8,5264
2600 DATA 162,100,169,0,32,157,69,160,
0,140,18,73,140,17,73,169,4665
2610 DATA 200,205,17,73,169,0,237,18,7
3,176,3,76,249,73,238,17,7674
2620 DATA 73,208,236,238,18,73,76,223,
73,173,16,73,208,3,76,61,5914
2630 DATA 74,56,173,16,73,233,1,141,16
,73,173,16,73,133,163,160,7082
2640 DATA 8,162,140,169,0,32,157,69,16
0,0,140,18,73,140,17,73,3439
2650 DATA 169,100,205,17,73,169,0,237,
18,73,176,3,76,58,74,238,6559
2660 DATA 17,73,208,236,238,18,73,76,3
2,74,76,249,73,169,0,133,7103
2670 DATA 163,160,0,162,0,169,0,32,157
,69,96,236,255,23,10,76,6065
2680 DATA 82,74,142,76,74,141,75,74,16
0,1,140,77,74,136,140,78,6101
2690 DATA 74,169,0,205,75,74,169,0,237
,76,74,48,3,76,160,74,4894
2700 DATA 56,173,75,74,237,77,74,141,7
5,74,173,76,74,233,0,141,7626
2710 DATA 76,74,24,173,77,74,105,2,141
,77,74,173,75,74,201,0,5233
2720 DATA 173,76,74,233,0,16,3,76,157,
74,238,78,74,76,97,74,5296
2730 DATA 173,78,74,133,160,96,1,79,0,
48,76,173,74,141,166,74,5830
2740 DATA 160,0,132,77,24,173,221,63,1
09,166,74,133,174,173,222,63,886
2750 DATA 105,0,133,175,169,0,141,168,
74,177,174,141,167,74,24,173,9507
2760 DATA 229,63,109,166,74,133,174,17
3,230,63,105,0,133,175,177,174,1349
2770 DATA 141,169,74,56,173,167,74,233
,26,133,160,173,168,74,233,0,59
2780 DATA 133,161,56,173,169,74,233,26
,133,162,164,162,166,161,165,160,3235
2790 DATA 32,92,69,24,173,167,74,105,2
6,133,160,173,168,74,105,0,6729
2800 DATA 133,161,56,173,169,74,233,26
,133,162,164,162,166,161,165,160,3255
2810 DATA 32,76,69,24,173,167,74,105,2
6,133,160,173,168,74,105,0,6717
2820 DATA 133,161,24,173,169,74,105,26
,133,162,164,162,166,161,165,160,2283
2830 DATA 32,76,69,56,173,167,74,233,2
6,133,160,173,168,74,233,0,9809
2840 DATA 133,161,24,173,169,74,105,26
,133,162,164,162,166,161,165,160,2303
2850 DATA 32,76,69,56,173,167,74,233,2
6,133,160,173,168,74,233,0,9829
2860 DATA 133,161,56,173,169,74,233,26
,133,162,164,162,166,161,165,160,3315
2870 DATA 32,76,69,173,253,2,208,3,76,
221,75,160,0,140,169,74,8028
2880 DATA 169,15,205,169,74,176,3,76,2
08,75,173,169,74,133,163,160,807
2890 DATA 10,162,60,169,0,32,157,69,16
0,0,140,168,74,140,167,74,7530
2900 DATA 169,200,205,167,74,169,0,237
,168,74,176,3,76,202,75,238,1005
2910 DATA 167,74,208,236,238,168,74,76
,176,75,238,169,74,76,144,75,483
2920 DATA 169,0,133,163,160,0,162,0,16
9,0,32,157,69,96,10,0,2222
2930 DATA 76,227,75,142,223,75,141,222
,75,173,222,75,201,0,173,223,3104
2940 DATA 75,233,0,48,3,76,8,76,56,169
,0,237,222,75,133,160,8337
2950 DATA 169,0,237,223,75,133,161,96,
173,223,75,133,161,173,222,75,3043
2960 DATA 133,160,96,6,30,17,0,18,0,24
4,255,244,255,23,0,18,6219
2970 DATA 0,17,76,37,76,160,0,132,77,2
00,140,253,2,169,5,141,7768
2980 DATA 20,76,169,25,205,20,76,176,3
,76,103,79,169,0,141,22,4376
2990 DATA 76,173,20,76,141,21,76,160,0
,140,24,76,140,23,76,140,4517
3000 DATA 26,76,140,25,76,173,21,76,20
5,23,76,173,22,76,237,24,6147
3010 DATA 76,16,3,76,91,79,24,173,25,7
6,109,23,76,133,174,173,6600
3020 DATA 26,76,109,24,76,133,175,24,1
65,174,109,23,76,133,172,165,8986
3030 DATA 175,109,24,76,133,173,24,165
,172,105,1,141,29,76,165,173,7975
3040 DATA 105,0,141,30,76,56,173,29,76
,237,21,76,133,174,173,30,7284
3050 DATA 76,237,22,76,133,175,56,165,
174,237,21,76,133,172,165,175,1888
3060 DATA 237,22,76,133,173,24,165,172
,105,1,141,27,76,165,173,105,8044
3070 DATA 0,141,28,76,173,22,76,141,32
,76,173,21,76,141,31,76,4243
3080 DATA 173,23,76,141,33,76,160,0,14
0,19,76,169,5,205,19,76,4582
3090 DATA 176,3,76,239,78,24,173,221,6
3,109,19,76,133,174,173,222,1059
3100 DATA 63,105,0,133,175,24,160,0,17
7,174,109,31,76,133,160,169,8902
3110 DATA 0,109,32,76,133,161,24,173,2
29,63,109,19,76,133,174,173,9257
3120 DATA 230,63,105,0,133,175,24,177,
174,109,33,76,133,162,164,162,70
3130 DATA 166,161,165,160,32,92,69,24,
173,221,63,109,19,76,133,174,7998
3140 DATA 173,222,63,105,0,133,175,24,
160,0,177,174,109,33,76,133,7203
3150 DATA 160,169,0,133,161,24,173,229
,63,109,19,76,133,174,173,230,1390
3160 DATA 63,105,0,133,175,24,177,174,
109,31,76,133,162,164,162,166,826
3170 DATA 161,165,160,32,92,69,24,173,
221,63,109,19,76,133,174,173,8969
3180 DATA 222,63,105,0,133,175,24,160,
0,177,174,109,33,76,133,160,8046
3190 DATA 169,0,133,161,24,173,229,63,
109,19,76,133,174,173,230,63,412
3200 DATA 105,0,133,175,56,177,174,237
,31,76,133,162,164,162,166,161,2772
3210 DATA 165,160,32,92,69,24,173,221,
63,109,19,76,133,174,173,222,717
3220 DATA 63,105,0,133,175,24,160,0,17
7,174,109,31,76,133,160,169,9022
3230 DATA 0,109,32,76,133,161,24,173,2
29,63,109,19,76,133,174,173,9377
3240 DATA 230,63,105,0,133,175,56,177,
174,237,33,76,133,162,164,162,1694
3250 DATA 166,161,165,160,32,92,69,24,
173,221,63,109,19,76,133,174,8118
3260 DATA 173,222,63,105,0,133,175,56,
160,0,177,174,237,31,76,133,9215
3270 DATA 160,169,0,237,32,76,133,161,
24,173,229,63,109,19,76,133,7723
3280 DATA 174,173,230,63,105,0,133,175
,56,177,174,237,33,76,133,162,710
3290 DATA 164,162,166,161,165,160,32,9
2,69,24,173,221,63,109,19,76,6927
3300 DATA 133,174,173,222,63,105,0,133
,175,56,160,0,177,174,237,33,9912
3310 DATA 76,133,160,169,0,133,161,24,
173,229,63,109,19,76,133,174,8863
```

```
3320 DATA 173,230,63,105,0,133,175,56,
177,174,237,31,76,133,162,164,1249
3330 DATA 162,166,161,165,160,32,92,69
,24,173,221,63,109,19,76,133,7239
3340 DATA 174,173,222,63,105,0,133,175
,56,160,0,177,174,237,33,76,9153
3350 DATA 133,160,169,0,133,161,24,173
,229,63,109,19,76,133,174,173,9839
3360 DATA 230,63,105,0,133,175,24,177,
174,109,31,76,133,162,164,162,288
3370 DATA 166,161,165,160,32,92,69,24,
173,221,63,109,19,76,133,174,8238
3380 DATA 173,222,63,105,0,133,175,56,
160,0,177,174,237,31,76,133,9335
3390 DATA 160,169,0,237,32,76,133,161,
24,173,229,63,109,19,76,133,7843
3400 DATA 174,173,230,63,105,0,133,175
,24,177,174,109,33,76,133,162,9006
3410 DATA 164,162,166,161,165,160,32,9
2,69,238,19,76,76,219,76,173,9869
3420 DATA 30,76,141,26,76,173,29,76,14
1,25,76,238,23,76,208,3,6100
3430 DATA 238,24,76,174,28,76,173,27,7
6,32,224,75,24,165,160,105,7733
3440 DATA 0,133,174,165,161,105,0,133,
175,165,175,72,165,174,72,174,1846
3450 DATA 30,76,173,29,76,32,224,75,10
4,133,174,104,133,175,165,174,1873
3460 DATA 197,160,165,175,229,161,48,3
,76,88,79,173,28,76,141,26,6111
3470 DATA 76,173,27,76,141,25,76,56,17
3,21,76,233,1,141,21,76,5029
3480 DATA 173,22,76,233,0,141,22,76,76
,85,76,24,173,20,76,105,4472
3490 DATA 5,141,20,76,76,50,76,96,80,0
,58,0,1,101,0,100,506
3500 DATA 0,10,0,76,118,79,32,154,65,1
04,79,4,24,173,221,63,5943
3510 DATA 109,108,79,133,174,173,222,6
3,105,0,133,175,169,0,141,114,9214
3520 DATA 79,160,0,177,174,141,113,79,
173,105,79,141,110,79,173,104,9729
3530 DATA 79,141,109,79,56,173,109,79,
237,113,79,141,109,79,173,110,9949
3540 DATA 79,237,114,79,141,110,79,174
,110,79,173,109,79,32,224,75,9087
3550 DATA 165,161,141,110,79,165,160,1
41,109,79,24,173,229,63,109,108,9866
3560 DATA 79,133,174,173,230,63,105,0,
133,175,169,0,141,114,79,160,9362
3570 DATA 0,177,174,141,113,79,173,107
,79,141,112,79,173,106,79,141,9591
3580 DATA 111,79,56,173,111,79,237,113
,79,141,111,79,173,112,79,237,1385
3590 DATA 114,79,141,112,79,174,112,79
,173,111,79,32,224,75,165,161,521
3600 DATA 141,112,79,165,160,141,111,7
9,173,110,79,133,133,173,109,79,89
3610 DATA 133,132,173,110,79,170,173,1
09,79,32,15,65,141,109,79,138,7192
3620 DATA 141,110,79,173,112,79,133,13
3,173,111,79,133,132,173,112,79,153
3630 DATA 170,173,111,79,32,15,65,141,
111,79,138,141,112,79,24,173,7317
3640 DATA 109,79,109,111,79,141,109,79
,173,110,79,109,112,79,141,110,8585
3650 DATA 79,173,109,79,201,113,173,11
0,79,233,2,48,3,76,134,80,6524
3660 DATA 174,110,79,173,109,79,32,79,
74,169,0,141,114,79,165,160,8529
3670 DATA 141,113,79,76,144,80,160,0,1
40,114,79,169,26,141,113,79,7466
3680 DATA 173,114,79,133,161,173,113,7
9,133,160,96,76,158,80,169,0,8590
3690 DATA 32,245,68,160,4,162,2,169,19
8,32,30,70,76,201,80,25,6088
3700 DATA 83,104,97,114,112,32,83,104,
111,111,116,101,114,44,32,98,5646
3710 DATA 121,32,77,97,116,42,82,97,11
6,162,80,169,175,32,40,66,6647
3720 DATA 76,238,80,26,40,99,41,32,49,
57,56,57,44,32,65,110,2019
3730 DATA 97,108,111,103,32,67,111,109
,112,117,116,105,110,103,162,80,8295
3740 DATA 169,211,32,40,66,76,249,80,0
,162,80,169,248,32,40,66,7612
3750 DATA 76,37,81,33,67,111,110,110,1
01,99,116,32,76,105,103,104,6152
3760 DATA 116,32,71,117,110,32,102,105
,114,115,116,32,103,97,109,101,6701
3770 DATA 32,112,111,114,116,162,81,16
9,3,32,40,66,76,48,81,0,2740
3780 DATA 162,81,169,47,32,40,66,76,89
,81,30,80,114,101,115,115,5631
3790 DATA 32,212,210,201,199,199,197,2
10,32,102,111,114,32,78,69,88,8776
3800 DATA 84,32,103,97,109,101,32,111,
114,162,81,169,58,32,40,66,5331
3810 DATA 76,133,81,33,116,121,112,101
,32,32,197,211,195,193,208,197,4241
3820 DATA 160,32,107,101,121,32,116,11
1,32,101,120,105,116,32,112,114,6604
3830 DATA 111,103,114,97,109,162,81,16
9,99,32,40,66,173,120,2,73,5883
3840 DATA 15,240,10,173,252,2,73,28,24
0,3,76,140,81,173,120,2,7077
3850 DATA 73,14,240,10,173,252,2,73,28
,240,3,76,157,81,96,0,5898
3860 DATA 1,0,76,181,81,160,0,140,175,
81,169,5,205,175,81,176,748
3870 DATA 3,76,211,81,173,175,81,174,1
75,81,157,231,63,238,175,81,3812
3880 DATA 76,186,81,160,0,140,175,81,1
69,5,205,175,81,176,3,76,8628
3890 DATA 27,82,169,6,32,138,69,165,16
0,141,177,81,173,177,81,77,346
3900 DATA 175,81,208,3,76,226,81,174,1
75,81,189,231,63,141,176,81,2533
3910 DATA 174,177,81,189,231,63,174,17
5,81,157,231,63,173,176,81,174,3896
3920 DATA 177,81,157,231,63,238,175,81
,76,216,81,96,4,0,76,33,5877
3930 DATA 82,160,0,140,28,82,169,15,20
5,28,82,176,3,76,164,82,6841
3940 DATA 56,169,15,237,28,82,133,163,
160,10,162,60,169,0,32,157,7425
3950 DATA 69,56,169,15,237,28,82,133,1
63,160,10,162,64,169,1,32,6535
3960 DATA 157,69,160,0,140,29,82,169,2
50,205,29,82,176,3,76,103,8256
3970 DATA 82,238,29,82,76,87,82,56,169
,15,237,28,82,133,163,160,9414
3980 DATA 10,162,80,169,0,32,157,69,56
,169,15,237,28,82,133,163,8391
3990 DATA 160,10,162,84,169,0,32,157,6
9,160,0,140,29,82,169,250,9278
4000 DATA 205,29,82,176,3,76,158,82,23
8,29,82,76,142,82,238,28,8704
4010 DATA 82,76,38,82,169,0,133,163,16
0,0,162,0,169,0,32,157,6177
4020 DATA 69,169,0,133,163,160,0,162,0
,169,1,32,157,69,96,16,4818
4030 DATA 251,76,196,82,160,0,140,191,
82,169,15,205,191,82,176,3,29
4040 DATA 76,65,83,56,169,15,237,191,8
2,133,163,160,8,162,240,169,3298
4050 DATA 0,32,157,69,173,191,82,133,1
63,160,6,162,245,169,1,32,9665
4060 DATA 157,69,160,0,140,192,82,169,
250,205,192,82,176,3,76,7,9591
4070 DATA 83,238,192,82,76,247,82,56,1
69,15,237,191,82,133,163,160,2920
4080 DATA 8,162,180,169,0,32,157,69,17
3,191,82,133,163,160,12,162,567
4090 DATA 194,169,0,32,157,69,160,0,14
0,192,82,169,250,205,192,82,3491
4100 DATA 176,3,76,59,83,238,192,82,76
```

```
,43,83,238,191,82,76,201,1459
4110 DATA 82,169,0,133,163,160,0,162,0
,169,0,32,157,69,169,0,5749
4120 DATA 133,163,160,0,162,0,169,1,32
,157,69,96,58,11,0,6,1833
4130 DATA 10,0,12,124,3,57,0,3,0,29,4,
6,0,33,20,0,6221
4140 DATA 76,115,83,160,0,140,108,83,2
00,140,107,83,136,140,109,83,9659
4150 DATA 140,106,83,140,105,83,32,155
,80,169,31,32,245,68,169,12,7797
4160 DATA 141,200,2,160,0,140,98,83,14
0,100,83,140,99,83,140,102,8571
4170 DATA 83,140,101,83,140,104,83,140
,103,83,32,34,76,32,208,72,6418
4180 DATA 32,178,81,160,0,140,94,83,14
0,95,83,169,5,205,95,83,8452
4190 DATA 176,3,76,70,85,169,3,141,253
,2,160,1,140,93,83,169,8608
4200 DATA 10,205,93,83,176,3,76,64,85,
174,94,83,189,231,63,133,472
4210 DATA 160,165,160,32,170,74,169,83
,133,163,160,92,162,83,169,110,1703
4220 DATA 32,96,71,173,120,2,73,15,240
,10,173,252,2,73,28,240,9087
4230 DATA 3,76,230,83,173,252,2,73,28,
240,3,76,98,84,169,255,1044
4240 DATA 141,252,2,169,0,32,245,68,76
,53,84,25,83,104,97,114,6270
4250 DATA 112,32,83,104,111,111,116,10
1,114,44,32,98,121,32,77,97,5654
4260 DATA 116,42,82,97,116,162,84,169,
27,32,40,66,76,90,84,26,4305
4270 DATA 40,99,41,32,49,57,56,57,44,3
2,65,110,97,108,111,103,5031
4280 DATA 32,67,111,109,112,117,116,10
5,110,103,162,84,169,63,32,40,7138
4290 DATA 66,96,238,253,2,169,3,205,25
3,2,144,3,76,116,84,160,9308
4300 DATA 1,140,253,2,172,92,83,174,11
1,83,173,110,83,32,19,73,6765
4310 DATA 169,0,133,163,174,94,83,189,
231,63,133,164,172,92,83,174,2750
4320 DATA 111,83,173,110,83,32,115,79,
165,161,141,97,83,165,160,141,1455
4330 DATA 96,83,169,5,205,96,83,169,0,
237,97,83,16,3,76,183,7404
4340 DATA 84,238,98,83,32,30,82,169,25
,205,96,83,169,0,237,97,9423
4350 DATA 83,16,3,76,241,84,238,101,83
,208,3,238,102,83,56,169,709
4360 DATA 26,237,96,83,133,174,169,0,2
37,97,83,133,175,24,173,99,774
4370 DATA 83,101,174,141,99,83,173,100
,83,101,175,141,100,83,76,252,1753
4380 DATA 84,238,103,83,208,3,238,104,
83,32,193,82,173,120,2,73,8438
4390 DATA 14,240,3,76,252,84,160,0,140
,253,2,174,94,83,189,231,2896
4400 DATA 63,133,160,165,160,32,170,74
,238,94,83,169,5,205,94,83,339
4410 DATA 144,3,76,45,85,160,0,140,94,
83,32,178,81,169,3,32,5613
4420 DATA 138,69,24,165,160,105,1,141,
253,2,238,93,83,76,207,83,600
4430 DATA 238,95,83,76,187,83,32,220,7
2,165,161,141,111,83,165,160,2229
4440 DATA 141,110,83,169,0,133,133,169
,60,133,132,173,111,83,170,173,2128
4450 DATA 110,83,32,68,65,141,110,83,1
38,141,111,83,173,110,83,201,818
4460 DATA 16,173,111,83,233,14,144,3,7
6,151,85,173,111,83,133,133,9701
4470 DATA 173,110,83,133,132,169,14,17
0,169,16,32,68,65,141,110,83,7422
4480 DATA 138,141,111,83,76,159,85,160
,0,140,111,83,140,110,83,169,9700
4490 DATA 255,141,252,2,169,0,32,245,6
8,160,20,162,2,169,198,32,9070
4500 DATA 30,70,160,2,162,0,169,2,32,3
5,69,76,215,85,24,83,5149
4510 DATA 104,97,114,112,32,83,104,111
,111,116,101,114,32,66,121,32,6177
4520 DATA 77,97,116,42,82,97,116,162,8
5,169,190,32,40,66,76,248,9888
4530 DATA 85,22,70,105,110,97,108,32,8
3,99,111,114,101,32,102,111,6826
4540 DATA 114,32,82,111,117,110,100,32
,162,85,169,225,32,59,66,174,9492
4550 DATA 108,83,173,107,83,32,231,66,
238,107,83,208,3,238,108,83,1463
4560 DATA 76,35,86,15,66,117,108,108,1
01,116,115,47,77,105,110,32,6207
4570 DATA 32,61,32,162,86,169,19,32,59
,66,174,111,83,173,110,83,8217
4580 DATA 32,231,66,76,70,86,15,84,111
,116,97,108,32,72,105,116,6596
4590 DATA 115,32,32,32,61,32,162,86,16
9,54,32,59,66,174,102,83,6585
4600 DATA 173,101,83,32,231,66,76,105,
86,15,84,111,116,97,108,32,6453
4610 DATA 77,105,115,115,101,115,32,61
,32,162,86,169,89,32,59,66,6037
4620 DATA 174,104,83,173,103,83,32,231
,66,76,140,86,15,66,117,108,7556
4630 DATA 108,115,101,121,101,115,32,3
2,32,32,61,32,162,86,169,124,6922
4640 DATA 32,59,66,173,98,83,32,229,66
,76,172,86,15,83,99,111,7620
4650 DATA 114,101,32,32,32,32,32,32,32
,32,61,32,162,86,169,156,6026
4660 DATA 32,59,66,174,100,83,173,99,8
3,32,231,66,173,105,83,205,1349
4670 DATA 99,83,173,106,83,237,100,83,
144,3,76,217,86,173,100,83,213
4680 DATA 141,106,83,173,99,83,141,105
,83,76,236,86,15,65,99,99,8103
4690 DATA 117,114,97,99,121,32,32,32,3
2,32,61,32,162,86,169,220,8027
4700 DATA 32,59,66,169,0,133,133,173,9
8,83,133,132,169,100,162,0,9623
4710 DATA 32,15,65,141,110,83,138,141,
111,83,169,0,133,133,169,60,9447
4720 DATA 133,132,173,111,83,170,173,1
10,83,32,68,65,141,110,83,138,9027
4730 DATA 141,111,83,173,109,83,205,11
0,83,169,0,237,111,83,144,3,9486
4740 DATA 76,57,87,173,110,83,141,109,
83,174,111,83,173,110,83,32,9040
4750 DATA 214,66,76,71,87,1,37,162,87,
169,69,32,40,66,76,82,5116
4760 DATA 87,0,162,87,169,81,32,40,66,
76,108,87,15,72,105,103,5568
4770 DATA 104,32,83,99,111,114,101,32,
32,32,61,32,162,87,169,92,6779
4780 DATA 32,59,66,174,106,83,173,105,
83,32,231,66,76,143,87,15,7838
4790 DATA 66,101,115,116,32,65,99,99,1
17,114,97,99,121,61,32,162,7849
4800 DATA 87,169,127,32,59,66,173,109,
83,32,212,66,76,161,87,1,7262
4810 DATA 37,162,87,169,159,32,40,66,7
6,172,87,0,162,87,169,171,9859
4820 DATA 32,40,66,76,206,87,23,80,114
,101,115,115,32,32,212,242,384
4830 DATA 233,231,231,229,242,160,32,1
16,111,32,112,108,97,121,162,87,1080
4840 DATA 169,182,32,40,66,76,240,87,2
3,79,114,32,116,121,112,101,7924
4850 DATA 160,197,211,195,193,208,197,
160,32,116,111,32,101,120,105,116,1166
4860 DATA 162,87,169,216,32,40,66,173,
120,2,73,14,240,10,173,252,771
4870 DATA 2,73,28,240,3,76,247,87,173,
120,2,73,15,240,10,173,9086
4880 DATA 252,2,73,28,240,3,76,8,88,17
```

```
3,120,2,73,14,240,10,6052
4890 DATA 173,252,2,73,28,240,3,76,25,
88,173,252,2,73,28,240,9414
4900 DATA 3,76,137,83,169,255,141,252,
2,169,0,32,245,68,76,91,1
4910 DATA 88,25,83,104,97,114,112,32,8
3,104,111,111,116,101,114,44,7598
4920 DATA 32,98,121,32,77,97,116,42,82
,97,116,162,88,169,65,32,7679
4930 DATA 40,66,76,128,88,26,40,99,41,
32,49,57,56,57,44,32,2120
4940 DATA 65,110,97,108,111,103,32,67,
111,109,112,117,116,105,110,103,8882
4950 DATA 162,88,169,101,32,40,66,96,9
6,226,2,227,2,112,83,0,6538
```

LISTING 2: ACTION!

```
;
;         SHARP SHOOTER
;     by Matthew J.W. Ratcliff
;
;         COPYRIGHT 1989
;      BY ANALOG COMPUTING
;


;       CHECKSUM DATA
;[48 0B 1E 62 0B 10 10 12
; 78 1A 5B 6E 9E 58 23 EE
; D8 EB E2 41 52 46 14 71
; 0B 3A 83 1F A4 53 18 ]
BYTE RTS=[$60] ; This declaration
               ; must be the first
               ; compiled code if
               ; using this pgm with
               ; OS/A+ versions 2.2
               ; and before

; Game global target array

BYTE ARRAY XCS=[26 79 132 26 79 132]
BYTE ARRAY YCS=[48 48 48 144 144 144]
BYTE ARRAY Tgsel(6)

BYTE Jiffy  = 20
BYTE Jiffy1 = 19
BYTE CH     = 764
BYTE TRIGGER= 632
BYTE Attract= 77

;-------------------------
INCLUDE "D:GUNREAD.ACT"
;-------------------------

PROC ZeroTime()

CARD Timer=19

Timer = 0

RETURN

;-------------------------
; Get elapsed time in jiffies
;
CARD FUNC GetJTime()

CARD tic

tic = Jiffy1*256 + Jiffy

RETURN(tic)
;-------------------------
PROC Blast( CARD xb, BYTE yb )

BYTE s

CARD d

Attract = 0

IF xb < 1 THEN
  xb = 1
FI

IF xb >158 THEN
  xb = 158
FI

IF yb < 1 THEN
  yb = 1
FI

IF yb > 190 THEN
  yb = 190
FI

Plot(xb-1, yb)
Plot(xb+1, yb)
Plot(xb, yb-1)
Plot(xb, yb+1)

s = 16
Sound(0, 100, 8, 15)

FOR d = 0 TO 200
  DO
  ;
  OD
WHILE s # 0
  DO
  s = s - 1
  Sound(0, 140, 8, s)
  FOR d = 0 TO 100
    DO
    ;
    OD
  OD

Sound(0, 0, 0, 0)

RETURN

;-------------------------
; Return the integer square
; ROOT of the value passed.
;
; Algorithm:  The integer square
; root is the count of the total
; successive odd numbers, starting
; from 1, that can be subtracted
; from the parameter before it goes
; negative.

BYTE FUNC ISqrt( INT r )

BYTE i, j

i = 1
j = 0
WHILE (r > 0)
  DO
  r = r - i
  i = i + 2
  IF r >= 0 THEN
    j = j + 1
    FI
  OD

RETURN( j )

;-------------------------

PROC SelTarget( BYTE tg )
```

```
CARD X
BYTE y

Attract = 0

x = XCS(tg)
y = YCS(tg)

Plot(x-26,y-26)
DrawTo(x+26,y-26)
DrawTo(x+26,y+26)
DrawTo(x-26,y+26)
DrawTo(x-26,y-26)

IF color # 0 THEN
  FOR y = 0 TO 15
    DO
    Sound(0, 60, 10, y)
    FOR x = 0 TO 200
      DO
      ;
      OD
    OD
    Sound(0, 0, 0, 0)
  FI

RETURN

;-------------------------
INT FUNC ABS(INT NUMBER)
IF (NUMBER < 0) THEN
 RETURN(-NUMBER)
FI
RETURN(NUMBER)

;-------------------------

PROC GAMESCREEN()

BYTE I,R
INT DX
INT DY
INT PHI, PHIXY, PHIY

CARD X
BYTE Y

Attract = 0
color = 1

FOR R=5 TO 25 STEP 5
  DO
  DX=R
  DY=0
  PHI = 0
  WHILE DX >= DY
    DO
    PHIY = PHI+DY+DY+1
    PHIXY= PHIY-DX-DX+1
    X = DX
    Y = DY
    FOR I=0 TO 5
      DO
      Plot (XCS(I)+X,YCS(I)+Y)
      Plot (XCS(I)+Y,YCS(I)+X)
      Plot (XCS(I)+Y,YCS(I)-X)
      Plot (XCS(I)+X,YCS(I)-Y)
      Plot (XCS(I)-X,YCS(I)-Y)
      Plot (XCS(I)-Y,YCS(I)-X)
      Plot (XCS(I)-Y,YCS(I)+X)
      Plot (XCS(I)-X,YCS(I)+Y)
      OD
    PHI = PHIY
    DY = DY + 1
    IF ABS(PHIXY)+0<ABS(PHIY) THEN
      PHI= PHIXY
      DX = DX-1
      FI
    OD ; WHILE
  OD ; R LOOP
RETURN

; ------------------

INT FUNC GetRadius( INT x, INT y, BYTE
 tg)

INT xx, yy, rr

rr = XCS(tg)
xx = x
xx = xx-rr
xx = Abs(xx)

rr = YCS(tg)
yy = y
yy = yy-rr
yy = Abs(yy)

xx = xx * xx
yy = yy * yy
xx = xx + yy

IF xx < 625 THEN
  rr = ISqrt( xx )
ELSE
  rr = 26
FI

RETURN(rr)

;--------------------
PROC Title( )

Graphics( 0 )
Poke (710, 4)
PrintE("Sharp Shooter, by Mat*Rat")
PrintE("(c) 1989, Analog Computing")
PrintE("")
PrintE("Connect Light Gun first game p
ort")
PrintE("")
PrintE("Press TRIGGER for NEXT game or
")
PrintE("type  ESCAPE  key to exit prog
ram")

DO
  ;
UNTIL TRIGGER=15 OR CH = 28
OD

DO
  ;
UNTIL TRIGGER=14 OR CH = 28
OD

RETURN
;------------------
PROC RandTgts()
BYTE i, y, f



FOR i = 0 TO 5
  DO
  Tgsel(i) = i
  OD
FOR i = 0 TO 5
  DO
    DO
    f = Rand( 6 )
    UNTIL f # i
    OD
  y = Tgsel(i)
  Tgsel(i) = Tgsel(f)
```

```
    Tgsel(f) = y
    OD
  RETURN

  ;--------------------
  PROC BingBong()

  BYTE bi, bo

  FOR bi = 0 TO 15
    DO
    Sound(0, 60, 10, 15-bi)
    Sound(1, 64, 10, 15-bi)
    FOR bo = 0 TO 250
      DO
      ;
      OD
    Sound(0, 80, 10, 15-bi)
    Sound(0, 84, 10, 15-bi)
    FOR bo = 0 TO 250
      DO
      ;
      OD
    OD

  Sound(0, 0, 0, 0)
  Sound(1, 0, 0, 0)
  RETURN
  ;--------------------
  PROC BingBap()

  BYTE bi, ba

  FOR bi = 0 TO 15
    DO
    Sound(0, 240, 8, 15-bi)
    Sound(1, 245, 6, bi)
    FOR ba = 0 TO 250
      DO
      ;
      OD
    Sound(0, 180, 8, 15-bi)
    Sound(0, 194, 12, bi)
    FOR ba = 0 TO 250
      DO
      ;
      OD
    OD

  Sound(0, 0, 0, 0)
  Sound(1, 0, 0, 0)
  RETURN

  ;--------------------

  PROC MAIN()
    BYTE y, f, i, j
    INT radius
    BYTE bulls
    CARD score, hits, misses
    CARD hiscore, round
    BYTE hipct
    CARD x
    BYTE BK=712

    round   = 1
    hipct   = 0
    hiscore = 0
    Title( )
  DO
  ; Until ESCAPE
    Graphics(31)
    BK = 12
    bulls = 0
    score = 0
    hits  = 0
    misses= 0
    GAMESCREEN()
    ZeroTime()
  ; Randomize target selection
    RandTgts()
    i = 0
    FOR j = 0 TO 5
      DO
      color = 3
      FOR f = 1 TO 10
        DO
        SelTarget( tgsel( i ) )
          DO
          GunRead( @x, @y )
          UNTIL TRIGGER=15 OR CH=28
          OD
        IF CH = 28 THEN
          CH=255
          Graphics(0)
          PrintE("Sharp Shooter, by Mat*
  Rat")
          PrintE("(c) 1989, Analog Compu
  ting")
          RETURN
          FI
        color = color + 1
        IF color > 3 THEN
          color = 1
        FI
        Blast( x, y )
        radius = GetRadius(x,y,tgsel(i))
        IF radius <= 5 THEN
          bulls = bulls + 1
          BingBong()
          FI
        IF radius <= 25 THEN
          hits  = hits  + 1
          score = score + (26-radius)
        ELSE
          misses = misses + 1
          BingBap()
          FI
          DO
          UNTIL TRIGGER=14
          OD
        color = 0
        SelTarget( tgsel(i))
        i = i + 1

        IF i > 5 THEN
          i = 0
          RandTgts()
          FI
        color = Rand(3) + 1
        OD
      OD
    x = GetJTime()
    x = x/60
    IF x < 3600 THEN
      x = 3600/x
    ELSE
      x = 0
    FI
    CH = 255
    Graphics(0)
    Poke (710, 20)
    Position(2,2)
    PrintE("Sharp Shooter By Mat*Rat")
    Print("Final Score for Round ")
    PrintCE( round )
    round = round + 1
    Print("Bullets/Min  = ")
    PrintCE( x )
    Print("Total Hits   = ")
    PrintCE( hits )
    Print("Total Misses = ")
    PrintCE( misses )
    Print("Bullseyes    = ")
    PrintBE( bulls )
    Print("Score        = ")
```

```
  PrintCE( score )
  IF score > hiscore THEN
    hiscore = score
    FI
  Print("Accuracy      = ")
  x = 100 * bulls
  x = x/60
  IF x > hipct THEN
    hipct = x
    FI
  PrintC( x )
  PrintE("%")
  PrintE("")
  Print("High Score    = ")
  PrintCE( hiscore )
  Print("Best Accuracy= ")
  PrintB( hipct )
  PrintE("%")
  PrintE("")
  PrintE("Press  Trigger  to play")
  PrintE("Or type ESCAPE  to exit")
  DO
  UNTIL TRIGGER=14 OR CH=28
  OD
  DO
  UNTIL TRIGGER=15 OR CH=28
  OD
  DO
  UNTIL TRIGGER=14 OR CH=28
  OD

UNTIL (CH = 28)
OD

CH=255
Graphics(0)
PrintE("Sharp Shooter, by Mat*Rat")
PrintE("(c) 1989, Analog Computing")
RETURN
```

LISTING 3: ACTION!

```
; GUNREAD.ACT
;
;      CHECKSUM DATA
;[6A BA 3A 75 52 ]

; Read the Atari light gun
; and convert the readings
; of LPENH & LPENV to current
; graphics mode screen coordinates
;
; Algorithm developed by:
; Matthew J. W. Ratcliff
; Ratware Softworks
; (c) 1989
;
; For Analog Computing
;
; Algorithm:
; The DELTA-X gun readings were
; apparently DESIGNED to be 160
; with DELTA-Y at 96.  These values
; work out to be multiples of two,
; by powers of two, for each and
; EVERY possible Atari graphics mode
; 0 through 15 (full screen modes).
;
; The X reading starts at about 89
; at the far left of the display,
; increases to 227 at about text
; column 34, then drops to zero.
; It increases to about 22 at the
; far right of the display.
;
; The Y reading starts at about 17
; at the top to 112 at the bottom.
;
; GunRead normalizes the X reading
; to 0-159, inclusive and Y to a
; range of 0-95.  Then the XSHIFT
; and YSHIFT tables are accessed,
; based on the current graphics mode.
; If the value is less than 128, it
; is a right shift count (divide).
; A value of 128 indicates a single
; left shift (multiply by 2).
;
; The end result is a valid X,Y
; coordinate reading of the light
; gun for the present graphics
; mode.  It is up to the user
; to assure the screen intensity
; (COLOR*16+INTENSITY) is at a level
; to get valid gun readings.  A value
; of at least 10 is recommended.
; A "flash" technique may work best
; Set all playfield intensities to
; 14, call GunRead, and restore the
; original playfield colors.
;

PROC GunRead( CARD POINTER xx,
              BYTE POINTER yy)

BYTE ARRAY xshift=[2 3 3 1 1 1 0 0 128
 1 1 1 2 2 0 0]

BYTE ARRAY yshift=[2 2 3 2 1 1 0 0 128
 128 128 128 2 3 128 128]

CARD GunX
BYTE GunY

BYTE DINDEX= $57
BYTE LPENH = 564
BYTE LPENV = 565

BYTE shift, index


GunX  = LPENH
GunX  = GunX & $FF
GunY  = LPENV
index = DINDEX
IF GunX <= 40 THEN
 GunX = GunX + 227
 IF GunX > 255 THEN
  GunX = 255
 FI
FI

IF GunX <= 90 THEN
  GunX = 90
FI

GunX = GunX - 90

IF GunX > 159 THEN
 GunX = 159
FI

shift = xshift(index)

IF shift = 128 THEN
  GunX = GunX LSH 1
ELSE
  GunX = GunX RSH shift
FI

GunY = GunY - 17


IF GunY > 127 THEN
  GunY = 0
```

continued on page 61

M/L EDITOR

For use in machine-language entry.

by Clayton Walnum

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LISTING 1: BASIC LISTING

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

BASIC Editor II

by Clayton Walnum

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

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

Typing in the Editor

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

Disk version:

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

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

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

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

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

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

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

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

Cassette version:

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

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

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

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

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

(6) Rewind the cassette.

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

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

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

Using the Editor

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

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

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

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

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

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

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

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

Leaving the Editor

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

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

Large listings

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

The post-processor

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

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

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

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

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

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

Murphy's Law

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

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

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

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

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

Listing 1.
BASIC listing.

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

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

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

Listing 2.
BASIC listing.

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

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

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

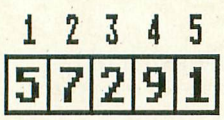
BASIC TRAINING: ARRAYS

by Clayton Walnum

So far we've talked about two kinds of variables: numeric and string. In this installment, I'd like to cover a variation on numeric variables, a powerful data structure known as an array. Arrays can be a little confusing at first, but in order to be a proficient BASIC programmer you must be able to handle them.

Actually, we've already talked a little about arrays, because a string is really nothing more than a character array. Last month, when we talked about strings, I provided this diagram:

Here you can see that we've got a series of values (in this case, character values) stored consecutively in memory. Each character in the string can be identified (indexed) according to its position in the string. Now let's take the above diagram and replace the character values with numeric values:



We have now converted the string—a character array—into a numeric array. Just like the string, each value in the array is identified by a number that represents its position. In other words, we can access the number 2 by referring to the third "element" of the array. This is where things get confusing, because with numeric arrays, we're always working with two numbers: the index (or position) of a value and the value itself.

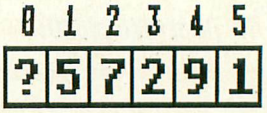
Programming with arrays

Just like strings, array variables have to be dimensioned so BASIC knows how much space to reserve for them. The DIM statement for a numeric array looks almost exactly like the DIM statement for a string. The only difference is we don't end the variable name with a dollar sign. Here's a DIM statement for the array illustrated above:

```
10 DIM NUMBERS(5)
```

This program line tells BASIC that we want to use an array called NUMBERS and that the array will need to store a maximum of six values.

Whoa! *Six* values? Yep. You see, array indexes actually start at zero. The first position of NUMBERS is actually position 0. But because most people tend to think of position 1 as the first position in a series, BASIC programmers (mostly being people too) like to ignore the 0 element and begin with element 1. Let's revise our illustration to show what our array *really* looks like:



I've placed a question mark in element 0 because, unless we've placed a number there, we can't be sure exactly what's stored in that position. Usually it'll be a 0, but it's a good programming practice not to trust the value of any variable we haven't first initialized (given a value to).

Now that we have a name for our array, we can use the indexes to access any values in the array. If the array was set up as illustrated, we could refer to the number 2 with the statement NUMBERS(3). We interpret this as meaning the value stored in the third element (skipping element 0) of NUMBERS. Remember: the number in parentheses is not a value stored in the array, but rather the position of the value we want.

How would we refer to the number 1 in the array NUMBERS? Everyone who said NUMBERS(5) gets a gold star.

Although our illustration shows the array already filled with values, in our program we still have an uninitialized array. To get the values into NUMBERS( ), add the following line to Line 10 from above:

```
20 NUMBERS(1)=5:NUMBERS(2)=7:
NUMBERS(3)=2:NUMBERS(4)=9:NUM
BERS(5)=1
```

This is only one way of getting the job done. It's not the best way, but until we learn about loops, it's the best we can do. In English, Line 20 reads "Place the value 5 into position 1 of the array NUMBERS; place the value 7 into position 2 of the array NUMBERS; place the value 2 into position 3 of the array NUMBERS. . ." etc.

To prove that we have indeed set up our array as shown in the original illustration, add these lines to our program:

```
30 PRINT NUMBERS(1)
40 PRINT NUMBERS(2)
50 PRINT NUMBERS(3)
60 PRINT NUMBERS(4)
70 PRINT NUMBERS(5)
```

If you were to run this program, you would get the following output:

```
80 X=1:PRINT NUMBERS(X)
90 X=2:PRINT NUMBERS(X)
100 X=3:PRINT NUMBERS(X)
110 X=4:PRINT NUMBERS(X)
120 X=5:PRINT NUMBERS(X)
```

An array's index doesn't have to be a constant (an explicit number). It can also be a

variable. For example, add the following lines to our program:

```
5
7
2
9
1
```

Now when you run the program, you'll get the following output:

```
5
7
2
9
1
5
7
2
9
1
```

The first five numbers in the list were printed by Lines 30 through 70, using constants as array indexes. The second five numbers were printed by Lines 80 through 120, using the variable X as the index and changing the value of X each time we used it.

Two-dimensional arrays

The above examples used a one-dimensional array. Unlike strings, though, numeric variables can have two "dimensions." Two-dimensional arrays are sometimes thought of as "tables" or "matrices" because they organize data in much the same way we would if we drew a table of values on a piece of graph paper, like this:

	1	2	3	4	5
1	5	2	4	1	8
2	7	1	6	2	9
3	1	4	7	2	1

To locate a value in a table like this, we need to have two indexes, the column and the row. For example, the value 6 can be found in column 3, row 2. This table is a graphic representation of a two-dimensional array. Let's use the array name TABLE,and dimension the array as a two-dimensional array:

```
10 DIM TABLE(5,3)
```

In the above line, we've told BASIC that we want a two-dimensional array with five columns and three rows (actually, six columns and four rows if we count the 0 elements). If we wanted to refer to the number 6 in the above table, we could call it TABLE(3,2).

Here's a simple way to get the values from the table into our array. Add these lines to Line 10:

```
20 TABLE(1,1)=5:TABLE(2,1)=2:
TABLE(3,1)=4:TABLE(4,1)=1:TAB
LE(5,1)=8
30 TABLE(1,2)=7:TABLE(2,2)=1:
TABLE(3,2)=6:TABLE(4,2)=2:TAB
LE(5,2)=9
40 TABLE(1,3)=1:TABLE(2,3)=4:
TABLE(3,3)=7:TABLE(4,3)=2:TAB
LE(5,3)=1
```

In Line 20 we initialize row 1 of the array, in Line 30 we initialize row 2, and in Line 40 we initialize row 3. As you can see, multi-dimensional arrays are much more complicated than single-dimensional arrays, and they can store a great deal more data.

Just as with a one-dimensional array, we may use numeric variables as indexes for a two-dimensional array. Frequently, the familiar X and Y are used as "coordinates" for the location of a piece of data.

What's the point?

Why are arrays so valuable to us as programmers? Because they allow us to quickly and conveniently access data. For example, suppose we had a class with five students in it. We could store each student's final grade average in an array, then use the student's ID number as an index for finding his grade. The following program illustrates this use of an array:

```
10 DIM GRADES(5)
20 GRADES(1)=88:GRADES(2)=75:
GRADES(3)=92:GRADES(4)=67:GRA
DES(5)=86
30 PRINT "ENTER STUDENT ID NU
MBER"
40 INPUT STUDENT
50 PRINT "STUDENT'S GRADE AVE
RAGE IS ";GRADES(STUDENT)
```

Line 10 dimensions the array GRADES( ). Line 20 initializes GRADES( ), putting each student's grade into one element of the array. Line 30 prompts the user for an ID number, and Line 40 retrieves this number from the keyboard. Line 50 displays on the screen the grade for the student ID entered in Line 40.

When you run this short program, you'll see something like this:

```
ENTER STUDENT ID NUMBER
?3
STUDENT'S GRADE AVERAGE IS 92
```

Just be careful not to enter a number lower than 1 or greater than 5. If you enter a 0, you'll get a value that doesn't mean anything because we never initialized element 0 of the array. If you enter a number greater than 5, you'll get an error because you'll be trying to access an array element that doesn't exist. Try it and see what happens.

Now how about an example of using a two-dimensional array. Let's take an even smaller class, say, three students. Now let's use a two-dimensional array to keep track of all of each student's test scores for the class. Table 1 shows what we find in the teacher's gradebook.

STUDENT	ID	TEST 1	TEST 2	TEST 3
Smith, Bill	1	85	72	92
Stowe, Jane	2	74	78	82
White, Alex	3	91	85	82

Table 1

This program is a computerized version of the gradebook:

```
10 DIM GRADES(3,3)
20 GRADES(1,1)=85:GRADES(2,1)
=72:GRADES(3,1)=92
30 GRADES(1,2)=74:GRADES(2,2)
=78:GRADES(3,2)=82
40 GRADES(1,3)=91:GRADES(2,3)
=85:GRADES(3,3)=82
50 PRINT "ENTER STUDENT ID"
60 INPUT STUDENT
70 PRINT "ENTER TEST #"
80 INPUT TEST
90 PRINT "THE SCORE IS ";GRAD
ES(TEST,STUDENT)
```

Line 10 dimensions the two-dimensional array GRADES( ). Lines 20 through 40 initialize the array with the students' test scores. Line 50 prompts the user for the student ID, and Line 60 retrieves that number from the keyboard. Lines 70 and 80 get the test number in the same way. Finally, Line 90 uses the values retrieved for TEST and STUDENT as indexes for accessing the appropriate score.

A typical run of the above program might look like this:

```
ENTER STUDENT ID
?2
ENTER TEST #
?3
THE SCORE IS 82
```

Some final words

In closing, I should mention that some BASICs allow arrays larger than two dimensions. A three-dimensional array—one that might be dimensioned as ARRAY(5,5,5), for example—can be visualized as a cube, wherein values are located by the column, row and depth at which they reside. Four- and five-dimensional arrays are also possible in some BASICs and other languages, but they are creatures that can reside only in the abstract environment of a computer's memory. There is no real-life table we can create to represent them.

Next time, we'll look at looping techniques.

BOOT CAMP

by Tom Hudson

In this month's *Boot Camp*, we're going to finish our discussion of X and Y register indexing and become proficient in multi-byte addition.

Regular *Boot Camp* readers will be happy to know that the introductory material will be completely covered in the next few isues. After that, we can start applying all the 6502 instructions to useful subroutines and full-scale programs!

Solution #2

I hope everyone at least tried to solve the indexing problem presented last issue. This problem asked readers to write the code necessary to copy the contents of the six-byte TABLE1 to TABLE2 in reverse order. This little brain-teaser is an excellent opportunity to gain more experience with the 6502 index registers.

Below is the code necessary to copy TABLE1 to TABLE2 in normal order. This code was shown last month.

```
10       *=  $600
20       LDX #5
30 COPY  LDA TABLE1,X
40       STA TABLE2,X
50       DEX
60       BPL COPY
70       BRK
80 TABLE1 .BYTE 10,20,30,40,50,60
90 TABLE2 *=*+6
0100      .END
```

In order for BASIC to reconstruct the number, it must multiply each byte by the value of its lowest-order bit.

I told you that only three changes to this code would allow it to copy the table in reverse order. The changed code is shown below.

```
10       *=  $600
20       LDX #5
30       LDY #0
40 COPY  LDA TABLE1,X
50       STA TABLE2,Y
60       INY
70       DEX
80       BPL COPY
90       BRK
0100 TABLE1 .BYTE 10,20,30,40,50,60
0110 TABLE2 *=*+6
0120        .END
```

How does it work? Let's step through the code and see.

Line 20 sets the X register to 5. This register will be used to point to different parts of TABLE1. With the index starting at 5, the register will point to the last byte of TABLE1.

Line 30 sets the Y register to 0. This register will be used to point to varying places in TABLE2. Unlike the X register, the Y register will start pointing at the *first* byte of TABLE2.

Lines 40-80 perform the table-data move function.

Line 40 loads the accumulator with a byte from TABLE1, indicated by the X register.

Line 50 stores the byte just loaded into a byte of TABLE2, indicated by the Y register.

Lines 60 and 70 are the heart of this routine. Note that the Y register is incremented each time the loop is executed, while the X register is decremented. Here are the X and Y register contents for each iteration of the loop.

TABLE1 (X)	TABLE2 (Y)
5	0
4	1
3	2
2	3
1	4
0	5

By looking at the above, you can see that the sixth byte (5+1) of TABLE1 will be moved to the first byte (0+1) of TABLE2, the fifth byte of TABLE1 to the second byte of TABLE2, and so on.

Line 80 loops back to the COPY label if the X register is positive (0-127). Once the X register is decremented past zero, it "wraps around" to binary 11111111, or −1 decimal, and the program stops at the BRK instruction in Line 90.

Line 100 sets up the initial values contained in TABLE1.

Line 110 tells the assembler to reserve six bytes for TABLE2. Remember, the "*=*+" directive allows you to set aside any number of bytes for tables, working areas, etc.

As a further example of the "reverse table" problem, below is the BASIC equivalent of the assembly code.

```
10 DIM TABLE1(5),TABLE2(5)
15 TABLE1(0)=10:TABLE1(1)=20:TABLE1(2)
=30:TABLE1(3)=40:TABLE1(4)=50:TABLE1(5
)=60
20 X=5
30 Y=0
40 A=TABLE1(X)
50 TABLE2(Y)=A
60 Y=Y+1
70 X=X-1
80 IF X>=0 THEN 40
90 END
```

Note that in BASIC it is necessary to initialize the TABLE1 array (Line 15). This does the same thing as the .BYTE directive in Line 100 of the assembly code.

This should give you a good idea of how indexing works. If you still have trouble, reread last month's discussion of indexing and try developing your own simple problems.

Math Revisited

As promised last month, we're going to start looking at multi-byte math operations, both in binary and binary-coded decimal (BCD).

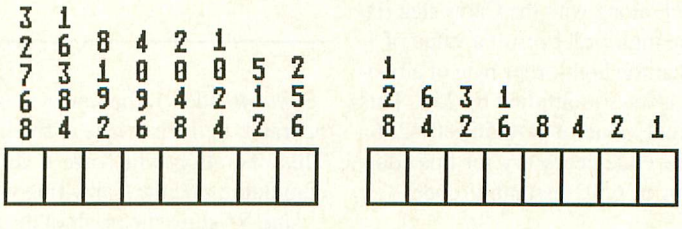
Why do we want to bother with multi-byte math? If you're only working with numbers from zero to 255, then single-byte math is fine. But what happens when you're writing the ultimate game program and need to show scores into the hundreds of thousands of points? Multi-byte math is the answer.

The simplest form of multi-byte math is probably the two-byte address storage. The 6502 can address 65536 (or 2^{16}) bytes of memory. Observant readers will note that this number will easily fit into two eight-bit bytes.

You've probably encountered two-byte addresses in BASIC. For example, if you need to know where your computer's display list is located, you can use the BASIC command:

```
DLIST=PEEK(560)+PEEK(561)*256
```

How does this work? Normally, we think of a byte as having bit values from one to 128 (left to right). In order to represent larger numbers, we add a second *high-order* byte to the first *low-order* byte. The high-order byte contains bit values from 2^8 (256) to 2^{15} (32768). This relationship is shown below.



In order for BASIC to reconstruct the number, it must multiply each byte by the value of its lowest-order bit. In the two-byte case, the low-order byte is multiplied by one, and the high-order byte is multiplied by 256. When the resulting numbers are added together, you have the value of the two-byte number.

Here are some decimal numbers, along with their two-byte binary equivalents.

DECIMAL	HIGH BYTE	LOW BYTE
128	00000000	10000000
255	00000000	11111111
256	00000001	00000000
257	00000001	00000001
511	00000001	11111111
512	00000010	00000000
65534	11111111	11111110
0	00000000	00000000

You don't have to stop with two bytes, either. For example, by using three bytes you can store numbers up to 2^{24}, or 16,277,216. Four bytes will give up to 2^{32}, or over four billion, and so on.

Carrying On

How is multi-byte math handled in 6502 assembly language? It's the same as single-byte, but with one difference. In multi-byte addition, the Carry flag is used to handle carries and borrows.

You've used carries and borrows all your life, but you probably don't think about them. Consider the addition of 13+9. When you add 3+9, you get 12. Since 12 is greater than the

maximum digit value of 9, you place the units portion (2) in the units portion of the result and carry the 10 to the next digit. This adds to the tens digit of 13, giving 20. When this is added to the units portion calculated earlier, we get a result of 22.

In subtraction, if you're subtracting 7 from 20, 7 is larger than 0, so a borrow from the next digit is necessary. The 2 in the tens position becomes a 1, and the 7 is subtracted from the borrowed 10, giving a result of 3 in the units position. The final result is 13.

These same principles apply in multi-byte math operations. The only difference is the base we are operating in. As you recall from a previous *Boot Camp*, the Carry flag is set to 1 if the result of an addition operation is greater than 255. In single-byte addition, we always clear the Carry flag before the ADC operation. In multi-byte adds, the Carry is only cleared before the *first* addition operation. This prevents any unwanted carries from giving incorrect results.

```
   HIGH        LOW
   ----        ---
  00000000    11111111   (255)
+ 00000000    00000001   (  1)
  --------    --------
  00000001    00000000   (256)
```

The above shows how carries work in binary. When 1 is added to 255, the resulting value of 256 is too large to fit in one byte. The low-order byte wraps around to 0 and the Carry flag is set. The high-order bytes are then added, along with the Carry flag (1). This gives the high-order result a value of 1. Remember that the high-order byte of a two-byte value is always multiplied by 256. This gives us a final value of (1×256)+0=256.

Below is the code necessary for this addition operation in 6502 assembly code.

```
01    *=  $600
10    CLD          ;BINARY MODE
20    LDA #255     ;GET 255
30    CLC          ;FIRST ADD!
40    ADC #1       ;ADD 1 TO 255
50    STA RESLO    ;STORE LOW RESULT
60    LDA #0       ;GET OP1 HIGH
70    ADC #0       ;ADD OP2 HIGH
80    STA RESHI    ;SAVE HIGH RESULT
90    BRK          ;ALL DONE!
0100 RESLO *=*+1 ;LOW RESULT BYTE
0110 RESHI *=*+1 ;HIGH RESULT BYTE
0120  .END       ;END OF ASSEMBLY
```

Line 10 clears the decimal mode to make sure we're working with binary numbers.

Line 20 loads 255, the low byte of the first operand, into the accumulator.

Line 30 clears the Carry flag for the first add operation. *Always* remember to clear the Carry flag for the first add of a multi-byte add operation.

Multi-byte subtraction works the same way as the single-byte version, except that the first subtract operation is preceded by a SEC (Set Carry) instruction.

Line 40 adds 1, the low byte of the second operand, to the low byte of the first operand. This operation will leave a zero in the accumulator, and the Carry flag will be set (1).

Line 50 stores the result of the low-byte add in the location labeled RESLO.

Line 60 loads 0, the high byte of the first operand, into the accumulator.

Line 70 adds 0, the high byte of the first operand, to the high byte of the second operand. Note that we *did not* clear the Carry before this operation, since we want the Carry status to be taken into account for all adds after the first one. In this case, with the Carry set, our result is 0+0+1, or 1.

Line 80 stores the result of the high-byte addition in the location labeled RESHI.

Line 90 stops the execution of the program with the BRK instruction.

Lines 100 and *110* set up the RESLO and RESHI storage areas. Note that these areas are set up with the low byte first, followed by the high byte. This is the standard 6502 storage format for two-byte values, and it's a good idea to get accustomed to it.

Multi-byte subtraction works the same way as the single-byte version, except that the first subtract operation is preceded by a SEC (Set Carry) instruction. Below is an example of the three-byte subtract operation $4203F5−$2E45FF. When finished, the result will be placed in RESL (low order), RESM (middle) and RESH (high order). Try executing this code and observe that the resulting number is $13BDF6.

```
01    *=  $600
10    CLD          ;BINARY MODE
20    LDA #$F5     ;GET OP1 LOW
30    SEC          ;FIRST SUBTRACT
40    SBC #$FF     ;SUB OP2 LOW
50    STA RESL     ;SAVE LOW RESULT
60    LDA #$03     ;GET OP1 MIDDLE
70    SBC #$45     ;SUB OP2 MIDDLE
80    STA RESM     ;SAVE MID RESULT
90    LDA #$42     ;GET OP1 HIGH
0100    SBC #$2E   ;SUB OP2 HIGH
0110    STA RESH   ;SAVE HIGH RESULT
0120    BRK        ;ALL DONE!
0130 RESL *=*+1  ;LOW RESULT BYTE
0140 RESM *=*+1  ;MID RESULT BYTE
0150 RESH *=*+1  ;HIGH RESULT BYTE
0160    .END       ;END OF ASSEMBLY
```

What About the Decimal Mode?

Remember how the 6502 uses two different methods of storing numbers? We have been looking at multi-byte operations in the binary mode. Multi-byte decimal-mode math works *exactly* like binary, but the data is stored in binary-coded decimal. All you have to do to select BCD math is use the SED (Set Decimal Mode) instruction at the start of your program. You can return to binary math at any time by using the CLD (Clear Decimal

Mode) instruction.

Now that we've looked at the basics of multi-byte math, let's make a few generalizations about the process.

```
10   LDA BYTE1A  ;BYTE 1
15   CLC         ;ON FIRST ONLY!
20   ADC BYTE1B
25   STA RESULT1
30   LDA BYTE2A  ;BYTE 2
35   ADC BYTE2B
40   STA RESULT2
45    .
50    .          ;ETC.
55    .
60   LDA BYTEnA  ;BYTE n
65   ADC BYTEnB
70   STA RESULTn
```

The above shows the procedure for a multi-byte add, where *n* is the number of bytes in the value. Note that the CLC instruction is used only for the *first* add of the group.

```
10   LDA BYTE1A  ;BYTE1
15   SEC         ;ON FIRST ONLY!
20   SBC BYTE1B
25   STA RESULT1
30   LDA BYTE2A  ;BYTE 2
35   SBC BYTE2B
40   STA RESULT2
45    .
50    .          ;ETC.
55    .
60   LDA BYTEnA  ;BYTE n
65   SBC BYTEnB
70   STA RESULTn
```

The above shows the procedure for a multi-byte subtract, where *n* is the number of bytes in the value. The subtract procedure is similar to the add in that the SEC instruction is only used for the *first* subtract.

What happens when you want to add or subtract two values of different length, such as adding a one-byte value to a three-byte value? The program below shows how this is done.

```
10   *=  $600
15   CLD           ;BINARY MODE
20   LDA SCORE     ;GET SCORE LOW
25   CLC           ;CLEAR 1ST TIME
30   ADC POINTS    ;ADD POINTS
35   STA SCORE     ;SAVE SCORE LOW
40   LDA SCORE+1   ;GET SCORE MID
45   ADC #0        ;ADD DUMMY ZERO
50   STA SCORE+1   ;SAVE SCORE MID
55   LDA SCORE+2   ;GET SCORE HIGH
60   ADC #0        ;ADD DUMMY ZERO
65   STA SCORE+2   ;SAVE SCORE HIGH
70   BRK           ;ALL DONE!
75 POINTS *=*+1    ;ONE BYTE
80 SCORE  *=*+3    ;THREE BYTES
85   .END          ;END OF ASSEMBLY
```

This program adds the one-byte value POINTS to the three-byte value SCORE. In this example, the three bytes of SCORE are not individually labeled, but are referenced as SCORE (low order), SCORE+1 (middle) and SCORE+2 (high order). The +1 and +2 added to the label SCORE simply indicate that the assembler is to add 1 and 2 to the address of SCORE for these operations. For example, if SCORE is located at $4000, SCORE+1 is address $4001, and SCORE+2 is $4002. If we had indicated SCORE−1, the address used would be $3FFF.

The subtract procedure is similar to the add in that the SEC instruction is only used for the first subtract.

By looking at this code, you will see that the first ADC operation adds the low byte of SCORE to POINTS, placing the result in SCORE. This is a typical first add, with a CLC operation before the addition.

The second and third adds are special in this case. Since POINTS is a one-byte field and SCORE is a three-byte field, we must complete the last two additions as if POINTS were three bytes long. As you can see from the example, the second and third adds simply add zeros to the second and third bytes of SCORE. This ensures that any carries out of the low bytes of SCORE will be properly taken care of. By adding zeros, the only factor affecting the result is the Carry flag.

The Challenge

No tutorial would be complete without a challenge to the readers. For next month, try to solve the following problems.

Problem 1: Subtract the two-byte field WITHD (withdrawals) from the three-byte field OLDBAL (old balance), placing the result in the three-byte field NEWBAL (new balance). All fields should be stored in BCD with standard data-storage formats. Start with OLDBAL = 108673 and WITHD = 4285. After the subtraction is complete, check NEWBAL to be sure it contains 104388.

Problem 2: Start with three ten-byte tables. Label these tables TABLE1, TABLE2 and TABLE3. Initialize TABLE1 and TABLE2 as follows:

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

Write the code necessary to subtract each byte of TABLE2 from the corresponding byte of TABLE1, placing the result in TABLE3. That is, subtract the first byte of TABLE2 from the first byte of TABLE1 and place it in the first byte of TABLE3. Repeat this process for each of the ten bytes in the tables. When complete, TABLE3 should contain the values:

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

These problems should get you thinking about multi-byte operations more deeply. Whatever you do, *don't give up*! Stick with it and you'll soon get the hang of it.

Next month we'll start looking at the many ways to manipulate our friend, the eight-bit byte.

PACKAGE DEALS FROM SAN JOSE COMPUTER

XE GS PACKAGE $149.95

» COMPUTER «» PRINTER «» PLOTTER «
» XE GS «» 1025 «» 1020 «

INCLUDES:

- 64K COMPUTER
- XG-1 LIGHT GUN
- CX40 JOYSTICK
- 1020 COLOR PRINTER PLOTTER
- 1025 DOT MATRIX PRINTER (80 COL.)
- BUG HUNT
- FLIGHT SIMULATOR
- MISSILE COMMAND

ALL ITEMS ARE NEW EXCEPT 1025 PRINTER

1200XL PACKAGE $99.95

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

INCLUDES:

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

ALL ITEMS ARE RECONDITIONED EXCEPT 1020 PRINTER

1200XL W/PLOTTER $59

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

INCLUDES:

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

ALL ITEMS ARE NEW EXCEPT 1200XL COMPUTER

DRIVE SPECIAL $99*

» 810 «

A SAVINGS OF $30

INCLUDES:

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

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

SOFTWARE PAK#1 $29.95

INCLUDES:

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

ST SOFTWARE TITLES AVAILABLE

CALL FOR LATEST PRICES AND TITLES

ASTRA 2001 DUAL DRIVE $199

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

12" GREEN SCREEN MONITOR $69.95

RECONDITIONED

NEW ARRIVALS

XE GS COMPUTER SYSTEM $99

W/PLOTTER FACTORY NEW

- 64K COMPUTER
- XG-1 LIGHT GUN
- DETATCHABLE KEYBOARD
- 1020 COLOR PRINTER PLOTTER
- FLIGHT SIMULATOR CARTRIDGE
- BUGHUNT, MISSILE COMMAND

1020 COLOR PRINTER PLOTTER

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

$14.98 Brand New

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

1200XL 64K COMPUTER

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

1027 PRINTER

- 80 Column
- Direct Connect
- Letter Quality

$79. Brand New

ATTN. DEALERS

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

800 48K COMPUTER

w/PAC-MAN $69.95 Reconditioned

1025 PRINTER

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

$79.98 Reconditioned

1050 DISK DRIVE $169.

Reconditioned

800 BOARDS

- Mother
- CPU
- Power
- ROM

YOUR CHOICE **$8.98** Brand New

8Bit I/O Cable . **$4.98**

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

810 DRIVES **$129.** Reconditioned

1200XL NEW **$69.98**

5.25" DISKS

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

MAJORITY ARE UNNOTCHED CONTAINING OLD SOFTWARE

NEW **DOS XE .. $9.95**

DOS 2.5 w/manual .. $4.98

1010 Tape drive ..... $29.

Books

Inside Atari basic $2.95
101 Tips and Tricks. $2.95

CARTRIDGES FOR 800, XL, XE

BASIC CARTRIDGE $4.95	MATH ENCOUNTER $7.98	DEFENDER $14.95	BALLBLAZER $19.98
BASIC TUTOR (2 BOOKS) $4.95	DANCE FANTASY $8.98	ROBOTRON $19.98	BLUE MAX $19.98
TURMOIL $4.95	LOGIC LEVELS $8.98	TENNIS $19.98	STAR RAIDERS II $19.98
PAC-MAN (no box) $4.95	MEMORY MANOR $8.98	FINAL LEGACY $19.98	DAVID'S MIDNIGHT MAGIC $19.98
DONKEY KONG (no box) $4.95	LINKING LOGIC $8.98	MARIO BROS. $19.98	ARCHON $19.98
GORF (400,800) $4.95	CHICKEN $8.98	DONKEY KONG JR. $19.98	KARATEKA $19.98
DEMON ATTACK (400,800)..... $4.95	CLAIM JUMPER $8.98	JUNGLE HUNT $19.98	CHOPLIFTER $19.98
DELUXE INVADERS $4.95	DELTA DRAWING $8.98	MOON PATROL $19.98	GATO $24.98
JOURNEY TO THE PLANETS... $4.95	HEY DIDDLE DIDDLE $8.98	BATTLEZONE $19.98	ACE OF ACES $24.98
STAR RAIDERS $4.95	SLIME (400/800) $8.98	FOOD FIGHT $19.98	LODE RUNNER $24.98
MISSLE COMMAND $4.95	ALPHABET ZOO $8.98	HARDBALL $19.98	BARNYARD BLASTER (gun req.) .. $24.98
ASTEROIDS $4.95	ALF $8.98	FIGHT NIGHT $19.98	DARK CHAMBERS $29.98
GALAXIAN $4.95	ADVENTURE CREATOR .. $8.98	ONE ON ONE BASKETBALL $19.98	AIRBALL $29.98
DEFENDER $4.95	DIG DUG $14.95	DESERT FALCON $19.98	SUMMER GAMES $29.98
E.T. $4.95	SKY WRITER $14.95	NECROMANCER $19.98	CROSSBOW (gun req.) $29.98
FACEMAKER $4.95	FOOTBALL $14.95	RESCUE ON FRACTALUS $19.98	CRIME BUSTERS (gun req.) $29.98

DISK SOFTWARE FOR 800, XL, XE

DAVID'S MIDNIGHT MAGIC $4.98	COMMBAT $4.98	LASER HAWK..... $4.98	STRANGE ODYSSEY $4.98
REPTON..... $4.98	PREPPIE I $4.98	ROCKET REPAIRMAN..... $4.98	VISICALC $24.98
BANDITS (48K 400,800) $4.98	PREPPIE II $4.98	ADVENTURELAND $4.98	BOOKKEEPER W/ num. keypad... $29.98
CLAIM JUMPER $4.98	THE COUNT $4.98	PIRATE ADVENTURE $4.98	
SYNTREND $4.98	DISK 50 (50 GAMES)..... $4.98	SECRET MISSION $4.98	
CROSSCHECK $4.98	FREAKY FACTORY..... $4.98	VOODOO CASTLE $4.98	
MISSION ASTEROID..... $4.98		TECHNICOLOR DREAM $4.98	

OTHER TITLES AVAILABLE PLEASE CALL FOR DETAILS

IN THE UNLIKELY EVENT THAT THE ITEM(S) YOU RECEIVE ARE DEFECTIVE, PLEASE CONTACT US FOR AN RA#.

Special buys on 8-BIT Hardware!

SAN JOSE COMPUTER

THE ATARI STORE

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

Light Gun

For use with pistol games on 8-Bit systems

$34.98

Light Gun Package

Includes:
- Light Gun
- Crime Busters
- Crossbow
- Barnyard Blaster

$109.

SHIPPING: ADD $5.00 TO ALL ORDERS. AIR AND INTERNATIONAL SHIPPING EXTRA. THAT'S IT.
WARRANTY: 90 DAY WARRANTY ON ALL ITEMS. **TAX:** CALIFORNIA RESIDENTS ADD 7% SALES TAX.
PREPAYMENT: USE VISA, MASTERCARD, MONEY ORDER, CASHIER'S CHECK OR PERSONAL CHECK.
PERSONAL CHECK MUST CLEAR PRIOR TO SHIPMENT. **C.O.D.:** CASH, CASHIER'S CHECK OR M.O. ONLY.
Prices subject to change without notice. Brand and/or product names are trademarks or registered trademarks of their respective holders.
Ad produced on an ATARI ST using Publishing Partner and printed on an ATARI SLM804 PostScript compatible laser printer.

CIRCLE #102 ON READER SERVICE CARD.

B&C ComputerVisions

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

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

800/XL/XE SOFTWARE
ALL TITLES ON DISK

ENTERTAINMENT

12 ADAMS ADVENTURES	14.95
ALIANTS	26.95
ALT. REALITY CITY	26.95
ALT.REAL DUNGEON	26.95
ASSULT FORCE	19.95
AUTO DUEL	35.95
BEYOND CASTLE WOLF	14.95
BISMARK	26.95
BOP & WRESTLE	26.95
BORDINO:1812	22.50
BOULDERDASH CONSTR.SET	17.95
BRUCE LEE	17.95
CASTLE WOLFENSTEIN	14.95
CHAMP. LODE RUNNER	26.95
CONFLICT IN VIET NAM	17.95
COSMIC TUNNELS	9.95
D-BUG	7.95
DALLAS QUEST	7.95
DELUXE INVADERS	7.95
F-15 STRIKE EAGLE	31.50
FIGHT NIGHT	17.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS	22.50
GREAT AMER.ROAD RACE	22.50
GUNSLINGER	26.95
HARD HAT MAC	7.95
JAWBREAKER	9.95
KARATEKA	13.50
KENNEDY APPROACH	17.95
KNICKERBOCKERS	13.50
KORONIS RIFT	13.50
LAST V-8	8.95
L.A. SWAT/PANTHER	8.95
LEADERBOARD	13.50
LODE RUNNER	13.50
MICROLEAGUE BASEBALL	35.95
NAPOLEON AT WATERLOO	22.50
MONTEZUMA'S REVENGE	14.95
MOUSEQUEST	17.95
MOON SHUTTLE	7.95
NATO COMMANDER	17.95
NIBBLER	12.95
NINJA	8.95
OGRE	26.95
OIL'S WELL	9.95
O'RILEY'S MINE	9.95
PIRATES OF BARB. COAST	22.50
PITFALL/DEMON ATTACK	13.50
PREPPIE I & II	9.95
RESCUE ON FRACTALAS	13.50
ROME & THE BARBARIANS	17.95
SILENT SERVICE	31.50
SOLO FLIGHT	17.95
SPEEDKING	8.95
SPITFIRE 40	31.50
STARFLEET I	44.95
STAR RAIDERS II	17.95
SPY VS. SPY III	17.95
STOCKMARKET	22.50
STRIP POKER	26.95
SUMMER GAMES	17.95
TAX DODGE	9.95
TEMPLE OF APSHAI	9.95
THE HULK	5.35
TOMAHAWK (64K)	26.95
TRAILBLAZER	26.95
ULTIMA II	35.95
ULTIMA III	35.95
ULTIMA IV	53.95
UNIVERSE	44.95
WINTER CHALLENGE	13.50
ZAXXON (400/800)	13.50

PROGRAMMING

ACTION!	71.95
ACTION! TOOLKIT	26.95
BASIC XL	53.95
BASIC XL TOOLKIT	26.95
BASIC XE	71.95
DIAMOND (GEM O/S)	69.95
DOS 2.5	7.95
DOS XE	10.00
DISK I/O	26.95
KYAN PASCAL	62.95
LIGHTSPEED C	35.95
LOGO	29.95
MAC/65	71.95
MAC/65 TOOLKIT	26.95
PILOT	19.95
SPARTA DOS X	71.95

PRODUCTIVITY

ANIMATION STATION	89.95
ATARIWRITER	29.95
ATARIWRITER (CART ONLY)	19.95
ATARIWRITER+	44.95
ATARI BOOKKEEPER	24.95
ATARI MUSIC II	14.95
AWARDWARE (1050)	13.50
BANK STREET WRITER	14.95
BLAZING PADDLES	31.50
COMPUTE YOUR ROOTS	35.95
DATAMANAGER	8.95
ELECTRONIC CHECKBOOK	8.95
FAMILY FINANCE	6.95
GUITAR WIZARD	26.95
HOME ACCOUNTANT	19.95
HOME BASE	12.95
HOME FILING MANAGER.	6.95
HOMEPAK	24.95
INVENTORY MASTER	80.95
LETTER WIZARD	29.95
MONEY MANAGER	8.95
MUSIC CONSTRUCTION SET	13.50
NEWSROOM (1050 - 64K)	44.95
NEWS STATION	26.95
NEWS STA. COMPANION.	26.95
PAGE DESIGNER	26.95
PAINT	12.95
PRINT POWER (1050)	13.50
PRINTKIT (1050)	13.50
PRINTSHOP	34.95
P.S. COMPANION (64K)	24.95
P.S.GRAPHICS LIBRARY 1	17.95
P.S.GRAPHICS LIBRARY 2	17.95
P.S.GRAPHICS LIBRARY 3	17.95
PROOF READER	17.95
PUBLISHING PRO	35.95
RUBBER STAMP	26.95
SYNTREND	14.95
SUPER MAILER	35.95
THE LOTTO PROGRAM	17.95
TIMEWISE	6.95
TURBOWORD/80 COLUMN REQUIRES XEP80	44.95
VIDEO TITLESHOP (64K)	26.95
GRAPHICS COMPANION.	17.95
VIRTUOSO	29.95
VISICALC	24.95

EDUCATION

ATARI LIGHT MODULE (REQ. STARTER SET)	9.95
BUZZWORD	35.95
GRANDMA'S HOUSE (-10)	9.95
HEY DIDDLE (AGE 3-10)	9.95
LINKWORD: SPANISH	22.50
LINKWORD: GERMAN	22.50
LINKWORD: FRENCH	22.50
LINKWORD: ITALIAN	22.50
MASTER TYPE	14.95
STATES AND CAPITALS	9.95
TOUCH TYPING	9.95
QUIZ MASTER CONSTR.	8.95
QUIZ MSTR. USA CONSTR.	8.95
AMERICAN EDUCATION:	
A+ BIOLOGY G10+	17.95
A+ GERMAN	17.95
A+ GRAMMER G4+	17.95
A+ LERN TO READ G1-4	35.95
A+ READING COMP G1-8	35.95
A+ SCIENCE G3-4	17.95
A+ SCIENCE G5-6	17.95
A+ SCIENCE G7-8	17.95
A+ SPANISH	17.95
A+ SPELLING G2-8	35.95
A+ U.S. GEOG. G8+	17.95
A+ U.S. GOV. G10+	17.95
A+ U.S. HIST. G5+	17.95
A+ VOCABULARY G4+	17.95
A+ WORLD GEOG. G8+	17.95
A+ WORLD HIST. G8+ (G = GRADE LEVEL)	17.95
CBS (AGE 3-6):	
ASTROGROVER	8.95
BIG BIRD SPEC DELIVE	8.95
ERNIE'S MAGIC SHAPE.	8.95
DAVIDSON:	
MATH BLASTERS G1-6	44.95
SPELL IT! AGE 10+	44.95
DESIGNWARE:	
MATHMAZE (6-11)	26.95
MISSION ALGEBRA (13+)	13.50
SPELLICOPTER (6-11)	26.95
TINK TONK (AGE 4-6):	
ABC'S	8.95
COUNT AND ADD	8.95
SMART THINKER	8.95
SPELLING	8.95
SUBTRACTION	8.95
THINKING SKILLS	8.95
ALL 6 TINK TONKS	39.95
UNICORN:	
10 LITTLE ROBOTS (PRE-SCHOOL)	26.95
FUN BUNCH (6-ADULT)	26.95
RACECAR RITHMETIC (AGE 6+)	26.95
WEEKLY READER (PRE-SCHOOL):	
STICKY BEAR SHAPES	26.95
STICKY BEAR NUMBERS.	26.95
STICKY BEAR ABC'S	26.96
STICKY BEAR OPPOSITE	26.95
SB BASKET BOUNCE	26.95
STICKY BEAR BOP	26.95
RUN FOR IT	26.95
PIC BUILDER	26.95

****

WE CARRY A FULL LINE OF SOFTWARE FOR THE 520/1040 AND MEGA ST COMPUTERS.

800/XL/XE SOFTWARE
ALL TITLES ON CARTRIDGE

ENTERTAINMENT

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

PRODUCTIVITY

ATARIWRITER	29.95
FUN WITH ART	14.95
MICROFILER	22.50

EDUCATION

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

(* = REQ. LIGHT GUN)

****

SPECIAL VALUE
PAC-MAN AND QIX CARTRIDGES IN SPECIAL STORAGE CASE
ALL FOR ONLY
$9.95

****

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

SUPER SPECIALS
RECONDITIONED ATARI MERCHANDISE 30 DAY WARRANTY

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

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

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

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

CIRCLE #103 ON READER SERVICE CARD.

REVIEW

CROSSBOW

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
XL/XE cartridge: $34.95

Reviewed by Matthew J.W. Ratcliff

Crossbow is a swashbuckling graphics adventure that is played with the Atari light gun. Like Robin Hood, you lead a band of merry friends on a quest to recover treasures stolen by the Evil Master. As you and your friends trek across eight detailed scenes, the Evil Master sends hoards of creatures to dispatch you. With the aid of your trusty crossbow (light gun) you can vanquish the foes, pick up new friends (extra lives) and discover the path to the final battle within the castle hall.

The original *Crossbow* is a coin-op video game from Exidy, copyrighted 1983. The Atari version was completed and copyrighted in 1988 by Atari Corp., developed by Sculptured Software Inc. Since I have never played the coin-op original, I cannot give you a direct comparison.

The graphics are quite good, although I have seen better. The screens remind me of Koala pictures, with little or no special effects for added shading, details or depth. The screens seem "flat," but they are plenty detailed and the animation of the characters is smooth and predictable. I like the sound effects of *Crossbow*, especially the digitized one of the scream when one of your friends is decimated by the enemy.

The game begins with a parchment map display. The eight areas to be traveled are displayed as graphic images. Your adventurer is displayed to the left center of the screen, and at the bottom are red and green (and sometimes blue) flashing boxes. You must shoot one to select your path. The color chosen determines the path you'll take to the next adventure scene. You may waste some time traveling back and forth between the same areas before you learn the proper color sequence to travel through all seven treasure and adventure screens and on to the eighth and final battle screen in the castle hall. When the path is chosen, a dashed line is traced out from your current location to your next fight.

You may end up at the cactus for a challenge in the desert. The display is open desert, with cacti scattered all about and mountains in the distance. Your three friends will begin walking across the screen, left to right, spaced about ¼- to ¾-screen apart. As they walk, they are beset upon by vultures, ants and scorpions. You must shoot them with the trusty crossbow before a friend gets killed. If all your friends die, the game ends. Rabbits, snakes and the Master's evil eye may be shot for extra points. The first traveler to make it across the display usually picks up a treasure for more points. Help all your friends across the desert in order to advance to the next stage of the adventure. The first time each screen is completed (except for Village and Castle scenes) an extra friend joins you, providing another "life" to help you through the game.

A trek through the caverns requires that you shoot hanging stalactites to plug holes in your friends' path. Bats, a ghost and falling stalactites must also be eliminated. This is one of the most challenging sites to complete.

In the Volcano phase, lava rocks must be burst above your friends' heads. A large standing stone must be shot to make a bridge across a lava river. With only one basic obstacle (the lava rocks) to worry about, this is an easy level to complete with a steady hand and concentration.

The Jungle is nasty. There are two small pits your friends can safely walk across, so long as you don't allow the man-eating plants to grow up from them. You cannot concentrate too much on those nasty flowers though, because there are banana-tossing monkeys and ornery toucans flying from the trees above. With practice—and a careful eye on the plants—this screen can be mastered.

The Village, haunted by many evils, is a witches' haven. Ghosts fly out to bash your friends. Warlocks appear on the rooftops and rain fireballs on you. Lightening bolts are a constant threat. Gangsters shoot from the windows of the houses, evil faces pop up everywhere. You need to shoot out the street lights so darkness will provide some cover for your friends. You'll be lucky to get all your friends through this phase.

Down at the River, you simply have to escort your friends across a bridge while avoiding the pterodactyls and bouncing boulders. The myriad other creatures on this screen, such as trolls and alligators, can be blasted for bonus points. This phase is not as easy as it might seem.

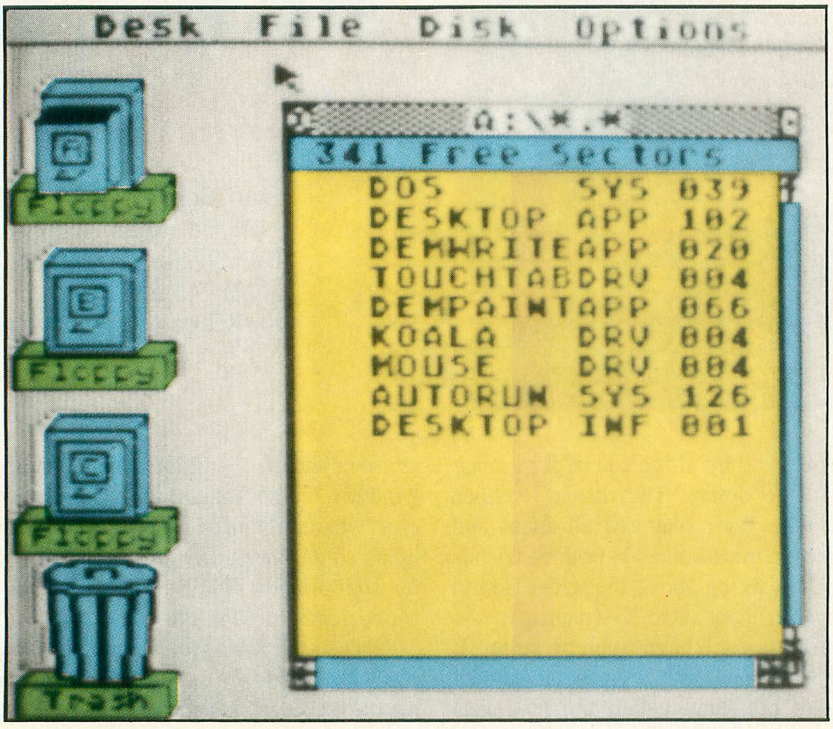
At the Drawbridge, the entrance to the castle, you must first shoot the ropes holding the door up. It will lower across the moat, allowing you to enter. Watch for deadly archers along the ramparts of the castle.

Once you have conquered all seven preliminary screens, you have to choose the correct path to finally enter the Castle Hall and shoot the Evil Master while his eyes are glowing red. If you complete the game, it starts over again for a repeat of the challenge.

Gun accuracy is imperfect but livable. If the light gun is held near the display, you find that it shoots slightly to the left, just as in Atari's *Bug Hunt*. It isn't a particular problem with *Crossbow*, however, since you have unlimited shots with no penalty. That, coupled with the fact your crossbow is a rapid-fire unit (hold down the trigger for continuous firing), makes the game playable despite the minor inaccuracy.

Crossbow is a good light-gun game. The graphics are a bit blocky, but well done overall. Playability is good and the sound effects are entertaining. This game will definitely make a nice addition to anyone's light-gun game collection.

Matthew Ratcliff, a frequent contributor to ANALOG Computing, *lives in St. Louis, Missouri, with his wife and two children.*



DIAMOND GRAPHICS OPERATING SYSTEM

Reeve Software
29W 150 Old Farm Lane
Warrenville, IL 60555
(312) 393-2317
Disk version: $29.95
Cartridge version: $79.95

Reviewed by James F. Patterson

I remember the first time I saw a GEM-type operating system on the Apple Macintosh, with its menu, icons, and point-and-click replacement for typed commands. I was sure at that time I was seeing the future, sure that this was what was going to bring computing to the masses. I was also convinced that this meant the end of the 8-bit computer line. Then in May 1986 I saw a magazine displaying the GEOS system from Berkeley Softworks. At last this GEM-type desktop operating system was available on a 6502 8-bit machine, even though it was on the Commodore 64. I knew it was only a matter of time before it would be available on the Atari 8-bits. Little did I know that it would be two and a half years.

There are, as I write this, two new GEM-type operating systems available for the 8-bit Atari line of computers: *Diamond OS* from Reeve Software, distributed by USA Media, which we'll be examining here, and *GOE* from Total Control Systems.

An introduction

First, in case you are new to computing or you've been living in a cave for the last few years, the GEM-type operating system consists of an easy-to-use and-learn operating environment with icons, windows, drop-down menus, dialog boxes and a pointer (mouse). As I stated, these have been available for some time on other, more expensive systems like the Atari ST, Apple Macintosh, Commodore Amiga, and even the IBM XT computers.

These new graphic-oriented operating systems are, unlike many of the improvements in the 8-bit world, not hardware enhancements; in other words, you aren't required to open your computer and remove chips or cut tracings on the circuit boards. They use the bank-select cartridges made popular by ICD (except the disk version of GOS); therefore, they should work on all 8-bit Atari computers. Before we look at the operating system, let's first get acquainted with a few of the terms used.

Desktop: The screen display.

Icons: Graphic representations, images or objects used to show a file or utility on the screen or monitor.

Menu: As the name implies, a display listing choices or functions from which the user may select.

Pointer: Used by the operator to make menu selections. This can be any input device, such as a mouse, joystick, trakball, light pen, or even a keyboard used to move the cursor around on the display screen.

Window: A rectangular section or area of a display screen that is dedicated to a specific activity or application. Windows allow the screen to display more information than one screen allows. Think of them as a screen within a screen.

Dialog box: An on-screen form that prompts the user for information and allows the input of that information.

An overview

The *Diamond Graphics Operating System* is a great implementation of the GEM-type operating system on the Atari 8-bit computers. The desktop is available in two formats: The disk version, which requires 64K of RAM, supports DOS 2.5. The piggyback super cartridge version, which requires only 48K of RAM, also includes the Diamond Programmer's Kit. The cartridge, according to the manufacturer, supports most Atari DOS systems, including the new Atari DOS XE and Sparta DOS X. Like the ST, GOS performs the following functions:

1.) Time/Date DOS support.

2.) Folder support, including deleting a folder file.

3.) Exit to Basic; just type DOS to return to the desktop.

4.) Added icons, like a stop sign after commands with "Are you sure?" in a dialog box before manipulating files.

5.) Multiple-window handling; the title bar of the current window is highlighted.

6.) Window sliders; these allow the user to choose the size of the on-screen window display.

7.) Compatibility with all DOS systems (cartridge version only).

8.) Print or show disk files from the

desktop.

9.) Multiple-file deleting and copying (Tagging).

10.) Direct icon-to-icon disk copying; just click on one icon and drag to the other.

11.) Direct window-to-window file copying and deleting.

12.) Auto saving of window positions using one or two windows.

13.) Window-full reversing.

14.) Command line files.

15.) Memory expansion support up to 16 megabytes.

As can be seen from the above list, almost every need has been foreseen and met by the system developers. I, on the other hand, cannot verify all the features mentioned, as I was furnished with the disk version of GOS and don't own an extended-memory machine. I can tell you, however, that the version I worked with seemed to do all that the authors promised and was quite impressive.

The *Diamond GOS* supports all available cursor-movement devices, with selection made at the graphics configuration screen. To choose the preferred device, the user simply clicks on the device icon and the appropriate driver is installed. The available drivers include joystick, ST mouse, Touch Tablet and trakball. This procedure is repeated for the desired memory-expansion driver for those 8-bit users with memory expansions of 256K or more.

Once the desktop is set up for your system (i.e. the number of drives and the system icons are placed as you want them on the screen, and you've set your preference for Text or Icons for file display when the windows are open), you then save your desktop to disk. Once done, the desktop comes up this way every time you boot your system. This feature gives you the option of different setups for different system configurations.

The desktop

Upon booting up the system, you are greeted with the GOS desktop, named for its similarity to the office desktop. You have file cabinets with drawers that open and close as you open and close files (in this case, the directory for the disks), the desktop work surface area, and a trash can for garbage disposal. The visual elements of the desktop are the menu bar along the top of the screen, the icons for the disk drives as set up (described earlier) by the user, the trash can and the cursor or mouse pointer. Moving or dragging the mouse pointer across the menu bar shows you the options available by highlighting each heading (Desk, File, Disk and Option), each with its own drop-down menu containing more options.

Disk icons are represented by the picture of a file cabinet, the drawer of which opens when you choose one of the disk icons. When you activate a disk icon, a window opens displaying the directory of that disk. At this point you move the mouse pointer to the desired file and double-click (press the input device's trigger twice quickly). The file is then opened.

It should be stated here that the mouse pointer may also be moved around the desktop with no input device by using the four arrow keys and substituting the space bar for the fire button on a joystick device or the button on the mouse.

The specifics

The mouse is used to make most of the choices with GOS. There are two ways to make a selection. The first is to move the mouse pointer (cursor) to a menu on the screen. You can then double-click to both choose that option and open it. Second, you can move to the menu and single-click; that is, hit the button once then move across the menu bar to FILE, where a drop-down menu appears. You then single-click on the OPEN command. The method is entirely up to you.

To copy a disk with GOS, you move the mouse pointer across the menu bar at the top of the desktop to the DISK option, which causes a drop-down menu with the choices of DUPLICATE and FORMAT to appear. After choosing DUPLICATE, a dialog box appears that asks you for the source and destination disks, and whether or not the destination disk needs to be formatted. You then either proceed by clicking on OK to continue or CANCEL to go back to the desktop.

To copy a file, you double-click on the desired source disk—or use the single-click method as described above—to display the files on that disk, then move the mouse pointer to the desired file, click, and drag it over to the destination disk icon. It should be noted here that the disk drives are no longer referred to as D1: and D2:, etc. GOS uses the method used on most major systems; that is D1: is now A:, and D2: is B:.

The re-sizing of a window can be done in two ways. You may fill the entire screen with the window by clicking the mouse pointer on the button found in the window's upper right-hand corner. To change the window's size to something other than full screen, you can use the mouse pointer to drag on the button found in the window's lower right-hand corner. You can drag the window's corner in any direction, making the window wider, narrower, taller or shorter. When re-sizing a window, you will see a flashing outline of the window as you drag it. This outline shows the size that the window will be when you release the mouse button. Sizing of windows is important when you want to copy a file from one drive to another.

The grid at the top of the window is where you click and drag to reposition the entire window in the desktop. The button in the upper right-hand corner of the window is the same as the CLOSE option under the FILE selection of the desktop menu bar. This closes the window and puts you back in the desktop.

The trash can icon is used to delete files when they are no longer needed. To do this, you simply click on a file and drag it to the trash icon. If, at this point, you have the CONFIRM option active under the OPTIONS menu (which, in my opinion, should always be active), a DIALOG box appears and asks "ARE YOU SURE?" You then choose either OK to delete the file, or CANCEL to return to the desktop. If, on the other hand, you make an accidental choice, you simply move out of the window or away from the icon and single-click. The choice is cancelled.

Conclusion

I consider *Diamond GOS* a significant development for the 8-bit line of Atari computers. No 8-bit Atari user should be without it.

Jim Patterson is a former product support representative for Texas Instruments. He holds a degree in electromechanical engineering and is currently working toward a degree in computer science. He has been an avid Atari user since 1981.

XF551 Commands

by Jerry van Dijk

The new XF551 drive is probably the best thing that's happened to Atari's 8-bit computer line since the birth of the 130XE. But it also follows what seems to be normal Atari policy in that it comes with an ancient DOS and no documentation of the new features (true double density, double-sided, high-speed) whatsoever. Which in effect means that if you don't want (or can't get) the new DOS XE, you're stuck with what amounts to a slightly faster 1050 in a new housing.

So when I got my new XF551 drive, the first thing I did was take a deep dive into the file-management system to see if I couldn't somehow use, in my own programs, some of the new features. The following is the result of my explorations.

I must warn you, though, that this is fairly complicated stuff and best suited for the hard-core Atari addict. If you want to know more about using the SIO commands, I can't do any better than to direct you to previous ANALOGs and the Master Memory Map.

Conventions

Since all the XF551 commands and offsets mentioned below are new, there are no fixed names for them yet. I was therefore forced to make up my own. When I use a new name, I will explain its meaning and put "NEW" after its value or offset. In naming them, I tried to stay as close to the standard Atari naming conventions as possible.

The SIO interface

The SIO (Serial Input Output) system is the last link between the operating system and the XF551. Communicating with the drive will therefore, in practice, mean setting up a Device Control Block (DCB) with the proper information and calling the SIO system through its vector (SIOV, $E459). This way, you can format a disk, read or write a sector, etc. More information about SIO and the DCB can be found in the Master Memory Map.

The new features

To control the new features, the XF551 adds two new commands to the list of SIO commands. The new commands are GETREC (GET drive set-up RECord, $4E, NEW) and PUTREC (PUT drive set-up RECord, $4F, NEW). These commands will read or write a DRVREC (DRiVe RECord, NEW). The 12 bytes that make up a DRVREC are the real secret behind the new features of the XF551.

Since all the XF551 commands and offsets mentioned are new, there are no fixed names for them yet.

The DRVREC format is (all names: NEW): See Figure 1

NOTE: DRVREC uses a high-byte first format

Explanations

DRVTRC: This byte specifies the number of tracks. Normal value is $28, which means the disk contains 40 tracks.

DRVSTP: Defines that the step rate is the time the drive needs to access a new sector. In other words, how long it takes to execute a command. Normal value is $0000. I've put in the question marks to indicate that I have no information on other possible values.

DRVSEC: The number of sectors on a track. Values here are $12 for an 18-sector single-density disk or $1A for a 26-sector dual-density disk.

DRVSID: The number of sides used. The $00 stands for single-sided operation, $01 for double-sided. To format a disk on both sides, simply set the DRVSID to 1 and issue an SIO format command. You can access the extra sectors on the second side through SIO. The drive numbers these sectors consecutively. So if you formatted the disk double-sided, single-density, the last sector on side one is number 720, and the first on the second side is 721.

DRVDEN: The density used. Possible values are $00 for single-density (DOS 2.0) and $04 for double-density (DOS 2.5).

DRVBYT: The number of bytes in a sector. Normally, this is $0080 for 128 bytes/sector. In double density, you set the value to $0100 for true 256-byte sectors. In the latter case, you have under any 2.X compatible DOS 253 data bytes a sector. As with normal 128-byte sectors, the last three bytes are used by DOS for its own bookkeeping.

DRVSEL: Set the drive-select number. If you booted from the XF551, this will normally be 1. This byte gives you the possibility of overriding the drive-select switch.

DRVSER: Controls the baud rate used in communication between SIO and the XF551. Its default value is $41 (on a PAL system). In theory you can speed up disk access by increasing the drive's baud rate. But to synchronize with SIO you would have to modify its timing values also. There is, however, no information yet about the values possible.

DRVMSC: These two bytes ($0000) seem to serve no useful purpose yet. They're probably here for compatibility reasons or future extensions.

NAME	USAGE	OFFSET	VALUES	
DRVTRC	# of tracks	$00	$28	(40 track)
DRVSTP	step rate	$01	$0000	(???????)
DRVSEC	# sectors/track	$03	$12	(18 sectors)
			$1A	(26 sectors)
DRVSID	# of sides	$04	$00	(single-sided)
			$01	(double-sided)
DRVDEN	density	$05	$00	(single-density)
			$04	(double density)
DRVBYT	# bytes/sector	$06	$0080	(128-byte sector)
			$0100	(256-byte sector)
DRVSEL	drive select	$08	$01	(Drive 1 select)
DRVSER	serial rate	$09	$41	(???????)
DRVMSC	misc	$0A	$0000	(???????)

FIGURE 1

Using the new commands

Use of the DRVREC is fairly straightforward. With the new DCB GETREC command, you read the drive record in a buffer somewhere, modify the bytes for the new setup and finally write the record back to the drive with the PUTREC command. That's all there is to it.

It would be wise, in a practical program, to issue a disk status call (DSKINV, $E453) after changing the DRVREC. If all went well, the DCB status byte (STATUS, $30) will contain the value of 1.

Conclusion

There is, of course, nothing final about the data given above. The final word can only come from Atari, if and when they'll decide to publish the XF551 full-interface specifications. In the meantime, however, you have at least a starting point for using the full power of the XF551—and for exploring new, as yet unseen horizons on your Atari.

Jerry van Dijk uses his Atari both as a study tool and for recreation. His main interests are system-level programming and the use of computers in the practice of law.

ST NOTES

by Frank Cohen

Atari is aggressively building a U.S. dealer base. Since Michael Dendo joined Atari as the company's vice president of sales, Dendo has been working to change the apathy many industry insiders have toward Atari.

Dendo joined Atari Corp. in August '88 and has survived his position longer than any of his predecessors. Dendo was previously the western regional and national military sales manager for Star Micronics, a manufacturer of printers. At Star, Dendo managed distribution to key accounts and volume merchants. Prior to Star, Dendo was the vice president of dealer sales for National Business Systems, a manufacturer of point-of-sale terminals and high-speed embossing equipment. National also produced ion deposition printers, which use ion beams instead of lasers to produce computer-generated printouts.

Dendo lists Atari Corp.'s limited chain of retail outlets as the number one reason the ST has failed to make significant inroads into the U.S. personal computer market. He cites the limited supply of machines and poor dealer relations as the major culprits behind Atari's current retail woes.

Supplies of ST computers became limited in 1988 due to an industrywide shortage of memory chips. Like most personal computers, the ST uses a special high-capacity memory chip called a DRAM (Dynamic Random Access Memory). American protectionist legislation and supply restrictions by the Japanese electronics cartel, MITI, caused a worldwide shortage of DRAM chips in 1988. Recently, Atari's supply crisis seemed to be reduced.

"The DRAM crisis is over," said Sam Tramiel, president of Atari Corp., at a trade show late last year. Tramiel noted three contracts with major DRAM chip manufacturers to ensure an adequate supply of memory chips during 1989. Tramiel quoted current production quantities of ST computers to be between 50,000 and 70,000 machines per month. "Ninety percent of our production went to Europe last year," said Tramiel, acknowledging the starved supply conditions of the U.S. market. With increased production of ST computers, the U.S. market will again be readdressed with advertising and promotional plans absent since 1986.

Languages

The most popular development language for the ST is C. The language began as a solution to the problem of high-level programming languages becoming too isolated from a host computer's operating system and hardware—as is the case with Pascal. Programming in C is very much a hybrid between assembly and high-level language programming. This can be a mixed blessing when a programmer looks at the pitfalls built into the C language. To help combat the pitfalls, most C manufacturers have begun including a new type of debugging software with their C packages.

Source-level debuggers evaluate C source code for errors before the source code is compiled. The traditional method of finding bugs in programs is to compile the source code, run the program and evaluate its performance. With a source-level debugger, a programmer is alerted to syntax errors, variable manipulation problems and program tracing.

Mark Williams' C ($179.95 List) has a new source-level debugger, *CSD* ($69.95 List) available for its ST development language. Mark Williams' *CSD* operates like a C interpreter. Programs may be interrupted and variables or memory may be checked. In separate GEM windows, the source code, program evaluation and runtime history are displayed. Using *CSD* is advertised to cut C programming time in half.

Deutschland BASIC

There is a new standard BASIC language for the ST in Europe. Atari Germany began shipping Omikron BASIC as the standard Atari ST BASIC language late last year. Since then, 11 other European countries have followed the Omikron standard.

Omikron BASIC is close to MBASIC and GW BASIC for MS-DOS machines. Omikron's development package comes with an interpreter, compiler, and a large library of precompiled programs for use in specialized applications. The Omikron library has routines for GEM, MIDI, numerics, statistics, complex numbers and financial mathematics. The libraries make it easier for beginners to understand how to develop complex applications.

Omikron is a powerful BASIC. The program allows screen editing using the GEM system with menus and windows, and the language is rich with mathematical operators: 19-digit precision, matrix operations, factorial and hyperbolic functions. Built-in commands also support QuickSorting of arrays, and Indexed Sequential Access Method (ISAM) file indexing methods for business database applications.

Omikron recently attended COMDEX, in part to find an American distributor to handle U.S. and Canadian marketing of their products. If the U.S. version of Omikron BASIC is well packaged and supported, it could give GFA BASIC a run for its money.

Mainframe communications

Tozd Kooperacija—with a name like that, it must be a Yugoslavian company—showed an interface box recently developed to permit a Mega ST computer to emulate an IBM 3270 terminal. Sounds like fun, doesn't it? When you consider the $30,000 IBM charges its customers for a 3270 terminal, the $6,500 price of a Mega ST and the Tozd interface box becomes appealing.

The Tozd 3270 emulator box allows up to eight computers to emulate IBM terminals. Engineering shops looking for a low-cost alternative to the Digital Equipment or IBM solution are finding the Mega ST to be a powerful workstation.

Eating frenzy

Computer trade shows in general are a gigantic curio emporium. Almost every booth has something to take home. At a recent trade show we found WordPerfect giving away hats, while Intersect pushed out its business cardholders. Lots and lots of brochures were given out, which created a high demand for plastic bags. The inevitable outcome is a bag frenzy. The little Microsoft bag holds about ten brochures and fits snugly into the larger Packard Bell bag. The Xerox bag swallows up the Packard Bell bag with ease. But the MacWorld bag, which boasts dimensions that exceed two yards of plastic, gobbled up all the competition.

Frank Cohen has been developing Atari programs since his first commercial product, Clowns & Balloons. *You may contact him directly on CompuServe (76004,1573) and GEnie (FRANK.COHEN), or by writing to P.O. 14628, Long Beach, CA 90803-1208.*

REVIEW

CRIME BUSTER

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
XL/XE cartridge: $34.95

Reviewed by Matthew J.W. Ratcliff

Crime Buster is a hot new light-gun game from James Zalewski, the author of the popular *Barnyard Blaster*. In this game, the mobsters are trying to take over the city, and it is up to you, the hottest cop in town, to clean it up.

This one- or two-player game sports first-rate graphics, excellent sound effects, and light-gun accuracy that is top-notch (noticeably better than *Bug Hunt* and *Crossbow*, the same as *Barnyard Blaster*).

After blasting the one- or two-player sign and shooting to select one of 12 precincts on the city map, you hit the road in your police car, lights flashing. Your patrol car drives along the road, right to left, viewed from overhead. Some cars pass you, or you drive past them. Cars with stripes on them have crooks inside, and they are out to get you. The most logical thing to do is point your gun at the offending mobster's car and blast away, but nothing happens. There is a series of five arrows at the bottom right of the display, pointing from west to east at various angles. Shoot at one of these arrows to determine the direction of the bullets from your car at the criminals' car.

This is a frustrating screen to navigate. While you try to shoot the right arrows or hit the slider arrows to their left to adjust your car's position on the screen, the bad guys are blasting away with their tommy guns. It is difficult to get past this screen without simply holding your gun directly in front of the arrows and rapid-firing. Author James Zalewski admits he is disappointed with this one game screen, but it was Atari's specifications he had to meet.

At the end of a short and harrowing drive, you are at your first destination. After cleaning up the first precinct, you will no longer have to drive again, so long as you continue to select adjacent areas from the city map that are within walking distance.

You must clean up four types of hideouts. At the warehouse you will find gangsters popping up from behind crates and boxes, appearing in windows above and entering a door below. You might think it's time to shoot first and ask questions later. It isn't quite so simple, however, since there are innocent bystanders mixed up in this gun battle. Any time a pretty lady or a buxom blonde pops up, hold your fire. Ladies should never be shot at. Sometimes a kid will pop up, wearing a beanie and firing away like a big-time hood. Shoot him before you get plugged. Always make certain he has a gun in his hand and not a lollypop. Little kids with candy should not be shot at. Sometimes a fella will walk through a doorway, and then move on. Don't shoot him unless he pulls a gun and begins firing at you.

Any time the wrong person is shot, it costs you five bullets from your limited supply, which can result in an early demise for you. Blast all the bad guys with bullets to spare and move on to the next precinct. In the warehouse, in the alleys or in the downtown scenes, you will find street lights. The documentation doesn't mention it, but if you shoot these lights out, the gangsters' shooting accuracy will drop by 50% so that you die less often.

Down at the pier you may have the most fun, blasting the bad guys who pop up in the windows on the Sea Witch. Sometimes scuba diving hoods will surface in the water off to the right of the pier. Occasionally a young beauty floating in an inner tube will come along to distract you as well. On this screen there are fewer places for the hoods to pop up, so it seems easier to clean up. Zalewski has paid great attention to detail: The hats fly off the heads of blasted gangsters, they show painful expressions when you plug them, and even the scuba mask cracks when one of the divers is shot.

Over in the alley there are many windows and an open doorway to keep a sharp eye on. The gangsters are heavily armed and will stop at nothing to prevent you from busting up their ring. Lots of innocent bystanders make the alley tough to clean up.

When you get downtown, the battle rages to a peak. There is an Army Surplus store and Z's Bar and Grill brimming with bad guys. They appear in the windows, doorways, and even from under a manhole cover in the street. This screen seems to be the toughest to complete.

Each time you are killed, an impressive skull-and-crossbones is displayed. You are a cat of three lives, but that can be extended. After about the sixth area has been cleaned up, some special characters will begin to pop up occasionally. Blasting them may result in a bonus life or extra bullets. Some can cost you dearly as well, according to the author. These are not mentioned in the documentation and are left for the player to discover. There is one character you will recognize, however, if you have ever played *Barnyard Blaster*. These tidbits from James Zalewski enticed me to keep playing the game long and hard until I could complete all 12 precincts and finish the game, receiving a *Crime Buster* rating.

The game is played until either the hoodlums get the best of you, or you have cleaned up the city. A final rating, from Mobster through Detective, Unpluggable, and ultimately *Crime Buster* is presented. This is a thoroughly enjoyable game, even after completing it once. It is a lot of fun to set up a friendly competition with friends in the two-player mode as well.

Crime Buster is an impressive light-gun game. I am looking forward to more excellent works from James Zalewski and Atari Corp., which has been turning out some fine titles since the beginning of 1989.

Matthew Ratcliff, a frequent contributor to ANALOG Computing, *lives in St. Louis, Missouri, with his wife and two children.*

MACRO EDITOR

A macro is a sequence of keys that can be called up by pressing a single key. For instance, I could define a macro to type out LOAD "D:", and from that point on I would need only to type one keystroke to type that string. If you're a slow typist, macros can improve your efficiency when it comes to programming.

Using the Macro Editor

Macro Editor is designed to be as easy to use as possible. All you need to do is type in Listing 1, check your work with *BASIC Editor II* (found elsewhere in this issue), save it to disk and type RUN. *Macro Editor* allows a maximum of ten macro keys. Macros are activated by typing Shift+Control+#, where # is a number between 0 and 9. When you run *Macro Editor*, you will be prompted for each macro key. Use Start to tell the computer you are done entering one macro and to go to the next.

When you're finished, the program will ask if you wish to create an AUTORUN.SYS file. If you answer no, the macros will work, but will be lost when you turn the computer off. By creating an AUTORUN.SYS file, you are telling the computer to set up your macros every time you turn on your computer and boot DOS. It should be noted that the AUTORUN.SYS can be placed on a disk without *Macro Editor*; both are stand-alone programs. You can also run *Macro Editor* repeatedly; for instance, if I booted with certain macros activated (with an AUTORUN.SYS on my disk), or had already run *Macro Editor* once, I could run it again and change the macros. But remember, you must answer "Y" to the prompt if you wish to make the change to an AUTORUN.SYS.

Obviously there are many uses for the macros other than entering BASIC commands. For instance, if you are a Sparta-DOS or DOS XL user, you could define a macro to type out DIRECTORY <Return> for you. If you are a hard-drive user, a macro can type out a long pathname to get to a file.

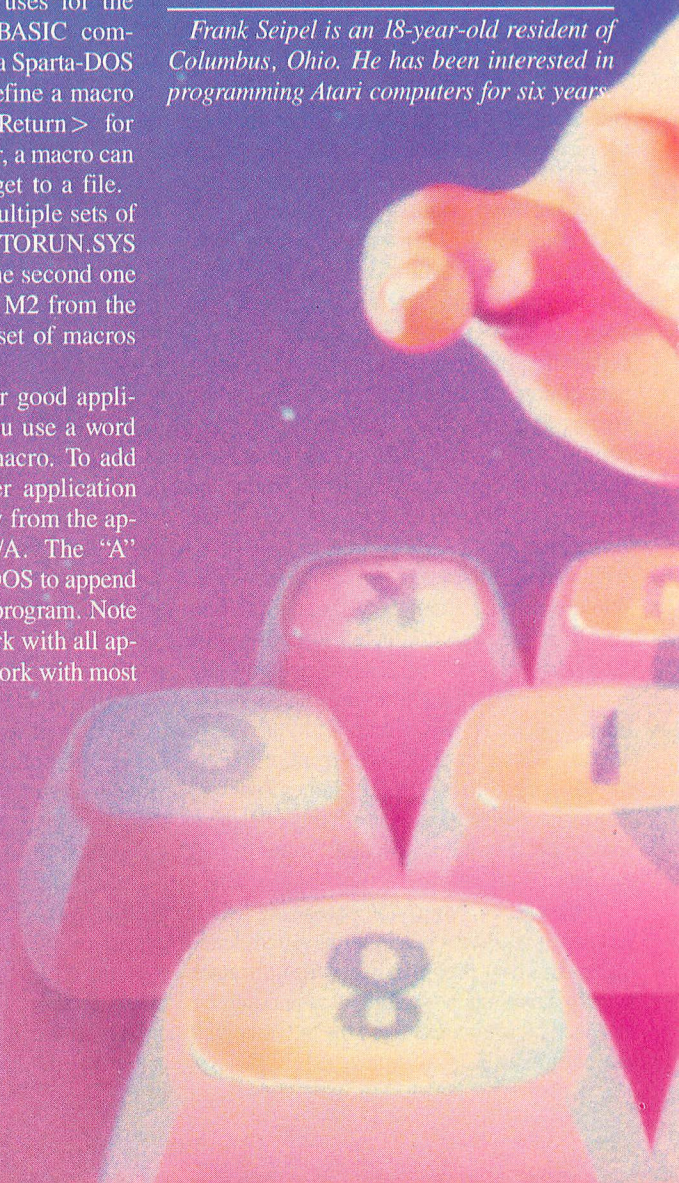
It is also possible to have multiple sets of macros. Rename the first AUTORUN.SYS created to M1. Then rename the second one generated as M2. Type M1 or M2 from the Dn: prompt to change which set of macros is currently active.

A word processor is another good application for using macros. If you use a word often, you could assign it a macro. To add the macro program to another application program, use DOS copy. Copy from the application to AUTORUN.SYS/A. The "A" stands for "append" and tells DOS to append the application over the macro program. Note that *Macro Editor* may not work with all application programs. It should work with most languages, however.

Macro Editor is Resetproof. In addition, if you don't intend to use all ten macros, you can define only the ones you want and then hit Break to save time.

Macro Editor resides on page six—almost *all* of page six—so keep this in mind before attempting to use other machine-language utilities, which also use page six.

Frank Seipel is an 18-year-old resident of Columbus, Ohio. He has been interested in programming Atari computers for six years.




5
6
7
R
T
Y
U
S
F
G
H
V

LISTING 1: BASIC

```
WO 1 REM ********************************
FE 2 REM *          MACRO EDITOR        *
DF 3 REM *        by Frank Seipel       *
ZD 4 REM *                              *
BT 5 REM *        COPYRIGHT 1989        *
RX 6 REM *     BY ANALOG COMPUTING      *
WU 7 REM ********************************
NN 8 REM
OI 18 GRAPHICS 17:POKE 710,14:POSITION 0,
   6:? #6;"   KEYBOARD MACROS":ACTIVE=PEE
   K(1536)=169
IX 19 ? #6:? #6;"   BY FRANK SEIPEL":? #6
   :? #6:? #6:? #6;"     please wait"
HN 20 FOR I=1536 TO 1672:READ D:POKE I,D:
   NEXT I
KV 25 GRAPHICS 0:IF  NOT ACTIVE THEN TRAP
    28:OLD=PEEK(1535):POKE 1535,104:X=USR
   (1535):POKE 1535,OLD
XC 28 ? :? "Enter macros.  Hit <Start> af
   ter":? "entering each macro.  Any"
HN 29 ? "<Return> keypresses will be part
    of":? "the macro.":?
PF 30 ? "A macro is a string of text --":
   ? "for instance, you could define"
DE 31 ? "Shift+Control+0 as LIST <Return>
   ":? "with this program -- and then typ
   e"
TG 32 ? "Shift+Control+0 instead of LIST.
   ":? :? "This program can be used to"
DM 33 ? "redefine the macros in memory, o
   r":? "write out an AUTORUN.SYS file to
   "
GO 34 ? "your disk, which will automatica
   lly":? "install your macros every time
    you":? "boot-up.":?
IJ 35 OPEN #1,4,0,"K:":OFFSET=0:OFFTABLE=
   1673:DTABLE=1686
HS 36 ? :? "Hit <Return> for next page":G
   ET #1,X:? CHR$(125):? :? "Sum length o
   f all macros may"
WA 37 ? "not exceed 128.  After using a":
   ? "macro, you may not use it again unt
   il"
XZ 38 ? "you have typed some other key, o
   r":? "used another macro key.":? "(If
   this is a problem, just hit"
PM 39 ? "<Return> before executing the ma
   cro)":?
GW 40 FOR I=0 TO 9
HA 42 POKE OFFTABLE+I,OFFSET
KB 50 ? :? "Enter macro for Shift+Control
   +";CHR$(I+48+128);:? "-->";
QR 60 IF PEEK(53279)=6 THEN GOTO 140:REM
   *SAVE*
QX 70 IF PEEK(764)=255 THEN GOTO 60
WK 80 POKE DTABLE+OFFSET,PEEK(764):GET #1
   ,X:? CHR$(X);
IR 130 OFFSET=OFFSET+1:GOTO 60
SC 140 POKE DTABLE+OFFSET,255:OFFSET=OFFS
   ET+1:FOR D=1 TO 500:NEXT D:NEXT I
ZB 200 ? :? :? "Would you like to make th
   ese":? "your default macros [i.e., wou
   ld you"
WE 210 ? "like to write an AUTORUN.SYS":?
    "consisting of these macros and the":
   ? "macro program to D1: ? <Y/N> -->";
IV 220 GET #1,X:IF X=ASC("y") OR X=ASC("Y
   ") THEN ? "Yes":? :? "Working...":GOTO
    30000
5M 230 ? "No":END
PU 29010 DATA 169,0,141,149,6,165,12,141,
   46,6,165,13,141,47,6,169,23,133,12,169
   ,6,133,13,160,48
LS 29020 DATA 162,6,169,6,32,92,228,173,1
   49,6,201,1,240,6,169,1,141,149,6,96,76
   ,224,7,72,138,72,173,147,6
KM 29030 DATA 208,44,173,9,210,197,0,240,
   17,133,0,162,0,189,127,6,205,9,210,240
   ,11,232,224,10,208,243
UG 29060 DATA 104,170,104,76,95,228,169,1
   ,141,147,6,189,137,6,141,148,6,76,80,6
   ,174,148,6,189,150,6,201,255,240,9
XP 29200 DATA 141,252,2,238,148,6,76,80,6
   ,169,0,141,147,6,76,80,6,242,223,222,2
   18,216,221,219,243,245,240
YU 30000 CLOSE #1:OPEN #1,8,0,"D:AUTORUN.
   SYS"
KF 30010 START=300:XEND=318:GOSUB 31000
WF 30020 START=1536:XEND=1791:GOSUB 31000
ES 30030 PUT #1,226:PUT #1,2
FM 30040 PUT #1,227:PUT #1,2
JQ 30050 PUT #1,0:PUT #1,6
IR 30060 CLOSE #1:END
JM 31000 PUT #1,255:PUT #1,255
PO 31010 CELL=START:GOSUB 31200
TV 31020 PUT #1,LOW:PUT #1,HI
IF 31030 CELL=XEND:GOSUB 31200
UD 31040 PUT #1,LOW:PUT #1,HI
TP 31050 FOR I=START TO XEND
FJ 31060 PUT #1,PEEK(I):NEXT I
WJ 31200 HI=INT(CELL/256):LOW=CELL-HI*256
   :RETURN
```

LISTING 2: ASSEMBLY

```
0100 ;******************************
0110 ;*                            *
0120 ;* ———Macro keys——— *
0130 ;*                            *
0140 ;* Written by: Frank Seipel *
0150 ;*                            *
0160 ;* December 30, 1988          *
0170 ;*                            *
0180 ;******************************
0190 ;
0200 SYSVBV = $E45F
0210 SETVBV = $E45C
0220 CH  =  $02FC
0230 KBCODE = $D209
0240 LASTKEY = $00
0250      *=  $0600
0260      LDA #0
0270      STA FIRSTRUN
0280      LDA 12
0290      STA DOSJUMP+1
0300      LDA 13
0310      STA DOSJUMP+2
0320      LDA #INIT&255
0330      STA 12
0340      LDA #INIT/256
0350      STA 13
0360 INIT LDY #START&255
0370      LDX #START/256
0380      LDA #6
0390      JSR SETVBV
0400      LDA FIRSTRUN
0410      CMP #1
0420      BEQ DOSJUMP
0430      LDA #1
0440      STA FIRSTRUN
0450      RTS
0460 DOSJUMP JMP $FFFF
0470 ;
0480 ; Actual code starts here
0490 ;
0500 START PHA        ; Save A
0510      TXA
0520      PHA         ; Save X
0530      LDA INPROGRESS ;already going
0540      BNE TYPEITOUT ; do next key
0550      LDA KBCODE  ;  compare key
0560      CMP LASTKEY ;  to last-
0570      BEQ EXIT    ;  quit if same
0580      STA LASTKEY ;   store last
0590      LDX #0      ; zero index
0600 LOOP LDA KEYCODES,X ; check if
0610      CMP KBCODE  ; key is a
0620      BEQ MACROPRESSED ; macro key
0630      INX         ; inc X
0640      CPX #10     ; done?
0650      BNE LOOP    ; no; do nxt
0660 EXIT
0670 ; Code to exit interrupt
0680      PLA
0690      TAX         ; Restore X
0700      PLA         ; Restore A
0710      JMP SYSVBV
0720 MACROPRESSED
0730 ; Initiate macro typing code
0740      LDA #1      ; Tell interrupt
0750      STA INPROGRESS ; to get going
0760      LDA OFFSETS,X ; get offset
0770      STA CUROFFSET ; store offset
0780      JMP EXIT    ; quit
0790 TYPEITOUT
0800 ; code to key macro
0810      LDX CUROFFSET ; get offset
0820      LDA DATA,X  ; get data
0830      CMP #255    ; end of macro?
0840      BEQ DONE    ; yes; quit
0850      STA CH      ; no; type
0860      INC CUROFFSET ; inc offset
0870      JMP EXIT    ; quit
0880 DONE
0890 ; End macro code
0900      LDA #0      ; Tell interrupt
0910      STA INPROGRESS ; quit typing
0920      JMP EXIT    ; and quit
0930 KEYCODES .BYTE 242,223,222,218,21
6,221,219,243,245,240 ; Codes for Macr
o keys (internal)
0940 OFFSETS .BYTE 1,2,3,4,5,6,7,8,9,1
0 ; Reserve RAM for offsets
0950 INPROGRESS .BYTE $00
0960 CUROFFSET .BYTE $00
0970 FIRSTRUN .BYTE $00
0980 DATA
0990 ; Macro key data table
```

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

```
THE ANALOG #76 DISKETTE CONTAINS 14
MAGAZINE FILES.  THEY ARE LISTED BELOW:

SIDE 1:

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

MACROEDT.BAS  BASIC    LOAD   MACRO EDITOR
MACROEDT.M65  MAC/65   LOAD   MACRO EDITOR SOURCE
SHOOTER .OBJ  ML       (#3)   SHARP SHOOTER
GUN     .ACT  ACTION!  (#1)   SHARP SHOOTER SOURCE
GUNREAD .ACT  ACTION!  (#1)   SHARP SHOOTER SOURCE
HANOI   .BAS  BASIC    LOAD   RECURSION
HEAPSORT.BAS  BASIC    LOAD   RECURSION
QUIKSORT.BAS  BASIC    LOAD   RECURSION
RAMDISK .OBJ  ML       (#4)   RAMDISK 800XL
RAMDISK .SYN  ASSEM.   LOAD   RAMDISK 800XL SOURCE
SKEET   .OBJ  ML       (#3)   SKEET SHOOT
SKEET   .M65  MAC/65   LOAD   SKEET SHOOT SOURCE
MLEDITOR.BAS  BASIC    LOAD   M/L EDITOR
EDITORII.LST  BASIC    ENTER  BASIC EDITOR II

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

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

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

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

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

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

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

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

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

continued from page 15

RECURSION

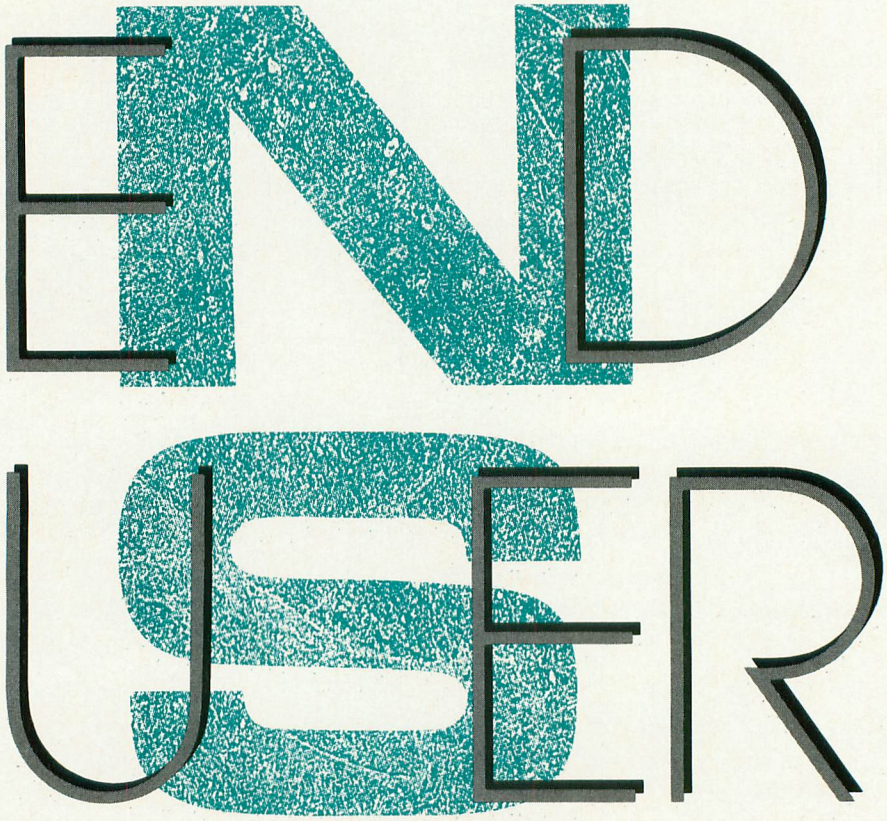
```
   R$(LEVEL/2+64);:REM Display level
MG 59 REM If there are entries to the lef
   t of PIVOT, change FIRST and LAST to t
   he new limits and immediately sort
EH 60 IF FIRST<PIVOT-1 THEN LAST=PIVOT-1:
   GOTO 20
QU 69 REM Restore the positions of unsort
   ed arrays (to the right) and sort
TP 70 IF LEVEL THEN FIRST=ASC(FIRST$(LEVE
   L-1))*256+ASC(FIRST$(LEVEL))
GP 75 IF LEVEL THEN LAST=ASC(LAST$(LEVEL-
   1))*256+ASC(LAST$(LEVEL)):LEVEL=LEVEL-
   2:? "{";:GOTO 20:REM ESC/BACK SPACE={
BA 79 REM ARRAY$ is already sorted, so a
   simple print is sufficient
UG 80 FOR A=1 TO COUNT:? ARRAY$((A-1)*SIZ
   E+1,A*SIZE):NEXT A
DG 85 ? :? COUNT;" records used",RAM-COUN
   T;" records left"
II 90 ? "Entry: ";:INPUT A$:IF A$="" THEN
    FIRST=1:LAST=COUNT:LEVEL=0:GOTO 20:RE
   M Call recursive sorting algorithm
OR 94 REM Simple routine to tack entry to
    the end of ARRAY$
ZU 95 A=SIZE*COUNT:FOR B=1 TO SIZE:ARRAY$
   (A+B)=" ":NEXT B:ARRAY$(A+1,A+LEN(A$))
   =A$:COUNT=COUNT+1:GOTO 90
```

continued from page 38

Sharp Shooter

```
FI

IF GunY > 95 THEN
  GunY = 95
FI

shift = yshift(index)

IF shift = 128 THEN
  GunY = GunY LSH 1
ELSE
  GunY = GunY RSH shift
FI

xx ^= GunX
yy ^= GunY

RETURN
```

END USER



by Arthur Leyenberger

Atari returns with new products and renewed purpose

Attending the latest COMDEX show is one of the many jobs of this Atari reporter. I'm not complaining, but over the last seven years I have attended approximately 16 COMDEX and Consumer Electronics Shows. I say approximately because in retrospect they all tend to blur together. Over the years, Atari, the computer industry and the technology have changed, but there is always something new to see and report on.

The 1989 Spring COMDEX (COMputer Dealers EXposition) was held in Chicago rather than Atlanta, the customary location. As a result, show attendance was down because of the cool weather. However, Atari made headlines with two new-product announcements and their renewed vigor for recapturing the U.S. ST market.

COMDEX is held twice each year, and at the last show in Las Vegas, Atari was talking but not showing. There wasn't much for Atari *to* show then. But that was due in part to their new policy of not discussing new products unless they will be shipping in 60 days. Although we have heard these claims before, Atari was saying (at the time) that 1989 would be the year they would return in force to the U.S. market.

It is well known that for the last several years Atari has been concentrating on the European market. With their limited human resources (Atari is a small company) and the recent DRAM (Dynamic Random Access Memory)-chip shortage, Atari was unable (and unwilling) to support ST sales in the United States. No advertising, fewer and fewer dealers and increasingly fewer new ST products from third-party vendors has left the domestic ST market to virtually wither away. And then there is the 8-bit market, which has seen little support from Atari for quite some time.

Well, all of this is old news. Atari has been claiming they will "shine in '89," so to speak, and from what I saw at COMDEX they might do just that. National media attention was given to Atari's ST laptop computer and the new "pocket" MS-DOS computer called Portfolio. More important is Atari's new attitude toward product availability.

I've briefly discussed Atari's renewed purpose above. According to Sam Tramiel, president of Atari Corp., the DRAM shortage is over (at least for Atari), so more STs can be manufactured and therefore be available for the U.S. Further, with product availability comes a reason for advertising. Although we may not see much in the way of television and radio advertising, Atari says they are committed to advertise nationally in the print media.

Atari readily concedes that they sacrificed the U.S. market in 1988 in order to maintain their position of leadership overseas. This means that they will need to work doubly hard in the areas of distribution and marketing to increase sales, attract new dealers and court developers. One area of continued success in the U.S. is the MIDI (Musical Instrument Digital Interface) market. Atari claims to have 35 percent of this market. Interestingly, the majority of new ST and Mega dealers are music stores rather than computer stores.

Atari was showing what will no doubt be a major success with musicians—the ST laptop. Originally named Stacy, and now renamed the Transportable, it was first dis-

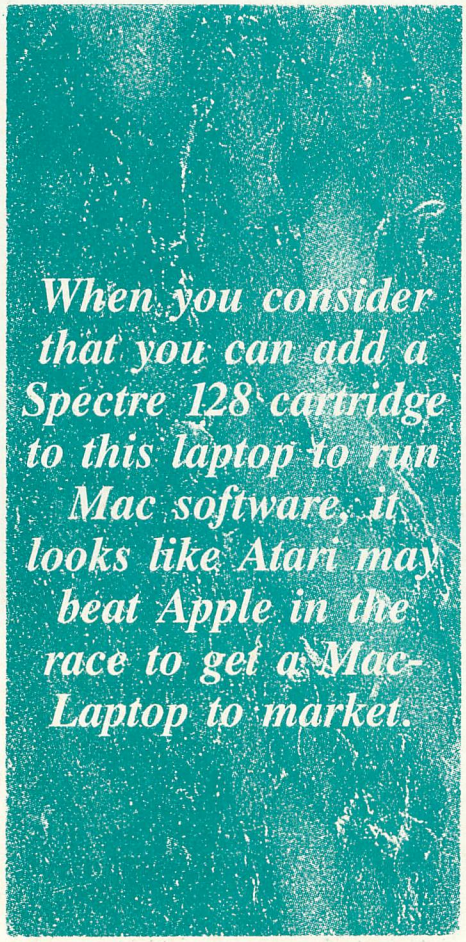
cussed last November at the Fall COMDEX in Las Vegas. However, it was not shown officially and consisted of a working prototype with exposed circuit cards and cables. Even the LCD screen was separate from the unit. A foam mockup of the final design was also seen last year, which, as it turns out, was similar to the final design.

The Transportable being shown was a working pre-production unit housed in a locked Plexiglas display case. A series of continuously running demos could be seen on the LCD screen. I had heard that the Transportable would be at COMDEX, but I feared the worst: that it would be too big, too heavy and unattractive. I'm happy to say I was wrong. The Transportable is attractive and about the size of other MS-DOS laptop computers.

The Transportable weighs in at 15.2 pounds, which is at the upper end of the weight range of comparable PC laptop computers. Using a 640- by 400-pixel supertwist LCD screen, the laptop offers the same resolution as the monochrome ST monitor. In addition, one megabyte of memory and a single 3½-inch (double-sided) floppy-disk drive are provided. According to Atari, an optional second floppy drive or hard disk can be added to the unit.

The Transportable has all of the ports and interfaces of a regular ST or Mega ST, including monitor, serial and parallel floppy and hard disk, MIDI, mouse and joystick. It can run on AC power or use its nonreplaceable internal battery pack. I have no idea how long the laptop will operate once the battery is fully charged, but I suspect it will be approximately 2-3 hours.

One of the unique features of the Atari Transportable is a built-in trakball on the lower right side of the keyboard. It is slightly larger than a ping-pong ball and used in place of a mouse to control the screen pointer. Two keys that function identically to mouse buttons are placed immediately above it. The trakball is a thoughtful addition, since using a separate mouse on a laptop is somewhat cumbersome.



I was able to spend a couple of minutes using the Transportable and came away impressed. The keyboard was surprisingly good—similar to that on a Mega ST. Although the built-in trakball seemed strange at first, I'm sure I could get used to it. A mouse port is provided in case you want to use a normal ST mouse.

With a list price of $1,500, the Atari Transportable should be a success, especially with musicians. When you consider that you can add a Spectre 128 cartridge to this laptop to run Mac software, it looks like Atari may beat Apple in the race to get a MacLaptop to market. And don't forget PC Ditto that gives MS-DOS compatibility. Having virtually three computers in one makes the Atari laptop unique and should increase its appeal.

The Atari Portfolio was the other new Atari product at COMDEX. Billed as a hand-held MS-DOS computer, it contains DOS 2.11 in ROM, 128K of RAM (expandable to 640K), an 8-line by 40-character LCD display and a 63-key QWERTY keyboard. The Portfolio uses an 80C88 processor like the original IBM PC and sells for $400. The unit is about the size of a videotape and weighs under a pound. Two standard "AA" batteries power the Portfolio for up to 48 hours of continuous use.

Built-in software includes a word processor, a spreadsheet that creates *Lotus 1-2-3*-compatible files, an address/phone list program and an appointment calendar. An interface jack is provided for exchanging data directly with a PC via a "smart cable." The Portfolio can also accept either ROM cards for software or RAM cards for data storage. Even standard PC peripherals, such as modems and printers, are said to be usable with the Portfolio by means of "card-cables" that insert into the RAM/ROM wafer slot.

The Atari Portfolio is an attractive product that appears to be functional too. I was permitted to "use" the Portfolio for a couple of minutes and was impressed. The unit felt solid, the overall design was clean and the keys had a good response. I doubt if I could type an entire article using the teeny-weeny keyboard, but I could certainly enter short notes with practice.

According to Atari, both the Transportable and the Portfolio should be available by the time you read this. Products like these demonstrate that Atari is trying to move forward with innovative products. Both the laptop and the Portfolio are niche products that should appeal to more than just the tradi-

with more Atari end users, Atari will be a stronger company. We'll all benefit from that.

Atari was showing another new product of interest to ST users. The Megafile 44 is a hard-disk unit that uses a removable 44-megabyte cartridge. The cartridge sells for $150 and has a relatively fast access time of 25 milliseconds. The unit itself sells for $1,200 and will be available by the time you read this.

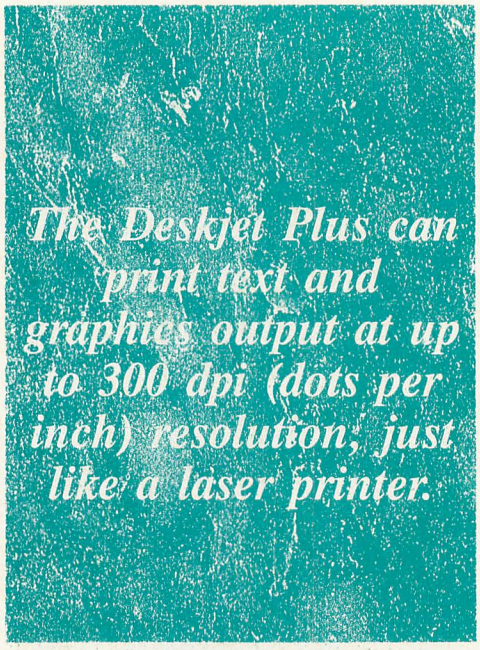
Other things

Technology is still the watchword at COMDEX. Intel introduced their new 80486 microprocessor, which contains a math coprocessor on the chip itself. It is primarily meant for the workstation and minicomputer markets. IBM had a prototype machine using the new chip at the show.

Several manufacturers of laptop computers—Mitsubishi, Toshiba, Sharp—were showing laptops with color LCD screens. Seeing color on a laptop screen is amazing, although the technology is still about a year from production. All of the models that were displayed were "under glass," and the companies refused to discuss any technical details.

Hewlett-Packard was showing a new version of the excellent Deskjet Printer at COMDEX called the Deskjet Plus. The Plus offers all of the features of the original model that has been available for about a year, plus much more. Using inkjet technology that Hewlett-Packard pioneered a few years ago with their ThinkJet printer, the new model yields laser-printer-quality output at the price of a high-end dot-matrix printer.

The Deskjet Plus can print text and graphics output at up to 300 dpi (dots per inch) resolution, just like a laser printer. Output is essentially indistinguishable from laser-printed output. However, the speed of printing is slower because the printer prints as it receives output from the a computer. The printer can print in draft mode at a speed of 240 characters per second (cps). The speed of the letter-quality mode is 120 cps. Graphics output is slower.



The most significant difference between the HP Deskjet and the new Deskjet Plus is throughput speed. Although draft- and letter-quality printing speed is still rated the same, throughput is said to be two to five times faster due to a faster microprocessor, paper pick-up mechanism and motor, which moves the paper through the printer in half the time of the original.

The Deskjet Plus contains more built-in fonts: six portrait and four landscape. Further, landscape printing is now possible without the need for an optional font cartridge. In addition, larger fonts are also included (up to 30 points) and the Plus can print on legal-size paper. The Deskjet Plus sells for $995 and the original Deskjet has been reduced to $795.

Atari shuffles the deck, and other exciting tidbits

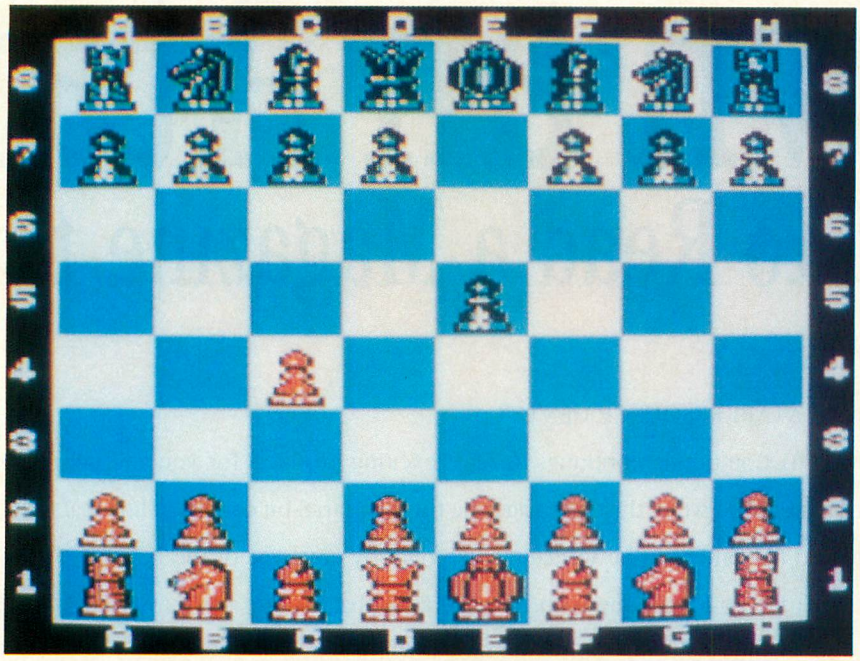
Sig Hartmann, longtime sidekick of Jack Tramiel and veteran of Atari Corp., has recently assumed the role of executive vice president of Atari Corp. and president of O.E.M. Sales. This post also encompasses government and institutional sales, but it is no secret that Atari has had difficulty breaking into the mainstream business market, which makes Sig's new job even more challenging. Sig has held just about every post at Atari Corp. and has the energy to get things done. We wish him the best of luck in his new position.

Other new faces at Atari include Joe Mendolia, new V.P. of marketing, and Tony Salerno, V.P. of U.S. Software. Joe came from Imagen and is now responsible for user-group support as well as Atari marketing. He will be primarily responsible for the strategy and not-so-trivial implementation of Atari's return to the U. S. market. I have him to thank for allowing me to get my paws on both the ST laptop and the Portfolio, even if it was only for a few fleeting minutes. Thanks, Joe.

Tony Salerno comes from Borland International, a company specializing in utility and language software for the PC and Macintosh. Tony will be responsible for technical support, equipment sales and developer support. Let's hope he is successful in these areas, especially with developer support. It would be great if Atari could support developers the way other major companies do. A world-class effort would surely keep the existing developers in the Atari fold as well as attract new ones—something we clearly need.

It's official. Shiraz Shivji has left Atari. Who is Shiraz, you might ask? Oh, just the so-called father of the Atari ST computer. Shiraz's engineering brilliance allowed Jack Tramiel to introduce the original ST four years ago. He has also been actively involved in the Mega series and other peripheral products. We wish Shiraz success in his new endeavors and hope Atari can find an equally talented engineer to replace him.

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



THE CHESSMASTER 2000

The Software Toolworks
One Toolworks Plaza
13557 Ventura Boulevard
Sherman Oaks, CA 91423
(818) 885-9000
XL/XE cartridge: $39.95

Reviewed by Matthew J.W. Ratcliff

The Chessmaster 2000 is the most sophisticated chess program since *Sargon III*. This finely crafted program and its complete documentation will help you learn to play, from the basics through tournament-level expertise, with some 100 classic sample games on the data disk. The sample games begin with Greco in 1620 and go through the Karpov versus Kasparov world championship in 1985, ending with two examples of *Chessmaster* vanquishing *Sargon III* in 1986.

Chessmaster comes with a book that introduces chess basics, with all the moves and terminology explained. A basic point system for each captured piece is presented to help you keep track of their value during game play. Information on joining the U.S. Chess Federation is also provided, if you want to get really serious about the game.

The reference continues with a history of chess as it developed into its modern form. Another brief history of the game is presented in terms of the world champions and their playing styles. Next, a section on chess and machines leads us from the earliest mechanical players through the latest computer-based game-playing algorithms. Finally, the reference guide presents the 100 classic sample games, which are on the enclosed data disk, followed by sample *Chessmaster* problems, solutions and a bibliography. I found this entire book fascinating reading.

A handy reference guide provides information on booting the program and executing game controls. The escape key toggles between the main menu screen and the finely detailed graphic display of the chess board and pieces. Game control may be carried out with the joystick or via keyboard input. An alphanumeric grid is displayed around the board, which is used for positional references of the pieces.

Chessmaster accommodates newcomers by allowing them to turn on the easy mode and select a play level of zero. Castling, *en passant* (a move that allows the capture of an opponent's pawn "in passing") and pawn promotion are fully supported. At any time, you may simply press Return to change sides and take on the opponent's pieces as your own. A classic game may be loaded and played for your learning enjoyment. You may also save a game to disk to finish at a later time if so desired. Complete disk-management functions are provided for cataloging games, deleting, loading, solving mates and printing a game history.

Chessmaster will play in a "coffeehouse" mode, if you choose, playing a more relaxed style suitable for casual players, rather than adhering to a strict tournament format.

As you become more adept at the game, you may program *Chessmaster* to play from level 0 through 19. The higher the level of play, the tougher *Chessmaster* is to beat and the longer it will take to make each move. You may toggle the easy mode at any time as well, thus disabling *Chessmaster*'s think-ahead capability while waiting on your move.

Play modes may be selected for human against *Chessmaster*, human against human or *Chessmaster* against itself. If you want to see how this program considers each move, you may "show thinking." When this is enabled, each possible move *Chessmaster* considers will be displayed. As you learn the game, you may request a hint, and the best possible move for you to make will be revealed.

The teach mode is excellent for novices or rusty players like myself. Whenever you select a piece, all possible moves are highlighted on the board. Teaching may be toggled on or off at any time, as can sound effects, which are simple audio cues for each move.

Chessmaster provides complete control over screen and chessboard colors, and you may rotate the board for a different perspective. According to the documentation, either a two- or three-dimensional graphic display may be selected, but it seems the Atari 8-bit version works only in the 2-D mode.

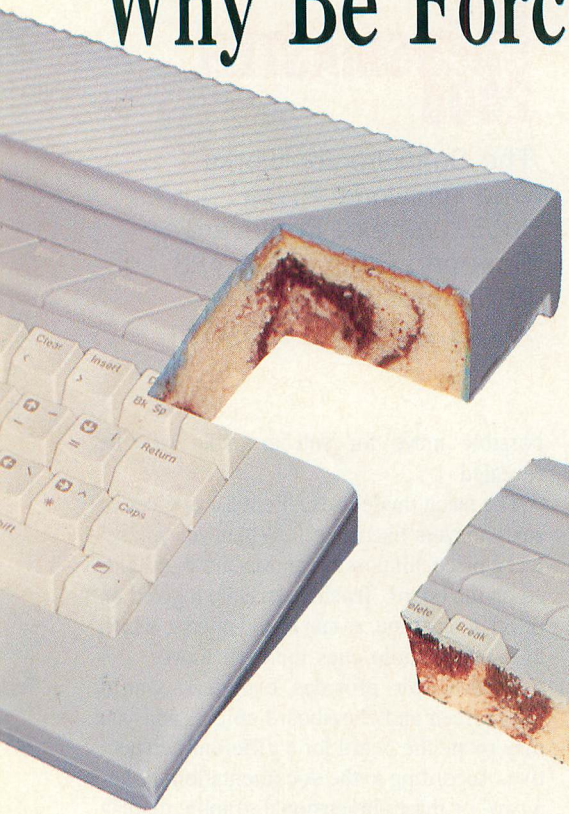
To play out hypothetical situations, you may set up the board. An additional menu and set of controls make customizing a board layout simple. This can be used to set up a handicap against a better player or to help you plan out strategies alone.

The Chessmaster 2000 is a superb game for two players who don't want to mess with the clutter of a real chessboard and pieces, and who want the convenience of a quick and safe way to store an incomplete game. With a printout of the game play, you can go back and study where you went wrong or simply play it back on the screen. *Chessmaster* will also serve as a first-rate chess tutor and help boost your status in the chess club. It is a finely crafted product, with complete documentation and near-perfect game play to hone your skills and make you a first-rate chess player.

Matthew Ratcliff, a frequent contributor to ANALOG Computing, *lives in St. Louis, Missouri, with his wife and two children.*

You Own an Atar

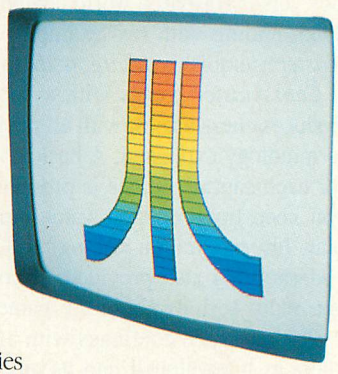
Why Be Forced to Read a Magazine that



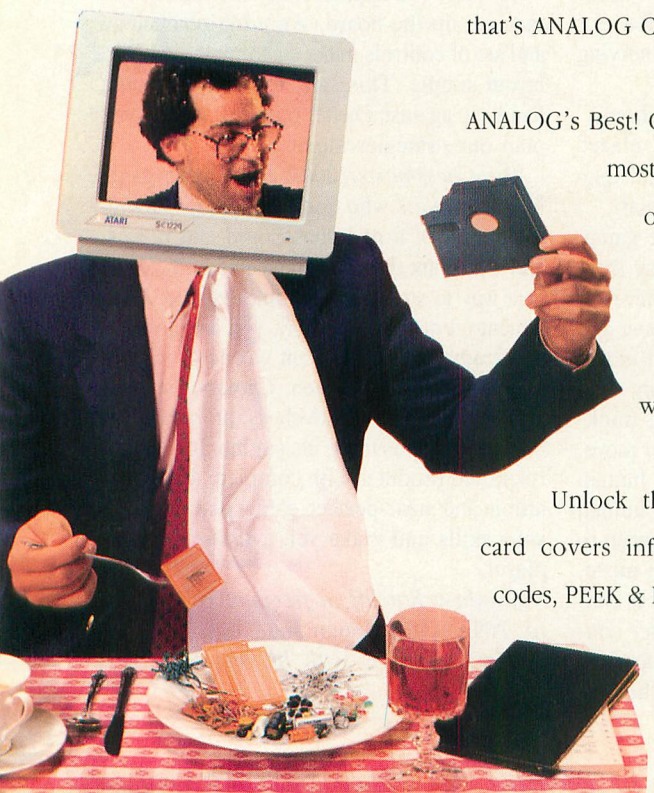
YOUR ATARI RESOURCE CENTER

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

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



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

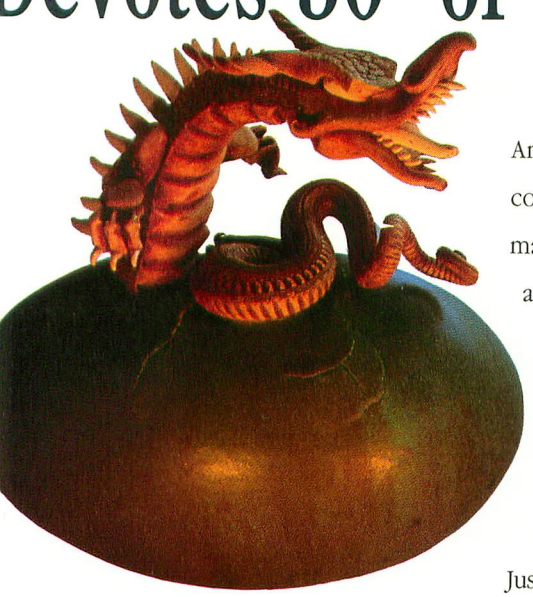


ANALOG's Best! Over 88 of ANALOG Computing's best and most requested programs are now available on this series of ten diskettes. The programs are all ready to run and come with complete documentation on the flip side of each floppy diskette. Select from Graphics, Educational, Utilities 1, Utilities 2,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

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

i 6502 Computer.

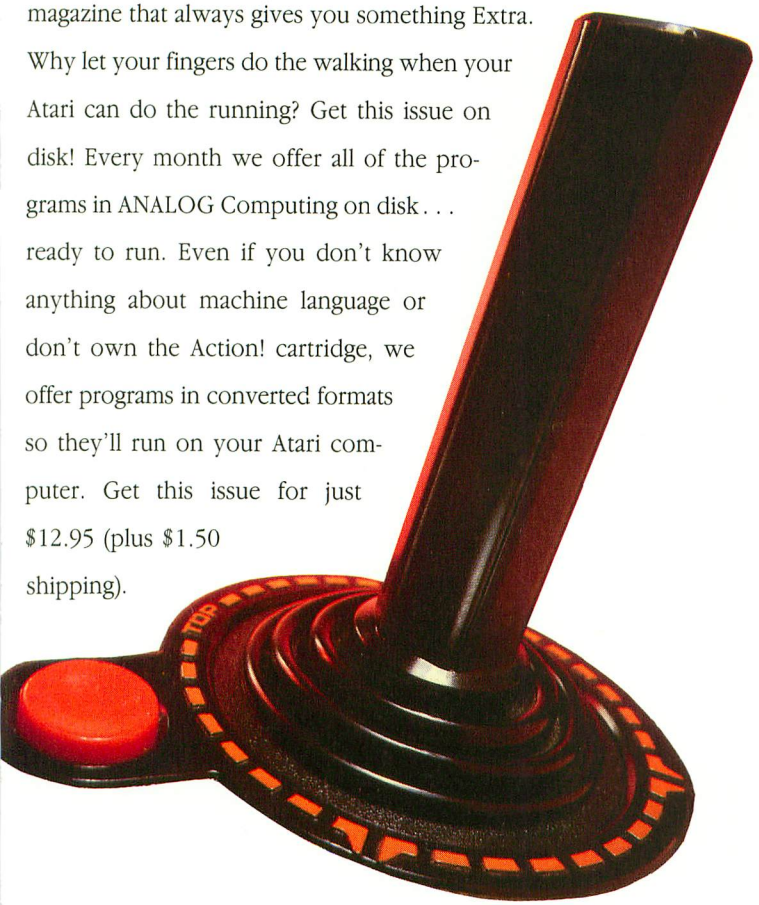
Devotes 50% of Its Pages to the Atari ST?



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

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

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



ANALOG COMPUTING

ANALOG COMPUTING OFFICIAL ORDER FORM

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

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

☐ Payment Enclosed Charge My ☐ VISA ☐ Master Card

Card # _____ Exp. _____

Signature _____

Name _____

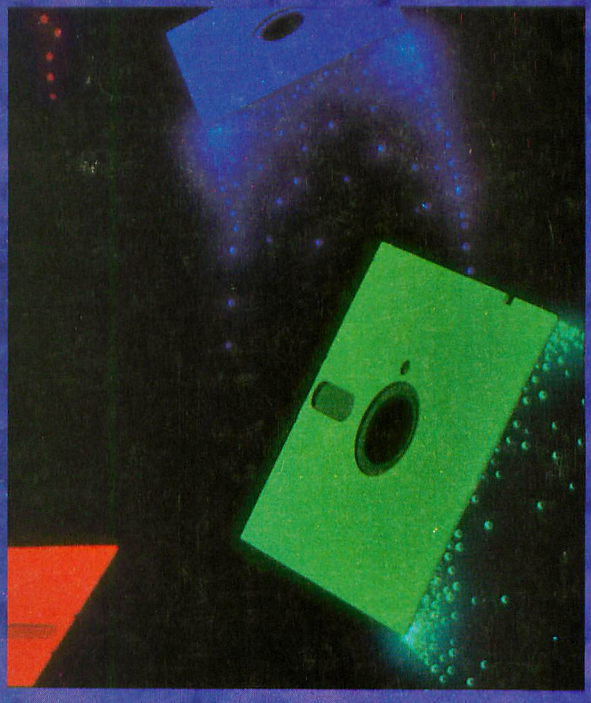
Address _____

City _____ State _____ Zip _____

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

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

INSIDE THIS ISSUE:



MORE

BOOT CAMP

PLUS

END USER

DATABASE DELPHI