The #1 Magazine For Atari Computer Owners

ANALOG COMPUTING T.M.

ISSUE 74
JULY 1989

DISK VERSION $12.95

SPECIAL GRAPHICS ISSUE:

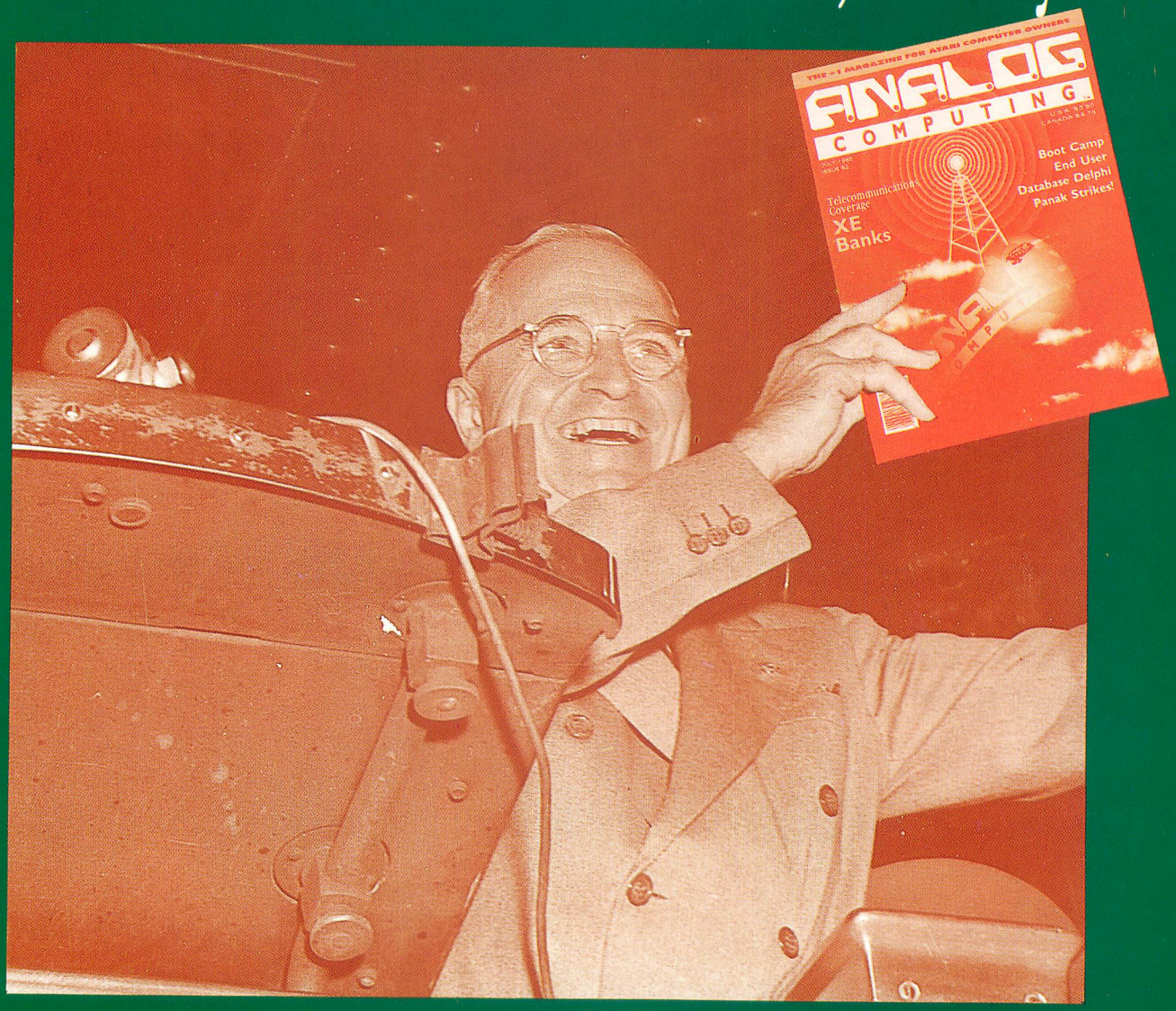
CHAOS
CHARACTER SET
DISPLAY UTILITY
DEGAS VIEW

REVIEWS:

CRYSTAL CASTLES
INTO THE EAGLE'S NEST

0 74369 50077 8 07

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 Andy Eddy

It's an understatement to say that the Atari community is in turmoil—Atari computer owners are consistently barraged by soothsayers ringing the death knell for their favorite computers and the company that makes them—even though the machines have given many thousands of people good service down through the years. And through it all, Atari has survived, in one form or another, to bring further surprises to the marketplace.

Having just returned from the World of Atari show in Anaheim, California, it's nice to see a few good companies still supporting the 8-bit line of computers. Though many *power users* feel lost without an "-est" computer—lat*est*, fast*est*, bigg*est* and so on—we all know how capable the Atari 8-bit line has been for so long. And with a few companies still striving to better them, we can continue to be productive and entertained.

But the most important support that Atari computer owners get is from fellow users, as the pages of ANALOG Computing testify. Each month we present informative reviews and articles, as well as type-in programs and tutorials, all written by your fellow Atari users. And this month, of course, is no different.

For example, much has been said over the last few years about the graphics abilities of the ST line of computers. Even so, we all know that the 8-bit line is no slouch in the graphics department. To prove it, Matt Ratcliff has put together a program that lets 8-bit users look at high-resolution, DEGAS-format ST pictures, with virtually no loss of detail.

When you consider the power of the hardware, it's not hard to understand how Matt and others have stretched it beyond what most expect it could handle. After all, the Nintendo Entertainment System uses a 6502 processor, the same one that's used in the Atari 8-bit line, and we all know how popular the NES is these days! Along the same lines, there's a new game machine called the TurboGrafx-16—known in Japan as the PC Engine—due to be released shortly with an updated version of the 6502, and the output of this console is nothing short of phenomenal.

The capabilities are there, waiting to be tapped. All it takes is a mixture of ingenuity, imagination and time. And don't forget that your programming efforts can earn cold, hard cash by being published in ANALOG Computing. If you have something that 8-bit owners can use, please send it to ANALOG Computing, P.O. Box 1413- MO, Manchester, CT 06040-1413 for consideration. Aside from the fame and fortune you'll receive, it'll give your fellow Atarians a little more "-est" for their computer.

Cont

on page 10

on page 18

on page 74

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

ANALOG COMPUTING STAFF

Publisher
LEE H. PAPPAS

Executive Editor
CLAYTON WALNUM

Art Director
KRISTEL PECKHAM

Associate Editor
ANDY EDDY

Managing Editor
DEAN BRIERLY

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
CHARLES F. JOHNSON

Contributing Editors
MICHAEL BANKS, FRANK COHEN, MATTHEW J. W. RATCLIFF

Cover Photography
GARRY BROD

Models
JOHN BERADO
NORMA EDWARDS

Cover Illustration
ALAN HUNTER

Illustrations
JOHN BERADO

Copy Chief
KATRINA VEIT

Copy Editors
NORMA EDWARDS
RANDOLPH HEARD
KIM TURNER
SARAH WEINBERG

Editorial Assistant
PATRICIA KOURY

Chief Typographer
ALICE NICHOLS

Typographers
DAVID BUCHANAN
B. MIRO JR.
QUITA SAXON

Contributors
ALFRED L. ACOSTA
DAVE ARLINGTON
MATT FRUIN
TOM HUDSON
BARRY KOLBE
BRYAN SCHAPPEL
DAVID SCHOCH

Vice President, Production
DONNA HAHNER

Executive Art Director
PAMELA CARR

Advertising Production Director
JANICE ROSENBLUM

Advertising Production Coordinator
MAGGIE CHUN

National Advertising Director
JAY EISENBERG
(213) 467-2266
(For regional numbers, see right)

Corporate Ad Director
PAULA THORNTON

Subscriptions Director
IRENE GRADSTEIN

Vice President, Sales
JAMES GUSTAFSON

JULY 1989
ISSUE 74

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**

READER

COM-DOS Plus

I want you to know that I typed in and enjoy using COM-DOS, which was featured in your June 1988 issue. Unfortunately, this version formats disks only in single density, and I have a 1050 drive and like to use enhanced density.

Through a fortunate error on my part, I now have a version of COM-DOS that will, to my great delight, format in enhanced density. Here is how it happened. In your July 1988 issue, there appeared a short binary program called *Boot Directory*. I placed this on my utility disk with DOS 2.5. I wanted to try it out under COM-DOS, so I placed a disk with COM-DOS on it in my drive and ran *Boot Directory*, which was still in memory.

Since the disk operating system was DOS 2.5, the *Boot Directory* file rewrote DOS 2.5 to my COM-DOS disk; thus my COM-DOS became enhanced. The DUP.SYS that accompanied COM-DOS works along with the enhanced version as if nothing happened. I was shocked and surprised to see "999+ FREE SECTORS" when I formatted a disk with what I thought was my old version of COM-DOS.

Thanks, ANALOG, for supplying the programs that make my computing more enjoyable—even when I foul up.

—Ed Thoma
Georgetown, PA

A Plea to Abacus

For quite some time I have wanted a FORTRAN compiler for my 130XE. To my knowledge one does not exist. Recently, I came across an article about a company called Abacus that sells a FORTRAN-64 compiler for the Commodore 64. I thought that if such a compiler could be written for a C-64, maybe one could be written for an Atari 8-bit. The response from Abacus was a very cold "we have never written anything for the Ataris and never will." Perhaps if they were to receive a truckload of letters from interested Atari users, their attitude would change. The address is 5370 52nd Street SE, Grand Rapids, MI 49508.

—Bill Tart
Houston, TX

Actually, it's not true that Abacus has never written anything for Ataris. They don't have any products for the 8-bit computers, true, but they have an excellent set of books for people who own STs.

The Dark Side

I am tired of the gloom and doom coming from the pages of your magazine. I have just read another article on the fine programs of the "past." In the previous issues I have read a few articles telling me of the hatred between the users of the older line of computers and the newer STs.

I am writing this letter using *The Writer's Tool*. This program helped me get through

COMMENT

college. I learned how to program by using Atari BASIC and now use BASIC XL. I use *Chessmaster 2000* daily to stay ahead of my fellow chess players. The list of great game software is endless.

Because of Atari's computers, OSS's and other third-party software, and books and magazines such as yours, many of us are more computer literate. I will be updating to a more powerful computer soon, but I won't quit using my Atari. Please keep the whining out of your editorials and keep up the good work.

—Chuck Yates
Memphis, TN

Whining? Hey, we don't make the world the way it is, we just live in it. If you don't want us to talk about "old" software, (much of which, I might add, is new to a majority of our readers, since they haven't had their computers that long), and there aren't any new products to talk about, what software should we cover? The large majority of letters we get would also be classified as "whining" by you. The number of these letters tells us that there's a problem out there, one that ANALOG Computing *surely didn't create. The real questions are: Do we ignore the situation? Or do we report on it?*

Double-sided Disks?

I was pleased to see the what and the how of Atari's XF551 brought to light, but one minor error caught my eye. In his review Mr. Ratcliff attributes the XF551's inability to format double-sided disks to an intentional design consideration on Atari's part. Not so. As the owner of an ATR8000 (remember those?), I discovered long ago that standard drives will not format disks without a timing hole (unlike the 810 and 1050 drives, which use an internal timer).

This proves once and for all that the XF551 does indeed use a standard mechanism, and that the formatting limitation is not the result of any kindheartedness by Atari. On the few occasions when I've absolutely had to format a double-sided disk, I've found it is indeed a feasible, if somewhat dicey, proposition to create a second index hole in the disk jacket with a hole punch (taking care to line up the puncher with the hole already in the disk itself).

—Clay Halliwell
Springfield, MO

Game Design Fan

I have owned my Atari 800XL for about three years, but I had owned an Atari 600XL for about two years before that. When I got these computers, I was interested in playing and writing game programs. So when I saw that ANALOG had the *Game Design Workshop*, I decided to check into it. I was introduced to Player/Missile Graphics earlier, but I only understood a small amount of it. When I read GDW it answered a lot of questions I had.

GDW is helping me design my first BASIC game, so please go back in your files and bring out old articles that will help me better understand my computer.

Also keep putting those wonderful game programs in your magazine, no matter how long they are. I love them.

—Dennis Debro
Birmingham, AL

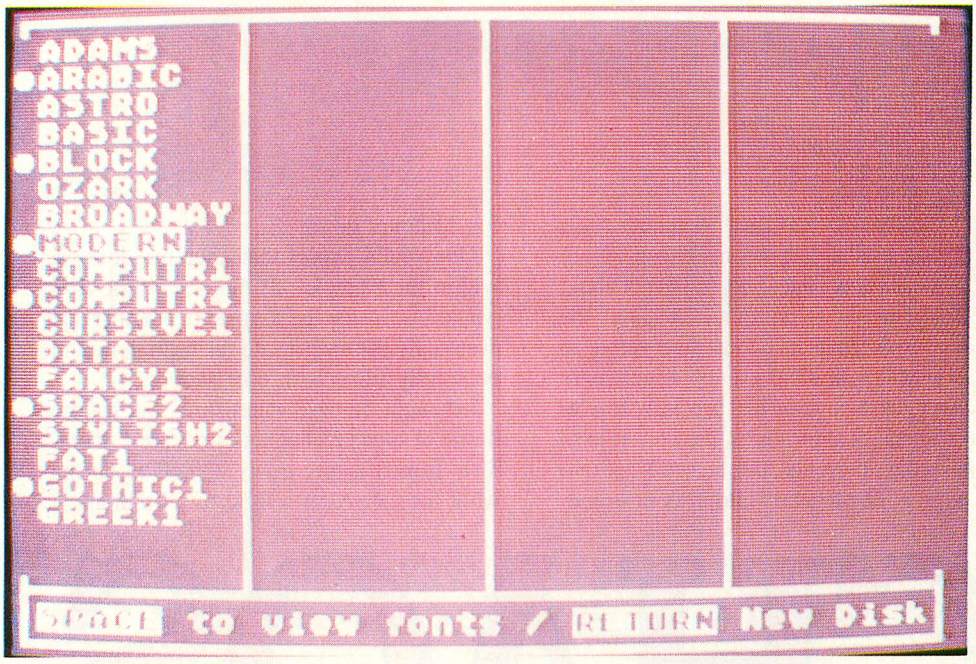
We thank you for your kind words. Hopefully, many other readers also found the Game Design Workshop *to be a helpful series.*

Character Set Display Utility

by Dave Arlington

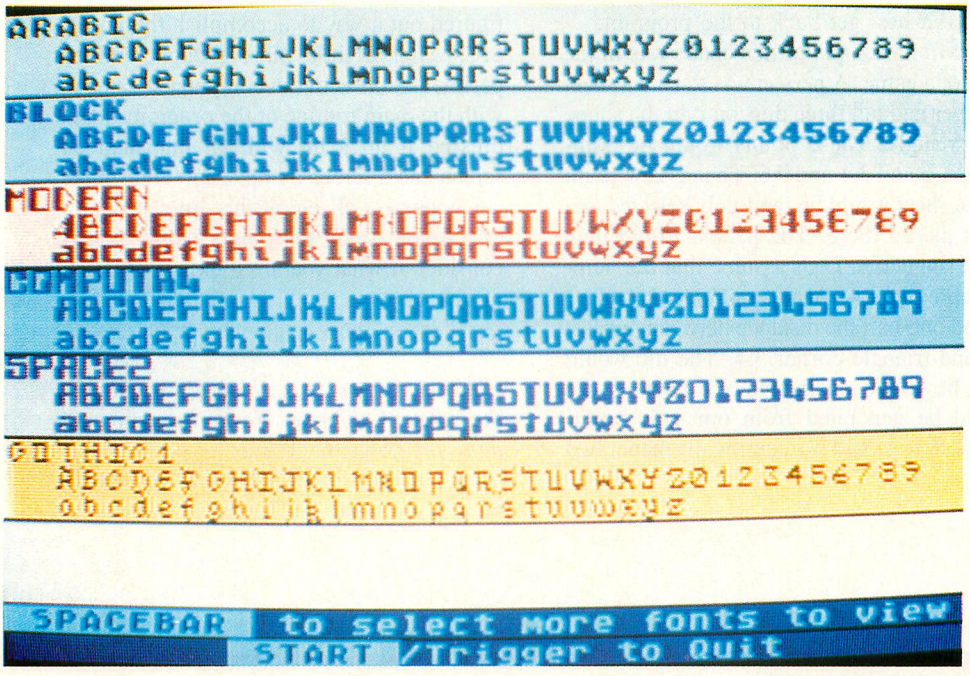
Ever since the dawn of time (well, at least since the beginning of Atari 8-bit computers), one of the most popular type of utility programs for the Atari has been the character set editor. Every Atari magazine worth its salt has published at least one, and several commercial ones have existed as well. One of the best was *Create-A-Font* by Vince Erceg published here in ANALOG a long while back.

Equal or exceeding in quantity are the programs that *use* redefined character sets. In addition to using them in your own programs, many public domain and commercial programs exist that employ them. The result, for most Atari owners, is a vast collection of character sets—if you're like me, you've ended up with a disk containing 40-50 different fonts. To make things worse, many have confusing or similar names.



I had fonts called STANDARD, ATARI, ROM, and DEFAULT that I had a sneaking suspicion were all just the standard built-in Atari character set. You might also have FANCY1, FANCY2, FANCY3 and FANCY4, or maybe ARCHAIC, ANCIENT, ROMAN and ARABIC, all sounding similar.

continued on page 44



CHAOS

by Alfredo L. Acosta

A few weeks ago I was watching *Nova* on public television. The program was about chaos; no, it was not about freeways or politics, but about the chaos that is being discovered in physics. Many physical events, which were once thought to be governed precisely by elegant mathematical formulas, are now seen to contain chaotic events. But before I get into the physics of planetary motions or something, I'd best get back to the program.

The first example of this chaos was in the form of a game. A person was to take a sheet of paper and put three dots on it in the shape of a triangle. These points would be numbered 1,2 and 3,4 and 5,6. Then a new point on the sheet would be randomly selected and a dot placed there. The person would then roll a single die. Let's assume that the number rolled was 4. A new dot would then be placed one-half the distance between our last dot and triangle corner 3,4. The die would then be rolled again and yet another dot would be generated from our new point. These dots would be continually generated for 24 hours or so.

One would expect that once the dots got within the triangle, all the rest of the dots would stay inside the triangle (true), and that eventually the area inside the triangle would be solidly filled with dots (not true).

I thought this was interesting, but had no desire to spend 24 hours to find out. I therefore turned to my trusty 130XE to do the work for me. After a little bit of thought, I figured out a way to accomplish this in BASIC. After a little more thought and experimentation with the computer, I came up with the main routine of the program. I then dressed it up a bit with an informational screen and a screen dump. Short and simple.

I hope you all enjoy this little program. While you watch the points being plotted, consider at what you are looking: order out of chaos!

Alfredo L. Acosta is a biologist working for the state of California. He has been programming for about six years.

continued on page 57

New Home Productivity Software

JVB Software has released *Laserteller*, an all-in-one program that includes the ability to handle a checking and savings account, as well as keep an on-disk note and address book. Modeled after ATMs (automatic teller machines), the program allows you to quickly handle checking transactions. It also includes a "core" version of the program that lets you enter deposits and withdrawals, or check previous entries, without having to load the entire program.

Laserteller's graphics are unusual for a home-productivity program, with each screen looking like a futuristic ATM. The program sells for $14.95 plus $2.50 shipping and handling.

JVB Software
6538 Hazeltine Avenue
Van Nuys, CA 91401

CIRCLE #111 ON READER SERVICE CARD.

8 BIT NEWS

BBS Listing

Now available from Bubeck Publishing is the *1989 BBS Bible*, a comprehensive listing of almost 5,000 bulletin-board systems throughout the country. According to Bubeck Publishing, "The goal of the *1989 BBS Bible* is to make BBSs more accessible, both to those who already use them and those who have a computer and a modem but haven't yet contacted any. There is no other such publication available today."

Also included in the book are articles on accessing BBSs, low-cost long-distance services, modems, computer viruses and other topics.

The *BBS Bible* will be updated annually and is available for $24.95 plus $3 shipping and handling. (Pennsylvania residents must add $1.68 sales tax.)

Bubeck Publishing
P.O. Box 104
Collegeville, PA 19426
(215) 287-6356

CIRCLE #112 ON READER SERVICE CARD.

More No Frills

No Frills software has just released their second program for the Atari 8-bit computers. Called *PS Users Utility Disk*, the package not only includes various utility programs for use with *Print Shop*, but also two complete printing utilities for creating labels, bookmarks and coupons.

Some of the *Print Shop* utilities include an icon viewer, which can load up to four icons at a time; an icon cataloger, which will print icons to Epson- and Prowriter-compatible printers; a font cataloger; a border cataloger; and a utility to delete, undelete or rename *Print Shop* files.

The *PS Users Utility Disk* sells for $29.95 plus $2.00 shipping.

No Frills Software
800 East 23rd Street
Kearney, Nebraska 68847 (308) 234-6250

CIRCLE #113 ON READER SERVICE CARD.

APX Program Now Shareware

Mastermatch, a board game requiring logic and concentration, was originally released by the now-defunct Atari Program Exchange (APX). This program, which is a version of the popular puzzle game, Mastermind, originally sold for $24.95. It has now been given an extensive upgrade and is available as shareware.

Anyone interested in obtaining this program, along with complete documentation should contact Chuck Mullally at the address given below. Note that, because it is shareware, this program may be copied and freely distributed, though the author requests a $10 donation from anyone who enjoys the program and who would like the complete documentation.

Chuck Mullally
2925 Scenic Drive
Muskegon, MI 49445

CIRCLE #114 ON READER SERVICE CARD.

ST NOTES

by Frank Cohen

DRAM Solutions

"The DRAM crisis is over," said Sam Tramiel, president of Atari Corp, at last fall's COMDEX computer trade show in Las Vegas, Nevada. The limited supply of ST computers in the U.S. has been linked to a worldwide shortage of memory chips, Dynamic Random Access Memory (DRAM) chips. The ST depends on large numbers of these special high-capacity memory chips. American protectionist legislature and supply restrictions by the Japanese electronics cartel MITI caused a worldwide shortage of DRAM chips in 1988. The result was a 500% increase in DRAM chip prices, and the limited availability of STs in the US.

The solution to Atari's supply problems seems to have been worked out. Tramiel told of three contracts with major DRAM-chip manufacturers to insure an adequate supply of memory chips during 1989. Unfortunately, this does not lower the price of DRAM chips; it only insures availability.

520ST owners have been stunned to find the economic reality of the DRAM crisis. DRAM prices have put several ST computer memory-board-expansion manufacturers out of the market. E. Arthur Brown Company, formerly the manufacturer of a two- and four-megabyte expansion board for the 520 and 1040 ST, discontinued their memory board because of the limited availability of DRAM chips. Brown also noted that the high price kept customers from taking the plunge.

One solution to the DRAM crisis is to use surplus DRAM chips, which are found in close-out lots from chip manufacturers, removed from old computers, and taken from faulty memory boards. Although surplus DRAMs are not as reliable as new guaranteed chips, yields of good chips make up for the time and effort required to get the surplus chips. California Development Labs has developed a portable DRAM-chip tester for this purpose.

The RT1 is a hand-held battery-powered DRAM-chip tester that weighs less than one pound. A nine-volt Alkaline battery powers the unit. Inside the RT1 is a microcomputer system that can test and find faulty DRAM chips.

Some DRAM-chip suppliers have become sneaky. You might find a DRAM chip marked as a 256K chip, when in reality it is a 65K chip. The RT1 identifies all types of 64K, 256K and one-megabyte DRAM chips, indicating the chip's actual internal configuration on several LED indicators.

The RTI is easy to use. Just plug a DRAM chip into the RTI socket and press TEST. An audible alarm and a couple of LED indicators tell you within five seconds if the DRAM chip is good or bad.

California Design Labs
18323 Parthenia Place
Northridge, CA 91325
(818) 885-0058

Local Area ST

Atari continues to call the ST a business machine, and the rest of the world ignores it. Why is Atari's claim of the ST as a business machine taken so lightly? Because the ST lacks two vital ingredients: expansion slots and local area networking (LAN). The IBM PC is virtually swimming in expansion boards that add extra memory, hard-disk controllers, and other external hardware expansion boxes. Now consider for a moment that the most popular hardware expansion for the ST is a monitor A-B switch box and a bag of springs to stiffen the keyboard.

LAN provides a seamless interface between several computers. To the computer user, a distant computer's hard disk appears as just another device on the user's desktop. When a file is read, the LAN sends and receives messages across a transmission line between the two computers. LAN software patches itself into the deepest part of the operating system, so it works transparently with all applications, such as word processors, spreadsheets, databases, etc.

Several attempts have been made at creating a LAN for the ST computer. John Demar of QMI began writing *ST NET* in 1985. DeMar's system used the MIDI ports to communicate information between ST systems. From the GEM Desktop, a floppy diskette appeared as another disk icon on a networked system. If completed, *ST NET* would have become the most popular networking system. *ST NET* has not been completed due to several technical constraints.

DeMar found himself trying to pry open an operating system locked into read-only memory (ROM). The TOS operating system was not designed with a LAN in mind, so patching TOS is a very complicated matter.

The second difficulty was in using the MIDI ports as a communication system. DeMar found the transmission time to be too slow. MIDI transmits data at 31.5 thousand bits per second. When transmitting a file of 10,000 bytes, *ST NET* took more than 40 seconds to complete the transmission. This might not seem very slow, but a file copy function of 10,000 bytes takes less than five seconds to complete.

A new version of TOS, the ST's ROM-based operating system, has recently been released to ST developers. DeMar is once again looking into the operating system to see if a software-based LAN can be created using the new TOS. If the software can be created, the only other obstacle will be the physical connection between ST computers.

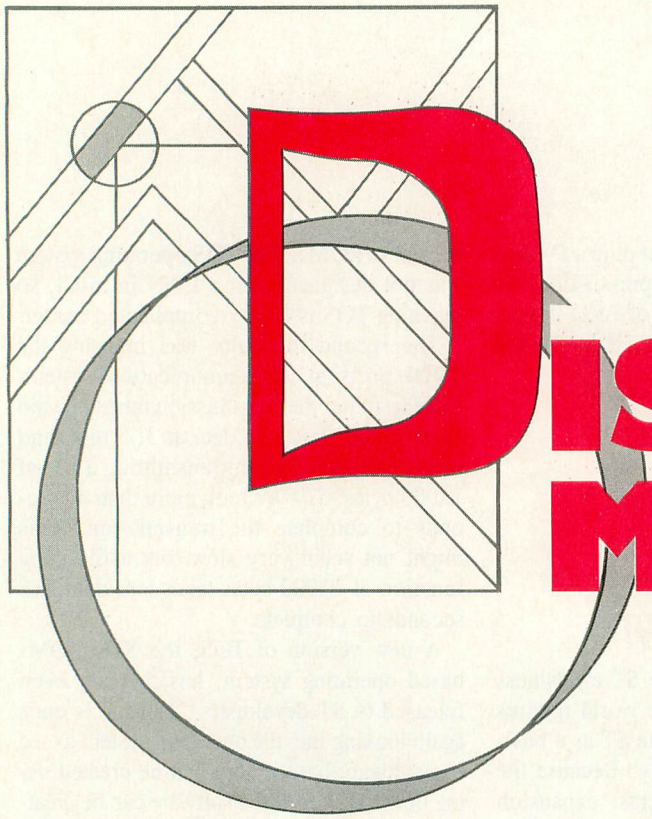
Use of the MIDI ports for LAN transmissions is attractive because of the low cost of MIDI cables and not having to build a hardware interface board for the ST. Just plug two $5 MIDI cables between two or more STs and you have a LAN. However, there are additional ports which might result in faster LAN data transmission.

The ST floppy-disk drives are serviced by a high-speed floppy-disk-drive controller. Data transmission rates go up to 32 microseconds per byte across the floppy-disk-drive cables, about 1,000 times faster than the MIDI ports. The ST floppy-disk controller chip is also much more reliable than the MIDI port controller in data transmission.

The idea of using the floppy-disk controller as a LAN manager is a practical one. The data rate is high enough to be competitive with MS-DOS LAN systems, and the software should be straightforward.

Frank Cohen has been developing Atari programs since his first commercial product, Clowns & Balloons. *He later developed* Regent Base, *an SQL 4GL database, and is currently involved with several other ST-related productivity and small business software packages. You may contact Frank directly on CompuServe (76004,1573) or Genie (FRANK.COHEN).*

Use of the MIDI ports for LAN transmissions is attractive because of the low cost of MIDI cables. However, there are additional ports which might result in faster LAN data transmission.



DISK MASTER

by Bryan Schappel & Barry Kolbe

Have you ever had the urge to snoop around your disks? To see how files are stored on a disk? Or, to try to cheat at your favorite text adventure? Did you ever want to disassemble a few disk sectors or a file that you could not load into memory? If you would like to do these things then *Disk Master* is for you!

Disk Master (DM) is actually two programs in one. The editor allows you to examine and edit any sector on a single-density or 1050 double-density disk. The disassembler allows you to disassemble sectors or files (with or without labels) and send the output to the screen, printer or a disk.

Typing It In

Type DM (Listing 1) using the M/L Editor found elsewhere in this issue. Use the binary load option of your DOS to load and run DM. It would be best to remove any cartridges and/or disable BASIC on XL/XE models.

The Disk Editor

As you will see, the screen is divided into two sections: a section that is filled with numbers and a section filled with characters. This area displays all the data in one disk sector. There are 128 bytes in each sector—numbered from 0 to 127 (00-7F in hexadecimal). The left margin and the top of the screen have guide numbers to cross-reference the data. When you begin, the buffer that holds sector data is full of 0s, which you will see as hexadecimal 00s. To the right of the data is an 8 by 16 array of hearts. These are the ATASCII equivalents of those 0s. Below is information about the sector:

SEC: This is the current sector number displayed in hexadecimal.

FIL: This is the DOS 2.x file number. The file number ranges from 0-3F, for a maximum of 64 files per disk.

LNK: This is the DOS sector link. This number is the pointer to the next data sector in a file. This number is invalid on a boot disk (such as a game) and the directory sectors.

BYT: This number tells DOS how many bytes of actual data are contained in the sector. This number typically ranges between 0 and 125.

SRC: This is the source drive that DM uses to read all data from while in the editor.

DST: This is the destination drive that DM will write all information to. *Note*: The source and destination may be different; this allows users with multiple-drive systems to read from one disk and write to another.

INP: This tells you how your input will be interpreted. If the flag is set to "H," all input will be interpreted as hex strings; otherwise the input will be interpreted as character strings.

What is a hex string? Well, a hex string is simply input that is made up of hexadecimal digits that represent ATASCII characters. For example, the hex string "414243" is the same as the character string "ABC."

So, why do we use hex strings? Because, a hex string allows you to enter characters that you normally can't, like the EOL (ATASCII 155, $9B) and the delete keys. You will find a use for hex strings when you want to search for data on a disk.

A note on hex string interpretation: As you may have noticed, it takes two hex digits to represent one character, so what happens if you enter an odd number of digits? Well, an extra 0 is added to the beginning of your input, so, "1FE" becomes "01FE." We felt that this was the natural way to interpret the hex string.

DNS: This shows you which density the program is operating in. A 1 indicates single density, and a 2 indicates 1050 double density.

You may activate a built-in help screen by pressing the "?" (use Shift-/). The help screen lists all of the command keys along

with their associated function. Here is a brief description of each command.

Read a sector (R): Pressing "R" allows you to enter a sector number for DM to read into its memory buffers. This number may range from 1-2D0 in single density and 1-410 in 1050 double density. Simply press Return to reread the current sector.

Up one sector (+): This will increase the current sector number by 1 and read it into memory. If you are on the last sector of the disk, DM will keep reading it.

Down one sector (−): This reads in the sector before the one you're on. You may not read Sector Zero.

Edit sector (E): The edit routine requires two pieces of information: the starting byte and the new data. At the EDTB prompt, enter the starting byte number of the edit, and at the EDTD prompt, enter the data.

The data will be entered according to the INP flag. Additionally, character strings will be converted to either internal code or ATASCII depending on the setting of the character display mode (see the C command).

Write sector buffer (W): You are asked for the sector number to write the sector to, or you may press Return for the current sector. You are asked if you are sure before the write occurs; pressing Escape or "N" will abort. If you write the buffer to the current sector you may use Undo to restore the buffer and rewrite the buffer back to disk. If you used a different sector number you cannot restore the sector's original contents.

Undo (U): This restores the sector buffer contents to what they were when the sector was first read.

Number Base (N): This allows you to change the number base that is used to display the sector data. The default mode is hexadecimal—pressing this key toggles between hex and decimal.

Decimal numbers are displayed using only two digits. If the left digit is inversed, then add 200 to the number displayed. If the right digit is inversed, add 100 to the displayed number. If no digits are inversed, then the number on screen is the actual value.

If you write the buffer to the current sector you may use Undo to restore the buffer and rewrite the buffer back to disk.

Character Display (C): This toggles the character display array from ATASCII to internal code. The setting of this array is used to interpret your character-string input during searches and edits.

Block fill (B): This command fills the sector buffer with a hex byte, just enter the value and, *wham!*, the buffer is filled. (Excellent for erasing sectors; just use the value 00.)

Exclusive OR sector contents (J): This will perform a binary EOR on every byte in the sector with a value that you supply. This can be useful when some data has been encrypted in a sector. To recover from an EOR, just perform the EOR again with the same value.

Trace link (T): This will read in the sector pointed to by the LNK field. Use this command to follow a disk file.

New file number (F): This allows you to change the file number on the current sector. You are asked for the new file number; remember that only values 0-3F are allowed. You will receive an Error 181 if the value is out of range.

New sector link (L): This allows you to change the sector link on the current sector. Only values 0-2D0 (single) and 0-410 (double) are allowed. An Error 180 will be given if the new link is out of range.

Density toggle (V): Pressing "V" toggles between single and 1050 density. Having this set incorrectly cannot damage your disks in any way.

Main menu (M): This command just returns you to the main menu allowing you to enter the disassembler or exit to DOS.

Search for data (S): This command searches the current sector for data you enter. Remember that your input will be interpreted according to the INP flag. If the data that you are searching for is found, a "<" is placed on the screen at the beginning of each occurrence. This command only examines the current sector, so if you want to search another sector for the same data, just read in that sector and hit Return at the SRCH prompt. You may also want to clear the screen before doing another search (use Control-Clear to redraw the screen).

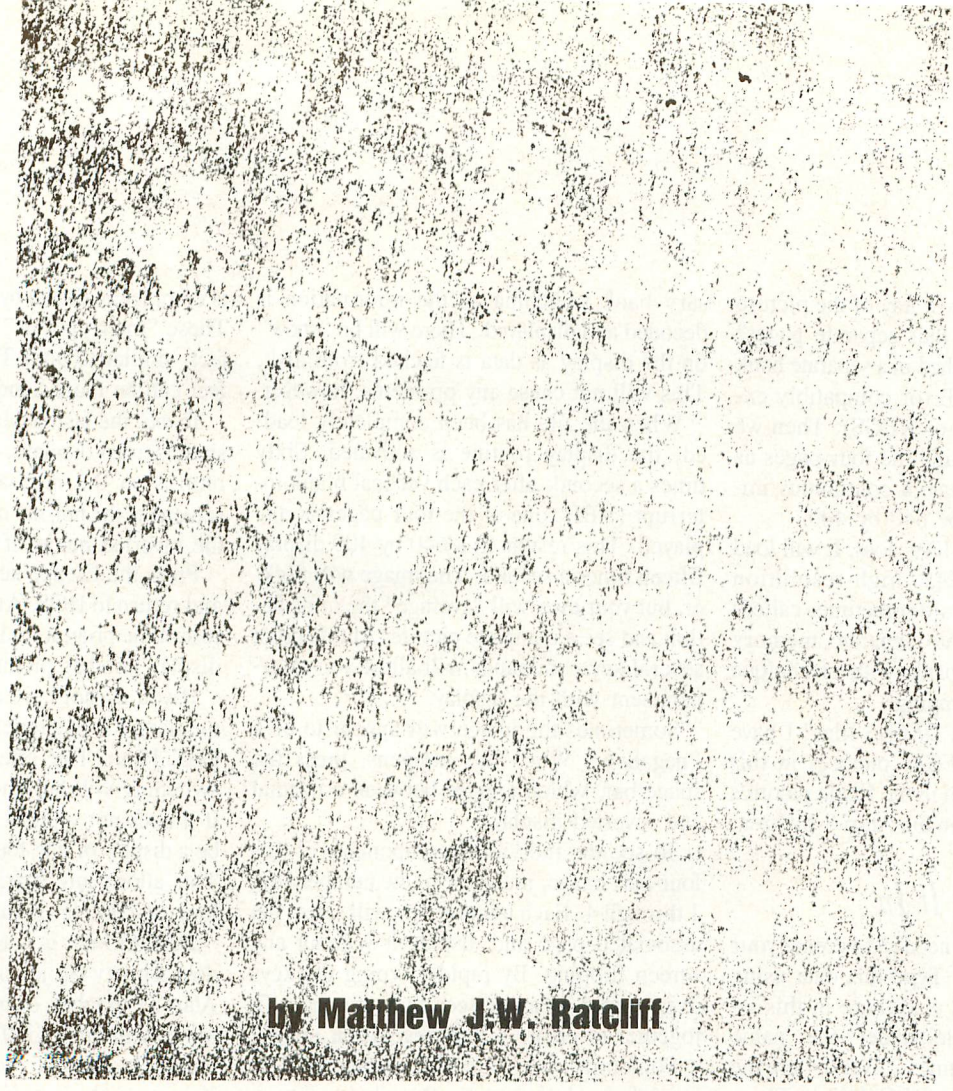
Hex to decimal (H): This converts a hexadecimal number to its decimal equivalent. Only values of 0-FFFF are allowed.

Decimal to hex (.): This converts a decimal number to its hex counterpart. Only values

continued on page 27

ILLUSTRATION BY JOHN BERADO

DEGAS View



by Matthew J.W. Ratcliff

The Atari ST can do some pretty amazing things. But many of us still cling to our faithful 8-bit machines, not wanting to give up such a good friend despite the enticements of the powerful, graphically exquisite Atari ST. Still we are curious. How can we get a peek at what the ST can do without purchasing one, or going to the local dealer (if that is even remotely possible) to peruse their wares, or reading about it in one of those ST magazines that we don't usually purchase?

DEGAS View (the first word is pronounced day-GAH) brings the ST's highest-resolution graphics capabilities to the Atari 8-bit machines, without dropping a single bit.

DEGAS by Tom Hudson is the most popular graphics art package for the Atari ST. With it, art work can be created in 320 by 200 pixels with 16 colors, 640 by 200 with four colors, or 640 by 400 with two colors. This last mode is similar to graphics mode 24 (8+16) on the 8-bits, which is 320 by 192 with two colors. If we were to load up a *DEGAS* high-resolution graphic and throw away every other pixel horizontally and every other scan line, or three-fourths of the image, we could display part of a *DEGAS* picture on a graphics 24 screen.

What if we didn't throw any of the picture away, but split it into four separate pages? Each page could be stashed in a separate bank of memory on a 130XE (or compatibly expanded 800XL, 65XE or XEGS). Then we could simply flip through the four pages as fast as possible, to create a continuous image that appeared to be 640 by 400.

This is what *DEGAS View* does. It will load uncompressed *DEGAS* high-resolution (monochrome, as they are sometimes called) pictures, sort them into separate memory banks, and display the banks with a technique referred to as page flipping.

Where can you find *DEGAS* files? I have accumulated some *DEGAS* pictures in this format and placed them in the Koala pictures database on DELPHI in the Atari 8-bit area.

Typing It In

Listing 1 is the data needed to create your copy of *DEGAS View*. Type this data using the *M/L Editor*, found elsewhere in this issue, creating the file under the name DEGASVUF.OBJ. Listings 2 3 and 4 are the MAC/65 source code for *DEGAS View*. These listings do not need to be typed; they are provided only for those people interested in assembly language programming.

Using the Program

Load DEGASVUF.OBJ from DOS. No external cartridges should be installed, and internal BASIC should be disabled. When run, *DEGAS View* first prompts for a drive number. Simply press Return to accept the default of 1, or type the drive number and press Return. A directory of all files with an extender of .PI3, the standard for all *DEGAS* high-resolution picture files, will be listed to the display. Press Escape twice and then Return to abort the program and return to DOS. Press Return only to enter a new drive number.

Type the name of the *DEGAS* file you wish to view and press Return. There is no need to specify the drive number or extender, if the current default drive and extender (.PI3) are valid. If the file is not found, or a load error occurs, the problem is detected and *DEGAS View* will restart.

DEGAS View will immediately begin loading the picture file. It will perform the necessary bank switching as the information is decoded and displayed. There will be "snow" on the display as data is fetched from disk. This will not cause any problems, however.

When the file has been completely loaded, the display routine is activated. Sixty times a second, after each vertical blank interrupt (VBI), one of the four pages is displayed. This results in a 640 by 400 display fifteen times a second. The image will flicker, but your eyes will "average" the information and see a complete picture. At this point several key commands will allow you to experiment with the display.

Sometimes the image will appear to be a "negative." When this happens, press the space bar, which toggles between a normal and inverted display.

To see the individual components of this four-part image, press one of the number keys 1 through 4. Each number key will select the associated page and display it without any screen flipping. By rapidly typing the keys in sequence it can be seen how the pages fit together to create a high-resolution image.

Sometimes only half of the image will be enough resolution to see the whole picture. Press the H key to see half the data. This will cause *DEGAS View* to flip through pages 1 and 3 only, resulting in an effective display rate of 30 times per second. This noticeably reduces the flicker. Press the S key to see the other half of the data, screens 2 and 4, in the same manner. Try alternating between S and H keys, to see which display pair looks best.

To get a better idea of what the complete image would look like, flicker free, press the F key. In this mode *DEGAS View* will flip through all four pages as fast as possible.

This will result in page flipping before each is completely drawn by the display hardware. These "interruptions" will create a snow effect, similar to poor TV reception. This effect can be more annoying than the flicker.

To exit the fast display mode, or any of the static screen displays, press the N key. This returns to the normal display of all four screens at the normal rate. This does not reset the inverted mode, if it has been enabled.

Press the Escape key to exit *DEGAS View* and return to DOS. Any other key press will restart the program, allowing you to load and display another picture.

If you do not have a 130XE or expanded-memory XL machine, you may still use *DEGAS View*. The effects of memory bank switching will be ignored by the computer, if it does not support it. The end result will be a display of one-fourth the original image. This allows any Atari 8-bit owner to take a look at *DEGAS* pictures.

DEGAS View is not an ST emulator, but it does a pretty fair job of presenting full-screen Atari ST graphics on an 8-bit machine. There are possibilities for *DEGAS View* enhancements. It could be modified to save any one page to disk as a simple graphics-mode-8 screen dump. It could be updated to load multiple 8-bit screens and save them as a composite *DEGAS* image, allowing you to send your best 8-bit graphics to ST-owning friends. If ANALOG receives sufficient reader requests for new *DEGAS View* features, I will gladly consider upgrading this program for a future issue of ANALOG.

DEGAS View *is not an ST emulator, but it does a pretty fair job of presenting full-screen Atari ST graphics on an 8-bit machine.*

Matthew Ratcliff is definitely an Atari "old-timer." He has been writing about Atari computers for as long as we can remember and recently celebrated his one-hundredth article sale. He lives in Missouri.

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,2,50,50,50,83,58,155
,0,72,162,96,169,12,157,4003
1010 DATA 66,3,32,86,228,162,96,169,3,
157,66,3,169,2,157,68,3685
1020 DATA 3,169,50,157,69,3,104,157,75
,3,41,240,73,16,9,12,22
1030 DATA 157,74,3,32,86,228,96,211,50
,215,50,112,48,79,0,64,2878
1040 DATA 59,51,61,51,79,0,80,161,51,1
56,52,65,211,50,0,0,645
1050 DATA 0,0,68,49,58,70,73,76,69,78,
65,77,69,46,69,88,303
1060 DATA 84,155,0,68,49,58,42,46,80,7
3,51,155,0,162,0,169,1824
1070 DATA 15,157,216,50,232,224,99,208
,248,162,0,157,62,51,232,224,1428
1080 DATA 99,208,248,169,0,32,6,50,169
,0,141,198,2,162,16,169,4335
1090 DATA 12,157,66,3,32,86,228,76,21,
52,160,196,229,231,225,243,2801
1100 DATA 160,214,233,229,247,172,160,
194,249,160,205,225,244,170,210,225,93
40
1110 DATA 244,172,160,168,227,169,160,
193,238,225,236,239,231,160,155,162,76
79
1120 DATA 0,169,9,157,66,3,169,240,157
,68,3,169,51,157,69,3,3662
1130 DATA 169,37,157,72,3,169,0,157,73
,3,32,86,228,162,0,169,4424
1140 DATA 9,157,66,3,169,185,157,68,3,
169,51,157,69,3,169,255,6987
1150 DATA 157,72,3,169,0,157,73,3,32,8
6,228,76,121,52,91,82,3159
1160 DATA 69,84,85,82,78,93,32,61,32,7
5,101,101,112,32,100,101,2021
1170 DATA 102,97,117,108,116,32,115,10
4,111,119,110,32,97,98,111,118,4627
1180 DATA 101,46,155,162,0,169,9,157,6
6,3,169,84,157,68,3,169,4052
1190 DATA 52,157,69,3,169,37,157,72,3,
169,0,157,73,3,32,86,965
1200 DATA 228,76,188,52,68,114,105,157
,52,152,53,118,101,32,35,32,2152
1210 DATA 68,101,103,97,115,32,46,80,7
3,51,32,102,105,108,101,115,2881
1220 DATA 32,97,114,101,32,111,110,32,
63,155,162,0,169,9,157,66,3677
1230 DATA 3,169,154,157,68,3,169,52,15
7,69,3,169,34,157,72,3,2550
1240 DATA 169,0,157,73,3,32,86,228,162
,0,169,5,157,66,3,169,3896
1250 DATA 0,157,68,3,169,4,157,69,3,16
9,2,157,72,3,169,0,1436
1260 DATA 157,73,3,32,86,228,173,0,4,2
01,155,240,3,141,186,51,6959
1270 DATA 162,16,169,3,157,66,3,169,6,
157,74,3,169,0,157,75,2763
1280 DATA 3,169,185,157,68,3,169,51,15
7,69,3,32,86,228,152,48,4631
1290 DATA 66,162,16,169,5,157,66,3,169
,0,157,68,3,169,4,157,2898
1300 DATA 69,3,169,255,157,72,3,169,0,
157,73,3,32,86,228,152,5373
1310 DATA 48,33,162,0,169,9,157,66,3,1
69,0,157,68,3,169,4,1562
1320 DATA 157,69,3,169,255,157,72,3,16
9,0,157,73,3,32,86,228,4594
1330 DATA 76,35,53,162,16,169,12,157,6
6,3,32,86,228,76,138,53,3671
1340 DATA 71,114,97,112,104,105,99,115
,32,102,105,108,101,32,116,111,4177
1350 DATA 32,108,111,97,100,32,63,155,
162,0,169,9,157,66,3,169,3831
1360 DATA 114,157,68,3,169,53,157,153,
53,148,54,69,3,169,24,157,4146
1370 DATA 72,3,169,0,157,73,3,32,86,22
8,76,201,53,91,82,69,4054
1380 DATA 84,85,82,78,93,32,61,32,82,1
01,115,116,97,114,116,44,3238
1390 DATA 32,110,101,119,32,100,114,10
5,118,101,155,162,0,169,9,157,5553
1400 DATA 66,3,169,171,157,68,3,169,53
,157,69,3,169,30,157,72,4195
1410 DATA 3,169,0,157,73,3,32,86,228,7
6,11,54,68,111,117,98,3016
1420 DATA 108,101,32,91,69,83,67,93,32
,97,110,100,32,91,82,69,1938
1430 DATA 84,85,82,78,93,32,101,120,10
5,116,115,46,155,162,0,169,5475
1440 DATA 9,157,66,3,169,234,157,68,3,
169,53,157,69,3,169,33,4051
1450 DATA 157,72,3,169,0,157,73,3,32,8
6,228,162,0,169,5,157,4466
1460 DATA 66,3,169,0,157,68,3,169,4,15
7,69,3,169,128,157,72,4502
1470 DATA 3,169,0,157,73,3,32,86,228,1
62,16,169,12,157,66,3,3002
1480 DATA 32,86,228,173,0,4,201,27,240
,7,201,155,208,64,76,217,9220
1490 DATA 51,169,0,32,6,50,76,125,54,6
8,101,103,97,115,32,86,2109
1500 DATA 105,101,119,32,98,121,32,77,
97,116,42,82,97,116,155,162,5629
1510 DATA 0,169,9,157,66,3,169,103,157
,68,3,169,54,157,69,3,2995
1520 DATA 169,22,157,72,3,169,0,149,54
,144,55,157,73,3,32,86,1975
1530 DATA 228,96,173,0,4,162,0,160,0,1
40,1,50,201,58,240,25,4177
1540 DATA 173,1,4,201,58,240,18,169,68
,141,168,51,173,186,51,141,8095
1550 DATA 169,51,169,58,141,170,51,160
,3,189,0,4,153,168,51,232,6705
1560 DATA 200,201,155,240,10,201,46,20
8,240,238,1,50,76,195,54,173,9276
1570 DATA 1,50,208,30,136,169,46,153,1
68,51,200,169,80,153,168,51,8423
1580 DATA 200,169,73,153,168,51,200,16
9,51,153,168,51,200,169,155,153,1035
1590 DATA 168,51,162,0,169,9,157,66,3,
169,168,157,68,3,169,51,4598
1600 DATA 157,69,3,169,255,157,72,3,16
9,0,157,73,3,32,86,228,4874
1610 DATA 162,16,169,3,157,66,3,169,4,
157,74,3,169,0,157,75,3085
1620 DATA 3,169,168,157,68,3,169,51,15
7,69,3,32,86,228,152,16,4408
1630 DATA 71,76,87,55,70,105,108,101,3
2,101,114,114,111,114,44,32,3009
1640 DATA 112,114,101,115,115,32,82,69
,84,85,82,78,155,162,0,169,5067
1650 DATA 9,157,66,3,169,62,157,68,3,1
69,55,157,69,3,169,25,3123
1660 DATA 157,72,3,169,0,157,73,3,32,8
6,228,169,255,205,252,2,9804
1670 DATA 240,251,141,252,2,76,217,51,
169,24,32,6,50,169,211,162,7194
1680 DATA 50,164,20,196,20,240,252,145
,55,140,56,141,48,2,142,49,5135
1690 DATA 2,169,0,133,88,169,64,133,89
,169,200,141,0,50,162,16,5297
1700 DATA 169,7,157,66,3,169,51,157,68
,3,169,50,157,69,3,169,4117
1710 DATA 34,157,72,3,169,0,157,73,3,3
2,86,228,152,16,60,162,4535
```

```
1720 DATA 16,169,12,157,66,3,32,86,228
,169,0,32,6,50,76,226,3658
1730 DATA 55,76,79,65,68,32,69,82,82,7
9,82,155,162,0,169,9,3180
1740 DATA 157,66,3,169,215,157,68,3,16
9,55,157,69,3,169,11,157,4939
1750 DATA 72,3,169,0,157,73,3,32,86,22
8,96,165,88,133,224,165,8931
1760 DATA 89,133,225,169,10,141,198,2,
169,0,141,197,2,169,4,141,5908
1770 DATA 200,2,162,16,169,7,157,66,3,
169,51,157,68,3,169,50,3461
1780 DATA 157,69,3,169,160,157,72,3,16
9,0,157,73,3,32,86,228,4579
1790 DATA 152,16,71,162,241,142,1,211,
169,0,32,6,50,162,16,169,4394
1800 DATA 12,157,66,3,32,86,228,76,97,
56,70,105,108,101,32,105,3657
1810 DATA 110,112,117,116,32,101,114,1
14,111,114,155,162,0,169,9,157,6236
1820 DATA 66,3,169,80,157,68,3,169,56,
157,69,3,169,17,157,72,4096
1830 DATA 3,169,0,157,73,3,32,86,228,9
6,162,0,189,51,50,160,5369
1840 DATA 4,24,42,38,226,42,38,141,56,
136,57,227,136,208,247,24,8930
1850 DATA 160,4,189,131,50,42,38,228,4
2,38,229,136,208,247,232,189,3276
1860 DATA 51,50,160,4,24,42,38,226,42,
38,227,136,208,247,160,4,8466
1870 DATA 24,189,131,50,10,38,228,10,3
8,229,136,208,247,134,230,162,2572
1880 DATA 193,142,1,211,165,226,145,22
4,162,241,142,1,211,145,224,162,4359
1890 DATA 197,142,1,211,165,227,145,22
4,162,201,142,1,211,165,228,145,4047
1900 DATA 224,162,205,142,1,211,165,22
9,145,224,230,224,208,2,230,225,6434
1910 DATA 166,230,232,224,80,240,3,76,
130,56,206,0,50,240,18,173,7641
1920 DATA 0,50,201,100,208,8,169,0,133
,224,169,80,133,225,76,24,7953
1930 DATA 56,162,16,169,12,157,66,3,32
,86,228,162,193,160,48,165,8231
1940 DATA 20,197,20,240,252,142,1,211,
140,212,50,232,232,232,232,224,7223
1950 DATA 201,208,2,160,64,173,252,2,2
01,255,208,6,224,209,240,219,6012
1960 DATA 208,221,162,241,142,1,211,16
2,64,169,12,157,66,3,32,86,4587
1970 DATA 228,162,64,169,3,157,66,3,16
9,4,157,74,3,169,0,157,3926
1980 DATA 75,3,76,106,57,75,58,0,169,1
03,157,68,3,169,57,157,4720
1990 DATA 69,3,32,86,228,152,16,13,162
,64,169,12,157,66,3,32,2396
2000 DATA 86,228,76,217,51,162,64,137,
57,132,58,169,7,157,66,3,4235
2010 DATA 169,0,157,68,3,169,4,157,69,
3,169,1,157,72,3,169,3555
2020 DATA 0,157,73,3,32,86,228,173,0,4
,201,32,208,3,76,90,4182
2030 DATA 58,201,72,208,3,76,107,58,20
1,78,208,3,76,29,57,201,5600
2040 DATA 70,208,3,76,181,58,201,83,20
8,3,76,144,58,201,49,208,8260
2050 DATA 4,162,193,208,22,201,50,208,
4,162,197,208,14,201,51,208,527
2060 DATA 4,162,201,208,6,201,52,208,2
0,162,205,142,1,211,162,255,2323
2070 DATA 236,252,2,240,251,162,241,14
2,1,211,76,135,57,201,27,208,689
2080 DATA 78,162,64,169,12,157,66,3,32
,86,228,76,46,58,68,101,3452
2090 DATA 103,97,115,32,86,105,101,119
,32,98,121,32,77,97,116,42,3333
2100 DATA 82,97,116,44,32,40,99,41,32,
65,110,97,108,111,103,155,4616
2110 DATA 162,0,169,9,157,66,3,169,12,
157,68,3,169,58,157,69,4299
2120 DATA 3,169,34,157,72,3,169,0,157,
73,3,32,86,228,96,162,5654
2130 DATA 64,169,12,157,66,3,32,86,228
,76,217,51,173,198,2,72,6470
2140 DATA 173,197,2,141,198,2,104,141,
197,2,76,29,57,162,193,165,7656
2150 DATA 20,197,20,240,252,142,1,211,
162,201,165,20,197,20,240,252,3387
2160 DATA 142,1,211,174,252,2,224,133,
58,215,58,255,240,227,162,241,6491
2170 DATA 142,1,211,76,135,57,162,197,
165,20,197,20,240,252,142,1,9864
2180 DATA 211,162,205,165,20,197,20,24
0,252,142,1,211,174,252,2,224,2967
2190 DATA 255,240,227,162,241,142,1,21
1,76,135,57,169,255,162,193,142,3445
2200 DATA 1,211,162,197,142,1,211,162,
201,142,1,211,162,205,142,1,280
2210 DATA 211,205,252,2,240,231,162,24
1,142,1,211,76,135,57,226,2,9739
2220 DATA 227,2,195,51,0,0,0,0,0,0,0,0
,0,0,0,0,3240
```

LISTING 2: ASSEMBLY

```
10 ********************************
20 *        DEGAS VIEW            *
30 *  by Matthew J.W. Ratcliff   *
40 *                              *
50 *      Copyright 1989          *
60 *    BY ANALOG COMPUTING       *
70 ********************************
80 *
1000 *SAVE#D:DEGASVUF.M65
1010 *ASM,#-,#D:DEGASVUF.COM
1020 * Degas *FULL* View, by Mat*Rat
1030 *
1040 * This program will load a degas
1050 * high resolution, .PI3 format
1060 * picture file and display 1/4
1070 * of it on 4 different pages
1080 * of 130XE (or compatible upgrade
1090 * upgrade XL), in Graphics mode 8
.
1100 * The software will then FLIP thr
ough
1110 * them as fast as possible. Other
1120 * keyboard control options are
1130 * as follows:
1140 *
1150 *  -- Key commands --
1160 * 1 - Display 1/4 of graphic, 1st
 page
1170 * 2 - second page
1180 * 3 - third page
1190 * 4 - fourth page
1200 * F - full display speed, not rea
l useful
1210 * H - show half image, alternate
screens 1 & 3, 30 Hertz
1220 * S - show half image, alternate
screens 2 & 4, 30 Hertz
1230 * N - Normal display, show all 4
screens, 15 hertz
1240 * SPACE - Invert display, show at
 normal speed
1250 *          screen will stay invert
ed, until inverted again
1260 * RETURN - Exit to main menu for
another picture selection
1270 * ESCAPE - Exit program
1280 *
1290 * At filename input prompt, press
 ESC twice, then return
1300 * to exit from there
1310 *
1320 *  >>>BEWARE<<< Don't expect to r
un
1330 * this with SpartaDOS X Cartridge
.
1340 * SDX likes all those RAM banks.
1350 * Go mucking about with them and
1360 * you are going to CRASH dude!
1370 * Important addresses for
1380 * loader program:
1390 ESCAPE = 27
1400 SCREEN = $E0
1410 * Byte splitting registers
1420 SHIFT1 = $E2
1430 SHIFT2 = $E3
1440 SHIFT3 = $E4
1450 SHIFT4 = $E5
1460 SAVEX = $E6
1470 BANKSCR = $4000 ;16K IS ADDR
1480 BANKSCR1 = $5000
1490 SCAN1 = 99
1500 SCAN2 = 99
1510 * Page bank select values
1520 PAGE1 = 193
1530 PAGE2 = 193+4
1540 PAGE3 = 193+8
1550 PAGE4 = 193+12
1560 MAIN =  241
1570 * Bank select register
1580 BANKSEL = 54017
1590 BLANK4 = 48
1600 BLANK5 = 64
1610     .ORG $3200
1620 LINECOUNT .BYTE 0
1630 EXFLG .BYTE 0
1640     .OPT OBJ
1650 * Mat*Rat's customized system equ
ates
1660     .INCLUDE #D:SYSEQU.M65
1670 * Mac/65's IO Macros library
1680     .INCLUDE #D:IOMAC.LIB
1690 * Mat*Rat's Graphics routine
1700     .INCLUDE #D:GRAPHICS.M65
1710 * 2 line graphic buffer
1720 BUF160 .DS 160
1730 * Custom display list work area
1740 DISPLAY .BYTE 112,48,79
1750     .WORD BANKSCR
1760 DLA .DS SCAN1   ; 15'S GO HERE
1770     .BYTE 79    ; SKIP 4K BOUNDAR
Y
1780     .WORD BANKSCR1
1790 DLB .DS SCAN2   ; ANOTHER 104 MOD
E 8 LINES
1800     .BYTE 65
1810     .WORD DISPLAY
1820     .BYTE 0,0,0,0
1830 * Working filename for parsing
1840 FILEN .BYTE "D1:FILENAME.EXT",155
,0
1850 * Default directory spec
1860 FILSPEC .BYTE "D1:*.PI3",155,0
1870 * Initialize custom, 200 line
1880 * graphics mode 8 display
1890 * list for starters
1900 INIZ LDX #0     ; Setup display l
ist
1910     LDA #15
1920 INIZ1 STA DLA,X
1930     INX
1940     CPX #SCAN1
1950     BNE INIZ1
1960     LDX #0
1970 INIZ2 STA DLB,X
1980     INX
1990     CPX #SCAN2
2000     BNE INIZ2
2010 * Clear screen, show title
2020 * and Start DEGAS VIEW program
2030 START
2040      GRAPHICS  0
2050     LDA #0
2060     STA 710
2070 * Get drive number first
2080 * or accept default displayed
2090 * by pressing return only
2100      CLOSE  1
2110      PRINT  0,"Degas View, By Ma
t*Rat, (c) Analog "
2120      PRINT  0,FILSPEC
2130      PRINT  0,"[RETURN] = Keep de
fault shown above."
2140      PRINT  0,"Drive # Degas .PI3
 files are on ?"
2150      INPUT  0,FILENAME,2
2160     LDA FILENAME
2170     CMP #155
2180     BEQ NONEWDR
2190     STA FILSPEC+1
2200 * Open a directory listing
2210 * and display it
2220 NONEWDR  OPEN  1,6,0,FILSPEC
2230     TYA
2240     BMI NOFILE
2250 NEXTFILE  INPUT  1,FILENAME
2260     TYA
2270     BMI NOFILE
2280      PRINT  0,FILENAME
2290     JMP NEXTFILE
2300 * Request filename to view
2310 * [RETURN] only restarts, allowin
g
2320 * the user to change drive spec
2330 * ESCAPE will exit the program to
 DOS
2340 * Or type filename (no drive or e
xt reqd) to load
2350 NOFILE
2360      CLOSE  1
2370      PRINT  0,"Graphics file to l
oad ?"
2380      PRINT  0,"[RETURN] = Restart
, new drive"
2390      PRINT  0,"Double [ESC] and [
RETURN] exits."
2400      INPUT  0,FILENAME,128
2410      CLOSE  1
2420     LDA FILENAME
2430     CMP #ESCAPE
```

```
2440      BEQ ESCEXIT
2450      CMP #155
2460      BNE GETFILE
2470      JMP START
2480 ESCEXIT  GRAPHICS  0
2490       PRINT  0,"Degas View by Mat*
Rat"
2500      RTS
2510 * Parse the filename
2520 * If no drive letter or number
2530 * prefixed, insert those
2540 * If no extender, paste on .PI3
2550 GETFILE LDA FILENAME
2560      LDX #0
2570      LDY #0       ; EXTENDER FOUND
FLAG
2580      STY EXFLG
2590      CMP #':
2600      BEQ GETF1
2610      LDA FILENAME+1
2620      CMP #':
2630      BEQ GETF1
2640      LDA #'D
2650      STA FILEN
2660      LDA FILSPEC+1
2670      STA FILEN+1
2680      LDA #':
2690      STA FILEN+2
2700      LDY #3
2710 * If a '.' is in the filename
2720 * don't mess with the extender
2730 GETF1 LDA FILENAME,X
2740      STA FILEN,Y
2750      INX
2760      INY
2770      CMP #155
2780      BEQ GETF2
2790      CMP #'.
2800      BNE GETF1
2810      INC EXFLG
2820      JMP GETF1
2830 GETF2 LDA EXFLG
2840      BNE GETF3
2850      DEY
2860      LDA #'.
2870      STA FILEN,Y
2880      INY
2890      LDA #'P
2900      STA FILEN,Y
2910      INY
2920      LDA #'I
2930      STA FILEN,Y
2940      INY
2950      LDA #'3
2960      STA FILEN,Y
2970      INY
2980      LDA #155
2990      STA FILEN,Y
3000 GETF3   PRINT  0,FILEN
3010       OPEN  1,4,0,FILEN
3020      TYA
3030      BPL LOADIT
3040 * Bad filename or drive?
3050 * Restart the program
3060       PRINT  0,"File error, press
RETURN"
3070      LDA #255
3080 HOLD CMP CH
3090      BEQ HOLD
3100      STA CH
3110      JMP START
3120 LOADIT
3130 * Let OS setup the GRAPHICS 24
3140 * originally, to get all system
3150 * variables set properly
3160 * Then point to custom display
3170 * list and memory.
3180       GRAPHICS  24
3190      LDA # <DISPLAY
3200      LDX # >DISPLAY
3210      LDY JIFFY
3220 LOADIT1 CPY JIFFY
3230      BEQ LOADIT1
3240      STA 560
3250      STX 561
3260      LDA # <BANKSCR
3270      STA SAVMSC
3280      LDA # >BANKSCR
3290      STA SAVMSC+1
3300      LDA #SCAN1+SCAN2+2
3310      STA LINECOUNT
3320 * Forget the 34 byte Degas header
3330       BGET  1,BUF160,34 ; Get and
forget header
3340      TYA
3350      BPL CNTNU
3360       CLOSE  1
3370       GRAPHICS  0
3380       PRINT  0,"LOAD ERROR"
3390      RTS
3400 * Setup screen RAM pointers
3410 CNTNU
3420      LDA SAVMSC
3430      STA SCREEN
3440      LDA SAVMSC+1
3450      STA SCREEN+1
3460      LDA #10
3470      STA 710
3480      LDA #0
3490      STA 709
3500      LDA #4
3510      STA 712
3520 * Get next 2 screen lines of pict
ure data
3530 GLINE  BGET  1,BUF160,160 ; Read
a line
3540      TYA
3550      BPL GLINE2
3560 GLINE1 LDX #MAIN
3570      STX BANKSEL
3580       GRAPHICS  0
3590       CLOSE  1
3600       PRINT  0,"File input error"
3610      RTS
3620 GLINE2 LDX #0   ; Buffer index
3630 * SHIFT1 will hold even pixels fo
r even lines
3640 * SHIFT2 will hold odd pixels for
 even lines
3650 GLINE3 LDA BUF160,X
3660      LDY #4
3670      CLC
3680 GLINE4 ROL A
3690      ROL SHIFT1
3700      ROL A
3710      ROL SHIFT2
3720      DEY
3730      BNE GLINE4
3740      CLC
3750      LDY #4
3760 * SHIFT3 will hold even pixels fo
r odd lines
3770 * SHIFT4 will hold odd pixels for
 odd lines
3780      LDA BUF160+80,X
3790 GLINE4A ROL A
3800      ROL SHIFT3
3810      ROL A
3820      ROL SHIFT4
3830      DEY
3840      BNE GLINE4A
3850      INX
3860      LDA BUF160,X
3870      LDY #4
3880      CLC
3890 GLINE5 ROL A
3900      ROL SHIFT1
3910      ROL A
3920      ROL SHIFT2
3930      DEY
3940      BNE GLINE5
3950      LDY #4
3960      CLC
3970      LDA BUF160+80,X
3980 GLINE5A ASL A
3990      ROL SHIFT3
4000      ASL A
4010      ROL SHIFT4
4020      DEY
4030      BNE GLINE5A
4040      STX SAVEX
4050      LDX #PAGE1
4060 * Stuff pixel data in PAGE1 bank
4070 * and main bank. Use PAGE1,2,3,4
4080 * for computational convenience
4090 * Shove copy of PAGE1 in MAIN
4100 * so display doesn't flash garbag
e
4110 * during keyboard inputs.
4120 *
4130 * Stuff appropriate shift bytes
4140 * into proper pages, as enabled.
4150      STX BANKSEL
```

```
4160      LDA SHIFT1
4170      STA (SCREEN),Y
4180      LDX #MAIN
4190      STX BANKSEL
4200      STA (SCREEN),Y
4210      LDX #PAGE2
4220      STX BANKSEL
4230      LDA SHIFT2
4240      STA (SCREEN),Y
4250      LDX #PAGE3
4260      STX BANKSEL
4270      LDA SHIFT3
4280      STA (SCREEN),Y
4290      LDX #PAGE4
4300      STX BANKSEL
4310      LDA SHIFT4
4320      STA (SCREEN),Y
4330      INC SCREEN
4340      BNE CHKDN
4350      INC SCREEN+1
4360 * Restore buffer index and check
4370 * if done reading picture file.
4380 CHKDN LDX SAVEX
4390      INX
4400      CPX #80
4410      BEQ UPDLINE
4420      JMP GLINE3
4430 UPDLINE
4440      DEC LINECOUNT
4450      BEQ SHOIT0
4460      LDA LINECOUNT
4470      CMP #100
4480      BNE JMGL
4490      LDA # <BANKSCR1
4500      STA SCREEN
4510      LDA # >BANKSCR1
4520      STA SCREEN+1
4530 JMGL
4540      JMP GLINE
4550 * Display routine
4560 * Enable respective banks
4570 * Stick BLANK4 lines command in
4580 * display list for even line page
s
4590 * and BLANK5 for odd line pages
4600 * This is done to minimixe vertic
al
4610 * jitter during page flipping.
4620 * Horizontal jitter is not elimin
ated,
4630 * however, as it would require so
me
4640 * extra fancy bit shifting at loa
d
4650 * time. Not fun.
4660 *
4670 SHOIT0  CLOSE  1
4680 SHOIT LDX #PAGE1
4690      LDY #BLANK4
4700 SHOIT1A LDA JIFFY
4710 SHOIT1 CMP JIFFY
4720      BEQ SHOIT1
4730      STX BANKSEL
4740      STY DISPLAY+1
4750      INX
4760      INX
4770      INX
4780      INX
4790      CPX #PAGE3
4800      BNE SHOIT2
4810      LDY #BLANK5
4820 SHOIT2 LDA CH
4830      CMP #255
4840      BNE SHOIT3
4850      CPX #PAGE4+4
4860      BEQ SHOIT
4870      BNE SHOIT1A
4880 * Always re-enable MAIN bank
4890 * before exiting any display loop
!
4900 SHOIT3 LDX #MAIN
4910      STX BANKSEL
4920 * Process key command
4930 * ESCAPE - Exit to dos
4940 * 1 - Display static screen, page
 1 only
4950 * 2 - Display page 2
4960 * 3 - Display page 3
4970 * 4 - Display page 4
4980 * H - Show half of image, flip sc
reens
4990 *      1 & 3 at 30 hertz
5000 * S - Show half of image, flip sc
reens
5010 *      2 & 4 at 30 hertz
5020 * F - Fastest display possible, a
 bit
5030 *      useless, but easy enuf to t
hrow in
5040 * N - Normal display (SHOIT above
)
5050 * SPACE - Invert displays, swap c
olor registers, simple
5060 *
5070 * Any other key loops back to STA
RT
5080 HOLDIT  CLOSE  4
5090       OPEN  4,4,0,"K:"
5100      TYA
5110      BPL HOLDNOW
5120       CLOSE  4
5130      JMP START
5140 HOLDNOW  BGET  4,FILENAME,1
5150      LDA FILENAME
5160 * SPACE - Invert display?
5170      CMP #32      ; SPACE?
5180      BNE CKSCL
5190      JMP INVERT
5200 * (H)alf display, 1&3 ?
5210 CKSCL CMP #'H
5220      BNE CKSNRM
5230      JMP SHOHALF
5240 * (N)ormal display, 4 pages?
5250 CKSNRM CMP #'N
5260      BNE CKSFAST
5270      JMP SHOIT
5280 * (F)astest possible flipping?
5290 CKSFAST CMP #'F
5300      BNE CKSCS
5310      JMP FASTEST
5320 CKSCS
5330 * (S)how half, pages 2&4?
5340      CMP #'S
5350      BNE CK1
5360      JMP SHOSECHAF
5370 * (1)-show screen 1 only?
5380 CK1 CMP #'1
5390      BNE CK2
5400      LDX #PAGE1
5410      BNE SVPG
5420 * (2)-show screen 2 only?
5430 CK2 CMP #'2
5440      BNE CK3
5450      LDX #PAGE2
5460      BNE SVPG
5470 * (3)-show screen 3 only?
5480 CK3 CMP #'3
5490      BNE CK4
5500      LDX #PAGE3
5510      BNE SVPG
5520 * (4)-show screen 4 only?
5530 CK4 CMP #'4
5540      BNE CKESC
5550      LDX #PAGE4
5560 SVPG STX BANKSEL
5570      LDX #255
5580 SVPG1 CPX CH
5590      BEQ SVPG1
5600      LDX #MAIN
5610      STX BANKSEL
5620      JMP HOLDNOW
5630 * (ESCAPE) from the program?
5640 CKESC CMP #ESCAPE
5650      BNE RESTART
5660       CLOSE  4
5670       PRINT  0,"Degas View by Mat*
Rat, (c) Analog"
5680      RTS
5690 RESTART  CLOSE  4
5700      JMP START
5710 * INVERT routine - swap color
5720 * registers 709 & 710
5730 INVERT LDA 710
5740      PHA
5750      LDA 709
5760      STA 710
5770      PLA
5780      STA 709
5790      JMP SHOIT
5800 * Show (H)alf, pages ! & 3
5810 SHOHALF LDX #PAGE1
5820      LDA JIFFY
```

continued on page 65

NEW ARRIVALS

NEW XL/XE SOFTWARE!

DARK CHAMBERS — AIRBALL SUMMER GAMES
CROSSBOW (XE gun) — CRIME BUSTERS (XE gun)
ONE ON ONE — KARATEKA
CHOPLIFTER — AIRBALL

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

1050 DISK DRIVE **$169.** Reconditioned

800 BOARDS
- Mother
- CPU
- Power
- ROM

YOUR CHOICE **$8.98**

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

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

810 DRIVES **$129.** Reconditioned

BASIC TUTOR 4 BOOKS **$4.98**

5.25" DISKS 20 CENTS EA.*

QTY.	PRICE
10	$1.00
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.
10' Joystick ext. ..... $ .99
Trackball $9.95

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

CARTRIDGES FOR 800, XL, XE

BASIC CARTRIDGE	$4.95	CHICKEN	$8.98	MARIO BROS.	$19.98	STAR RAIDERS II	$19.98
BASIC TUTOR (4 BOOKS)	$4.95	CLAIM JUMPER	$8.98	DONKEY KONG JR.	$19.98	DAVID'S MIDNIGHT MAGIC	$19.98
TURMOIL	$4.95	DELTA DRAWING	$8.98	JUNGLE HUNT	$19.98	ARCHON	$19.98
PAC-MAN (no box)	$4.95	HEY DIDDLE DIDDLE	$8.98	MOON PATROL	$19.98	KARATEKA	$19.98
DONKEY KONG (no box)	$4.95	SLIME (400/800)	$8.98	BATTLEZONE	$19.98	CHOPLIFTER	$19.98
GORF (400,800)	$4.95	ALPHABET ZOO	$8.98	FOOD FIGHT	$19.98	GATO	$24.98
DEMON ATTACK (400,800)	$4.95	ALF	$8.98	HARDBALL	$19.98	ACE OF ACES	$24.98
DELUXE INVADERS	$4.95	ADVENTURE CREATOR	$8.98	FIGHT NIGHT	$19.98	LODE RUNNER	$24.98
JOURNEY TO THE PLANETS	$4.95	SKY WRITER	$14.95	ONE ON ONE BASKETBALL	$19.98	BARNYARD BLASTER (XE gun)	$24.98
MATH ENCOUNTER	$7.98	FOOTBALL	$14.95	DESERT FALCON	$19.98	DARK CHAMBERS	$29.98
DANCE FANTASY	$8.98	DEFENDER	$14.95	NECROMANCER	$19.98	AIRBALL	$29.98
LOGIC LEVELS	$8.98	ROBOTRON	$19.98	RESCUE ON FRACTALUS	$19.98	SUMMER GAMES	$29.98
MEMORY MANOR	$8.98	TENNIS	$19.98	BALLBLAZER	$19.98	CROSSBOW (XE gun)	$29.98
LINKING LOGIC	$8.98	FINAL LEGACY	$19.98	BLUE MAX	$19.98	CRIME BUSTERS (XE gun)	$29.98

DISK SOFTWARE FOR 800, XL, XE

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

SAN JOSE COMPUTER

THE ATARI STORE

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

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 #109 ON READER SERVICE CARD.

continued from page 17

of 0-65535 are allowed.

Input mode toggle (I): This command toggles the INP flag between hex string and character-string modes.

Set SRC drive (O): Pressing this key increases the source-drive number. The program supports Drives 1-8.

Set DST drive (D): This command increases the destination drive by 1. You have access to all eight drives here also.

Redraw screen (Control-Clear): This forces DM to redraw the screen from scratch and removes any markers that are left behind from a search.

Print screen (Control-P): Pressing this key will perform a character dump of the screen to any printer. All offensive control characters and the like are removed from the output. Use this to produce a hard copy of a sector's data.

The Escape key may be used at any time to abort any function. The "less than" and "greater than" keys ("<" and ">") are used to change the screen hue and luminance, respectively.

Disassembler

The disassembler converts a binary file to assembly language source code. This code can be viewed on the screen (pressing the space bar pauses the output), dumped to a printer or saved as a disk file. Actually the disk file is in LIST format so it can be loaded into a word processor or entered into an assembler like MAC/65. To enter the resulting file into MAC/65 use: ENTER # D:FILENAME,A.

The ",A" adds line numbers to the source code. The code that is generated always has the same format:

```
MNEMONIC OPERAND ADDR B1 B2 B3
```

An example would be:

```
LDA $4503,X    ; $3000 BD 03 45
```

The data on the left is the generated code. The ";" marks the beginning of the comment field. ADDR is the memory location where LDA $4503,X would be assembled. B1, B2 and B3 are the hexadecimal values for the assembly opcode and the operand. When DM encounters data that isn't assembly code it writes it out as ".BYTE $hh." The headers for each segment of a file are displayed as:

```
* = $3000 ;END = $30FE
```

Disassembly does continue across sector boundaries, but does not cross segments of code. So one segment might end as:

```
LDA #$90     ;$4023 A9 90
.BYTE $20    ;$4025 20
 * =$4026    ;END = $4118
.BYTE $42    ;$4026 42
.BYTE $3A    ;$4027 3A
STA $5E12    ;$4028 8D 12 5E
        etc.
```

The series of bytes—$20, $42 and $3A—quite probably make the instruction JSR $3A42.

To begin disassembly, you must first set several parameters. You are reprompted if you type in an improper value; no error message appears.

All input requires using the Return key to finalize it. Escape aborts at any time.

You must first select the type of the disassembly, file (F) or disk (D). Pick the destination device which can be E (screen), P (printer) or D (disk file). If you choose D you must type in a complete file name as in: Dn:filename.ext.

Even for Drive 1 you must type D1:. The extension, of course, is optional. If the file is not a binary file, an error message will be given. Next choose whether you want labels. The labels are those which most assembly language programmers use. They follow closely those given in the technical manual and/or Compute!'s *Mapping the Atari*.

If you chose to disassemble files, type in a complete filename as your source file. The program takes control unless you press Escape to abort. If you are sending the code to a disk file or the printer, keep in mind that the output is kept in a large buffer. Consequently, there will be no printing or disk access until the buffer is full or you terminate the operation by pressing Escape. If the code is to be sent to disk, you will be asked to insert a destination disk when the buffer is full. Since the output code will be much larger than the object code, make sure you use a disk that is nearly empty. As you have seen, three bytes of data will become 26 bytes of source code!

If you selected disk disassembly (D), you must specify a source-drive number, density, starting-sector number, and the number of sectors to read. You cannot start at the last sector (2D0 for single and $410 for double density). Start at one less (2CF or $40F) and specify 2 as the number of sectors to disassemble. Then you must select an offset into the first sector (0-7F). For example, a boot sector does not have assembly code in the first six bytes. The real code begins at Byte 6 (bytes are numbered starting at 0!). 6 would be the offset. Specify an origin address (for example $706 for a boot sector). The data does not get put there—the address is just used for reference.

We must digress for a moment to explain the next option. Commercial boot disks generally use $80 bytes per sector, and the sectors are laid out sequentially on the disk. Sectors that are part of a file are not necessarily sequential. Each sector has a link to the next sector in the file. So Sector 123 may point to Sector 150. Therefore, when disassembling boot disks, type in 80 at the BYT/SEC prompt. But if you are disassembling a file or part of a file use 7D.

When disassembling part of a file you can find where the file begins by looking at the directory, sectors $169-$16F. Each entry is 16 bytes long. The first byte is a flag byte concerning the condition of the file, whether it is locked, deleted, etc. The next two bytes give the number of sectors in the file, while Bytes 3 and 4 have the starting-sector number. (The number of sectors and starting-sector number are stored in LO/HI format, so to calculate the starting sector take Byte 3 plus 256 times Byte 4.) With the editor, read the starting sector and use the trace function to follow the file around the disk. When you have found what you want, switch to the dissembler.

Warning!

If you are sending output to disk, you will need a separate diskette for this output. This is particularly true on any boot disk. Writing data to that disk would probably destroy programs stored there. Please put write protect tabs on the source disk, and always use a separate disk for output.

A final note: The input routine is very smart. If you are asked to supply a number, only legal hexadecimal digits are allowed to be typed, otherwise almost anything goes.

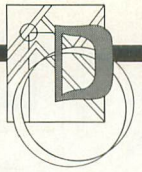
We hope that you can find a good use for this program, it is not a full-blown file editor (if you want an incredible DOS 2.0 file editor, contact me on DELPHI—my membername is BBKBRYAN), but it should provide you with the means to get a peek at those nasty machine-language games you have or fix the dreaded Error 164 (file number mismatch)! Well, so long and happy hacking.

Bryan and Barry are currently taking a break, partly so they have time to play with their new STs and partly so they can solve Stationfall *and thus save the universe.*

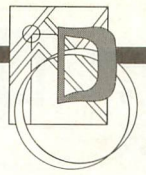
LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,48,212,79,76,96,69
,112,112,112,112,112,112,112,5604
1010 DATA 112,112,70,46,50,112,66,90,4
8,2,112,6,32,6,32,6,6676
1020 DATA 65,3,48,112,112,112,70,86,50
,6,64,66,40,40,2,2,7241
1030 DATA 2,2,2,2,2,2,2,65,29,48,112,1
12,112,70,86,50,9449
1040 DATA 32,66,230,48,32,0,66,0,40,2,
2,2,2,2,2,2,3250
1050 DATA 2,2,2,2,2,2,2,2,2,2,2,2,2,65
,52,48,3690
1060 DATA 128,128,128,226,249,154,128,
162,225,242,242,249,128,171,239,236,86
07
1070 DATA 226,229,128,225,238,228,128,
162,242,249,225,238,128,179,227,232,90
74
1080 DATA 225,240,240,229,236,128,128,
128,128,128,176,242,239,228,245,227,81
67
1090 DATA 229,228,128,230,239,242,154,
128,161,142,174,142,161,142,172,142,32
48
1100 DATA 175,142,167,142,128,163,239,
237,240,245,244,233,238,231,128,128,82
01
1110 DATA 239,240,244,233,239,238,0,0,
100,105,115,107,0,101,100,105,5209
1120 DATA 116,111,114,0,243,229,236,22
9,227,244,0,0,100,105,115,97,8403
1130 DATA 115,115,101,109,98,108,101,1
14,0,243,244,225,242,244,0,0,9347
1140 DATA 101,120,105,116,0,0,0,0,0,0,
0,0,128,128,128,173,404
1150 DATA 174,165,128,128,175,240,229,
242,225,238,228,128,128,128,128,128,42
77
1160 DATA 128,128,128,128,128,161,164,
164,178,128,162,145,128,162,146,128,10
80
1170 DATA 162,147,128,128,68,105,115,9
7,32,84,121,112,101,32,91,70,3122
1180 DATA 47,68,93,0,68,101,115,116,39
,110,32,91,69,47,80,47,723
1190 DATA 68,110,93,0,32,32,32,76,97,9
8,101,108,115,32,91,89,1933
1200 DATA 47,78,93,0,32,32,32,32,83,11
1,117,114,99,101,32,70,1327
1210 DATA 105,108,101,0,83,114,99,32,6
8,114,105,118,101,32,91,49,2115
1220 DATA 45,56,93,0,32,32,68,101,110,
115,105,116,121,32,91,49,2149
1230 DATA 45,50,93,0,32,83,101,99,116,
111,114,32,91,49,45,50,947
1240 DATA 68,48,93,0,32,32,32,35,32,11
1,102,32,83,101,99,116,1277
1250 DATA 111,114,115,0,32,32,79,102,1
02,115,101,116,32,91,48,45,1356
1260 DATA 55,70,93,0,79,114,105,103,10
5,110,32,91,48,45,70,70,1285
1270 DATA 70,70,93,0,66,121,116,47,83,
101,99,32,91,55,68,47,958
1280 DATA 56,48,93,0,112,112,16,70,66,
50,16,66,200,59,66,0,93
1290 DATA 40,2,2,2,2,2,2,2,2,2,2,2,2,2
,2,2,1600
1300 DATA 16,1,222,49,66,126,50,0,2,48
,66,0,45,16,66,128,8687
1310 DATA 42,32,66,190,51,65,190,49,66
,206,50,0,2,2,2,2,8061
1320 DATA 2,65,190,49,0,40,80,120,160,
200,240,24,64,104,144,184,7738
1330 DATA 224,8,48,88,128,168,208,248,
32,72,112,152,40,40,40,40,3538
1340 DATA 40,40,40,41,41,41,41,41,41,4
2,42,42,42,42,42,42,7001
1350 DATA 43,43,43,43,100,105,115,107,
0,109,97,115,116,101,114,0,2740
1360 DATA 72,99,73,0,81,89,88,89,0,0,0
,0,100,105,115,107,323
1370 DATA 0,0,101,100,105,116,111,114,
0,0,0,0,0,100,105,115,9798
1380 DATA 107,0,0,100,105,115,97,115,1
15,101,109,98,108,101,114,0,3649
1390 DATA 0,0,0,0,0,100,97,116,97,0,10
1,110,116,114,121,0,1820
1400 DATA 0,0,0,0,128,179,165,163,154,
128,128,128,128,128,128,166,9215
1410 DATA 169,172,154,128,128,128,128,
128,128,172,174,171,154,128,128,128,82
5
1420 DATA 128,128,128,162,185,180,154,
128,128,128,128,128,128,179,178,163,17
67
1430 DATA 154,145,128,128,128,128,128,
164,179,180,154,145,128,128,128,128,65
5
1440 DATA 128,169,174,176,154,168,128,
128,128,128,128,164,174,179,154,145,20
36
1450 DATA 128,128,128,128,164,233,243,
235,128,173,225,243,244,229,242,128,88
58
1460 DATA 168,229,236,240,128,179,227,
242,229,229,238,128,141,128,179,229,74
72
1470 DATA 236,229,227,244,128,163,239,
237,237,225,238,228,178,13,114,101,456
7
1480 DATA 97,100,0,0,139,13,117,112,0,
0,0,0,141,13,100,110,9540
1490 DATA 0,0,0,0,165,13,101,100,105,1
16,0,0,183,13,119,114,2175
1500 DATA 105,116,101,0,181,13,117,110
,100,111,0,0,174,13,3,98,889
1510 DATA 97,115,101,0,163,13,99,104,9
7,114,115,0,162,13,98,102,3226
1520 DATA 105,108,108,0,170,13,101,111
,114,0,0,0,180,13,116,114,1800
1530 DATA 97,99,101,0,166,13,102,105,1
08,101,3,0,172,13,108,105,2323
1540 DATA 110,107,3,0,182,13,100,101,1
10,115,0,0,173,13,109,101,2191
1550 DATA 110,117,0,0,179,13,115,101,1
14,99,104,0,168,13,40,127,2638
1560 DATA 36,0,0,0,142,13,36,127,40,0,
0,0,169,13,105,110,9726
1570 DATA 112,117,116,0,175,13,115,111
,114,99,101,0,164,13,100,101,3467
1580 DATA 115,116,110,0,253,13,114,102
,114,115,104,0,219,13,97,98,4586
1590 DATA 111,114,116,0,156,13,104,117
,101,0,92,0,158,13,108,117,2448
1600 DATA 109,0,92,0,128,128,176,242,2
29,243,243,128,163,175,174,180,5320
1610 DATA 178,175,172,141,176,128,244,
239,128,240,242,233,238,244,128,244,98
30
1620 DATA 232,233,243,128,243,227,242,
229,229,238,128,128,128,32,160,53,2407
1630 DATA 32,4,53,169,190,141,48,2,169
,49,141,49,2,169,6,133,3413
1640 DATA 185,169,45,133,186,162,0,134
,184,134,144,134,137,134,148,134,13
1650 DATA 154,134,161,232,134,143,134,
162,134,163,134,182,32,187,53,169,272
1660 DATA 0,32,185,56,32,11,55,32,113,
52,32,45,52,32,4,53,7831
1670 DATA 76,33,52,173,252,2,201,255,2
40,249,162,255,142,252,2,72,3427
1680 DATA 169,222,141,220,49,169,49,14
1,221,49,104,162,26,221,139,52,8242
```

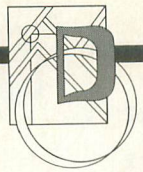
```
1690 DATA 240,5,202,16,248,48,220,138,
10,170,10,168,192,40,144,2,5946
1700 DATA 160,40,32,115,52,189,168,55,
141,111,52,189,169,55,141,112,7599
1710 DATA 52,32,32,57,32,255,255,160,4
0,138,72,162,0,185,91,60,6296
1720 DATA 157,0,45,200,232,224,4,208,2
44,169,154,141,4,45,104,170,9242
1730 DATA 96,42,57,34,21,1,56,62,0,40,
46,18,35,102,11,8,6514
1740 DATA 58,54,118,14,6,13,45,16,55,3
7,182,138,173,37,228,72,4729
1750 DATA 173,36,228,72,96,132,183,160
,0,132,129,169,128,145,185,32,8548
1760 DATA 166,52,164,129,201,155,240,5
7,201,27,208,2,56,96,201,126,8603
1770 DATA 208,11,192,0,240,227,169,0,1
45,185,136,16,220,166,182,240,2918
1780 DATA 17,41,127,201,58,144,2,41,22
3,32,216,56,224,0,16,2,3119
1790 DATA 48,199,196,183,240,195,153,8
5,45,32,235,54,145,185,200,208,2438
1800 DATA 184,153,85,45,169,0,145,185,
24,96,169,0,162,34,157,5,4117
1810 DATA 45,202,16,250,96,169,0,133,1
82,160,33,76,175,52,169,1,5932
1820 DATA 133,182,160,32,32,175,52,144
,1,96,162,0,189,85,45,201,5940
1830 DATA 155,240,3,232,208,246,138,41
,1,240,16,162,43,189,85,45,6941
1840 DATA 157,86,45,202,16,247,169,48,
141,85,45,162,0,160,0,189,6063
1850 DATA 85,45,201,155,240,26,133,153
,232,189,85,45,133,152,134,133,207
1860 DATA 132,130,32,196,56,164,130,16
6,133,153,85,45,200,232,208,223,3372
1870 DATA 24,96,169,1,133,182,160,4,32
,175,52,144,1,96,32,175,4481
1880 DATA 63,168,185,85,45,201,155,208
,2,24,96,32,216,56,6,212,6126
1890 DATA 38,213,6,212,38,213,6,212,38
,213,6,212,38,213,138,5,7134
1900 DATA 212,133,212,200,208,220,169,
0,160,0,153,0,40,153,0,41,3798
1910 DATA 153,0,42,153,0,43,200,208,24
1,165,82,133,85,152,133,84,9012
1920 DATA 96,173,254,49,133,138,173,22
,50,133,139,162,0,160,0,189,6717
1930 DATA 15,60,145,138,132,128,160,28
,145,138,164,128,232,200,189,15,740
1940 DATA 60,145,138,132,128,160,29,14
5,138,164,128,32,238,53,232,224,1769
1950 DATA 32,208,218,96,165,138,24,105
,40,133,138,144,2,230,139,96,7900
1960 DATA 173,254,49,133,138,173,22,50
,133,139,169,2,133,129,165,148,8450
1970 DATA 240,3,76,76,54,160,0,32,175,
63,132,128,185,0,44,133,4620
1980 DATA 212,32,122,63,230,128,164,12
9,200,173,130,5,32,235,54,145,8828
1990 DATA 138,173,131,5,32,235,54,200,
145,138,200,132,129,192,26,240,1499
2000 DATA 7,164,128,192,128,208,208,96
,32,238,53,169,2,133,129,76,7917
2010 DATA 59,54,160,0,230,129,169,255,
133,147,185,0,44,200,132,128,9906
2020 DATA 201,200,144,9,56,233,200,230
,147,230,147,208,9,201,100,144,2478
2030 DATA 5,56,233,100,230,147,162,48,
134,146,134,145,201,10,176,58,8997
2040 DATA 24,101,145,133,145,164,147,4
8,8,185,145,0,9,128,153,145,6396
2050 DATA 0,164,129,165,146,32,235,54,
145,138,200,165,145,32,235,54,11
2060 DATA 145,138,200,200,132,129,192,
27,240,7,164,128,192,128,208,166,2509
2070 DATA 96,32,238,53,169,3,133,129,2
08,240,56,233,10,230,146,208,2534
2080 DATA 187,96,32,11,55,173,254,49,1
33,138,173,22,50,133,139,162,8015
2090 DATA 0,160,30,189,0,44,134,132,16
6,137,208,3,32,235,54,166,7874
2100 DATA 132,145,138,232,200,192,38,2
08,234,32,238,53,224,128,208,225,5050
2110 DATA 96,32,253,54,29,164,55,166,1
31,96,32,253,54,29,160,55,6002
2120 DATA 166,131,96,134,131,72,42,42,
42,42,41,3,170,104,41,159,3199
2130 DATA 96,162,7,165,137,208,2,162,1
5,160,7,189,241,59,153,230,488
2140 DATA 59,202,136,16,246,96,165,143
,133,212,165,144,133,213,162,5,1261
2150 DATA 32,111,55,173,127,44,133,212
,162,35,32,115,55,173,125,44,6043
2160 DATA 74,74,133,212,162,15,32,115,
55,173,126,44,133,212,173,125,9104
2170 DATA 44,41,3,133,213,162,25,165,1
62,9,144,141,171,50,165,163,9199
2180 DATA 9,144,141,181,50,165,161,24,
105,145,141,201,50,164,154,185,757
2190 DATA 1,60,141,191,50,160,1,208,2,
160,2,132,134,134,135,32,5758
2200 DATA 122,63,164,134,166,135,185,1
28,5,32,235,54,9,128,157,126,7313
2210 DATA 50,232,200,192,4,208,239,76,
175,63,160,127,185,128,44,153,435
2220 DATA 0,44,136,16,247,96,32,64,0,9
6,64,0,32,96,240,55,3231
2230 DATA 125,56,57,56,177,56,236,56,7
1,57,119,57,35,58,100,58,3080
2240 DATA 89,59,222,55,231,55,11,57,26
,57,43,57,53,57,5,58,9802
2250 DATA 252,57,15,59,39,59,227,56,48
,59,191,59,15,58,96,69,2865
2260 DATA 252,57,132,72,165,137,73,1,1
33,137,76,188,54,165,148,73,7535
2270 DATA 1,133,148,76,250,53,32,108,5
3,144,1,96,165,213,208,251,1284
2280 DATA 165,212,201,128,176,245,133,
155,32,4,53,169,164,141,3,45,6315
2290 DATA 32,109,57,176,230,132,128,13
6,185,85,45,166,137,240,3,32,8041
2300 DATA 235,54,153,85,45,136,16,240,
164,155,162,0,189,85,45,153,8093
2310 DATA 0,44,232,200,192,128,240,4,2
28,128,208,240,76,2,57,169,409
2320 DATA 1,133,182,160,5,32,175,52,14
4,1,96,169,85,133,243,169,9337
2330 DATA 45,133,244,169,0,133,242,32,
0,216,32,210,217,144,3,32,7223
2340 DATA 175,63,165,213,72,165,212,72
,162,1,134,172,202,32,34,63,6996
2350 DATA 232,169,61,157,85,45,232,104
,133,212,104,133,213,32,161,56,9467
2360 DATA 76,148,56,32,108,53,176,30,1
62,1,134,172,202,32,161,56,6749
2370 DATA 169,61,141,90,45,162,6,32,34
,63,189,85,45,32,235,54,4396
2380 DATA 157,128,42,202,16,244,96,134
,134,32,122,63,166,134,169,36,7784
2390 DATA 157,85,45,232,76,106,63,32,1
08,53,144,1,96,165,212,160,7889
2400 DATA 127,153,0,44,136,16,250,76,2
,57,165,152,32,216,56,134,6794
2410 DATA 152,165,153,32,216,56,138,10
,10,10,10,5,152,96,162,15,2291
2420 DATA 221,75,60,240,3,202,16,248,9
6,165,154,73,1,133,154,76,7739
2430 DATA 32,55,32,108,53,144,1,96,160
,127,185,0,44,69,212,153,6915
2440 DATA 0,44,136,16,245,76,2,57,32,2
50,53,32,188,54,76,32,3758
2450 DATA 55,169,242,141,220,49,169,49
,141,221,49,104,104,76,45,52,6291
2460 DATA 32,148,55,76,2,57,160,39,169
,0,153,128,42,136,16,250,6471
```



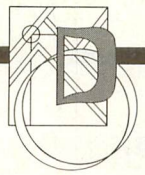
```
2470 DATA 96,166,162,32,63,57,134,162,
76,32,55,166,163,32,63,57,4428
2480 DATA 134,163,76,32,55,232,224,9,2
08,2,162,1,96,32,108,53,4453
2490 DATA 144,1,96,165,213,208,251,165
,212,201,64,144,5,160,181,76,1560
2500 DATA 9,59,10,10,133,212,173,125,4
4,41,3,5,212,141,125,44,5053
2510 DATA 76,2,57,165,154,208,3,76,24,
53,76,15,53,32,109,57,1514
2520 DATA 144,1,96,192,0,240,15,132,18
4,160,44,185,85,45,153,40,6953
2530 DATA 45,136,16,247,48,4,165,184,2
40,232,160,0,185,40,45,166,9310
2540 DATA 137,240,3,32,235,54,153,85,4
5,200,196,184,208,238,169,129,3948
2550 DATA 56,229,184,133,183,160,0,162
,0,132,128,189,85,45,217,0,7305
2560 DATA 44,208,13,200,232,228,184,20
8,242,165,128,133,155,32,210,57,2696
2570 DATA 164,128,200,196,183,208,224,
96,165,155,74,74,74,168,185,254,3763
2580 DATA 49,133,138,185,22,50,133,139
,152,10,10,10,133,136,165,155,6788
2590 DATA 56,229,136,133,136,10,24,101
,136,24,105,2,168,169,127,145,7178
2600 DATA 138,96,32,160,53,32,187,53,7
6,2,57,173,198,2,24,105,3905
2610 DATA 16,141,198,2,96,173,198,2,17
0,41,240,133,142,138,24,105,8424
2620 DATA 2,41,15,5,142,141,198,2,96,3
2,108,53,144,1,96,165,4701
2630 DATA 212,5,213,240,48,165,213,166
,161,221,5,60,144,15,240,2,8648
2640 DATA 176,35,165,212,221,3,60,144,
4,240,2,176,24,165,212,141,9552
2650 DATA 126,44,165,213,41,3,133,212,
173,125,44,41,252,5,213,141,9641
2660 DATA 125,44,76,2,57,160,180,76,9,
59,32,108,53,144,1,96,2797
2670 DATA 165,212,5,213,240,8,165,212,
133,143,165,213,133,144,32,134,1592
2680 DATA 58,16,3,32,9,59,32,148,55,76
,2,57,165,162,141,1,3219
2690 DATA 3,169,44,141,5,3,169,128,141
,4,3,169,82,208,17,165,6220
2700 DATA 163,141,1,3,169,44,141,5,3,1
69,0,141,4,3,169,87,2726
2710 DATA 141,2,3,165,143,5,144,208,2,
230,143,166,161,165,144,221,2923
2720 DATA 5,60,144,21,240,2,176,7,165,
143,221,3,60,144,10,189,7213
2730 DATA 3,60,133,143,189,5,60,133,14
4,165,143,141,10,3,165,144,7445
2740 DATA 141,11,3,169,128,141,8,3,169
,0,141,9,3,76,83,228,4330
2750 DATA 32,175,63,132,212,162,1,134,
172,162,7,189,7,60,157,85,7119
2760 DATA 45,202,16,247,162,8,32,34,63
,169,0,157,86,45,96,32,3440
2770 DATA 234,58,76,148,56,165,144,208
,7,165,143,201,2,176,1,96,7621
2780 DATA 198,143,165,143,201,255,208,
2,198,144,76,120,58,230,143,208,3283
2790 DATA 2,230,144,76,120,58,165,143,
133,159,165,144,133,160,173,125,2129
2800 DATA 44,41,3,133,144,173,126,44,1
33,143,5,144,208,5,160,182,8955
2810 DATA 76,9,59,76,120,58,165,159,13
3,143,165,160,133,144,96,32,8819
2820 DATA 108,53,144,1,96,165,143,133,
159,165,144,133,160,165,212,5,916
2830 DATA 213,240,8,165,212,133,143,16
5,213,133,144,165,143,133,212,165,4738
2840 DATA 144,133,213,32,122,63,160,3,
185,128,5,153,58,60,136,16,4875
2850 DATA 247,165,163,9,48,141,67,60,1
60,0,185,47,60,240,8,32,4798
2860 DATA 235,54,145,185,200,208,243,3
2,166,52,201,27,240,20,41,127,9179
2870 DATA 201,89,240,6,201,78,208,239,
240,8,32,153,58,16,3,32,4797
2880 DATA 9,59,76,80,59,165,161,73,1,1
33,161,76,32,55,128,128,5727
2890 DATA 128,139,144,128,139,145,128,
139,146,128,139,147,128,139,148,128,15
78
2900 DATA 139,149,128,139,150,128,139,
151,128,128,128,128,161,180,161,179,32
44
2910 DATA 163,169,169,128,128,128,0,16
9,174,180,165,178,174,161,172,161,4179
2920 DATA 180,161,179,163,169,169,128,
168,179,208,16,2,4,69,114,114,7153
2930 DATA 111,114,32,35,32,128,144,128
,152,145,144,145,152,146,144,146,1123
2940 DATA 152,147,144,147,152,148,144,
148,152,149,144,149,152,150,144,150,31
12
2950 DATA 152,151,144,151,152,87,114,1
05,116,101,32,83,69,67,58,32,3979
2960 DATA 48,48,48,48,32,116,111,32,68
,32,58,32,89,47,78,63,1276
2970 DATA 0,48,49,50,51,52,53,54,55,56
,57,65,66,67,68,69,1165
2980 DATA 70,165,228,244,131,128,168,2
55,164,128,164,255,168,162,230,233,892
9
2990 DATA 236,128,165,239,242,166,233,
236,131,179,242,227,232,172,233,238,17
40
3000 DATA 235,178,229,225,228,183,242,
233,244,128,128,128,128,160,0,132,3410
3010 DATA 172,132,200,32,131,64,16,3,7
6,88,61,133,168,160,0,217,7076
3020 DATA 250,66,240,5,200,192,151,208
,246,132,129,32,164,63,165,164,2465
3030 DATA 133,212,165,165,133,213,162,
23,32,34,63,32,175,63,165,168,8028
3040 DATA 133,212,162,28,32,34,63,169,
59,141,107,45,164,129,192,151,9244
3050 DATA 208,33,160,7,185,236,68,153,
85,45,136,16,247,32,175,63,8068
3060 DATA 164,168,132,212,32,122,63,16
9,1,133,171,162,8,32,80,63,5413
3070 DATA 76,54,61,190,145,67,160,0,18
9,82,66,153,86,45,232,200,9955
3080 DATA 192,3,208,244,133,171,164,12
9,185,40,68,168,185,217,68,133,2169
3090 DATA 171,152,10,168,185,191,68,14
1,52,61,185,192,68,141,53,61,7988
3100 DATA 166,171,202,240,20,32,131,64
,48,52,133,169,166,171,202,202,2152
3110 DATA 240,7,32,131,64,48,39,133,17
0,32,255,255,160,39,185,85,405
3120 DATA 45,201,32,208,3,136,16,246,1
69,155,153,86,45,169,0,153,8591
3130 DATA 87,45,165,164,24,101,171,133
,164,144,2,230,165,96,32,253,1160
3140 DATA 64,230,201,32,188,65,162,16,
32,105,64,162,32,32,105,64,4436
3150 DATA 169,59,160,69,32,54,71,32,63
,66,76,96,69,32,2,63,1028
3160 DATA 169,35,141,90,45,32,175,63,1
65,169,133,212,162,6,76,34,7384
3170 DATA 63,169,0,133,170,165,169,133
,212,165,170,133,213,162,6,169,3045
3180 DATA 23,133,166,169,73,133,167,16
5,195,208,6,160,0,177,166,208,2412
3190 DATA 3,76,34,63,72,41,15,133,134,
104,74,74,74,74,133,173,6183
3200 DATA 230,173,200,200,177,166,133,
174,24,101,173,133,176,136,177,166,360
8
```



```
3210 DATA 133,175,105,0,133,177,165,17
0,197,175,208,6,165,169,197,174,4383
3220 DATA 240,26,165,170,197,177,240,2
,176,67,165,169,197,176,176,61,3168
3230 DATA 165,170,197,175,144,187,165,
169,197,174,144,181,56,229,174,133,531
6
3240 DATA 173,202,160,2,200,232,177,16
6,157,85,45,16,247,41,127,157,416
3250 DATA 85,45,165,173,208,1,96,169,4
3,157,86,45,232,232,32,175,669
3260 DATA 63,165,173,133,212,230,172,3
2,34,63,198,172,96,165,166,24,214
3270 DATA 101,134,133,166,165,167,105,
0,133,167,76,165,61,32,2,63,5226
3280 DATA 76,139,61,32,2,63,32,139,61,
160,1,232,185,230,68,157,9770
3290 DATA 86,45,202,136,16,246,96,32,2
,63,32,14,63,76,143,61,3272
3300 DATA 32,81,62,76,67,62,32,81,62,1
69,44,157,86,45,169,89,5886
3310 DATA 157,87,45,96,169,0,133,170,3
2,2,63,32,26,63,232,160,5941
3320 DATA 2,208,200,169,0,133,170,32,2
,63,32,26,63,160,2,185,4619
3330 DATA 233,68,157,86,45,232,136,16,
246,96,169,65,141,90,45,96,8328
3340 DATA 32,2,63,230,171,32,175,63,16
5,169,16,12,41,127,73,127,6194
3350 DATA 24,105,1,133,181,76,180,62,1
33,181,165,169,16,16,165,164,9617
3360 DATA 56,229,181,133,212,165,165,2
33,0,133,213,76,211,62,24,101,190
3370 DATA 164,133,212,165,165,105,0,13
3,213,169,2,24,101,212,133,169,512
3380 DATA 133,212,165,213,105,0,133,17
0,133,213,32,151,61,198,171,96,1257
3390 DATA 32,2,63,32,14,63,32,26,63,16
9,41,157,86,45,96,32,2915
3400 DATA 2,63,32,139,61,76,99,62,32,1
75,63,165,169,133,212,162,672
3410 DATA 31,76,34,63,32,175,63,165,17
0,133,212,162,34,76,34,63,7078
3420 DATA 169,40,141,90,45,76,143,61,2
02,134,178,165,172,240,25,32,201
3430 DATA 170,217,32,230,216,166,178,1
60,255,200,232,177,243,157,85,45,5975
3440 DATA 16,247,41,127,157,85,45,96,3
2,122,63,166,178,232,224,23,442
3450 DATA 240,30,224,21,144,14,173,130
,5,157,85,45,232,173,131,5,8134
3460 DATA 157,85,45,96,169,36,157,85,4
5,232,165,171,201,2,240,230,3659
3470 DATA 160,0,185,128,5,157,85,45,23
2,200,192,4,208,244,202,96,3553
3480 DATA 160,0,162,1,134,131,169,240,
53,212,74,74,74,74,170,189,560
3490 DATA 75,60,153,128,5,200,166,131,
169,15,53,212,170,189,75,60,9830
3500 DATA 153,128,5,200,166,131,202,16
,219,96,169,32,162,39,157,85,9423
3510 DATA 45,202,16,250,96,169,0,133,2
12,133,213,96,162,16,32,105,8788
3520 DATA 64,165,194,240,23,32,207,64,
8,169,128,133,204,165,196,201,3608
3530 DATA 128,240,5,173,255,44,133,204
,40,48,52,96,162,16,169,3,6424
3540 DATA 157,66,3,169,130,157,68,3,16
9,45,157,69,3,169,4,157,6109
3550 DATA 74,3,169,0,157,75,3,133,187,
133,188,32,86,228,48,15,7192
3560 DATA 32,117,65,48,10,32,73,66,208
,3,160,1,96,160,152,192,8008
3570 DATA 136,208,7,169,0,133,200,76,8
8,61,32,234,58,169,88,160,9187
3580 DATA 69,32,54,71,169,85,160,45,32
,54,71,169,69,160,69,32,5315
3590 DATA 54,71,32,63,66,76,88,61,162,
32,32,105,64,162,32,169,5698
3600 DATA 3,157,66,3,169,150,157,68,3,
169,45,157,69,3,169,8,5213
3610 DATA 157,74,3,169,0,133,199,157,7
5,3,169,0,133,197,169,128,9681
3620 DATA 133,198,173,150,45,201,69,20
8,3,160,1,96,76,86,228,169,9952
3630 DATA 12,157,66,3,76,86,228,162,16
,169,7,157,66,3,169,0,5184
3640 DATA 157,72,3,157,73,3,76,86,228,
165,194,240,94,166,189,228,4926
3650 DATA 204,240,8,189,128,44,230,189
,160,1,96,169,1,133,189,165,1024
3660 DATA 196,201,125,208,33,173,253,4
4,41,3,133,144,173,254,44,133,974
3670 DATA 143,5,144,208,3,160,136,96,3
2,207,64,8,173,255,44,133,9547
3680 DATA 204,173,128,44,40,96,230,143
,208,2,230,144,32,207,64,8,8872
3690 DATA 173,128,44,40,96,165,202,5,2
03,240,218,165,202,56,233,1,2861
3700 DATA 133,202,165,203,233,0,133,20
3,76,134,58,165,187,5,188,240,3067
3710 DATA 42,165,187,56,233,1,133,187,
165,188,233,0,133,188,230,200,5404
3720 DATA 76,113,64,165,200,240,16,160
,151,32,163,60,32,130,65,165,8749
3730 DATA 169,133,168,198,200,208,236,
96,76,9,64,32,253,64,32,117,8528
3740 DATA 65,48,245,160,29,185,244,68,
153,85,45,136,16,247,32,73,8451
3750 DATA 66,208,5,32,117,65,48,224,16
5,152,133,164,133,212,165,153,3534
3760 DATA 133,165,133,213,32,122,63,16
2,4,32,106,63,32,117,65,48,4178
3770 DATA 199,165,152,133,212,56,229,1
64,133,187,165,153,133,213,229,165,710
2
3780 DATA 133,188,230,187,208,2,230,18
8,32,122,63,162,16,32,106,63,7292
3790 DATA 32,83,71,104,104,32,130,65,7
6,135,60,32,113,64,48,7,3034
3800 DATA 133,152,32,113,64,133,153,96
,169,85,160,45,32,54,71,160,7210
3810 DATA 0,132,20,165,20,201,5,144,25
0,173,150,45,201,69,208,1,172
3820 DATA 96,185,85,45,145,197,201,155
,240,3,200,208,244,200,152,24,4797
3830 DATA 101,197,133,197,165,198,105,
0,133,198,230,199,165,199,201,200,7501
3840 DATA 208,4,165,199,208,1,96,169,1
1,162,32,157,66,3,169,0,5807
3850 DATA 157,68,3,169,128,157,69,3,16
5,197,56,233,0,157,72,3,7110
3860 DATA 165,198,233,128,157,73,3,169
,0,133,197,133,199,169,128,133,2322
3870 DATA 198,32,23,66,208,12,169,11,1
60,69,32,54,71,32,63,66,3350
3880 DATA 162,32,32,86,228,48,27,32,23
,66,208,14,165,201,208,10,7981
3890 DATA 169,35,160,69,32,54,71,32,63
,66,160,1,96,165,163,197,8276
3900 DATA 162,96,32,234,58,169,79,160,
69,32,54,71,162,32,32,105,5524
3910 DATA 64,169,85,160,45,32,54,71,16
9,69,160,69,32,54,71,32,4118
3920 DATA 63,66,76,96,69,32,166,52,201
,155,208,249,162,32,96,165,2111
3930 DATA 152,201,255,208,2,197,153,96
,65,68,67,65,78,68,65,83,6163
3940 DATA 76,66,67,67,66,67,83,66,69,8
1,66,73,84,66,77,73,3830
3950 DATA 66,78,69,66,80,76,66,82,75,6
6,86,67,66,86,83,67,4081
3960 DATA 76,67,67,76,68,67,76,73,67,7
6,86,67,77,80,67,80,4052
```



```
3970 DATA 88,67,80,89,68,69,67,68,69,8
8,68,69,89,69,79,82,4252
3980 DATA 73,78,67,73,78,88,73,78,89,7
4,77,80,74,83,82,76,4673
3990 DATA 68,65,76,68,88,76,68,89,76,8
3,82,78,79,80,79,82,4768
4000 DATA 65,80,72,65,80,72,80,80,76,6
5,80,76,80,82,79,76,4448
4010 DATA 82,79,82,82,84,73,82,84,83,8
3,65,67,83,69,67,83,4413
4020 DATA 69,68,83,69,73,83,84,65,83,8
4,88,83,84,89,84,65,4910
4030 DATA 88,84,65,89,84,83,88,84,88,6
5,84,88,83,84,89,65,5095
4040 DATA 105,101,117,109,125,121,97,1
13,41,37,53,45,61,57,33,49,2800
4050 DATA 10,6,22,14,30,144,176,240,36
,44,48,208,16,0,80,112,5348
4060 DATA 24,216,88,184,201,197,213,20
5,221,217,193,209,224,228,236,192,2340
4070 DATA 196,204,198,214,206,222,202,
136,73,69,85,77,93,89,65,81,8920
4080 DATA 230,246,238,254,232,200,76,1
08,32,169,165,181,173,189,185,161,6499
4090 DATA 177,162,166,182,174,190,160,
164,180,172,188,74,70,86,78,94,1343
4100 DATA 234,9,5,21,13,29,25,1,17,72,
8,104,40,42,38,54,9624
4110 DATA 46,62,106,102,118,110,126,64
,96,233,229,245,237,253,249,225,261
4120 DATA 241,56,248,120,133,149,141,1
57,153,129,145,134,150,142,132,148,365
5
4130 DATA 140,170,168,186,138,154,152,
0,0,0,0,0,0,0,0,3,8584
4140 DATA 3,3,3,3,3,3,3,6,6,6,6,6,9,12
,15,18,5322
4150 DATA 18,21,24,27,30,33,36,39,42,4
5,48,51,51,51,51,51,228
4160 DATA 51,51,51,54,54,54,57,57,57,6
0,60,60,60,63,66,69,2380
4170 DATA 69,69,69,69,69,69,69,72,72,7
2,72,75,78,81,81,84,4445
4180 DATA 87,87,87,87,87,87,87,87,90,9
0,90,90,90,93,93,93,6447
4190 DATA 93,93,96,96,96,96,96,99,102,
102,102,102,102,102,102,102,7861
4200 DATA 105,108,111,114,117,117,117,
117,117,120,120,120,120,120,123,126,46
6
4210 DATA 129,129,129,129,129,129,129,
129,132,135,138,141,141,141,141,141,27
80
4220 DATA 141,141,144,144,144,147,147,
147,150,153,156,159,162,165,0,1,394
4230 DATA 2,3,4,5,6,7,0,1,2,3,4,5,6,7,
8,1,4814
4240 DATA 2,3,4,9,9,9,1,3,9,9,9,10,9,9
,10,10,5369
4250 DATA 10,10,0,1,2,3,4,5,6,7,0,1,3,
0,1,3,4618
4260 DATA 1,2,3,4,10,10,0,1,2,3,4,5,6,
7,1,2,4783
4270 DATA 3,4,10,10,3,11,3,0,1,2,3,4,5
,6,7,0,4817
4280 DATA 1,12,3,5,0,1,2,3,4,8,1,2,3,4
,10,0,4774
4290 DATA 1,2,3,4,5,6,7,10,10,10,10,8,
1,2,3,4,5056
4300 DATA 8,1,2,3,4,10,10,0,1,2,3,4,5,
6,7,10,5002
4310 DATA 10,10,1,2,3,4,5,6,7,1,12,3,1
,2,3,10,4960
4320 DATA 10,10,10,10,10,119,61,55,62,
61,62,81,62,90,62,96,3405
4330 DATA 62,110,62,125,62,148,62,154,
62,233,62,234,62,249,62,2,9794
4340 DATA 2,2,3,3,3,2,2,1,2,1,3,2,44,8
8,41,89,8344
4350 DATA 44,41,32,46,66,89,84,69,32,3
6,32,42,61,36,32,32,553
4360 DATA 32,32,32,32,59,69,78,68,61,3
6,32,32,32,32,155,155,3793
4370 DATA 0,155,73,110,115,101,114,116
,32,100,101,115,116,39,110,44,6433
4380 DATA 32,82,69,84,85,82,78,155,0,1
55,73,110,115,101,114,116,7970
4390 DATA 32,115,111,117,114,99,101,44
,32,82,69,84,85,82,78,155,6454
4400 DATA 0,155,67,111,109,112,108,101
,116,101,100,44,32,82,69,84,5761
4410 DATA 85,82,78,155,0,155,79,85,84,
80,85,84,32,0,155,73,5084
4420 DATA 78,80,85,84,32,0,216,162,255
,154,165,16,41,127,133,16,8621
4430 DATA 141,14,210,169,3,141,48,2,16
9,48,141,49,2,169,34,141,6416
4440 DATA 47,2,169,10,141,197,2,169,0,
141,200,2,169,2,133,82,7457
4450 DATA 141,198,2,169,148,141,199,2,
173,31,208,201,6,208,3,76,9482
4460 DATA 116,228,201,5,240,7,201,3,20
8,238,76,231,51,169,29,141,1908
4470 DATA 48,2,169,48,141,49,2,32,160,
53,169,14,133,166,169,49,7859
4480 DATA 133,167,32,190,71,32,6,72,14
4,3,76,96,69,173,85,45,5596
4490 DATA 41,223,162,0,201,70,240,5,20
1,68,208,233,232,134,194,32,4495
4500 DATA 190,71,32,17,72,176,227,160,
20,32,67,72,176,244,32,47,8318
4510 DATA 71,133,163,32,190,71,32,6,72
,176,207,173,85,45,41,223,9791
4520 DATA 162,0,201,89,240,5,201,78,20
8,236,232,134,195,165,194,208,8377
4530 DATA 23,32,190,71,32,17,72,176,17
7,160,0,32,67,72,176,244,9645
4540 DATA 32,47,71,133,162,76,229,70,3
2,223,71,32,190,71,32,6,6563
4550 DATA 72,176,151,173,85,45,41,127,
56,233,48,48,241,201,9,176,953
4560 DATA 237,133,162,32,190,71,32,6,7
2,176,204,173,85,45,41,127,8435
4570 DATA 56,233,49,48,241,201,2,176,2
37,133,161,170,189,3,60,133,2065
4580 DATA 212,189,5,60,133,213,32,122,
63,160,1,185,128,5,153,120,8915
4590 DATA 49,200,192,4,208,245,32,190,
71,32,29,72,176,203,32,35,8197
4600 DATA 72,176,246,165,212,133,143,1
65,213,133,144,32,190,71,32,29,224
4610 DATA 72,176,182,165,212,5,213,240
,245,165,212,133,202,24,101,143,5289
4620 DATA 133,212,165,213,133,203,101,
144,133,213,32,35,72,176,223,32,1622
4630 DATA 190,71,32,29,72,176,218,165,
213,208,247,165,212,201,128,176,8436
4640 DATA 241,133,189,32,190,71,32,29,
72,176,198,165,212,133,164,165,3958
4650 DATA 213,133,165,32,190,71,32,29,
72,176,182,165,213,208,247,165,6000
4660 DATA 212,201,125,240,4,201,128,20
8,237,133,196,32,160,53,132,201,4416
4670 DATA 169,52,141,48,2,169,48,141,4
9,2,32,182,63,32,50,64,4084
4680 DATA 32,135,60,32,130,65,169,0,13
3,200,162,255,173,252,2,201,4575
4690 DATA 255,240,237,142,252,2,201,28
,208,6,142,252,2,76,88,61,9511
4700 DATA 201,33,208,220,173,252,2,201
,28,240,239,201,33,208,245,142,7451
4710 DATA 252,2,76,250,70,173,86,45,56
,233,48,96,141,61,71,140,9050
4720 DATA 62,71,173,255,255,240,13,32,
79,71,238,61,71,208,243,238,5584
4730 DATA 62,71,208,238,96,201,155,208
```



```
,9,165,82,133,85,230,84,76,1975
4740 DATA 130,71,32,235,54,72,164,84,1
85,254,49,133,140,185,22,50,450
4750 DATA 133,141,104,164,85,145,140,2
00,192,39,144,9,240,7,230,84,1830
4760 DATA 32,130,71,164,82,132,85,96,1
64,84,192,21,144,53,169,0,8315
4770 DATA 133,138,169,40,133,139,169,4
0,133,140,169,40,133,141,162,0,9917
4780 DATA 160,39,177,140,145,138,136,1
6,249,165,141,133,139,165,140,133,4125
4790 DATA 138,24,105,40,133,140,165,14
1,105,0,133,141,232,224,21,208,3134
4800 DATA 223,198,84,96,32,83,71,169,6
4,141,190,2,169,0,141,182,9886
4810 DATA 2,165,166,164,167,32,54,71,1
69,58,32,79,71,169,32,32,5951
4820 DATA 79,71,32,235,71,165,166,24,1
05,16,133,166,144,2,230,167,1358
4830 DATA 96,165,85,164,84,24,121,254,
49,133,185,185,22,50,105,0,8197
4840 DATA 133,186,160,20,169,0,145,185
,136,16,251,96,32,235,71,160,1873
4850 DATA 0,132,182,200,76,175,52,32,2
35,71,160,0,132,182,160,15,9999
4860 DATA 76,175,52,32,235,71,76,108,5
3,165,212,5,213,240,22,166,2201
4870 DATA 161,165,213,221,5,60,144,15,
240,2,176,9,165,212,221,3,1097
4880 DATA 60,144,4,240,2,56,96,24,96,1
73,85,45,201,155,240,56,758
4890 DATA 201,68,240,34,201,80,240,4,2
01,69,208,44,169,58,141,86,1095
4900 DATA 45,169,155,141,87,45,162,0,1
89,85,45,153,130,45,200,232,2065
4910 DATA 224,20,208,244,24,96,173,87,
45,201,58,208,11,173,86,45,9501
4920 DATA 201,49,144,4,201,57,144,222,
56,96,32,184,72,48,43,169,8779
4930 DATA 200,160,59,32,196,72,48,34,1
69,0,133,138,166,138,189,254,3404
4940 DATA 49,188,22,50,32,196,72,48,17
,230,138,165,138,201,16,208,1982
4950 DATA 235,169,126,160,50,32,196,72
,16,3,76,9,59,96,169,32,5207
4960 DATA 141,192,3,169,155,141,193,3,
208,56,133,140,132,141,160,0,831
4970 DATA 162,0,134,134,177,140,16,2,2
30,134,41,127,201,123,176,19,587
4980 DATA 201,96,176,7,201,64,176,11,2
4,105,32,166,134,240,6,9,7584
4990 DATA 128,208,2,169,46,153,192,3,2
00,192,39,208,211,169,155,141,5067
5000 DATA 231,3,162,11,189,11,73,157,0
,3,202,16,247,32,89,228,9631
5010 DATA 96,64,1,87,128,192,3,48,0,40
,0,78,0,25,0,2,9500
5020 DATA 67,65,83,73,78,201,9,0,6,84,
82,65,77,83,218,9,5570
5030 DATA 0,8,87,65,82,77,83,212,8,0,9
,66,79,79,84,191,6128
5040 DATA 25,0,10,68,79,83,86,69,195,2
5,0,12,68,79,83,73,3966
5050 DATA 78,201,9,0,16,80,79,75,77,83
,203,9,0,17,66,82,3674
5060 DATA 75,75,69,217,41,0,18,82,84,6
7,76,79,203,9,0,32,3834
5070 DATA 73,67,72,73,68,218,9,0,33,73
,67,68,78,79,218,9,5610
5080 DATA 0,34,73,67,67,79,77,218,9,0,
35,73,67,83,84,65,4402
5090 DATA 218,9,0,36,73,67,66,65,76,21
8,9,0,37,73,67,66,3746
5100 DATA 65,72,218,9,0,38,73,67,80,84
,76,218,9,0,39,73,4156
5110 DATA 67,80,84,72,218,9,0,40,73,67
,66,76,76,218,9,0,4481
5120 DATA 41,73,67,66,76,72,218,9,0,42
,73,67,65,88,49,218,6509
5130 DATA 9,0,43,73,67,65,88,50,218,9,
0,65,83,79,85,78 4841
5140 DATA 68,210,9,0,66,67,82,73,84,73
,195,9,0,77,65,84,4681
5150 DATA 82,65,67,212,9,0,82,76,77,65
,82,71,206,9,0,83,4867
5160 DATA 82,77,65,82,71,206,9,0,84,82
,79,87,67,82,211,25,6646
5170 DATA 0,85,67,79,76,67,82,211,25,0
,88,83,65,86,77,83,5622
5180 DATA 195,9,0,106,82,65,77,84,79,2
08,86,0,212,70,82,176,9347
5190 DATA 86,0,218,70,82,197,86,0,224,
70,82,177,86,0,230,70,9834
5200 DATA 82,178,6,0,236,70,82,216,7,0
,237,69,69,88,208,8,8433
5210 DATA 0,238,78,83,73,71,206,8,0,23
9,69,83,73,71,206,10,7887
5220 DATA 0,240,70,67,72,82,70,76,199,
8,0,241,68,73,71,82,7174
5230 DATA 212,6,0,242,67,73,216,25,0,2
43,73,78,66,85,70,198,9342
5240 DATA 9,0,251,82,65,68,70,76,199,2
4,0,252,70,76,80,84,7734
5250 DATA 210,24,0,254,70,80,84,82,178
,25,2,0,86,68,83,76,5003
5260 DATA 83,212,25,2,6,86,66,82,69,65
,203,25,2,8,86,75,3946
5270 DATA 69,89,66,196,25,2,34,86,86,6
6,76,75,201,25,2,36,4301
5280 DATA 86,86,66,76,75,196,9,2,43,83
,82,84,73,77,210,9,6118
5290 DATA 2,47,83,68,77,67,84,204,25,2
,48,83,68,76,83,84,5220
5300 DATA 204,9,2,50,83,83,75,67,84,20
4,8,2,52,76,80,69,4654
5310 DATA 78,200,8,2,53,76,80,69,78,21
4,8,2,54,66,82,75,4663
5320 DATA 75,217,9,2,68,67,79,76,68,83
,212,9,2,111,71,80,5574
5330 DATA 82,73,79,210,121,2,112,80,65
,68,68,76,176,57,2,120,6637
5340 DATA 83,84,73,67,75,176,121,2,124
,80,84,82,73,71,176,57,7691
5350 DATA 2,133,83,84,82,73,71,176,9,2
,162,69,83,67,70,76,5950
5360 DATA 199,233,2,163,84,65,66,77,65
,208,9,2,182,73,78,86,7293
5370 DATA 70,76,199,9,2,190,83,72,70,7
6,79,203,9,2,191,66,7293
5380 DATA 79,84,83,67,210,9,2,192,80,6
7,79,76,82,176,9,2,5666
5390 DATA 193,80,67,79,76,82,177,9,2,1
94,80,67,79,76,82,178,8254
5400 DATA 9,2,195,80,67,79,76,82,179,9
,2,196,67,79,76,79,6771
5410 DATA 82,176,9,2,197,67,79,76,79,8
2,177,9,2,198,67,79,7080
5420 DATA 76,79,82,178,9,2,199,67,79,7
6,79,82,179,9,2,200,7605
5430 DATA 67,79,76,79,82,180,24,2,224,
82,85,78,65,196,25,2,6576
5440 DATA 226,73,78,73,84,65,196,9,2,2
28,82,65,77,83,73,218,9318
5450 DATA 25,2,229,77,69,77,84,79,208,
24,2,231,77,69,77,76,7745
5460 DATA 207,57,2,234,68,86,83,84,65,
212,9,2,240,67,82,83,8276
5470 DATA 73,78,200,6,2,242,67,72,177,
8,2,243,67,72,65,67,7367
5480 DATA 212,8,2,244,67,72,66,65,211,
9,2,251,65,84,65,67,7530
5490 DATA 72,210,5,2,252,67,200,9,2,25
4,68,83,80,70,76,199,9785
5500 DATA 9,2,255,83,83,70,76,65,199,9
,3,0,68,68,69,86,4658
5510 DATA 73,195,8,3,1,68,85,78,73,212
```

```
,9,3,2,68,67,79,3800
5520 DATA 77,78,196,9,3,3,68,83,84,65,
84,211,25,3,4,68,3927
5530 DATA 66,85,70,76,207,9,3,6,68,84,
73,77,76,207,25,3,4926
5540 DATA 8,68,66,89,84,76,207,24,3,10
,68,65,85,88,177,41,6058
5550 DATA 3,26,72,65,84,65,66,211,8,3,
64,73,67,72,73,196,6833
5560 DATA 8,3,65,73,67,68,78,207,8,3,6
6,73,67,67,79,205,6984
5570 DATA 8,3,67,73,67,83,84,193,8,3,6
8,73,67,66,65,204,6802
5580 DATA 8,3,69,73,67,66,65,200,8,3,7
0,73,67,80,84,204,7142
5590 DATA 8,3,71,73,67,80,84,200,8,3,7
2,73,67,66,76,204,7081
5600 DATA 8,3,73,73,67,66,76,200,8,3,7
4,73,67,65,88,177,6713
5610 DATA 8,3,75,73,67,65,88,178,9,3,1
92,80,82,78,66,85,6597
5620 DATA 198,9,3,253,67,65,83,66,85,1
98,8,5,128,76,66,85,6662
5630 DATA 70,198,9,208,0,72,80,79,83,8
0,176,9,208,1,72,80,7248
5640 DATA 79,83,80,177,9,208,2,72,80,7
9,83,80,178,9,208,3,7707
5650 DATA 72,80,79,83,80,179,9,208,4,7
2,80,79,83,77,176,9,7177
5660 DATA 208,5,72,80,79,83,77,177,9,2
08,6,72,80,79,83,77,6976
5670 DATA 178,9,208,7,72,80,79,83,77,1
79,9,208,8,83,73,90,7454
5680 DATA 69,80,176,9,208,9,83,73,90,6
9,80,177,9,208,10,83,7743
5690 DATA 73,90,69,80,178,9,208,11,83,
73,90,69,80,179,8,208,9247
5700 DATA 12,83,73,90,69,205,9,208,18,
67,79,76,80,77,176,9,7274
5710 DATA 208,19,67,79,76,80,77,177,9,
208,20,67,79,76,80,77,6996
5720 DATA 178,9,208,21,67,79,76,80,77,
179,9,208,22,67,79,76,7308
5730 DATA 80,70,176,9,208,23,67,79,76,
80,70,177,9,208,24,67,7632
5740 DATA 79,76,80,70,178,9,208,25,67,
79,76,80,70,179,8,208,9144
5750 DATA 26,67,79,76,66,203,8,208,27,
80,82,73,79,210,9,208,9970
5760 DATA 29,71,82,65,67,84,204,9,208,
30,72,73,84,67,76,210,9146
5770 DATA 9,208,31,67,79,78,83,79,204,
8,210,0,65,85,68,70,7033
5780 DATA 177,8,210,1,65,85,68,67,177,
8,210,2,65,85,68,70,6636
5790 DATA 178,8,210,3,65,85,68,67,178,
8,210,4,65,85,68,70,6688
5800 DATA 179,8,210,5,65,85,68,67,179,
8,210,6,65,85,68,70,6740
5810 DATA 180,8,210,7,65,85,68,67,180,
9,210,8,65,85,68,67,6754
5820 DATA 84,204,9,210;10,82,65,78,68,
79,205,8,210,14,73,82,7886
5830 DATA 81,69,206,8,210,15,83,75,67,
84,204,8,211,0,80,79,8018
5840 DATA 82,84,193,8,211,1,80,79,82,8
4,194,8,211,2,80,65,7773
5850 DATA 67,84,204,8,211,3,80,66,67,8
4,204,9,212,0,68,77,7693
5860 DATA 65,67,84,204,9,212,1,67,72,6
5,67,84,204,25,212,2,8244
5870 DATA 68,76,73,83,84,204,9,212,4,7
2,83,67,82,79,204,9,7893
5880 DATA 212,5,86,83,67,82,79,204,9,2
12,7,80,77,66,65,83,7170
5890 DATA 197,9,212,9,67,72,66,65,83,1
97,8,212,10,87,83,89,7892
5900 DATA 78,195,9,212,11,86,67,79,85,
78,212,7,212,12,80,69,8104
5910 DATA 78,200,7,212,13,80,69,78,214
,8,212,14,78,77,73,69,7706
5920 DATA 206,6,216,0,65,70,208,7,216,
230,70,65,83,195,6,217,2208
5930 DATA 170,73,70,208,6,217,210,70,8
0,201,7,218,68,90,70,82,579
5940 DATA 176,6,218,70,90,70,177,7,218
,96,70,83,85,194,7,218,1329
5950 DATA 102,70,65,68,196,7,218,219,7
0,77,85,204,7,219,40,70,619
5960 DATA 68,73,214,9,221,64,80,76,89,
69,86,204,8,221,137,70,767
5970 DATA 76,68,48,210,8,221,141,70,76
,68,48,208,8,221,152,70,1065
5980 DATA 76,68,49,210,8,221,156,70,76
,68,49,208,8,221,167,70,1419
5990 DATA 83,84,48,210,8,221,171,70,83
,84,48,208,8,221,182,70,2007
6000 DATA 77,79,86,197,6,221,192,69,88
,208,8,221,204,69,88,80,2363
6010 DATA 49,176,6,222,205,76,79,199,8
,222,209,76,79,71,49,176,2018
6020 DATA 9,228,0,69,68,73,84,82,214,9
,228,16,83,67,82,69,7850
6030 DATA 78,214,9,228,32,75,69,89,66,
68,214,9,228,48,80,82,9164
6040 DATA 73,78,84,214,9,228,64,67,65,
83,69,84,214,9,229,80,579
6050 DATA 68,73,83,75,73,214,9,228,83,
68,83,75,73,78,214,7,8952
6060 DATA 228,86,67,73,79,214,7,228,89
,83,73,79,214,9,228,92,1687
6070 DATA 83,69,84,86,66,214,9,228,95,
83,89,83,86,66,214,9,9444
6080 DATA 228,98,88,73,84,86,66,214,9,
228,116,87,65,82,77,83,9327
6090 DATA 214,9,228,119,67,79,76,68,83
,214,0,226,2,227,2,0,8200
6100 DATA 48,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,6148
```

LISTING 2: ASSEMBLY

```
0100 ;SAVE#D:DMPT1.M65
0110     .OPT NO LIST
0120 ;-----------------------------
0130 ;
0140 ;Disk Master
0150 ;the Disk Editor/Disassembler
0160 ;
0170 ;  (C) 1989 BY ST-LOG
0180 ;
0190 ;by: Barry Kolbe
0200 ;and Bryan Schappel
0210 ;
0220 ;-----------------------------
0230 ;
0240     *=  0
0250 PASS .= PASS+1
0260       .IF PASS=1
0270       .INCLUDE #D:SYSEQU.M65
0280       .ENDIF
0290 ;
0300 ;Zero Page Usage
0310 ;
0320     *=  $80
0330 Y1   .DS 1        ;save x/y hold
0340 Y2   .DS 1        ;areas
0350 Y3   .DS 1
0360 X1   .DS 1
0370 X2   .DS 1
0380 X3   .DS 1
0390 TMP1 .DS 1        ;temp variables
0400 TMP2 .DS 1
0410 TMP3 .DS 1
0420 AIFLAG .DS 1      ;asc/internal
0430 T    .DS 2        ;pntr
0440 L    .DS 2        ;pntr
0450 CTMP .DS 1        ;extra temp
0460 SECL .DS 1        ;sec # lo
0470 SECH .DS 1        ;sec # hi
0480 UNITS .DS 1       ;units digit
```

```
0490 TENS .DS 1       ;tens digit
0500 IV  .DS 1        ;inverse flag
0510 HDF .DS 1        ;hex/dex flag
0520 ICNT .DS 1       ;counter
0530 NMD .DS 1
0540 TOTD .DS 1
0550 HXL .DS 1        ;hex lo
0560 HXH .DS 1        ;hex hi
0570 HXORST .DS 1     ;hex/string
0580 FBYTE .DS 1      ;first byte
0590 MATFLG .DS 1     ;match flag
0600 LNKLO .DS 1      ;link lo
0610 LNKHI .DS 1      ;link hi
0620 SSECL .DS 1      ;save sec lo
0630 SSECH .DS 1      ;save sec hi
0640 DENSITY .DS 1    ;density flag
0650 SRCDRV .DS 1     ;source drive
0660 DESDRV .DS 1     ;dest'n drive
0670 ADDR .DS 2       ;disa address
0680 INDR .DS 2       ;pointer
0690 OP  .DS 1        ;first disa byte
0700 DTEMP .DS 2      ;next 2 bytes
0710 NBYTE .DS 1      ;# instr bytes
0720 HXDC .DS 1       ;hex/dec flag
0730 ADD_ON .DS 1     ;address offset
0740 LOC_S .DS 2      ;address start
0750 LOC_E .DS 2      ;address end
0760 CNTLO .DS 2      ;counter
0770 DIN5 .DS 2       ;disa hold
0780 IMODE .DS 1      ;input mode
0790 MAXLEN .DS 1     ;max length
0800 SRFLG .DS 1      ;srch data flag
0810 INPLOC .DS 2     ;input loc
0820 BYTCNT .DS 2     ;segment cnt
0830 GETPTR .DS 1     ;sec offset
0840 STLOAD .DS 2     ;start load
0850 ENLOAD .DS 2     ;end load
0860 TYPE .DS 1       ;disa/file
0870 LABELS? .DS 1    ;label flag
0880 S.SIZE .DS 1     ;sector size
0890 BUFPNT .DS 2     ;output pntr
0900 LINES .DS 1      ;#line in bigbuf
0910 BYTES .DS 1      ;bytes gotten
0920 SFLAG .DS 1      ;source flag
0930 SECCNT .DS 2     ;sector count
0940 SECBYTES .DS 1   ;bytes/sector
0950 SM  =   $2800    ;screen memory
0960 LINE0 = SM+16*40 ;text window
0970 BIGBUF = $8000   ;P: buffer
0980 PRNBUF = $03C0
0990 ;
1000 ;PRINT macro
1010 ;
1020     .MACRO PRINT
1030     LDA # <%1
1040     LDY # >%1
1050     JSR EPRINT
1060     .ENDM
1070 ;
1080 ;Work Buffers
1090 ;
1100     *=  $2C00
1110 MYBUF .DS 128    ;sec work buf
1120 DSBUF .DS 128    ;data buffer
1130 INPLIN .DS 40    ;input line
1140 SRBUF .DS 45     ;search buffer
1150 IBUF .DS 45      ;input buffer
1160 SRCFILE .DS 20   ;source file
1170 DSTFILE .DS 20   ;dest'n file
1180 ;
1190 ;Program Start
1200 ;
1210     *=  $3000    ;origin
1220 START JMP INTRO ;go intro!
1230 ;
1240 ;Intro Display List
1250 ;
1260 INDL .BYTE $70,$70,$70,$70,$70
1270     .BYTE $70,$70,$70,$70,$46
1280     .WORD CREDIT
1290     .BYTE $70,$42
1300     .WORD IL0
1310     .BYTE $02,$70,$06,$20
1320     .BYTE $06,$20,$06,$41
1330     .WORD INDL
1340 ;
1350 ;Info Display List
1360 ;
1370 INFODL .BYTE $70,$70,$70,$46
1380     .WORD CR2
1390     .BYTE $06,$40,$42
1400     .WORD SM+40
1410     .BYTE $02,$02,$02,$02,$02
1420     .BYTE $02,$02,$02,$02,$41
1430     .WORD INFODL
1440 ;
1450 ;Disassembler Display List
1460 ;
1470 DISADL .BYTE $70,$70,$70,$46
1480     .WORD CR2
1490     .BYTE $20,$42
1500     .WORD INFOLN
1510     .BYTE $20,$00,$42
1520     .WORD SM
1530     .BYTE $02,$02,$02,$02,$02
1540     .BYTE $02,$02,$02,$02,$02
1550     .BYTE $02,$02,$02,$02,$02
1560     .BYTE $02,$02,$02,$02,$02
1570     .BYTE $41
1580     .WORD DISADL
1590 ;
1600 IL0 .SBYTE "   by: Barry Ko"
1610     .SBYTE "lbe and Bryan S"
1620     .SBYTE "Chappel     "
1630     .SBYTE "  Produced for"
1640     .SBYTE " A.N.A.L.O.G. C"
1650     .SBYTE "omputing  "
1660     .SBYTE "Option  di"
1670     .SBYTE "sk editor "
1680     .SBYTE "select  di"
1690     .SBYTE "sassembler"
1700     .SBYTE " start  ex"
1710     .SBYTE "it        "
1720 ;
1730 INFOLN .SBYTE "   MNE  Operand"
1740     .SBYTE "          ADDR "
1750     .SBYTE "B1 B2 B3  "
1760 ;
1770 ;Disassembler prompts
1780 ;
1790 P0  .BYTE "Disa Type [F/D]",0
1800 P3  .BYTE "Dest'n [E/P/Dn]",0
1810 P4  .BYTE "   Labels [Y/N]",0
1820 P5  .BYTE "    Source File",0
1830 P1  .BYTE "Src Drive [1-8]",0
1840 P2  .BYTE "  Density [1-2]",0
1850 P6  .BYTE " Sector [1-2D0]",0
1860 P7  .BYTE "   # of Sectors",0
1870 P8  .BYTE "  Offset [0-7F]",0
1880 P9  .BYTE "Origin [0-FFFF]",0
1890 PA  .BYTE "Byt/Sec [7D/80]",0
1900 ;
1910 ;INCLUDE Support Files
1920 ;
1930     .INCLUDE #D1:DMPT2.M65
1940     .INCLUDE #D1:DMPT3.M65
1950     .INCLUDE #D1:DMPT4.M65
1960 ;
1970 ;Intro Screen
1980 ;
1990 INTRO CLD        ;chill decimal
2000     LDX #$FF     ;clear stack
2010     TXS
2020     LDA POKMSK
2030     AND #$7F
2040     STA POKMSK
2050     STA IRQEN
2060     LDA # <INDL ;turn on the
2070     STA SDLSTL  ;intro display
2080     LDA # >INDL ;list.
2090     STA SDLSTL+1
2100     LDA #$22     ;normal PF
2110     STA SDMCTL
2120     LDA #$0A     ;white text
2130     STA COLOR1
2140     LDA #$00     ;black border
2150     STA COLOR4
2160     LDA #2       ;grey backgrnd
2170     STA LMARGN   ;left marg=2
2180     STA COLOR2
2190     LDA #$94     ;blue gri text
2200     STA COLOR3
2210 I_LOOP LDA CONSOL ;get consol
2220     CMP #6       ;start?
2230     BNE T_5      ;no.
2240     JMP WARMSV   ;yes, so long!
2250 T_5 CMP #5       ;select?
2260     BEQ GO_DIS   ;yes. disa.
2270     CMP #3       ;option?
2280     BNE I_LOOP   ;no, loop
2290     JMP DISKED   ;Disk Editor.
2300 ;
2310 ;Disassembler Data Entry
2320 ;
2330 GO_DIS LDA # <INFODL ;install
2340     STA SDLSTL  ;the data
2350     LDA # >INFODL ;entry DLIST
2360     STA SDLSTL+1
2370     JSR CLRSCN  ;clear screen
2380     LDA # <P0   ;point to the
2390     STA INDR    ;first prompt
2400     LDA # >P0
2410     STA INDR+1
2420     JSR PROMPT  ;Disk/Files
2430 ;
2440 P00 JSR GETONE  ;get input
2450     BCC P01     ;ok?
2460 GO_INT JMP INTRO ;no way.
2470 ;
2480 P01 LDA IBUF    ;get char
2490     AND #223    ;make upper/norm
2500     LDX #0      ;0=files
2510     CMP #'F     ;was it F?
2520     BEQ P02     ;yes!
2530     CMP #'D     ;a D?
2540     BNE P00     ;no.
2550     INX         ;up flag
2560 P02 STX TYPE    ;save type
2570 ;
2580     JSR PROMPT  ;dest'n device
2590 P30 JSR GETFN   ;get name
2600     BCS GO_INT  ;bad input
2610     LDY #20     ;copy to pos 20
2620     JSR CHECKFN ;check filename
2630     BCS P30     ;oops, bad one.
2640     JSR G.DRV   ;get drive #
2650     STA DESDRV  ;save it.
2660 ;
2670     JSR PROMPT  ;want labels?
2680 P40 JSR GETONE  ;get response.
2690     BCS GO_INT  ;bad one.
2700     LDA IBUF    ;get char
2710     AND #223
2720     LDX #0      ;0=labels.
2730     CMP #'Y     ;a Y?
2740     BEQ P41     ;yes.
2750     CMP #'N     ;try N?
2760     BNE P40     ;no.
2770     INX         ;1=no labels
2780 P41 STX LABELS? ;save choice
2790 ;
2800     LDA TYPE    ;files or disk?
2810     BNE F_SECS  ;want disk.
2820     JSR PROMPT  ;filename
2830 ;
2840 P50 JSR GETFN   ;get filename
2850 G.0 BCS GO_INT  ;bad one.
2860     LDY #0      ;posit 0
2870     JSR CHECKFN ;check Fname
2880     BCS P50     ;oops.
2890     JSR G.DRV   ;get drive
2900     STA SRCDRV  ;save it.
2910     JMP DIS_IT  ;and disa!
2920 ;
2930 F_SECS JSR UP_INDR ;skip fname
2940     JSR PROMPT  ;source drive
2950 P10 JSR GETONE  ;get response
2960     BCS GO_INT  ;bad one.
2970     LDA IBUF    ;get the char
2980     AND #$7F
2990     SEC         ;make 1-8
3000     SBC #'0
3010     BMI P10     ;<0? yes!
3020     CMP #9      ;=>9?
3030     BCS P10     ;yes.
3040     STA SRCDRV  ;save it
3050 ;
3060     JSR PROMPT  ;density
3070 P20 JSR GETONE  ;input.
3080 G.1 BCS G.0     ;bad input.
3090     LDA IBUF    ;get char
3100     AND #$7F    ;mask inverse
3110     SEC         ;make 0-1
3120     SBC #'1
3130     BMI P20     ;<0?, yes.
3140     CMP #2      ;>1?
3150     BCS P20     ;yes.
3160     STA DENSITY ;save density
3170     TAX         ;X=A
3180     LDA SDLO,X  ;get max sec #
3190     STA FR0     ;and put it in
3200     LDA SDHI,X  ;FR0.
3210     STA FR0+1
3220     JSR BIN2HEX ;make hex.
3230     LDY #1      ;copy max #
3240 D.CP LDA LBUFF,Y ;into the next
3250     STA P6+10,Y ;prompt.
3260     INY
3270     CPY #4
3280     BNE D.CP
3290 ;
3300     JSR PROMPT  ;sector
3310 P60 JSR SPC_NUM ;get number
3320     BCS G.1
3330     JSR TESTFR0 ;in range?
3340     BCS P60     ;no.
3350     LDA FR0     ;yes, so we
3360     STA SECL    ;save the sector
3370     LDA FR0+1   ;number for
3380     STA SECH    ;later.
3390 ;
```

```
3400      JSR PROMPT  ;#secs
3410 P70  JSR SPC_NUM ;get number
3420 G.2  BCS G.1     ;bad one.
3430      LDA FR0     ;is number 0?
3440      ORA FR0+1
3450      BEQ P70     ;yes, that's bad
3460      LDA FR0     ;save the number
3470      STA SECCNT  ;of sectors for
3480      CLC         ;later & add on
3490      ADC SECL    ;start sector
3500      STA FR0     ;so we can test
3510      LDA FR0+1   ;to see if too
3520      STA SECCNT+1 ;many sec's
3530      ADC SECH    ;will be read.
3540      STA FR0+1
3550      JSR TESTFR0 ;in range?
3560      BCS P70     ;no way.
3570 ;
3580      JSR PROMPT  ;offset
3590 P80  JSR SPC_NUM ;get number
3600      BCS G.2
3610      LDA FR0+1   ;test hi byte
3620      BNE P80     ;)? yes!
3630      LDA FR0     ;get lo byte
3640      CMP #$80    ;=) $80?
3650      BCS P80     ;yes.
3660      STA GETPTR  ;save for later
3670 ;
3680      JSR PROMPT  ;origin
3690 P90  JSR SPC_NUM ;get number
3700      BCS G.2
3710      LDA FR0     ;copy the num
3720      STA ADDR    ;into the addr
3730      LDA FR0+1   ;for the disa-
3740      STA ADDR+1  ;ssembler.
3750 ;
3760      JSR PROMPT  ;sector size.
3770 PA0  JSR SPC_NUM ;get number
3780      BCS G.2
3790      LDA FR0+1   ;no hi byte
3800      BNE PA0     ;allowed!
3810      LDA FR0     ;test lo byte
3820      CMP #125    ;125 bytes?
3830      BEQ G.SZ    ;yes!
3840      CMP #128    ;128 bytes?
3850      BNE PA0     ;no.
3860 G.SZ STA S.SIZE ;save size
3870 ;
3880 ;Disassembly Entry Point
3890 ;
3900 DIS_IT JSR CLRSCN ;zap screen
3910      STY SFLAG   ;clear flag
3920      LDA # <DISADL ;install the
3930      STA SDLSTL  ;disa DLIST
3940      LDA # >DISADL
3950      STA SDLSTL+1
3960      JSR OPENIN  ;open input
3970      JSR OPENOUT ;open output
3980 ;
3990 D_LOOP JSR DISAMEM ;do instr.
4000      JSR PRINT.IT ;print it
4010      LDA #0      ;clear counter
4020      STA BYTES
4030      LDX #$FF    ;get a $FF
4040 C_LOOP LDA CH    ;key press?
4050      CMP #$FF
4060      BEQ D_LOOP  ;no.
4070      STX CH      ;clear CH
4080      CMP #28     ;escape?
4090      BNE T.SPC   ;no.
4100 END.IT STX CH    ;clear CH and
4110      JMP D_OVER  ;exit.
4120 ;
4130 T.SPC CMP #33    ;Space=pause
4140      BNE D_LOOP
4150 ;
4160 K.WAIT LDA CH    ;get keypress
4170      CMP #28     ;escape?
4180      BEQ END.IT  ;yes, exit
4190      CMP #33     ;wait for SPC
4200      BNE K.WAIT  ;to stop the
4210      STX CH      ;pause.
4220      JMP D_LOOP  ;loop again.
4230 ;
4240 ;Get drive number
4250 ;
4260 G.DRV LDA IBUF+1 ;get char
4270      SEC         ;set carry
4280      SBC #'0     ;subtract '0
4290      RTS         ;and leave
4300 ;
4310 ;E: Print routine
4320 ;
4330 EPRINT STA E.LP+1 ;save lo byte
4340      STY E.LP+2  ;save hi byte
4350 E.LP LDA $FFFF   ;get char
4360      BEQ E.DN    ;if 0, all done
4370      JSR E.PUT   ;put the char
4380      INC E.LP+1  ;bump pointer
4390      BNE E.LP
4400      INC E.LP+2
4410      BNE E.LP    ;and loop
4420 E.DN RTS         ;so long.
4430 ;
4440 ;E: Put byte routine
4450 ;
4460 E.PUT CMP #EOL   ;is it a CR?
4470      BNE E.EOL   ;no.
4480 PUTCR LDA LMARGN ;set xpos to
4490      STA COLCRS  ;the margin
4500      INC ROWCRS  ;bump ypos
4510      JMP SCROLL  ;check scroll
4520 ;
4530 E.EOL JSR ASC2IC ;make icode
4540      PHA         ;save it
4550      LDY ROWCRS  ;get ypos
4560      LDA GL,Y    ;look up the
4570      STA L       ;address of this
4580      LDA GH,Y    ;line and store
4590      STA L+1     ;in temp area
4600      PLA         ;get char back
4610      LDY COLCRS  ;get xpos
4620      STA (L),Y   ;on screen!
4630 ;
4640      INY         ;up xpos
4650      CPY #39     ;at rmargn?
4660      BCC E.LV    ;no.
4670      BEQ E.LV    ;just there.
4680      INC ROWCRS  ;next line
4690      JSR SCROLL  ;scroll screen
4700      LDY LMARGN  ;get margin
4710 E.LV STY COLCRS  ;set xpos
4720      RTS         ;and leave
4730 ;
4740 ;Check to see if screen must
4750 ;be scrolled.
4760 ;
4770 SCROLL LDY ROWCRS ;get ypos
4780      CPY #21     ;at line 21?
4790      BCC S.NO    ;no.
4800      LDA # <SM   ;get the addrs
4810      STA T       ;of line 0 & 1
4820      LDA # >SM
4830      STA T+1
4840      LDA # <[SM+40]
4850      STA L
4860      LDA # >[SM+40]
4870      STA L+1
4880      LDX #0      ;line counter
4890 S.M1 LDY #39     ;now move 1 line
4900 S.M2 LDA (L),Y   ;up 40 bytes
4910      STA (T),Y
4920      DEY
4930      BPL S.M2
4940      LDA L+1     ;now update the
4950      STA T+1     ;pointers.
4960      LDA L
4970      STA T
4980      CLC
4990      ADC #40
5000      STA L
5010      LDA L+1
5020      ADC #0
5030      STA L+1
5040      INX         ;up line count
5050      CPX #21     ;at end?
5060      BNE S.M1    ;no way.
5070      DEC ROWCRS  ;dec ypos
5080 S.NO RTS         ;and leave
5090 ;
5100 ;Print a Prompt
5110 ;
5120 PROMPT JSR PUTCR ;print CR
5130      LDA #$40    ;force uppercase
5140      STA SHFLOK
5150      LDA #0      ;and no inverse
5160      STA INVFLG
5170      LDA INDR    ;get addr of
5180      LDY INDR+1  ;the prompt
5190      JSR EPRINT  ;print it
5200      LDA #':     ;put a ':'
5210      JSR E.PUT
5220      LDA #$20    ;and a space
5230      JSR E.PUT
5240      JSR G_SCR   ;clear line
5250 UP_INDR LDA INDR ;add 16 to get
5260      CLC         ;to the next
5270      ADC #16     ;prompt.
5280      STA INDR
5290      BCC UP_HI
5300      INC INDR+1
5310 UP_HI RTS        ;so long.
5320 G_SCR LDA COLCRS ;get xpos
5330      LDY ROWCRS  ;and ypos
5340      CLC         ;add line addr
5350      ADC GL,Y
5360      STA INPLOC
5370      LDA GH,Y
5380      ADC #0
5390      STA INPLOC+1
5400      LDY #20     ;zap 21 chars
5410      LDA #0      ;a space
5420 G_CLR STA (INPLOC),Y ;on screen
5430      DEY         ;down counter
5440      BPL G_CLR   ;and loop
5450      RTS         ;so long.
5460 ;
5470 ;Get 1 byte via INPUT
5480 ;
5490 GETONE JSR G_SCR ;clear line
5500      LDY #0      ;force text mode
5510      STY IMODE
5520      INY         ;1 char
5530      JMP INPUT   ;get input
5540 ;
5550 ;Get a Filename
5560 ;
5570 GETFN JSR G_SCR ;zap line
5580      LDY #0      ;force text mode
5590      STY IMODE
5600      LDY #15     ;get 15 chars
5610      JMP INPUT
5620 ;
5630 ;Special get number
5640 ;
5650 SPC_NUM JSR G_SCR ;zap line
5660      JMP G_NUM   ;get a number
5670 ;
5680 ;Test FR0 for sector range
5690 ;
5700 TESTFR0 LDA FR0 ;is FR0 a 0?
5710      ORA FR0+1
5720      BEQ T.ERR   ;yes!
5730      LDX DENSITY ;get density
5740      LDA FR0+1   ;check to see
5750      CMP SDHI,X  ;if the sector
5760      BCC T.GD    ;number in FR0
5770      BEQ T.LO    ;is allowed.
5780      BCS T.ERR
5790 T.LO LDA FR0
5800      CMP SDLO,X
5810      BCC T.GD
5820      BEQ T.GD
5830 T.ERR SEC        ;signal error
5840      RTS
5850 T.GD CLC         ;signal good
5860      RTS
5870 ;
5880 ;Check Filename for Dx:
5890 ;
5900 CHECKFN LDA IBUF ;get 1st char
5910      CMP #EOL    ;is it EOL?
5920      BEQ CH.B    ;yes.
5930      CMP #'D     ;is it 'D'?
5940      BEQ FN.D    ;yes.
5950      CMP #'P     ;is it 'P'?
5960      BEQ CH.C    ;yes.
5970      CMP #'E     ;is it 'E'?
5980      BNE CH.B    ;no.
5990 CH.C LDA #':     ;make 2nd char
6000      STA IBUF+1  a ':' and the
6010      LDA #EOL    ;3rd char an EOL
6020      STA IBUF+2
6030 CH.R LDX #0      ;zero index
6040 CH.L LDA IBUF,X ;get input char
6050      STA SRCFILE,Y ;save in buf
6060      INY         ;up buf index
6070      INX         ;up input indx
6080      CPX #20     ;done 20 yet?
6090      BNE CH.L    ;no.
6100      CLC         ;signal good.
6110      RTS         ;and leave.
6120 ;
6130 FN.D LDA IBUF+2 ;get 3rd char
6140      CMP #':     ;3rd char a ':'?
6150      BNE CH.B    ;no, bad name
6160      LDA IBUF+1  ;get drive #
6170      CMP #'1     ;less than 1?
6180      BCC CH.B    ;yes.
6190      CMP #'9     ;greater than 8?
6200      BCC CH.R
6210 CH.B SEC         ;signal bad
6220      RTS
6230 ;
6240 ;Dump Editor Screen to Printer
6250 ;
6260 DMPSCN JSR PRNLFD
6270      BMI PRERR
```

LISTING 3: ASSEMBLY

```
6280      LDA # <BORDER ;print top
6290      LDY # >BORDER ;border
6300      JSR DMPSUB
6310      BMI PRERR
6320      LDA #0        ;count lines
6330      STA T         ;of sectr info
6340 DMLP LDX T         ;get index
6350      LDA GL,X      ;Lo/Hi bytes
6360      LDY GH,X      ;line address
6370      JSR DMPSUB    ;print it
6380      BMI PRERR
6390      INC T         ;next line
6400      LDA T
6410      CMP #16       ;done 16?
6420      BNE DMLP      ;nope
6430      LDA # <STATUS ;print
6440      LDY # >STATUS ;status line
6450      JSR DMPSUB
6460      BPL DMOK      ;Y has error
6470 PRERR JMP IOERR ;dble return
6480 DMOK RTS
6490 ;
6500 ;Printer Linefeed
6510 ;
6520 PRNLFD LDA #$20
6530      STA PRNBUF
6540      LDA #EOL
6550      STA PRNBUF+1
6560      BNE SAP
6570 ;
6580 ;Print a Line using SIO
6590 ;
6600 DMPSUB STA L     ;save address
6610      STY L+1     ;for indirect
6620      LDY #0
6630 DML LDX #0       ;use as an
6640      STX TMP1    ;inverse flag
6650      LDA (L),Y   ;get byte
6660      BPL DMJ     ;inversed?
6670      INC TMP1    ;yes-set flag
6680 DMJ AND #$7F     ;mask it off
6690      CMP #123    ;skip control
6700      BCS PERD    ;characters
6710      CMP #96
6720      BCS PRK
6730      CMP #64
6740      BCS PERD
6750      CLC         ;change to
6760      ADC #$20    ;ATASCII
6770 PRK LDX TMP1     ;was it inverse?
6780      BEQ PORK    ;no
6790      ORA #$80    ;add back in
6800      BNE PORK
6810 PERD LDA #'.     ;'.' for contrls
6820 PORK STA PRNBUF,Y ;into buffer
6830      INY
6840      CPY #$27    ;done w/line?
6850      BNE DML     ;no
6860      LDA #EOL    ;add EOL
6870      STA PRNBUF+$27
6880 ;
6890 SAP LDX #$0B     ;copy cmds
6900 SIP LDA PSIOCB,X ;to DDEVIC
6910      STA $0300,X
6920      DEX
6930      BPL SIP
6940      JSR SIOV    ;let SIO print
6950      RTS
6960 ;
6970 PSIOCB .BYTE $40 ;printer
6980      .BYTE $01   ;unit 1
6990      .BYTE $57   ;write
7000      .BYTE $80   ;send bit 7=out
7010      .WORD PRNBUF ;buffer
7020      .WORD $30   ;time out
7030      .WORD $28   ;buf length
7040      .BYTE $4E   ;normal prnt
7050      .BYTE $00   ;unused
7060 ;
7070 ;Get Label Data
7080 ;
7090      .INCLUDE #D:DMPT5.M65
7100 ;
7110 ;Find where program ends
7120 ;
7130      .OPT LIST
7140 ENDPROG = *
7150      .OPT NO LIST
7160 ;
7170 ;Add Run address
7180 ;
7190      *=  RUNAD
7200      .WORD START
7210      .END
```

```
0100 ;SAVE#D:DMPT2.M65
0110      .OPT NO LIST
0120 ;
0130 ;--------------------;
0140 ;  Disk Master part 2 ;
0150 ;   by: Barry Kolbe   ;
0160 ;       (c) 1989      ;
0170 ;--------------------;
0180 ;
0190 ;Graphics 0 Display List
0200 ;
0210 GR0DL .BYTE $70,$70,$10,$46
0220      .WORD CR1
0230      .BYTE $10,$42
0240      .WORD BORDER
0250      .BYTE $42
0260      .WORD SM
0270      .BYTE 2,2,2,2,2
0280      .BYTE 2,2,2,2,2
0290      .BYTE 2,2,2,2,2,$10
0300      .BYTE $01
0310 OLDD .WORD CONT ;flip bottom
0320 CONT .BYTE $42  ;of screen
0330      .WORD STATUS ;for HELP
0340      .BYTE 0,2    ;screen at
0350      .BYTE $30,$42
0360      .WORD INPLIN
0370      .BYTE $10,$42
0380      .WORD LINE0
0390      .BYTE $20,$42
0400      .WORD PRLINE
0410      .BYTE $41
0420      .WORD GR0DL
0430 ;
0440 ;The Help Screen Display
0450 ;Code used by HELPME
0460 ;
0470 HELPDL .BYTE $42
0480      .WORD HELP5
0490      .BYTE 0,2,2,2,2,2,$41
0500      .WORD GR0DL
0510 ;
0520 ;Low and High bytes of
0530 ;Screen Lines
0540 ;
0550 GL   .BYTE  <SM, <[SM+40]
0560      .BYTE  <[SM+80], <[SM+120]
0570      .BYTE  <[SM+160], <[SM+200]
0580      .BYTE  <[SM+240], <[SM+280]
0590      .BYTE  <[SM+320], <[SM+360]
0600      .BYTE  <[SM+400], <[SM+440]
0610      .BYTE  <[SM+480], <[SM+520]
0620      .BYTE  <[SM+560], <[SM+600]
0630      .BYTE  <[SM+640], <[SM+680]
0640      .BYTE  <[SM+720], <[SM+760]
0650      .BYTE  <[SM+800], <[SM+840]
0660      .BYTE  <[SM+880], <[SM+920]
0670 ;
0680 GH   .BYTE  >SM, >[SM+40]
0690      .BYTE  >[SM+80], >[SM+120]
0700      .BYTE  >[SM+160], >[SM+200]
0710      .BYTE  >[SM+240], >[SM+280]
0720      .BYTE  >[SM+320], >[SM+360]
0730      .BYTE  >[SM+400], >[SM+440]
0740      .BYTE  >[SM+480], >[SM+520]
0750      .BYTE  >[SM+560], >[SM+600]
0760      .BYTE  >[SM+640], >[SM+680]
0770      .BYTE  >[SM+720], >[SM+760]
0780      .BYTE  >[SM+800], >[SM+840]
0790      .BYTE  >[SM+880], >[SM+920]
0800 ;
0810 CREDIT .SBYTE "disk master "
0820      .BYTE "HcI",0,"QYXY"
0830 CR1 .SBYTE "    disk  "
0840      .SBYTE "editor    "
0850 CR2 .SBYTE " disk  dis"
0860      .SBYTE "assembler "
0870      .SBYTE "     data "
0880      .SBYTE "entry     "
0890 ;
0900 STATUS .SBYTE " SEC:      FI"
0910      .SBYTE "L:      LNK:     "
0920      .SBYTE "BYT:     "
0930 ;
0940 DRLIN .SBYTE " SRC:1     DS"
0950      .SBYTE "T:1    INP:H     "
0960      .SBYTE "DNS:1    "
0970 ;
0980 ;Help Screen
0990 ;
1000 HELP5 .SBYTE "Disk Master Help"
1010      .SBYTE " Screen - Select C"
```

```
1020      .SBYTE "ommand"
1030      .SBYTE "R-read  +-up    "
1040      .SBYTE "--dn    E-edit  "
1050      .SBYTE "W-write "
1060      .SBYTE "U-undo  N-#base "
1070      .SBYTE "C-chars B-bfill "
1080      .SBYTE "X-eor   "
1090      .SBYTE "T-trace F-file# "
1100      .SBYTE "L-link# V-dens  "
1110      .SBYTE "M-menu  "
1120      .SBYTE "S-serch H-H)D   "
1130      .SBYTE "A-D)H   I-input "
1140      .SBYTE "O-sorce "
1150      .SBYTE "D-destn R-rfrsh "
1160      .SBYTE EOL,"-abort K-hue ↑ "
1170      .SBYTE "Y-lum ↑ "
1180 ;
1190 PRLINE .SBYTE "  Press CONTROL-"
1200      .SBYTE "P to print this "
1210      .SBYTE "screen    "
1220 ;
1230 ;The Beginning of the
1240 ;Disk Editor
1250 ;
1260 DISKED JSR CLRSCN ;clear screen
1270      JSR ERAINP  ;clear input
1280      LDA # <GR0DL ;set up DL
1290      STA SDLSTL
1300      LDA # >GR0DL
1310      STA SDLSTL+1
1320      LDA # <[INPLIN+6]
1330      STA INPLOC
1340      LDA # >[INPLIN+6]
1350      STA INPLOC+1
1360      LDX #0        ;initialize
1370      STX SRFLG
1380      STX SECH
1390      STX AIFLAG  ;ASC/INT flag
1400      STX HDF     ;hex/dec flag
1410      STX HXORST  ;input flag
1420      STX DENSITY ;0=sngl 1 =dbl
1430      INX
1440      STX SECL
1450      STX SRCDRV
1460      STX DESDRV
1470      STX IMODE
1480      JSR DOLEFT  ;show border
1490      LDA #0      ;show empty
1500      JSR BFC
1510      JSR MASIN
1520 JG   JSR C_OFF
1530      JSR GETCOM  ;get comand
1540      JSR ERAINP  ;zap input
1550      JMP JG      ;more
1560 ;
1570 ;Get a Command
1580 ;
1590 GETCOM LDA CH    ;force uppercase
1600      CMP #$FF
1610      BEQ GETCOM  ;get key
1620      LDX #$FF
1630      STX CH
1640      PHA         ;save it
1650      LDA # <CONT ;reset DLIST
1660      STA OLDD
1670      LDA # >CONT
1680      STA OLDD+1
1690      PLA         ;get key
1700      LDX #26     ;check it
1710 CC1 CMP COMKEY,X
1720      BEQ CC2     ;valid
1730      DEX
1740      BPL CC1
1750      BMI GETCOM  ;overs
1760 CC2 TXA          ;x 2 for offset
1770      ASL A
1780      TAX
1790      ASL A
1800      TAY
1810      CPY #40
1820      BCC N40
1830      LDY #40
1840 N40 JSR SHO_COM
1850      LDA COMTAB,X ;get address
1860      STA JOFF+1  ;of routine
1870      LDA COMTAB+1,X
1880      STA JOFF+2
1890      JSR ERATXT  ;window
1900 JOFF JSR $FFFF   ;carry out
1910 ;
1920 C_OFF LDY #40
1930 ;
1940 ;Show the Command
1950 ;
1960 SHO_COM TXA
1970      PHA
```

```
1980      LDX #0
1990 SHC  LDA CTXT,Y
2000      STA INPLIN,X
2010      INY
2020      INX
2030      CPX #4
2040      BNE SHC
2050      LDA #$9A
2060      STA INPLIN+4
2070      PLA
2080      TAX
2090      RTS
2100 ;
2110 ;Command Keys
2120 ;
2130 COMKEY .BYTE $2A,$39,$22,$15,$01
2140      .BYTE $38,$3E,$00,$28,$2E
2150      .BYTE $12,$23,$66,$0B,$08
2160      .BYTE $3A,$36,$76,$0E,$06
2170      .BYTE $0D,$2D,$10,$37,$25
2180      .BYTE $B6,$8A
2190 ;
2200 ;Get a Key
2210 ;
2220 GETKEY LDA $E425 ;the easy way
2230      PHA         ;to get keys
2240      LDA $E424   ;from the OS!
2250      PHA
2260      RTS
2270 ;
2280 ;Get Input until ESCape or
2290 ;RETURN is pressed
2300 ;
2310 INPUT STY MAXLEN ;save length
2320      LDY #0      ;counter
2330 INLOOP STY Y2    ;save it
2340      LDA #$80    ;cursor is a
2350      STA (INPLOC),Y ;block.
2360      JSR GETKEY  ;get a key
2370      LDY Y2      ;restore Y
2380      CMP #EOL    ;is it EOL?
2390      BEQ INDONE  ;yes!
2400      CMP #27     ;is it ESC?
2410      BNE NOESC   ;NO!
2420      SEC         ;signal abort
2430      RTS         ;so long.
2440 NOESC CMP #126   ;delete?
2450      BNE NODEL   ;nope.
2460      CPY #0      ;on 1st char?
2470      BEQ INLOOP  ;yes!
2480      LDA #0      ;remove cursor
2490      STA (INPLOC),Y
2500      DEY         ;decrease cntr
2510      BPL INLOOP  ;and loop
2520 ;
2530 NODEL LDX IMODE  ;get mode?
2540      BEQ IS_OK   ;string mode.
2550      AND #$7F    ;mask inverse
2560      CMP #'9+1   ;>9?
2570      BCC ITS     ;no.
2580      AND #223    ;make uppercase.
2590 ITS  JSR CHEX    ;is legal?
2600      CPX #0      ;test X.
2610      BPL IS_OK   ;legal!
2620      BMI INLOOP  ;no good.
2630 ;
2640 IS_OK CPY MAXLEN ;at max len?
2650      BEQ INLOOP  ;yes.
2660      STA IBUF,Y  ;no save char
2670      JSR ASC2IC  ;and put it
2680      STA (INPLOC),Y ;on screen
2690      INY         ;up cntr
2700      BNE INLOOP  ;and loop.
2710 ;
2720 INDONE STA IBUF,Y ;save EOL
2730      LDA #0      ;erase cursor
2740      STA (INPLOC),Y
2750      CLC         ;show good.
2760      RTS         ;boogie.
2770 ;
2780 ;Erase Input Line
2790 ;
2800 ERAINP LDA #0
2810      LDX #34
2820 ER1  STA INPLIN+5,X
2830      DEX
2840      BPL ER1
2850      RTS
2860 ;
2870 ;Get a String
2880 ;
2890 G_STR LDA #0
2900      STA IMODE
2910      LDY #33
2920      JMP INPUT
2930 ;
2940 ;Get a Hexstring
2950 ;
2960 G_HEX LDA #1     ;hex mode.
2970      STA IMODE
2980      LDY #32     ;32 chars max
2990      JSR INPUT   ;get input.
3000      BCC MEV     ;ok?
3010      RTS         ;no.
3020 MEV  LDX #0      ;get length of
3030 G4E  LDA IBUF,X  ;the input.
3040      CMP #EOL
3050      BEQ G4L
3060      INX
3070      BNE G4E
3080 ;
3090 G4L  TXA         ;is the input
3100      AND #1      ;even?
3110      BEQ G4R     ;yes!
3120 ;
3130      LDX #43     ;no. add a
3140 MDN  LDA IBUF,X  ;leading zero
3150      STA IBUF+1,X ;to the input
3160      DEX         ;buffer.
3170      BPL MDN
3180      LDA #'0
3190      STA IBUF
3200 G4R  LDX #0      ;input pntr
3210      LDY #0      ;output pntr
3220 GH2  LDA IBUF,X  ;get char
3230      CMP #EOL    ;all done?
3240      BEQ GHID    ;yes!
3250      STA HXH     ;save it
3260      INX         ;get next char
3270      LDA IBUF,X
3280      STA HXL     ;save it
3290      STX X3      ;save x
3300      STY Y3      ;save y
3310      JSR HEX2BIN ;make binary
3320      LDY Y3      ;get Y
3330      LDX X3      ;get X
3340      STA IBUF,Y  ;save binary
3350      INY         ;up Y
3360      INX         ;up X
3370      BNE GH2     ;and loop
3380 GHID CLC         ;signal good
3390      RTS         ;later.
3400 ;
3410 ;Get a Number
3420 ;
3430 G_NUM LDA #1     ;hex mode.
3440      STA IMODE
3450      LDY #4      ;4 chars
3460      JSR INPUT   ;grab it.
3470      BCC GNK     ;good?
3480      RTS         ;no.
3490 GNK  JSR ZFR     ;zap FR0
3500      TAY         ;Y=0
3510 G4LOOP LDA IBUF,Y ;get a char
3520      CMP #EOL    ;done?
3530      BNE TESTIT  ;nope.
3540 G4G  CLC         ;signal good
3550      RTS         ;spater.
3560 ;
3570 TESTIT JSR CHEX ;get value
3580      ASL FR0     ;*2
3590      ROL FR0+1
3600      ASL FR0     ;*4
3610      ROL FR0+1
3620      ASL FR0     ;*8
3630      ROL FR0+1
3640      ASL FR0     ;*16
3650      ROL FR0+1
3660      TXA         ;get value
3670      ORA FR0     ;add to FR0
3680      STA FR0     ;save it
3690      INY         ;up count
3700      BNE G4LOOP  ;and loop
3710 ;
3720 ;Clear out Screen Memory
3730 ;
3740 CLRSCN LDA #0
3750      LDY #0
3760 CLRS STA SM,Y
3770      STA SM+$0100,Y
3780      STA SM+$0200,Y
3790      STA SM+$0300,Y
3800      INY
3810      BNE CLRS
3820      LDA LMARGN
3830      STA COLCRS
3840      TYA
3850      STA ROWCRS
3860      RTS
3870 ;
3880 ;Show the Screen Borders
3890 ;
3900 DOLEFT LDA GL    ;first line
3910      STA T       ;of memory
3920      LDA GH
3930      STA T+1
3940      LDX #0
3950 LFB  LDY #0      ;left column
3960      LDA LEFTB,X
3970      STA (T),Y
3980      STY Y1
3990      LDY #28     ;repeat in
4000      STA (T),Y   ;middle
4010      LDY Y1
4020      INX         ;over 1
4030      INY
4040      LDA LEFTB,X ;column 2
4050      STA (T),Y
4060      STY Y1
4070      LDY #29     ;and in middle
4080      STA (T),Y
4090      LDY Y1
4100      JSR ADD40   ;down 1 line
4110      INX
4120      CPX #32     ;16 lines?
4130      BNE LFB
4140      RTS
4150 ;
4160 ;Add 40 onto pointer to
4170 ;move down one line
4180 ;
4190 ADD40 LDA T
4200      CLC
4210      ADC #$28
4220      STA T
4230      BCC A40
4240      INC T+1
4250 A40  RTS
4260 ;
4270 ;Display the Sector Data
4280 ;
4290 SHWSEC LDA GL    ;start at top
4300      STA T
4310      LDA GH
4320      STA T+1
4330      LDA #2      ;y pos =2
4340      STA Y2
4350      LDA HDF     ;hex or dec?
4360      BEQ H9      ;HEX
4370      JMP SDEC
4380 H9   LDY #0
4390 H1   JSR ZFR
4400      STY Y1
4410      LDA MYBUF,Y ;get a byte
4420      STA FR0
4430      JSR BIN2HEX ;to hex
4440      INC Y1
4450      LDY Y2      ;screen pos Y
4460      INY
4470      LDA LBUFF+2 ;hi byte of hex
4480      JSR A5C2IC
4490      STA (T),Y   ;show it
4500      LDA LBUFF+3 ;now low byte
4510      JSR A5C2IC
4520      INY
4530      STA (T),Y
4540      INY
4550      STY Y2
4560      CPY #26     ;end of line?
4570      BEQ H4      ;yes
4580 H5   LDY Y1      ;end of buffer?
4590      CPY #$80
4600      BNE H1      ;no
4610      RTS
4620 H4   JSR ADD40   ;down 1 line
4630      LDA #2      ;reset LMargin
4640      STA Y2
4650      JMP H5
4660 ;
4670 ;Show Sector in Decimal Mode
4680 ;If 'tens' digit is inversed
4690 ;the number is over 200,if the
4700 ;'units' digit is inversed the
4710 ;number is between 100 and 200
4720 ;
4730 SDEC LDY #0
4740      INC Y2      ;3
4750 SD1  LDA #$FF    ;set inverse
4760      STA IV      ;flag
4770      LDA MYBUF,Y ;get byte
4780      INY
4790      STY Y1      ;save buf pos
4800      CMP #200    ;over 200?
4810      BCC DD1
4820      SEC         ;- 200
4830      SBC #200
4840      INC IV      ;set flag
4850      INC IV      ;to 1
4860      BNE DD3
4870 DD1  CMP #100    ;over 100?
4880      BCC DD3
```

```
4890      SEC
4900      SBC #100
4910      INC IV        ;set flag to 0
4920 DD3  LDX #'0       ;default '00'
4930      STX TENS
4940      STX UNITS
4950 DD5  CMP #$0A      ;over 10?
4960      BCS DD4       ;yes
4970      CLC
4980      ADC UNITS     ;add units digit
4990      STA UNITS
5000      LDY IV        ;see if any are
5010      BMI DD2       ;inversed
5020      LDA UNITS,Y   ;do inverse
5030      ORA #$80
5040      STA UNITS,Y
5050 DD2  LDY Y2        ;get screen pos
5060      LDA TENS      ;convert to
5070      JSR ASC2IC    ;screen code
5080      STA (T),Y
5090      INY
5100      LDA UNITS     ;same for
5110      JSR ASC2IC    ;units digit
5120      STA (T),Y
5130      INY           ;move right
5140      INY
5150      STY Y2
5160      CPY #27       ;done w/line?
5170      BEQ DD6
5180 DD7  LDY Y1        ;done with
5190      CPY #$80      ;sector?
5200      BNE SD1
5210      RTS
5220 DD6  JSR ADD40     ;next line
5230      LDA #3
5240      STA Y2        ;reset LMargin
5250      BNE DD7
5260 DD4  SEC           ;get tens digit
5270      SBC #$0A
5280      INC TENS
5290      BNE DD5
5300      RTS
5310 ;
5320 ;Show Data as ATASCII or
5330 ;Internal Code
5340 ;
5350 ASCINT JSR MASIN ;show type
5360      LDA GL        ;top line
5370      STA T
5380      LDA GH
5390      STA T+1
5400      LDX #0        ;data pos
5410 AIM  LDY #30       ;screen Y
5420 AIL  LDA MYBUF,X ;get byte
5430      STX X2
5440      LDX AIFLAG    ;ATASCII or
5450      BNE AIN       ;Internal
5460      JSR ASC2IC    ;convert
5470 AIN  LDX X2
5480      STA (T),Y     ;show it
5490      INX
5500      INY
5510      CPY #$26      ;end of line?
5520      BNE AIL
5530      JSR ADD40     ;next line
5540      CPX #$80      ;end of buffer?
5550      BNE AIM
5560      RTS
5570 ;
5580 ;Show it as ATASCII on screen
5590 ;
5600 ASC2IC JSR BITER
5610      ORA A2I,X
5620      LDX X1
5630      RTS
5640 ;
5650 ;Show as Internal Code on screen
5660 ;
5670 IC2ASC JSR BITER
5680      ORA I2A,X
5690      LDX X1
5700      RTS
5710 ;
5720 ;Do bit work
5730 ;
5740 BITER STX X1
5750      PHA
5760      ROL A
5770      ROL A
5780      ROL A
5790      ROL A
5800      AND #3
5810      TAX
5820      PLA
5830      AND #$9F
5840      RTS
5850 ;
5860 ;Do heading of 'ATASCII' or
5870 ;'INTERNAL' code
5880 ;
5890 MASIN LDX #7
5900      LDA AIFLAG  ;get flag
5910      BNE IN      ;no, internal
5920      LDX #15
5930 IN   LDY #7      ;copy header
5940 MA1  LDA INMS,X  ;on screen
5950      STA BORDER+30,Y
5960      DEX
5970      DEY
5980      BPL MA1
5990      RTS
6000 ;
6010 ;Display the Sector Data
6020 ;
6030 SECINF LDA SECL ;sector #
6040      STA FR0
6050      LDA SECH
6060      STA FR0+1
6070      LDX #5      ;at position 5
6080      JSR PH1     ;put it on.
6090      LDA MYBUF+127 ;# of bytes
6100      STA FR0
6110      LDX #35     ;posit. 35
6120      JSR PH2     ;put it on.
6130      LDA MYBUF+125 ;file #
6140      LSR A       ;is in the
6150      LSR A       ;6 hi bits
6160      STA FR0
6170      LDX #15     ;posit. 15
6180      JSR PH2     ;display.
6190      LDA MYBUF+126 ;sector link
6200      STA FR0     ;low
6210      LDA MYBUF+125 ;sector link
6220      AND #3      ;high
6230      STA FR0+1
6240      LDX #25     ;posit 25
6250      LDA SRCDRV  ;get source
6260      ORA #$90    ;make inverse
6270      STA DRLIN+5 ;on screen
6280      LDA DESDRV  ;dest drive
6290      ORA #$90
6300      STA DRLIN+15 ;on screen
6310      LDA DENSITY ;disk density
6320      CLC
6330      ADC #$91
6340      STA DRLIN+35 ;on screen
6350      LDY HXORST  ;input mode
6360      LDA INPMS,Y ;get letter
6370      STA DRLIN+25 ;on screen
6380 ;
6390 ;Put Hex # on Status Line
6400 ;enter with X = position
6410 ;
6420 PH1  LDY #1      ;start at dig#2
6430      BNE P_IN
6440 PH2  LDY #2      ;digit #3
6450 P_IN STY TMP1    ;save Y
6460      STX TMP2    ;save X
6470      JSR BIN2HEX ;make hex
6480      LDY TMP1    ;restore X&Y
6490      LDX TMP2
6500 P_LP LDA LBUFF,Y ;get char
6510      JSR ASC2IC  ;make ICODE
6520      ORA #$80    ;inverse it
6530      STA STATUS,X ;on screen
6540      INX
6550      INY
6560      CPY #4      ;at end?
6570      BNE P_LP    ;no.
6580      JMP ZFR     ;zap FR0
6590 ;
6600 ;Copy DSBUF to MYBUF
6610 ;
6620 TOMYBUF LDY #$7F
6630 TOM  LDA DSBUF,Y
6640      STA MYBUF,Y
6650      DEY
6660      BPL TOM
6670      RTS
6680 ;
6690 ;Tables for ATASCII and
6700 ;Internal code conversions
6710 ;
6720 I2A .BYTE $20,$40,$00,$60
6730 A2I .BYTE $40,$00,$20,$60
6740 ;
6750 ;The Command Table
6760 ;
6770 COMTAB .WORD EDITS ;block edit
6780      .WORD HX2DEC ;hex>dec
6790      .WORD DECH  ;dec>hex
6800      .WORD BFILL ;block fill
6810      .WORD EORS  ;eor sector
6820      .WORD CHNFIL ;file #
6830      .WORD SEARCHS ;search
6840      .WORD SLINKS ;link #
6850      .WORD READS ;read sec
6860      .WORD WRITES ;write sec
6870      .WORD AIFLIP ;flip chars
6880      .WORD HDFLIP ;flip #'s
6890      .WORD HELPME ;help scrn
6900      .WORD UNDO  ;undo
6910      .WORD CHNSRC ;src drive
6920      .WORD CHNDES ;dest drive
6930      .WORD KOLOR ;next hue
6940      .WORD REFRESH ;clear scrn
6950      .WORD DOWN1 ;sec=sec-1
6960      .WORD UP1   ;sec=sec+1
6970      .WORD HEXSTR ;input flip
6980      .WORD TRACES ;trace link
6990      .WORD CHNDENS ;new density
7000      .WORD LUMIN ;luminance
7010      .WORD INTRO ;menu screen
7020      .WORD REFRESH ;clear scrn
7030      .WORD DMPSCN ;prnt scrn
7040 ;
7050 ;Routines
7060 ;
7070 ;Show Sector as ATASCII (0) or
7080 ;Internal Code (1)
7090 ;
7100 AIFLIP LDA AIFLAG
7110      EOR #1
7120      STA AIFLAG
7130      JMP ASCINT
7140 ;
7150 ;Show Sector as Hex (0) or
7160 ;Decimal (1) data
7170 ;
7180 HDFLIP LDA HDF
7190      EOR #1
7200      STA HDF
7210      JMP SHWSEC
7220 ;
7230 ;Edit the sector:Set INP to
7240 ;H(hex) or S(string) input
7250 ;
7260 EDITS JSR G_NUM ;get start byte
7270      BCC EE1
7280 EDN  RTS
7290 EE1  LDA FR0+1   ;is the #($80?
7300      BNE EDN
7310      LDA FR0
7320      CMP #$80
7330      BCS EDN     ;no way.
7340      STA FBYTE   ;save it
7350      JSR ERAINP  ;clear INPLIN
7360      LDA #$A4    ;a 'D'
7370      STA INPLIN+3
7380      JSR G_HS    ;get data
7390      BCS EDN
7400      STY Y1      ;save length
7410      DEY
7420 MIC  LDA IBUF,Y  ;make the data
7430      LDX AIFLAG  ;ASC/ICODE
7440      BEQ MIO
7450      JSR ASC2IC
7460 MIO  STA IBUF,Y
7470      DEY
7480      BPL MIC
7490 EE2  LDY FBYTE   ;Y= pos
7500      LDX #0
7510 CPM  LDA IBUF,X  ;get byte
7520 EE3  STA MYBUF,Y ;store it
7530      INX
7540      INY
7550      CPY #$80    ;end of sec?
7560      BEQ ERT     ;yes.
7570      CPX Y1      ;end of data?
7580      BNE CPM     ;no.
7590 ERT  JMP SHOWALL ;show changes
7600 ;
7610 ;Decimal to Hex converter
7620 ;
7630 DECH LDA #1      ;hex mode
7640      STA IMODE
7650      LDY #5      ;5 chars max
7660      JSR INPUT
7670      BCC G_IB
7680      RTS
7690 G_IB LDA # <IBUF ;point FP
7700      STA INBUFF  ;to buffer
7710      LDA # >IBUF
7720      STA INBUFF+1
7730      LDA #0
7740      STA CIX
7750      JSR AFP     ;ASCII to FP
7760      JSR FPI     ;FP to Integer
7770      BCC U5      ;CLC->good.
7780      JSR ZFR     ;'0000' on error
7790 U5   LDA FR0+1
```

```
7800      PHA          ;save the #
7810      LDA FR0
7820      PHA
7830      LDX #1
7840      STX HXDC     ;force decimal
7850      DEX          ;posit 0
7860      JSR COPYNUM
7870      INX          ;insert =
7880      LDA #'=
7890      STA IBUF,X
7900      INX
7910      PLA          ;get # back
7920      STA FR0
7930      PLA
7940      STA FR0+1
7950      JSR COP_ALL ;copy hex
7960      JMP SHO_IT
7970 ;
7980 ;Hex to Decimal Converter
7990 ;
8000 HX2DEC JSR G_NUM ;get number
8010      BCS ANRTS
8020 ;
8030      LDX #1       ;force decimal
8040      STX HXDC
8050      DEX          ;insert hex
8060      JSR COP_ALL
8070      LDA #'=      ;insert "="
8080      STA IBUF+5
8090      LDX #6       ;posit #6
8100      JSR COPYNUM ;copy number
8110 SHO_IT LDA IBUF,X ;put the str
8120      JSR A5C2IC  ;on the
8130      STA LINE0,X ;screen.
8140      DEX
8150      BPL SHO_IT
8160 ANRTS RTS
8170 ;
8180 ;Copy all 4 Hex digits to IBUF
8190 ;
8200 COP_ALL STX TMP1
8210      JSR BIN2HEX
8220      LDX TMP1
8230      LDA #'$
8240      STA IBUF,X
8250      INX
8260      JMP ALL
8270 ;
8280 ;Block Fill
8290 ;
8300 BFILL JSR G_NUM
8310      BCC BFB
8320      RTS
8330 BFB LDA FR0
8340 BFC LDY #$7F     ;fill mybuf
8350 BFF STA MYBUF,Y ;with value
8360      DEY
8370      BPL BFF
8380 BFD JMP SHOWALL ;Show it
8390 ;
8400 ;Convert Hex to Binary value
8410 ;returns with A = value
8420 ;
8430 HEX2BIN LDA HXL ;check if
8440      JSR CHEX
8450      STX HXL
8460      LDA HXH      ;same for hi
8470      JSR CHEX
8480      TXA          ;x16
8490      ASL A
8500      ASL A
8510      ASL A
8520      ASL A
8530      ORA HXL      ;add low
8540      RTS          ;=A
8550 ;
8560 ;See if digit is in the Hex
8570 ;table
8580 ;
8590 CHEX LDX #$0F    ;16 digits
8600 HH4 CMP HXTAB,X
8610      BEQ HH3
8620      DEX
8630      BPL HH4
8640 HH3 RTS
8650 ;
8660 ;Sets flag for Hex or
8670 ;String (Character) Input
8680 ;
8690 HEXSTR LDA HXORST
8700      EOR #1
8710      STA HXORST
8720      JMP SECINF
8730 ;
8740 ;EOR the data with a byte
8750 ;
8760 EORS JSR G_NUM
8770      BCC E_GO
8780      RTS
8790 E_GO LDY #$7F    ;eor whole
8800 ERP LDA MYBUF,Y ;sector
8810      EOR FR0
8820      STA MYBUF,Y
8830      DEY
8840      BPL ERP
8850      JMP SHOWALL ;show it
8860 ;
8870 ;Show Sector Data, Characters
8880 ;and Sector information
8890 ;
8900 SHOWALL JSR SHWSEC ;data
8910      JSR A5CINT  ;characters
8920      JMP SECINF  ;inofrmation
8930 ;
8940 ;Install the HELP screen
8950 ;
8960 HELPME LDA # <HELPDL ;change
8970      STA OLDD     ;display list
8980      LDA # >HELPDL ;part way
8990      STA OLDD+1   ;down
9000      PLA
9010      PLA
9020      JMP GETCOM
9030 ;
9040 ;Restore the Sector's Data
9050 ;
9060 UNDO JSR TOMYBUF
9070      JMP SHOWALL
9080 ;
9090 ;Erase text in Window
9100 ;
9110 ERATXT LDY #39
9120      LDA #0
9130 ET1 STA LINE0,Y
9140      DEY
9150      BPL ET1
9160      RTS
9170 ;
9180 ;Change Source Drive
9190 ;
9200 CHNSRC LDX SRCDRV
9210      JSR DRV_UP
9220      STX SRCDRV
9230      JMP SECINF
9240 ;
9250 ;Change Destination Drive
9260 ;
9270 CHNDES LDX DESDRV
9280      JSR DRV_UP
9290      STX DESDRV
9300      JMP SECINF
9310 ;
9320 ;Increment Drive Number
9330 ;
9340 DRV_UP INX
9350      CPX #9
9360      BNE SHX
9370      LDX #1
9380 SHX RTS
9390 ;
9400 ;Change the File Number
9410 ;
9420 CHNFIL JSR G_NUM
9430      BCC FF1
9440 FF3 RTS
9450 FF1 LDA FR0+1
9460      BNE FF3
9470      LDA FR0
9480      CMP #64      ;0-63 allowed
9490      BCC FF4
9500      LDY #181
9510      JMP IOERR
9520 FF4 ASL A         ;high 6 bits
9530      ASL A        ;used for file
9540      STA FR0      ;number
9550      LDA MYBUF+125 ;get byte
9560      AND #3       ;mask out low
9570      ORA FR0      ;add in File #
9580      STA MYBUF+125
9590      JMP SHOWALL ;show it
9600 ;
9610 ;Get Hex or String
9620 ;
9630 G_HS LDA HXORST
9640      BNE G_ST
9650      JMP G_HEX
9660 ;
9670 G_ST JMP G_STR
9680 ;
9690 ;Search for Hex or String.
9700 ;String can be either
9710 ;ATASCII or Internal Code
9720 ;
9730 SEARCHS JSR G_HS ;get data
9740      BCC AO2
9750 AOR RTS
9760 ;
9770 AO2 CPY #0       ;just EOL?
9780      BEQ IS_OLD? ;yup, check buf
9790      STY SRFLG   ;save length
9800      LDY #44     ;copy bufr over
9810 AOCP LDA IBUF,Y ;to the save
9820      STA SRBUF,Y ;buffer for
9830      DEY         ;next time
9840      BPL AOCP
9850      BMI AORI
9860 IS_OLD? LDA SRFLG ;old data?
9870      BEQ AOR     ;no way!
9880 ;
9890 AORI LDY #0      ;convert to
9900 ALI LDA SRBUF,Y ;internal
9910      LDX AIFLAG
9920      BEQ A_ST
9930      JSR A5C2IC
9940 A_ST STA IBUF,Y
9950      INY
9960      CPY SRFLG
9970      BNE ALI
9980 ;
9990 SSTR LDA #$81    ;compute the
010000    SEC         ;last posit we
010010    SBC SRFLG   ;can look at.
010020    STA MAXLEN  ;save it.
010030    LDY #0      ;sec pntr
010040 SEL LDX #0     ;bufr pntr
010050    STY Y1      ;save it
010060 TRYF LDA IBUF,X ;get buffer
010070    CMP MYBUF,Y ;in sector?
010080    BNE GS_UP   ;no increment.
010090    INY         ;get next char
010100    INX
010110    CPX SRFLG   ;at end?
010120    BNE TRYF    ;no yet...
010130    LDA Y1      ;first byte
010140    STA FBYTE   ;save
010150    JSR SHO_POS ;sho match!
010160 ;
010170 GS_UP LDY Y1    ;get posit
010180    INY         ;increment
010190    CPY MAXLEN  ;at end?
010200    BNE SEL     ;no loop.
010210    RTS         ;so long
010220 ;
010230 ;find place on screen to
010240 ;mark the match
010250 ;
010260 SHO_POS LDA FBYTE ;byte #
010270    LSR A       ;div by 8
010280    LSR A       ;to get row
010290    LSR A
010300    TAY
010310    LDA GL,Y    ;get memory
010320    STA T       ;position
010330    LDA GH,Y
010340    STA T+1
010350    TYA
010360    ASL A
010370    ASL A
010380    ASL A       ;x8
010390    STA TMP3
010400    LDA FBYTE   ;get remainder
010410    SEC
010420    SBC TMP3    ;go right by
010430    STA TMP3    ;threes!
010440    ASL A       ;x2
010450    CLC
010460    ADC TMP3    ;x3
010470    CLC
010480    ADC #2      ;for margin
010490    TAY
010500    LDA #127    ;wedge
010510    STA (T),Y   ;show it
010520    RTS
010530 ;
010540 ;Refresh the screen to
010550 ;erase the arrows
010560 ;
010570 REFRESH JSR CLRSCN
010580    JSR DOLEFT
010590    JMP SHOWALL
010600 ;
010610 ;Change Screen Color
010620 ;
010630 KOLOR LDA COLOR2
010640    CLC
010650    ADC #$10
010660    STA COLOR2
010670    RTS
010680 ;
010690 ;Change Luminance
010700 ;
```

```
010710 LUMIN LDA COLOR2
010720   TAX
010730   AND #$F0
010740   STA CTMP
010750   TXA
010760   CLC
010770   ADC #2
010780   AND #$0F
010790   ORA CTMP
010800   STA COLOR2
010810   RTS
010820 ;
010830 ;Change Sector Link
010840 ;
010850 SLINKS JSR G_NUM
010860   BCC SL_G
010870   RTS
010880 SL_G LDA FR0  ;chk for 0
010890   ORA FR0+1
010900   BEQ LN_ER
010910   LDA FR0+1
010920   LDX DENSITY ;make sure the
010930   CMP SDHI,X  ;new link is
010940   BCC G_LNK   ;allowed in
010950   BEQ G_TLO   ;this density.
010960   BCS LN_ER
010970 G_TLO LDA FR0
010980   CMP SDLO,X
010990   BCC G_LNK
011000   BEQ G_LNK
011010   BCS LN_ER
011020 G_LNK LDA FR0 ;store it
011030   STA MYBUF+126
011040   LDA FR0+1
011050   AND #3
011060   STA FR0
011070   LDA MYBUF+125 ;mask off
011080   AND #$FC     ;file #
011090   ORA FR0+1    ;add in link
011100   STA MYBUF+125
011110   JMP SHOWALL ;show it
011120 ;
011130 LN_ER LDY #180
011140   JMP IOERR
011150 ;
011160 ;Read a Sector
011170 ;
011180 READS JSR G_NUM ;get sector
011190   BCC G_RD
011200   RTS
011210 G_RD LDA FR0  ;just EOL?
011220   ORA FR0+1
011230   BEQ DORED   ;yes, reread
011240   LDA FR0     ;copy the new
011250   STA SECL    ;# to area
011260   LDA FR0+1
011270   STA SECH
011280 ;
011290 DORED JSR READIT ;do the read
011300   BPL G.Q
011310   JSR IOERR
011320 G.Q JSR TOMYBUF ;move to mybuf
011330   JMP SHOWALL ;show it
011340 ;
011350 ;Set up Reading
011360 ;
011370 READIT LDA SRCDRV ;source drv
011380   STA DUNIT
011390   LDA # >DSBUF ;read into
011400   STA DBUFHI  ;dsbuf
011410   LDA # <DSBUF
011420   STA DBUFLO
011430   LDA #$52
011440   BNE REDWRT
011450 ;
011460 ;Set up Write
011470 ;
011480 WRITIT LDA DESDRV ;dest. drv
011490   STA DUNIT
011500   LDA # >MYBUF ;write out
011510   STA DBUFHI  ;mybuf
011520   LDA # <MYBUF
011530   STA DBUFLO
011540   LDA #$57
011550 ;
011560 ;general read,write
011570 ;
011580 REDWRT STA DCOMND ;save cmd
011590   LDA SECL    ;sector 0?
011600   ORA SECH
011610   BNE C_D     ;no.
011620   INC SECL    ;now sector 1.
011630 C_D LDX DENSITY ;test to be
011640   LDA SECH    ;sure that the
011650   CMP SDHI,X  ;sector is on
011660   BCC S_OK    ;the disk.
011670   BEQ T_LO
011680   BCS T_NO
011690 T_LO LDA SECL ;test lo.
011700   CMP SDLO,X
011710   BCC S_OK
011720 T_NO LDA SDLO,X ;move max #
011730   STA SECL    ;into sec #
011740   LDA SDHI,X
011750   STA SECH
011760 S_OK LDA SECL ;save the
011770   STA DAUX1   ;sector in SIO
011780   LDA SECH    ;work area
011790   STA DAUX2
011800   LDA #$80    ;128 bytes
011810   STA DBYTLO
011820   LDA #0
011830   STA DBYTHI
011840   JMP DSKINV  ;do SIO
011850 ;
011860 ;Format an I/O Error #
011870 ;
011880 F_ERR JSR ZFR ;zap FR0
011890   STY FR0     ;save err #
011900   LDX #1      ;force decimal
011910   STX HXDC
011920   LDX #7      ;copy error msg
011930 DS.1 LDA ERMS,X ;to buffer
011940   STA IBUF,X
011950   DEX
011960   BPL DS.1
011970   LDX #8      ;copy number in
011980   JSR COPYNUM
011990   LDA #0
012000   STA IBUF+1,X
012010   RTS
012020 ;
012030 ;Disk editor Error Routine
012040 ;
012050 IOERR JSR F_ERR
012060   JMP SHO_IT  ;show error.
012070 ;
012080 ;Read Down One Sector
012090 ;
012100 DOWN1 LDA SECH
012110   BNE DA2     ;ok
012120   LDA SECL
012130   CMP #2      ;if LT 2 no
012140   BCS DA2     ;decrease
012150   RTS
012160 DA2 DEC SECL  ;go down 1
012170   LDA SECL
012180   CMP #$FF
012190   BNE DA1
012200   DEC SECH
012210 DA1 JMP DORED ;read it
012220 ;
012230 ;Read Up One Sector
012240 ;
012250 UP1 INC SECL
012260   BNE UPA
012270   INC SECH
012280 UPA JMP DORED
012290 ;
012300 ;Trace a file
012310 ;
012320 TRACES LDA SECL ;save cur
012330   STA SSECL   ;number just
012340   LDA SECH    ;in case
012350   STA SSECH
012360   LDA MYBUF+125 ;get link
012370   AND #3      ;hi
012380   STA SECH
012390   LDA MYBUF+126 ;and low
012400   STA SECL
012410   ORA SECH
012420   BNE D_R     ;if 0 end of
012430   LDY #182
012440   JMP IOERR
012450 D_R JMP DORED ;trace
012460 ;
012470 EOFT LDA SSECL ;reset current
012480   STA SECL    ;sector number
012490   LDA SSECH
012500   STA SECH
012510   RTS
012520 ;
012530 ;Write a Sector
012540 ;
012550 WRITES JSR G_NUM ;sector to
012560   BCC WR1     ;write.
012570   RTS
012580 WR1 LDA SECL  ;save old #
012590   STA SSECL   ;for later.
012600   LDA SECH
012610   STA SSECH
012620   LDA FR0     ;sector 0?
012630   ORA FR0+1
012640   BEQ WR2     ;yes.
012650   LDA FR0     ;copy new sec#
012660   STA SECL    ;into pntr
012670   LDA FR0+1
012680   STA SECH
012690 WR2 LDA SECL  ;use old sector
012700   STA FR0     ;number.
012710   LDA SECH
012720   STA FR0+1
012730   JSR BIN2HEX ;make hex
012740   LDY #3      ;copy to the
012750 WR3 LDA LBUFF,Y ;message.
012760   STA SUREMS+11,Y
012770   DEY
012780   BPL WR3
012790   LDA DESDRV  ;put in the
012800   ORA #$30    ;destn drive
012810   STA SUREMS+20
012820   LDY #0
012830 WR4 LDA SUREMS,Y
012840   BEQ GYN
012850   JSR ASC2IC
012860   STA (INPLOC),Y
012870   INY
012880   BNE WR4
012890 GYN JSR GETKEY
012900   CMP #27
012910   BEQ WRU
012920   AND #$7F
012930   CMP #'Y
012940   BEQ WR5
012950   CMP #'N
012960   BNE GYN
012970   BEQ WRU
012980 ;
012990 WR5 JSR WRITIT
013000   BPL WRU
013010   JSR IOERR
013020 WRU JMP EOFT  ;restore
013030 ;
013040 ;Change Density
013050 ;
013060 CHNDENS LDA DENSITY
013070   EOR #1
013080   STA DENSITY
013090   JMP SECINF
013100 ;
013110 BORDER .SBYTE "   +0 +1 +2 +3"
013120   .SBYTE " +4 +5 +6 +7    A"
013130   .SBYTE "TASCII   "
013140 ;
013150 INMS .SBYTE "INTERNAL"
013160 ATMS .SBYTE "ATASCII "
013170 INPMS .SBYTE "HS"
013180 SDLO .BYTE $D0,$10
013190 SDHI .BYTE $02,$04
013200 ERMS .BYTE "Error # "
013210 ;
013220 ;Left Border
013230 ;
013240 LEFTB .SBYTE " 0 810182028303"
013250   .SBYTE "8404850586068707B"
013260 ;
013270 ;Various Messages & Prompts
013280 ;
013290 SUREMS .BYTE "Write SEC: 0000"
013300   .BYTE " to D : Y/N?",0
013310 HXTAB .BYTE "0123456789ABCDEF"
013320 CTXT .SBYTE "Edt# HJD D)H"
013330   .SBYTE "Bfil EorFil#"
013340   .SBYTE "SrchLinkRead"
013350   .SBYTE "Writ   "
```

LISTING 4: ASSEMBLY

```
0100 ;SAVE#D:DMPT3.M65
0110 ;--------------------
0120 ;
0130 ;Memory disassembler
0140 ;
0150 ; by: Bryan Schappel
0160 ;
0170 ;--------------------
0180 ;
0190 DISAMEM LDY #0  ;zero Y
0200      STY HXDC   ;in hex
0210      STY BYTES
0220      JSR GETBYT ;get opcode
0230      BPL D.1    ;error?
0240      JMP D_OVER ;yes!
0250 ;
0260 D.1 STA OP      ;save op
0270      LDY #0     ;zero Y
0280 FNDOP CMP OPCODE,Y ;is op in
0290      BEQ HAVOP  ;the opcode
0300      INY        ;table?
0310      CPY #151   ;151 yet?
```

```
0320      BNE FNDOP    ;no.
0330 ;
0340 HAVOP STY Y2      ;save Y
0350      JSR ZBUF     ;clear buffer
0360      LDA ADDR     ;copy addr
0370      STA FR0      ;to FP
0380      LDA ADDR+1
0390      STA FR0+1
0400      LDX #23      ;pos. 23
0410      JSR COPYNUM  ;copy it
0420      JSR ZFR      ;clear FP
0430      LDA OP       ;get op
0440      STA FR0      ;in FP
0450      LDX #28      ;pos 28
0460      JSR COPYNUM  ;copy it
0470      LDA #';
0480      STA IBUF+22
0490      LDY Y2       ;get Y
0500      CPY #151     ;legal instr?
0510      BNE N151     ;yes.
0520 ;
0530      LDY #7       ;copy 8 chars
0540 CP.B LDA POINTB,Y ;get char
0550      STA IBUF,Y   ;put in buffer
0560      DEY          ;decrease index
0570      BPL CP.B     ;and loop
0580      JSR ZFR      ;zap FR0
0590      LDY OP       ;get the byte
0600      STY FR0      ;put in FR0
0610      JSR BIN2HEX  ;make it hex
0620      LDA #1       ;force 1 byte
0630      STA NBYTE
0640      LDX #8       ;posit 8
0650      JSR SOME     ;copy number
0660      JMP AEOL1    ;and add EOL.
0670 ;
0680 N151 LDX OFFSET,Y ;MNE offset
0690      LDY #0       ;copy out the
0700 ADMNE LDA MNE,X   ;Mnemonic and
0710      STA IBUF+1,Y ;put in
0720      INX          ;buffer.
0730      INY
0740      CPY #3       ;3 chars?
0750      BNE ADMNE    ;no.
0760      STA NBYTE
0770      LDY Y2       ;restore Y
0780      LDA ITYPE,Y  ;get Itype
0790      TAY          ;transfer to Y
0800      LDA NBTAB,Y  ;get # bytes for
0810      STA NBYTE    ;this instr.
0820      TYA          ;restore Acc.
0830      ASL A        ;times 2
0840      TAY          ;in Y
0850      LDA DCOMT,Y  ;look up the
0860      STA DJSR+1   ;routine addr
0870      LDA DCOMT+1,Y ;an put in
0880      STA DJSR+2   ;JSR
0890      LDX NBYTE    ;get # bytes
0900      DEX          ;sub 1.
0910      BEQ DJSR     ;if 0, skip!
0920      JSR GETBYT   ;get next one
0930      BMI D_OVER   ;oops, error.
0940      STA DTEMP    ;save byte
0950      LDX NBYTE    ;get count again
0960      DEX          ;sub 2.
0970      DEX
0980      BEQ DJSR     ;if 0, skip
0990      JSR GETBYT   ;get 3rd byte
1000      BMI D_OVER   ;oops, error.
1010      STA DTEMP+1  ;save this byte
1020 ;
1030 DJSR JSR $FFFF    ;go command!
1040 AEOL1 LDY #39     ;now insert
1050 AEOL3 LDA IBUF,Y  ;the EOL
1060      CMP #32      ;character
1070      BNE AEOL2
1080      DEY
1090      BPL AEOL3
1100 AEOL2 LDA #EOL
1110      STA IBUF+1,Y
1120      LDA #0
1130      STA IBUF+2,Y
1140      LDA ADDR     ;now add the
1150      CLC          ;number of
1160      ADC NBYTE    ;byte to the
1170      STA ADDR     ;adr pointer
1180      BCC AE1      ;to get next
1190      INC ADDR+1   ;instr addr.
1200 AE1 RTS           ;bye.
1210 ;
1220 ;Disassembly Over
1230 ;
1240 D_OVER JSR ANY_LEFT ;send extra.
1250      INC SFLAG    ;set swap flag
1260      JSR SEND_OUT ;write buffer
1270      LDX #$10     ;slam IOCB #1
1280      JSR CLOSE
1290      LDX #$20     ;slam IOCB #2
1300      JSR CLOSE
1310       PRINT  ALL.DONE ;done mess.
1320      JSR GET_RET ;get a return
1330      JMP INTRO    ;goto intro.
1340 ;
1350 ;Immediate Mode
1360 ;
1370 IMMED JSR LOBYTE ;insert LSB
1380      LDA #'#      ;a '#'
1390      STA IBUF+5   ;in buffer
1400      JSR ZFR      ;zap FR0
1410      LDA DTEMP    ;get byte
1420      STA FR0      ;into FR0
1430      LDX #6       ;and add to the
1440      JMP COPYNUM  ;output.
1450 ;
1460 NUMBR LDA #0      ;zero MSB
1470      STA DTEMP+1
1480 NUMBR2 LDA DTEMP ;put in FR0
1490      STA FR0
1500      LDA DTEMP+1
1510      STA FR0+1
1520 NUMBR3 LDX #6     ;# in pos. 6
1530      LDA # <LAB   ;point to the
1540      STA INDR     ;label table
1550      LDA # >LAB
1560      STA INDR+1
1570      LDA LABELS?  ;want labels?
1580      BNE GO_OUT   ;nope.
1590 ;
1600 ;This is the Location Label
1610 ;search algorithm.  The label
1620 ;data are stored as follows:
1630 ;
1640 ;hi nybble-byte 1 loc length
1650 ;lo nybble-byte 1 label length
1660 ;location hi,location lo, and
1670 ;the label name.
1680 ;
1690 LAB_SRCH LDY #0 ;zero index
1700      LDA (INDR),Y ;get a byte.
1710      BNE N_FF     ;<>0 then cont.
1720 GO_OUT JMP COPYNUM ;just number
1730 ;
1740 N_FF PHA          ;save byte
1750      AND #$0F     ;get label len
1760      STA TMP1     ;save it.
1770      PLA          ;get byte back
1780      LSR A        ;shift it down
1790      LSR A
1800      LSR A
1810      LSR A
1820      STA ADD_ON   ;save it
1830      INC ADD_ON   ;add one.
1840      INY          ;move up 2
1850      INY          ;bytes.
1860      LDA (INDR),Y ;get addr lo
1870      STA LOC_S    ;save it.
1880      CLC          ;add on location
1890      ADC ADD_ON   ;length & save.
1900      STA LOC_E
1910      DEY          ;down index
1920      LDA (INDR),Y ;get addr hi
1930      STA LOC_S+1  ;save
1940      ADC #0       ;add carry
1950      STA LOC_E+1  ;and store.
1960 ;
1970      LDA DTEMP+1  ;instr hi
1980      CMP LOC_S+1  ;equal to lochi?
1990      BNE N.TST    ;no.
2000      LDA DTEMP    ;test lo
2010      CMP LOC_S
2020      BEQ D_CP     ;exact match!
2030 ;
2040 N.TST LDA DTEMP+1 ;now we see if
2050      CMP LOC_E+1  ;the instr addr
2060      BEQ TST.L    ;is in the range
2070      BCS GO_UP    ;of the location
2080 TST.L LDA DTEMP
2090      CMP LOC_E
2100      BCS GO_UP
2110 ;
2120      LDA DTEMP+1  ;test hi
2130      CMP LOC_S+1
2140      BCC GO_OUT   ;not in range
2150      LDA DTEMP    ;test lo
2160      CMP LOC_S
2170      BCC GO_OUT   ;not in range
2180 D_CP SEC          ;calc actual
2190      SBC LOC_S    ;offset for the
2200      STA ADD_ON   ;+x add on.
2210 ;
2220      DEX          ;posit 5
2230      LDY #2       ;start at pos 2
2240 C_CP INY          ;up Y
2250      INX          ;up X
2260      LDA (INDR),Y ;get label name
2270      STA IBUF,X   ;from the table
2280      BPL C_CP     ;and insert into
2290      AND #$7F     ;disa line.
2300      STA IBUF,X
2310      LDA ADD_ON   ;exact match?
2320      BNE PLUS     ;no.
2330      RTS          ;so long.
2340 ;
2350 PLUS LDA #'+      ;put in a '+'
2360      STA IBUF+1,X
2370      INX
2380      INX
2390      JSR ZFR      ;zap FR0
2400      LDA ADD_ON   ;make offset to
2410      STA FR0      ;a decimal #
2420      INC HXDC
2430      JSR COPYNUM  ;insert number
2440      DEC HXDC     ;force hex
2450      RTS          ;and leave.
2460 ;
2470 GO_UP LDA INDR    ;get the label
2480      CLC          ;pointer and
2490      ADC TMP1     ;add on the len
2500      STA INDR     ;of the label so
2510      LDA INDR+1   ;we can search
2520      ADC #0       ;more.
2530      STA INDR+1
2540      JMP LAB_SRCH ;loop!
2550 ;
2560 ;Zero page
2570 ;
2580 ZPAG JSR LOBYTE ;do LSB
2590      JMP NUMBR    ;add arg
2600 ;
2610 ;Zero page,X
2620 ;
2630 ZPAGX JSR LOBYTE ;LSB
2640      JSR NUMBR    ;add arg
2650 ADDX2 LDY #1      ;just ,X
2660 ADDX INX          ;up X
2670 ADDX3 LDA COMMAX,Y ;copy the
2680      STA IBUF+1,X ;,X into
2690      DEX          ;the buffer
2700      DEY
2710      BPL ADDX3
2720      RTS          ;done.
2730 ;
2740 ;Absolute
2750 ;
2760 ABSOL JSR LOBYTE ;LSB
2770      JSR HIBYTE   ;MSB
2780      JMP NUMBR2   ;add arg
2790 ;
2800 ;Absolute ,X
2810 ;
2820 ABSOLX JSR ABSOL ;set up absol
2830      JMP ADDX2    ;copy in ,X
2840 ;
2850 ;Absolute ,Y
2860 ;
2870 ABSOLY JSR ABSOL ;do absol
2880 ACY LDA #$2C      ;a comma
2890      STA IBUF+1,X ;in buf
2900      LDA #'Y      ;a Y
2910      STA IBUF+2,X ;in buf
2920      RTS          ;done
2930 ;
2940 ;Indirect X
2950 ;
2960 INDX LDA #0       ;no MSB
2970      STA DTEMP+1
2980      JSR LOBYTE   ;do LSB
2990      JSR OPENP    ;add open '('
3000      INX          ;up X
3010      LDY #2       ;3 bytes
3020      BNE ADDX     ;copy ',X)'
3030 ;
3040 ;Indirect Y
3050 ;
3060 INDY LDA #0       ;no MSB
3070      STA DTEMP+1
3080      JSR LOBYTE   ;do LSB
3090      JSR OPENP    ;left (
3100      LDY #2       ;copy in the
3110 ADDY LDA COMMAY,Y ;'),Y'
3120      STA IBUF+1,X
3130      INX
3140      DEY
3150      BPL ADDY
3160      RTS          ;done
3170 ;
3180 ;Accumulator Mode
3190 ;
3200 ACCUM LDA #'A     ;an 'A'
3210      STA IBUF+5   ;in buffer
3220      RTS          ;done
```

```
3230 ;
3240 ;Branch instructions
3250 ;
3260 RELA JSR LOBYTE ;put LSB
3270     INC NBYTE   ;make 3 byte
3280     JSR ZFR     ;zero FR0
3290     LDA DTEMP   ;get operand
3300     BPL NOBACK  ;not inverse
3310     AND #$7F    ;mask inverse
3320     EOR #$7F    ;flip it
3330     CLC         ;add a 1
3340     ADC #1
3350     STA DIN5+1  ;save it
3360     JMP TOFP    ;done
3370 NOBACK STA DIN5+1
3380 TOFP LDA DTEMP  ;operand
3390     BPL FORWARD ;bran forward
3400     LDA ADDR    ;now subtract
3410     SEC         ;from addr
3420     SBC DIN5+1  ;to find out
3430     STA FR0     ;where this
3440     LDA ADDR+1  ;branch goes.
3450     SBC #0
3460     STA FR0+1
3470     JMP BACK    ;done.
3480 FORWARD CLC     ;add # to
3490     ADC ADDR    ;addr to calc
3500     STA FR0     ;forward branch
3510     LDA ADDR+1
3520     ADC #0
3530     STA FR0+1
3540 BACK LDA #2     ;now add a 2
3550     CLC         ;to skip the
3560     ADC FR0     ;instruction
3570     STA DTEMP   ;address
3580     STA FR0
3590     LDA FR0+1
3600     ADC #0
3610     STA DTEMP+1
3620     STA FR0+1
3630     JSR NUMBR3  ;add number
3640     DEC NBYTE   ;make 2 bytes
3650 ;
3660 ;Implied Mode
3670 ;
3680 IMP RTS         ;do nothing
3690 ;
3700 ;Indirect instructions
3710 ;
3720 INDI JSR LOBYTE ;copy in LSB
3730     JSR HIBYTE  ;and MSB
3740     JSR OPENP   ;do open (
3750     LDA #')     ;and add
3760     STA IBUF+1,X ;closing one
3770     RTS         ;leave.
3780 ;
3790 ;Zero page ,Y
3800 ;
3810 ZPAGY JSR LOBYTE ;do LSB
3820     JSR NUMBR   ;add arg
3830     JMP ACY     ;copy ,Y
3840 ;
3850 ;Insert LSB
3860 ;
3870 LOBYTE JSR ZFR  ;zero FR0
3880     LDA DTEMP   ;do lobyte
3890     STA FR0
3900     LDX #31     ;at pos. 10
3910     JMP COPYNUM ;do it!
3920 ;
3930 ;Insert MSB
3940 ;
3950 HIBYTE JSR ZFR  ;no FR0
3960     LDA DTEMP+1 ;hibyte
3970     STA FR0     ;in FP
3980     LDX #34     ;at pos. 14
3990     JMP COPYNUM ;copy it.
4000 ;
4010 ;Add Open (
4020 ;
4030 OPENP LDA #'(   ;open (
4040     STA IBUF+5  ;in buffer
4050     JMP NUMBR2  ;add number
4060 ;
4070 ;Copy number to string
4080 ;
4090 COPYNUM DEX
4100     STX CNTLO   ;save X
4110     LDA HXDC    ;hex/dec?
4120     BEQ INHEX   ;have Hex.
4130     JSR IFP     ;to FP
4140     JSR FASC    ;to ASC
4150     LDX CNTLO   ;get index
4160     LDY #$FF    ;and copy
4170 CNLOOP INY      ;the number
4180     INX
4190     LDA (INBUFF),Y
4200     STA IBUF,X
4210     BPL CNLOOP
4220     AND #$7F    ;mask bit 7
4230     STA IBUF,X
4240     RTS         ;leave
4250 ;
4260 INHEX JSR BIN2HEX ;make hex
4270     LDX CNTLO   ;get pos
4280     INX         ;up it.
4290     CPX #23
4300     BEQ ALL     ;if 25 do all
4310     CPX #21     ;is X => 30?
4320     BCC ARG     ;yes.
4330 SOME LDA LBUFF+2 ;no. just copy
4340     STA IBUF,X  ;the LSB part
4350     INX         ;of the Hex
4360     LDA LBUFF+3 ;number.
4370     STA IBUF,X
4380     RTS         ;bye.
4390 ARG LDA #'$     ;install a '$'
4400     STA IBUF,X
4410     INX         ;up X
4420     LDA NBYTE   ;Get # of bytes
4430     CMP #2      ;is it 2?
4440     BEQ SOME    ;no.
4450 ALL LDY #0      ;no. copy it
4460 ALOOP LDA LBUFF,Y ;all to buf
4470     STA IBUF,X
4480     INX
4490     INY
4500     CPY #4      ;done yet?
4510     BNE ALOOP   ;nope.
4520     DEX         ;dec X
4530     RTS         ;so long.
4540 ;
4550 ;Binary to hexadecimal
4560 ;
4570 BIN2HEX LDY #0  ;buf indx=0
4580     LDX #1      ;do 2 nums
4590 NXTBYT STX X1   ;save it
4600     LDA #240    ;get HI bits
4610     AND FR0,X
4620     LSR A       ;divide by 16
4630     LSR A
4640     LSR A
4650     LSR A
4660     TAX         ;to index
4670     LDA HXTAB,X ;get char
4680     STA LBUFF,Y ;and save
4690     INY         ;up buf index
4700     LDX X1      ;get X back
4710     LDA #15     ;now LO bits
4720     AND FR0,X
4730     TAX
4740     LDA HXTAB,X ;get char and
4750     STA LBUFF,Y ;put in buffer
4760     INY         ;up buf indx
4770     LDX X1      ;get X
4780     DEX         ;down by one
4790     BPL NXTBYT  ;done?
4800     RTS         ;yes. bye.
4810 ;
4820 ;Fill buffer with blanks
4830 ;
4840 ZBUF LDA #32    ;space
4850     LDX #39     ;do 40 spaces
4860 ZLOOP STA IBUF,X ;in buf
4870     DEX         ;down count
4880     BPL ZLOOP   ;go again
4890     RTS         ;bye.
4900 ;
4910 ;Zero FR0
4920 ;
4930 ZFR LDA #0
4940     STA FR0
4950     STA FR0+1
4960     RTS
4970 ;
4980 ;Open Input File
4990 ;
5000 OPENIN LDX #$10 ;slam IOCB #1
5010     JSR CLOSE
5020     LDA TYPE    ;get disa type
5030     BEQ O_FILE  ;if 0, then file
5040     JSR UP.READ ;read 1st sector
5050     PHP         ;save status
5060     LDA #$80    ;default size
5070     STA SECBYTES
5080     LDA S.SIZE  ;get actual
5090     CMP #$80    ;is 128 bytes?
5100     BEQ O.SO    ;yes.
5110     LDA DSBUF+127 ;get data byte
5120     STA SECBYTES ;count.
5130 O.SO PLP        ;pull status
5140     BMI INERR   ;oops, error.
5150     RTS         ;so long.
5160 O_FILE LDX #$10 ;IOCB #1
5170     LDA #3      ;open command
5180     STA ICCOM,X
5190     LDA # <SRCFILE ;point to the
5200     STA ICBAL,X ;source file-
5210     LDA # >SRCFILE ;name.
5220     STA ICBAH,X
5230     LDA #4      ;read only.
5240     STA AUX1,X
5250     LDA #0
5260     STA AUX2,X
5270     STA BYTCNT  ;zero byte cnt
5280     STA BYTCNT+1
5290     JSR CIOV    ;open it!
5300     BMI INERR   ;oops.
5310     JSR GET.TWO ;get file header
5320     BMI INERR   ;empty file!
5330     JSR CHKFF   ;is it binary?
5340     BNE CIO.B   ;no!
5350     LDY #1      ;signal good
5360     RTS
5370 CIO.B LDY #152  ;not binary file
5380 ;
5390 ;Input error
5400 ;
5410 INERR CPY #136  ;EOF?
5420     BNE PI      ;no, show error
5430     LDA #0      ;kill instr
5440     STA BYTES
5450     JMP D_OVER  ;and exit.
5460 PI  JSR F_ERR
5470      PRINT  IN_P ;print INPUT
5480      PRINT  IBUF
5490      PRINT  ALL.DONE+10
5500     JSR GET_RET
5510     JMP D_OVER
5520 ;
5530 ;Open Output file
5540 ;
5550 OPENOUT LDX #$20 ;close #2
5560     JSR CLOSE
5570     LDX #$20
5580     LDA #3      ;open command
5590     STA ICCOM,X
5600     LDA # <DSTFILE ;point to
5610     STA ICBAL,X ;the dest'n
5620     LDA # >DSTFILE ;filename.
5630     STA ICBAH,X
5640     LDA #8      ;write only
5650     STA AUX1,X
5660     LDA #0      ;no lines.
5670     STA LINES
5680     STA AUX2,X
5690     LDA # <BIGBUF ;point to the
5700     STA BUFPNT  ;start of the
5710     LDA # >BIGBUF ;buffer.
5720     STA BUFPNT+1
5730     LDA DSTFILE ;is destn to
5740     CMP #'E     ;the screen?
5750     BNE C.CIO   ;no, call cio
5760     LDY #1      ;show good.
5770     RTS
5780 C.CIO JMP CIOV  ;open it!
5790 ;
5800 ;Close IOCB
5810 ;
5820 CLOSE LDA #12   ;close command
5830     STA ICCOM,X
5840     JMP CIOV    ;slam!
5850 ;
5860 ;CIO get byte routine
5870 ;
5880 CIOGET LDX #$10 ;on ch #1
5890     LDA #7      ;get bytes
5900     STA ICCOM,X
5910     LDA #0      ;no length says
5920     STA ICBLL,X ;put byte in the
5930     STA ICBLH,X ;accumulator.
5940     JMP CIOV    ;get it!
5950 ;
5960 ;Get byte routine
5970 ;
5980 GETBYT LDA TYPE ;get type.
5990     BEQ G_FILE  ;if 0, from file
6000     LDX GETPTR  ;get index
6010     CPX SECBYTES ;at end?
6020     BEQ NX.SEC  ;yes.
6030     LDA DSBUF,X ;get byte.
6040     INC GETPTR  ;bump pointer.
6050     LDY #1      ;signal good.
6060     RTS         ;bye.
6070 ;
6080 NX.SEC LDA #1   ;start on byte 1
6090     STA GETPTR
6100     LDA S.SIZE  ;trace links?
6110     CMP #125
```

continued on page 63

Character Set Display Utility

continued from page 9

The trouble is that most character set editors are great for editing sets but not for looking at them and comparing them. At most, you can usually only look at one or two sets at a time and by the time you load in the new set, you have forgotten just what the last one looked like.

The *Character Set Display Utility* (CSDU) allows you to compare up to seven different character sets on the screen at one time. And it's easy to go back and reselect more fonts. By using this utility, you will be able to clean up your font disks quickly—and easily select which font suits your needs for a particular program.

Typing It In

CSDU is written in Action!. To use the program, carefully type in Listing 1 using the *M/L Editor*, found elsewhere in this issue.

Listings 2 and 3 are the Action! source code for CSDU. If you have Action!, you may type these listings instead, using *D:CHECK in Action!* (ANALOG, December '88) to check your work. To create a copy of CSDU from the Action! listings, load Listing 3 into the Action! editor and compile it, making sure that Listing 2 (you should name it CSET2.ACT) is on the disk in your drive.

By using this utility, you will be able to clean up your font disks quickly—and easily select which font suits your needs.

Using the Program

At the title screen, insert your font disk into the drive and press Start. The fonts must be the standard 1,024-byte, nine-sector redefined character set fonts, and the extender must be .FNT, as in D:FILENAME.FNT.

Next, you will be presented with the font-selection screen. Here you may use either the joystick or the console keys to select fonts. The font-selection screen will display up to 80 font names. Pressing Select or Option, or moving the joystick, changes the highlighted filename. Pressing Start or the joystick button will toggle a marker next to the font. You may select up to seven fonts. You will not be allowed to mark more than seven or mark an empty space.

When you have marked from one to seven font names, pressing the space bar will take you to the font display screen. If you press Return or press the space bar without selecting any font names, you will be prompted to insert a new font disk.

When the font display screen is displayed, you will see seven multicolored windows. Each font selected will have a window containing the font name and the full set of alphanumeric characters—that is, the upper and lower case letters and the number characters. Pressing the space bar will allow you to select more fonts to view. If you wish to view fonts on a new disk, insert the disk before pressing the space bar. Pressing Start or the joystick trigger will end the program.

There are a couple cases where you might see something funny on the screen. If you attempt to load in a nonstandard Atari character set font (for instance, a *Daisy Dot* font file), the display window for that font will contain garbage.

Using Action! as a Macro Assembler

In a review of mine published in ANALOG #66, the biographical blurb at the end of the article mentioned my hunt for gainful employment. Since that time, I am happy to report, I've found employment as a C

programmer. Programming in C on a daily basis has had two effects on me. The first of these was a renewed respect and enthusiasm for the Action! programming language on the Atari.

Second, now as a "professional" (yeah, right) programmer, I decided to learn everything there was to know about programming my Atari. I dug out every *Boot Camp* article, starting from the first one in ANALOG #13. Soon I had worked my way up to Karl Wieger's excellent series of articles on macro assemblers and was ready to put my MAC/65 cartridge to work.

By that time, however, my work with C led me to realize one great advantage of the Action! language. Like C, Action! is both a high-level language and a low-level language at the same time.

It is a high-level language when I want to take advantage of built-in Print or Locate routines or use flexible IF-THEN-ELSE, DO-WHILE, REPEAT-UNTIL constructs. It is a low-level language when I want to directly manipulate hardware addresses, set or flip individual bytes, or set up custom screens or interrupts. This gives a programmer a lot of power to do fancy things very easily without wading through tons of assembly code.

For instance, the beginning of the Action! listing looks a lot like the equates at the beginning of an assembly program. Most of this I lifted right out of the *Boot Camp* series. The DEFINEs are used for the actual seven bytes of machine language used. About the only thing Action! can't do is manipulate the Atari status registers like the X, Y registers and the accumulator.

The global BYTE variables are used for one-byte memory locations and hardware registers. The global CARD variables are used for addresses. The nice thing about this is that Action! automatically takes care of storing and reading these numbers in low-byte, high-byte format for us. No bothering with LDA #START&255, LDA #START/256, etc.

The global arrays are used as storage areas for filenames, the custom display list, the character set pointers, and the screen color pointers. While Action! allows you to store variables wherever you like, you have to take some steps to make sure your variable area doesn't overwrite a variable area used by the language itself. By setting up a 512-byte array called DLSPACE, we can put our display list, DL, in there and be assured it does not cross a page boundary.

Similarly, we have to step back MEMTOP far enough to hold the seven character sets and the screen memory.

On the other hand, all the routines that read the joystick and the keyboard, select the files, read the filenames off the screen, and invert the selected filenames are not any that I would want to implement in assembly language, macros or not. Action! is more than fast enough to handle any of these routines. In fact, I had to slow things down for the font selection screen to keep pace with human reflexes.

If you want the power of machine language without having to reinvent the wheel, give Action! a try.

The nice thing about this is that Action! automatically takes care of storing and reading these numbers in low-byte, high-byte format for us.

A programmer for Bell Communications Research, Dave still finds enough time to use his 8-bit Atari. He enjoys programming in Action! and MAC/65, and is interested in computer simulations of all types.

Character Set Display Utility

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,254,52,49,72,93,226,
84,168,0,167,4,0,226,84,5097
1010 DATA 161,0,167,4,21,80,0,0,56,0,8
0,114,105,110,116,73,838
1020 DATA 69,61,42,40,73,78,84,32,105,
41,226,84,133,0,167,4,2321
1030 DATA 25,91,36,50,48,32,95,83,72,7
0,84,82,32,36,52,67,9334
1040 DATA 32,80,114,105,110,116,73,68,
69,93,226,84,101,0,0,0,653
1050 DATA 20,0,84,94,0,69,79,76,32,36,
52,67,32,95,68,79,9327
1060 DATA 80,82,50,72,0,2,90,53,7,0,53
,0,226,124,76,140,1508
1070 DATA 108,44,92,156,101,53,112,112
,112,66,0,0,2,130,14,2,8425
1080 DATA 2,130,14,2,2,130,14,2,2,130,
14,2,2,130,14,2,5880
1090 DATA 2,130,14,2,2,130,14,2,14,2,6
5,0,0,110,53,133,7630
1100 DATA 160,134,161,132,162,24,104,1
33,132,105,3,168,104,133,133,105,6461
1110 DATA 0,72,152,72,160,1,177,132,13
3,130,200,177,132,133,131,200,663
1120 DATA 177,132,168,185,160,0,145,13
0,136,16,248,96,133,192,134,193,439
1130 DATA 56,169,0,229,192,72,169,0,22
9,193,170,104,96,134,194,224,1742
1140 DATA 0,16,3,32,196,53,133,130,134
,131,165,133,69,194,133,194,9217
1150 DATA 166,133,16,9,165,132,32,196,
53,133,132,134,133,96,165,134,7634
1160 DATA 166,135,164,194,16,3,76,196,
53,96,32,213,53,169,0,133,4590
1170 DATA 134,133,135,165,130,208,4,16
5,131,240,34,165,132,208,4,165,9142
1180 DATA 133,240,26,70,133,102,132,14
4,13,24,165,134,101,130,133,134,6556
1190 DATA 165,135,101,131,133,135,6,13
0,38,131,56,176,214,76,246,53,7773
1200 DATA 164,132,240,10,134,134,70,13
4,106,136,208,250,166,134,96,164,1124
1210 DATA 132,240,10,134,134,10,38,134
,136,208,250,166,134,96,169,112,9915
1220 DATA 170,169,12,157,66,3,32,86,22
8,138,56,233,16,208,241,108,8959
1230 DATA 10,0,86,54,0,32,86,228,16,26
,192,136,240,6,104,104,4908
1240 DATA 152,108,106,54,72,138,72,74,
74,74,74,170,169,1,157,192,6324
1250 DATA 5,104,170,104,96,201,8,144,7
,104,104,169,134,108,106,54,5266
1260 DATA 10,10,10,10,170,96,24,105,1,
157,68,3,152,105,0,157,2115
1270 DATA 69,3,96,72,138,168,104,170,1
73,108,54,96,72,169,0,133,5520
1280 DATA 165,104,96,72,169,1,133,165,
104,96,134,163,132,164,32,141,7405
1290 DATA 54,169,0,157,73,3,168,177,16
3,240,26,157,72,3,24,165,5300
1300 DATA 163,105,1,157,68,3,165,164,1
05,0,157,69,3,169,11,157,3711
1310 DATA 66,3,32,109,54,165,165,240,1
8,169,0,157,72,3,157,73,4486
1320 DATA 3,169,11,157,66,3,169,155,32
,109,54,96,32,180,54,76,3179
1330 DATA 194,54,32,171,54,76,4,55,32,
187,54,76,194,54,32,171,3764
1340 DATA 54,76,16,55,134,160,32,141,5
4,169,0,157,72,3,157,73,3357
1350 DATA 3,169,11,157,66,3,165,160,32
,109,54,96,170,173,108,54,5375
1360 DATA 76,28,55,169,155,76,52,55,13
4,164,132,165,32,141,54,24,4230
1370 DATA 165,164,105,1,157,68,3,165,1
65,105,0,157,69,3,165,163,5157
1380 DATA 240,24,157,72,3,169,0,157,73
,3,169,5,157,66,3,32,840
1390 DATA 109,54,189,72,3,240,3,56,233
,1,160,0,145,164,165,166,7565
1400 DATA 240,7,165,164,166,165,76,64,
55,96,72,169,0,133,166,104,5960
1410 DATA 76,64,55,32,141,54,169,0,157
,72,3,157,73,3,169,7,1807
1420 DATA 157,66,3,32,109,54,133,160,9
6,134,161,32,141,54,165,161,6925
1430 DATA 32,158,54,165,163,157,74,3,1
65,164,157,75,3,169,3,157,5613
1440 DATA 66,3,32,109,54,138,74,74,74,
74,170,169,0,157,192,5,4714
1450 DATA 96,32,141,54,169,12,157,66,3
,32,109,54,96,72,162,96,3209
1460 DATA 169,12,157,66,3,32,86,228,10
4,157,75,3,73,16,9,12,9888
1470 DATA 157,74,3,169,83,133,160,169,
58,133,161,169,160,157,68,3,7142
1480 DATA 169,0,157,69,3,169,3,157,66,
3,32,109,54,96,133,85,2387
1490 DATA 134,86,132,84,96,32,6,56,169
,6,174,253,2,76,28,55,2611
1500 DATA 32,6,56,169,6,76,139,55,133,
160,134,161,132,162,160,0,6874
1510 DATA 165,162,208,4,165,163,240,22
,165,164,145,160,200,208,2,230,2160
1520 DATA 161,198,162,165,162,201,255,
208,231,198,163,56,176,226,96,72,3256
1530 DATA 169,0,133,164,104,76,32,56,1
33,160,134,161,132,162,160,0,6989
1540 DATA 165,164,208,4,165,165,240,24
,177,162,145,160,200,208,4,230,2340
1550 DATA 161,230,163,198,164,165,164,
201,255,208,229,198,165,56,176,224,644
1
1560 DATA 96,76,124,56,72,169,224,141,
9,212,104,64,96,50,70,76,4801
1570 DATA 138,56,169,192,141,14,212,17
3,49,2,141,134,56,173,48,2,4274
1580 DATA 141,133,56,24,173,133,56,105
,14,133,174,173,134,56,105,0,4693
1590 DATA 133,175,24,160,0,177,174,105
,128,145,174,140,47,2,173,123,7303
1600 DATA 56,141,1,2,173,122,56,141,0,
2,169,34,141,47,2,96,1410
1610 DATA 76,203,56,169,2,32,213,55,32
,135,56,160,0,140,198,2,4205
1620 DATA 200,140,240,2,136,132,82,56,
173,229,2,233,0,133,174,173,9227
1630 DATA 230,2,233,4,133,175,165,174,
41,0,141,10,53,165,175,41,5161
1640 DATA 252,141,11,53,169,8,133,132,
173,11,53,170,173,10,53,32,3285
1650 DATA 56,54,141,244,2,173,11,53,14
1,230,2,173,10,53,141,229,7080
1660 DATA 2,169,226,133,163,169,2,133,
165,169,0,133,164,160,0,174,8044
1670 DATA 11,53,173,10,53,32,80,56,169
,224,133,163,169,0,133,165,7823
1680 DATA 169,8,133,164,160,0,174,11,5
3,173,10,53,32,80,56,24,739
1690 DATA 173,10,53,105,8,133,160,173,
11,53,105,0,133,161,169,225,7706
1700 DATA 133,163,169,0,133,165,169,8,
```

```
133,164,160,24,166,161,165,160,9900
1710 DATA 32,80,56,24,173,10,53,105,40
,133,160,173,11,53,105,0,2288
1720 DATA 133,161,169,225,133,163,169,
0,133,165,169,8,133,164,160,32,8147
1730 DATA 166,161,165,160,32,80,56,24,
173,10,53,105,24,133,160,173,5419
1740 DATA 11,53,105,0,133,161,169,225,
133,163,169,0,133,165,169,8,8174
1750 DATA 133,164,160,152,166,161,165,
160,32,80,56,24,173,10,53,105,4386
1760 DATA 32,133,160,173,11,53,105,0,1
33,161,169,225,133,163,169,0,8250
1770 DATA 133,165,169,8,133,164,160,16
8,166,161,165,160,32,80,56,169,8804
1780 DATA 3,133,84,169,5,133,85,76,236
,57,9,33,72,65,82,65,2308
1790 DATA 67,84,69,82,160,57,162,226,1
69,6,32,16,55,76,2,58,1506
1800 DATA 9,32,32,32,32,32,32,35,69,84
,160,57,162,248,169,6,5067
1810 DATA 32,16,55,76,29,58,14,32,32,3
2,32,32,32,32,37,73,7121
1820 DATA 83,80,76,65,89,160,58,162,14
,169,6,32,16,55,76,57,954
1830 DATA 58,15,32,32,32,32,32,32,32,3
2,36,84,73,76,73,84,9438
1840 DATA 89,160,58,162,41,169,6,32,16
,55,160,0,140,144,2,140,3148
1850 DATA 146,2,140,145,2,76,109,58,28
,32,32,32,32,32,32,32,7857
1860 DATA 32,32,32,32,32,98,121,32,68,97,
118,101,32,65,114,108,105,3453
1870 DATA 110,103,116,111,110,162,58,1
69,80,32,22,55,76,154,58,34,2758
1880 DATA 32,32,32,32,32,40,99,41,32,4
9,57,56,57,32,102,111,193
1890 DATA 114,32,65,78,65,76,79,71,32,
67,111,109,112,117,116,105,4478
1900 DATA 110,103,162,58,169,119,32,22
,55,76,165,58,0,162,58,169,4501
1910 DATA 164,32,22,55,76,213,58,37,32
,32,73,110,115,101,114,116,3990
1920 DATA 32,68,105,115,107,32,119,105
,116,104,32,70,111,110,116,115,5102
1930 DATA 44,32,80,114,101,115,115,32,
211,212,193,210,212,162,58,169,2250
1940 DATA 175,32,10,55,173,31,208,73,6
,240,8,173,132,2,240,3,5530
1950 DATA 76,220,58,96,88,0,0,166,76,1
84,76,245,58,32,151,53,5407
1960 DATA 236,58,2,56,173,236,58,233,1
,133,174,169,0,133,133,169,8935
1970 DATA 8,133,132,165,174,162,0,32,2
,54,141,239,58,173,239,58,8064
1980 DATA 141,240,58,24,173,239,58,105
,7,141,47,59,173,47,59,205,6186
1990 DATA 240,58,176,4,76,94,59,0,172,
238,58,162,0,173,237,58,7662
2000 DATA 32,24,56,165,160,41,127,141,
241,58,173,241,58,73,32,208,9099
2010 DATA 3,76,85,59,173,241,58,174,24
0,58,157,14,53,238,237,58,9904
2020 DATA 238,240,58,76,36,59,96,86,38
,76,38,76,102,59,142,96,3360
2030 DATA 59,141,95,59,172,96,59,162,0
,173,95,59,32,24,56,165,3752
2040 DATA 160,141,97,59,173,97,59,73,3
2,240,3,76,211,59,24,173,5783
2050 DATA 95,59,105,1,133,160,169,0,13
3,161,172,96,59,166,161,165,9387
2060 DATA 160,32,24,56,165,160,141,98,
59,173,98,59,73,160,208,3,6540
2070 DATA 76,208,59,238,254,52,169,7,2
05,254,52,176,3,76,203,59,8673
2080 DATA 169,20,141,253,2,172,96,59,1
62,0,173,95,59,32,13,56,2717
2090 DATA 76,208,59,169,7,141,254,52,7
6,246,59,173,97,59,73,20,5881
2100 DATA 240,3,76,246,59,56,173,254,5
2,233,1,141,254,52,169,32,9010
2110 DATA 141,253,2,172,96,59,162,0,17
3,95,59,32,13,56,96,240,5192
2120 DATA 59,20,76,253,59,169,32,141,2
53,2,160,1,140,247,59,169,9056
2130 DATA 20,205,247,59,176,3,76,121,6
0,160,0,140,249,59,169,3,6401
2140 DATA 205,249,59,176,3,76,115,60,1
69,0,133,133,169,10,133,132,6504
2150 DATA 173,249,59,162,0,32,2,54,141
,248,59,169,0,133,133,169,7271
2160 DATA 10,133,132,173,249,59,162,0,
32,2,54,133,172,138,133,173,7686
2170 DATA 24,169,8,101,172,141,91,60,1
73,91,60,205,248,59,176,4,8124
2180 DATA 76,109,60,36,172,247,59,162,
0,173,248,59,32,13,56,238,7261
2190 DATA 248,59,76,80,60,238,249,59,7
6,22,60,238,247,59,76,7,6756
2200 DATA 60,96,36,70,48,36,76,129,60,
142,123,60,141,122,60,160,5894
2210 DATA 0,140,125,60,200,140,124,60,
173,122,60,141,173,60,173,123,9074
2220 DATA 60,141,174,60,173,173,60,205
,124,60,173,174,60,237,125,60,9927
2230 DATA 176,5,76,186,60,53,36,238,12
4,60,208,232,238,125,60,76,9910
2240 DATA 156,60,96,65,65,36,65,53,76,
195,60,142,188,60,141,187,7873
2250 DATA 60,172,188,60,162,0,173,187,
60,32,24,56,165,160,141,191,8327
2260 DATA 60,160,0,140,189,60,169,7,20
5,189,60,176,3,76,107,61,5935
2270 DATA 173,191,60,73,32,208,3,76,64
,61,24,173,187,60,109,189,6790
2280 DATA 60,133,160,169,0,133,161,172
,188,60,166,161,165,160,32,24,8362
```

Character Set Display Utility

```
2290 DATA 56,165,160,141,190,60,173,19
0,60,73,32,208,3,76,61,61,4873
2300 DATA 24,173,187,60,109,189,60,133
,85,173,190,60,73,128,133,160,9235
2310 DATA 165,160,32,52,55,24,173,187,
60,109,189,60,133,174,24,165,7819
2320 DATA 174,105,1,133,85,76,101,61,2
4,173,187,60,109,189,60,133,7129
2330 DATA 85,173,191,60,73,128,133,160
,165,160,32,52,55,24,173,187,7617
2340 DATA 60,109,189,60,133,174,24,165
,174,105,1,133,85,238,189,60,9077
2350 DATA 76,222,60,96,36,56,53,36,65,
50,36,67,54,76,120,61,1436
2360 DATA 169,255,141,252,2,160,0,140,
254,52,169,22,133,84,200,132,9506
2370 DATA 85,169,0,133,163,169,0,133,1
65,169,32,133,164,160,56,162,9145
2380 DATA 53,169,14,32,32,56,76,182,61
,20,211,208,193,195,197,32,9697
2390 DATA 116,111,32,118,105,101,119,3
2,102,111,110,116,115,32,162,61,5495
2400 DATA 169,161,32,10,55,76,210,61,1
7,47,32,210,197,212,213,210,1295
2410 DATA 206,32,78,101,119,32,68,105,
115,107,162,61,169,192,32,10,5565
2420 DATA 55,160,0,140,112,61,200,140,
111,61,136,140,114,61,200,140,9162
2430 DATA 113,61,174,113,61,173,111,61
,32,192,60,173,31,208,73,5,5681
2440 DATA 240,14,173,120,2,41,2,133,17
4,165,174,240,3,76,117,62,6901
2450 DATA 162,9,169,196,32,126,60,173,
111,61,141,108,61,173,113,61,6983
2460 DATA 141,109,61,238,111,61,208,3,
238,112,61,169,20,205,111,61,8087
2470 DATA 169,0,237,112,61,48,3,76,99,
62,160,0,140,112,61,200,5794
2480 DATA 140,111,61,24,173,113,61,105
,10,141,113,61,173,114,61,105,5846
2490 DATA 0,141,114,61,169,31,205,113,
61,169,0,237,114,61,48,3,4915
2500 DATA 76,99,62,136,140,114,61,200,
140,113,61,174,108,61,173,109,8661
2510 DATA 61,32,192,60,174,111,61,173,
113,61,32,192,60,173,31,208,8076
2520 DATA 73,3,240,14,173,120,2,41,1,1
33,174,165,174,240,3,76,7418
2530 DATA 1,63,162,9,169,196,32,126,60
,173,111,61,141,108,61,173,7683
2540 DATA 113,61,141,109,61,56,173,111
,61,233,1,141,111,61,173,112,7640
2550 DATA 61,233,0,141,112,61,173,111,
61,201,1,173,112,61,233,0,7117
2560 DATA 48,3,76,239,62,160,0,140,112
,61,169,20,141,111,61,56,5103
2570 DATA 173,113,61,233,10,141,113,61
,173,114,61,233,0,141,114,61,7083
2580 DATA 173,113,61,201,1,173,114,61,
233,0,48,3,76,239,62,140,6460
2590 DATA 114,61,169,31,141,113,61,174
,108,61,173,109,61,32,192,60,6533
2600 DATA 174,111,61,173,113,61,32,192
,60,173,120,2,41,4,133,174,5544
2610 DATA 165,174,240,3,76,94,63,162,9
,169,196,32,126,60,173,111,7696
2620 DATA 61,141,108,61,173,113,61,141
,109,61,56,173,113,61,233,10,6890
2630 DATA 141,113,61,173,114,61,233,0,
141,114,61,173,113,61,201,1,6949
2640 DATA 173,114,61,233,0,48,3,76,76,
63,160,0,140,114,61,169,5182
2650 DATA 31,141,113,61,174,108,61,173
,109,61,32,192,60,174,111,61,6979
2660 DATA 173,113,61,32,192,60,173,120
,2,41,8,133,174,165,174,240,9995
2670 DATA 3,76,186,63,162,9,169,196,32
,126,60,173,111,61,141,108,7674
2680 DATA 61,173,113,61,141,109,61,24,
173,113,61,105,10,141,113,61,5041
2690 DATA 173,114,61,105,0,141,114,61,
169,31,205,113,61,169,0,237,8219
2700 DATA 114,61,48,3,76,168,63,160,0,
140,114,61,200,140,113,61,6818
2710 DATA 174,108,61,173,109,61,32,192
,60,174,111,61,173,113,61,32,6137
2720 DATA 192,60,173,31,208,73,6,240,8
,173,132,2,240,3,76,224,8279
2730 DATA 63,162,39,169,16,32,126,60,5
6,173,113,61,233,1,133,160,7351
2740 DATA 174,111,61,165,160,32,99,59,
173,252,2,73,12,240,10,173,7545
2750 DATA 252,2,73,33,240,3,76,243,61,
173,252,2,73,12,240,8,6971
2760 DATA 173,254,52,240,3,76,88,64,32
,250,59,169,22,133,84,160,7589
2770 DATA 1,132,85,76,39,64,24,32,73,1
10,115,101,114,116,32,68,3505
2780 DATA 105,115,107,32,119,105,116,1
04,32,70,111,110,116,115,44,162,6332
2790 DATA 64,169,14,32,10,55,76,63,64,
13,32,80,114,101,115,115,3257
2800 DATA 32,211,212,193,210,212,32,16
2,64,169,49,32,10,55,173,31,5684
2810 DATA 208,73,6,240,3,76,70,64,169,
255,141,252,2,76,218,64,9645
2820 DATA 160,0,140,254,52,140,112,61,
200,140,111,61,169,20,205,111,9269
2830 DATA 61,169,0,237,112,61,16,3,76,
218,64,160,0,140,114,61,5373
2840 DATA 140,113,61,169,30,205,113,61
,169,0,237,114,61,16,3,76,4498
2850 DATA 207,64,172,111,61,174,114,61
,173,113,61,32,24,56,165,160,6653
2860 DATA 141,110,61,173,110,61,73,20,
240,3,76,187,64,238,254,52,9759
2870 DATA 24,173,113,61,105,1,133,161,
172,111,61,166,161,173,254,52,1051
2880 DATA 32,242,58,24,173,113,61,105,
10,141,113,61,173,114,61,105,6391
2890 DATA 0,141,114,61,76,123,64,238,1
11,61,208,144,238,112,61,76,9646
2900 DATA 100,64,96,90,53,20,0,0,167,4
,29,91,36,50,57,36,9714
2910 DATA 48,51,36,65,65,36,66,68,36,5
5,56,76,246,64,169,0,4006
2920 DATA 32,213,55,160,1,140,240,2,16
9,82,141,198,2,169,12,141,7820
2930 DATA 197,2,76,53,65,39,17,18,18,1
8,18,18,18,18,18,23,6273
2940 DATA 18,18,18,18,18,18,18,18,18,2
3,18,18,18,18,18,18,5438
2950 DATA 18,18,18,23,18,18,18,18,18,1
8,18,18,5,162,65,169,386
2960 DATA 13,32,22,55,160,1,132,84,169
,20,197,84,176,3,76,106,5787
2970 DATA 65,169,9,133,85,169,29,197,8
5,176,3,76,101,65,169,124,7362
2980 DATA 32,52,55,24,165,85,105,9,133
,85,76,77,65,230,84,76,5867
2990 DATA 64,65,169,21,133,84,160,0,13
2,85,76,157,65,39,1,18,2516
3000 DATA 18,18,18,18,18,18,18,18,24,18,1
8,18,18,18,18,18,18,5496
```

```
3010 DATA 18,24,18,18,18,18,18,18,18,1
8,18,24,18,18,18,18,5542
3020 DATA 18,18,18,18,4,162,65,169,117
,32,22,55,76,207,65,39,3759
3030 DATA 124,32,32,32,32,32,32,32,32,
32,32,32,32,32,32,32,7474
3040 DATA 32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,7392
3050 DATA 32,32,32,32,32,32,124,162,65
,169,167,32,22,55,76,1,2594
3060 DATA 66,39,26,18,18,18,18,18,18,1
8,18,18,18,18,18,18,5622
3070 DATA 18,18,18,18,18,18,18,18,18,1
8,18,18,18,18,18,18,5518
3080 DATA 18,18,18,18,18,18,18,18,3,16
2,65,169,217,32,10,55,2417
3090 DATA 160,0,140,242,64,169,1,32,20
1,55,76,29,66,7,68,58,2682
3100 DATA 42,46,70,78,84,169,6,133,163
,169,0,133,164,160,66,162,9003
3110 DATA 21,169,1,32,161,55,160,1,140
,240,64,140,241,64,173,240,2371
3120 DATA 64,133,84,173,241,64,133,85,
169,20,133,163,160,64,162,219,1644
3130 DATA 169,1,32,130,55,169,64,205,2
22,64,144,3,76,175,66,238,9788
3140 DATA 242,64,169,3,141,239,64,169,
10,205,239,64,176,3,76,124,8959
3150 DATA 66,174,239,64,189,219,64,133
,160,165,160,32,52,55,238,239,2382
3160 DATA 64,76,95,66,238,240,64,169,2
0,205,240,64,144,3,76,165,9687
3170 DATA 66,160,1,140,240,64,24,173,2
41,64,105,10,141,241,64,169,210
3180 DATA 31,205,241,64,144,3,76,165,6
6,136,140,241,64,173,240,64,1454
3190 DATA 133,84,173,241,64,133,85,169
,127,205,83,3,144,10,173,242,660
3200 DATA 64,73,80,240,3,76,64,66,169,
1,32,201,55,32,117,61,4246
3210 DATA 173,252,2,73,33,240,3,76,8,6
6,169,255,141,252,2,96,8997
3220 DATA 57,68,36,52,76,223,66,32,151
,53,216,66,3,169,1,32,4154
3230 DATA 201,55,169,4,133,163,169,0,1
33,164,172,217,66,174,216,66,1813
3240 DATA 169,1,32,161,55,169,7,141,82
,3,173,219,66,141,85,3,6071
3250 DATA 173,218,66,141,84,3,169,4,14
1,89,3,169,0,141,88,3,3836
3260 DATA 162,16,169,0,32,86,228,169,1
,32,201,55,96,36,56,53,4225
3270 DATA 36,76,44,67,32,151,53,37,67,
2,160,0,140,40,67,169,4063
3280 DATA 0,133,163,160,20,162,53,169,
70,32,71,56,169,0,133,133,6193
3290 DATA 169,8,133,132,173,39,67,162,
0,32,2,54,133,174,138,133,6619
3300 DATA 175,166,174,189,14,53,73,32,
208,3,76,58,68,169,68,141,6200
3310 DATA 71,53,169,58,141,72,53,169,0
,133,133,169,8,133,132,173,8621
3320 DATA 39,67,162,0,32,2,54,133,174,
138,133,175,24,165,174,109,9078
3330 DATA 40,67,133,172,165,175,105,0,
133,173,166,172,189,14,53,73,8634
3340 DATA 32,208,3,76,227,67,173,40,67
,201,8,144,3,76,227,67,7178
3350 DATA 24,173,40,67,105,3,133,174,1
69,0,133,133,169,8,133,132,7970
3360 DATA 173,39,67,162,0,32,2,54,133,
172,138,133,173,24,165,172,8941
3370 DATA 109,40,67,133,170,165,173,10
5,0,133,171,166,170,189,14,53,9300
3380 DATA 166,174,157,70,53,238,40,67,
76,111,67,24,173,40,67,105,5467
3390 DATA 6,141,70,53,24,173,40,67,105
,3,133,174,169,46,166,174,8715
3400 DATA 157,70,53,24,173,40,67,105,4
,133,174,169,70,166,174,157,30
3410 DATA 70,53,24,173,40,67,105,5,133
,174,169,78,166,174,157,70,9528
3420 DATA 53,24,173,40,67,105,6,133,17
4,169,84,166,174,157,70,53,8801
3430 DATA 173,38,67,133,163,172,37,67,
162,53,169,70,32,220,66,76,7443
3440 DATA 81,68,169,224,133,163,169,0,
133,165,169,8,133,164,160,0,9113
3450 DATA 174,38,67,173,37,67,32,80,56
,96,68,17,32,76,88,68,2348
3460 DATA 160,0,140,82,68,169,5,205,82
,68,176,3,76,189,68,56,6337
3470 DATA 173,229,2,233,0,133,174,173,
230,2,233,4,133,175,165,174,2578
3480 DATA 41,0,141,83,68,165,175,41,25
2,141,84,68,169,8,133,132,8993
3490 DATA 173,84,68,170,173,83,68,32,5
6,54,174,82,68,157,94,53,6092
3500 DATA 173,84,68,141,230,2,173,83,6
8,141,229,2,24,173,82,68,7263
3510 DATA 105,1,133,162,164,162,174,84
,68,173,83,68,32,41,67,238,8220
3520 DATA 82,68,76,93,68,169,224,141,1
00,53,56,173,229,2,233,0,9010
3530 DATA 133,174,173,230,2,233,4,133,
175,165,174,41,0,141,83,68,7888
3540 DATA 165,175,41,252,141,84,68,169
,8,133,132,173,84,68,170,173,515
3550 DATA 83,68,32,56,54,141,244,2,173
,84,68,141,230,2,173,83,8707
3560 DATA 68,141,229,2,160,0,174,84,68
,173,83,68,32,41,67,56,4257
3570 DATA 173,229,2,233,0,133,174,173,
230,2,233,5,133,175,165,174,2690
3580 DATA 41,0,133,88,165,175,41,240,1
33,89,169,5,133,163,160,0,8871
3590 DATA 166,89,165,88,32,71,56,96,76
,51,69,72,138,72,152,72,5578
3600 DATA 238,2,53,173,2,53,73,7,240,3
,76,74,69,160,0,140,4879
3610 DATA 2,53,24,173,108,53,109,2,53,
133,174,173,109,53,105,0,5650
3620 DATA 133,175,160,0,177,174,141,24
,208,174,2,53,189,94,53,141,8785
3630 DATA 9,212,104,168,104,170,104,64
,96,76,116,69,169,192,141,14,8779
3640 DATA 212,24,173,90,53,105,0,141,9
0,53,173,91,53,105,1,141,5567
3650 DATA 91,53,173,90,53,41,0,141,90,
53,173,91,53,41,255,141,8044
3660 DATA 91,53,173,150,53,133,163,169
,0,133,165,169,39,133,164,172,1286
3670 DATA 149,53,174,91,53,173,90,53,3
2,80,56,24,173,90,53,105,5144
3680 DATA 4,133,174,173,91,53,105,0,13
3,175,165,88,41,255,160,0,8993
3690 DATA 145,174,24,173,90,53,105,5,1
33,174,173,91,53,105,0,133,6709
3700 DATA 175,169,8,133,132,165,89,170
,165,88,32,56,54,160,0,145,7053
3710 DATA 174,24,173,90,53,105,37,133,
174,173,91,53,105,0,133,175,8122
3720 DATA 173,90,53,41,255,160,0,145,1
74,24,173,90,53,105,38,133,7437
```

Character Set Display Utility

```
3730 DATA 174,173,91,53,105,0,133,175,
169,8,133,132,173,91,53,170,9277
3740 DATA 173,90,53,32,56,54,160,0,145
,174,173,91,53,141,49,2,5574
3750 DATA 173,90,53,141,48,2,173,50,69
,141,1,2,173,49,69,141,4981
3760 DATA 0,2,96,55,36,67,52,36,76,67,
70,169,0,32,213,55,4181
3770 DATA 169,255,141,252,2,160,0,140,
47,2,32,85,68,160,1,140,5164
3780 DATA 240,2,169,12,141,198,2,169,2
,141,197,2,169,6,141,2,5885
3790 DATA 53,32,113,69,169,34,141,47,2
,165,89,141,63,70,165,88,6955
3800 DATA 141,62,70,160,0,132,87,169,0
,205,254,52,144,3,76,79,7454
3810 DATA 71,140,59,70,56,173,254,52,2
33,1,141,160,70,173,160,70,560
3820 DATA 205,59,70,176,4,76,79,71,79,
160,0,132,84,132,85,140,7004
3830 DATA 60,70,169,7,205,60,70,176,3,
76,214,70,173,59,70,10,6114
3840 DATA 10,10,133,174,24,165,174,109
,60,70,133,172,166,172,189,14,557
3850 DATA 53,133,160,165,160,32,52,55,
238,60,70,76,170,70,32,59,6143
3860 DATA 55,76,243,70,22,32,32,65,66,
67,68,69,70,71,72,73,3114
3870 DATA 74,75,76,77,78,79,80,81,82,8
3,84,162,70,169,220,32,8226
3880 DATA 10,55,76,14,71,16,85,86,87,8
8,89,90,48,49,50,51,2616
3890 DATA 52,53,54,55,56,57,162,70,169
,253,32,22,55,76,53,71,5123
3900 DATA 28,32,32,97,98,99,100,101,10
2,103,104,105,106,107,108,109,7660
3910 DATA 110,111,112,113,114,115,116,
117,118,119,120,121,122,162,71,169,685
3920 DATA 24,32,22,55,24,165,88,105,16
0,133,88,165,89,105,0,133,7333
3930 DATA 89,238,59,70,76,149,70,24,17
3,62,70,105,96,133,88,173,8313
3940 DATA 63,70,105,4,133,89,160,0,132
,84,132,85,76,144,71,40,6002
3950 DATA 32,83,80,65,67,69,66,65,82,3
2,160,244,239,160,243,229,4781
3960 DATA 236,229,227,244,160,237,239,
242,229,160,230,239,238,244,243,160,39
16
3970 DATA 244,239,160,246,233,229,247,
160,162,71,169,103,32,22,55,24,8900
3980 DATA 173,62,70,105,176,133,88,173
,63,70,105,4,133,89,160,0,6430
3990 DATA 132,84,132,85,76,216,71,40,1
60,160,160,160,160,160,160,160,160,3519
4000 DATA 160,32,83,84,65,82,84,32,175
,212,242,233,231,231,229,242,9167
4010 DATA 160,244,239,160,209,245,233,
244,160,160,160,160,160,160,160,160,81
13
4020 DATA 162,71,169,175,32,10,55,173,
31,208,73,6,240,15,173,132,8791
4030 DATA 2,240,10,173,252,2,73,33,240
,3,76,223,71,173,252,2,140
4040 DATA 73,33,240,3,76,12,72,169,255
,141,252,2,160,1,140,6,8010
4050 DATA 53,76,17,72,160,0,140,6,53,9
6,76,21,72,32,200,56,4227
4060 DATA 160,0,140,6,53,32,243,64,32,
64,70,173,6,53,240,3,5576
4070 DATA 76,29,72,169,0,32,213,55,96,
96,226,2,227,2,18,72,5954
```

LISTING 2 : ACTION!

```
;
;Character Set Display Utility
; by Dave Arlington
; COPYRIGHT 1989 BY ANALOG COMPUTING

; IMPORTANT NOTE: AN "*" AT THE END
; OF A LINE MEANS THAT THE LINE WRAPS
; AROUND TO THE NEXT LINE OF THE
; LISTING.  THE TWO LINES SHOULD BE
; TYPED AS ONE, WITHOUT THE "*".

;      CHECKSUM DATA
;[34 56 26 A0 52 CB 56 B0
; 7E ED CA 6D 57 AB 80 12
; 89 32 82 ]

;
;Global variables and defines
;

DEFINE RTI="$40",
       PHA="$48",
       PLA="$68",
       TXA="$8A",
       TAX="$AA",
       TYA="$98",
       TAY="$A8",
       START="6",
       SELECT="5",
       OPTION="3"

BYTE dindex=$57,   sdmctl=$22F,
     nmien =$D40E, consol=$D01F,
     chbase=$D409, chbas =$2F4,
     color2=$2C6,  colpf2=$D018,
     rowcrs=$54,   colcrs=$55,
     crsinh=$2F0,  lmargn=$52,
     color1=$2C5,  icsta =$353,
     stick0=$278,  strig0=$284,
     ch    =$2FC,  iccom =$352,
     txtrow=$290,  selfiles,
     wsync =$D40A, count,
     finish

CARD savmsc=$58,   sdlstl=$230,
     vdslst=$200,  memtop=$2E5,
     icbadr=$354,  icblen=$358,
     txtcol=$291,  RamSet

BYTE ARRAY fnames(56), filenm(20),
           dlspace(512), cset(7),
           cols=[$7C $4C $8C
                 $6C $2C $5C $9C],
           dl=[112 112 112 $42 0 0 2
               130 $0E 2 2 130 $0E
               2 2 130 $0E 2 2 130
               $0E 2 2 130 $0E 2 2
               130 $0E 2 2 130 $0E
               2 $0E 2 $41 0 0]

INCLUDE "D:CSET2.ACT"

PROC CIO=$E456(BYTE areg, xreg)


PROC LoadFont(BYTE ARRAY namefile
              CARD addr)

  Close(1)
  Open(1,namefile,4,0)
  iccom=7
```

```
    icbadr=addr
    icblen=1024
    CIO(0,$10)
    Close(1)
  RETURN


  PROC GetFont(CARD addr BYTE filenum)

    BYTE letcnt

    letcnt=0
    Zero(filenm,20)
  IF fnames(filenum*8)#32 THEN
    filenm(1)='D
    filenm(2)=':
    WHILE fnames(filenum*8+letcnt)#32
          AND letcnt<8
    DO
      filenm(letcnt+3)=fnames
                       (filenum*8+letcnt)
      letcnt==+1
    OD
    filenm(0)=letcnt+6
    filenm(letcnt+3)='.
    filenm(letcnt+4)='F
    filenm(letcnt+5)='N
    filenm(letcnt+6)='T
    LoadFont(filenm,addr)
  ELSE
    MoveBlock(addr,$E000,8)
  FI
RETURN


PROC AllocMem()

  BYTE cnt
  CARD temp

  FOR cnt=0 TO 5
  DO
    temp=((memtop-$400)&$FC00)
    cset(cnt)=temp RSH 8
    memtop=temp
    GetFont(temp,cnt+1)
  OD
  cset(6)=$E0
  temp=((memtop-$400)&$FC00)
  chbas=temp RSH 8
  memtop=temp
  GetFont(temp,0)
  savmsc=(memtop-$500)&$F000
  Zero(savmsc,1280)
RETURN

PROC Dli2()

  [PHA TXA PHA TYA PHA]
  count==+1
  IF count=7 THEN
    count=0
  FI
  colpf2=cols(count)
  chbase=cset(count)
  [PLA TAY PLA TAX PLA RTI]
RETURN


PROC Dlisetup2()

    nmien=192
    dlspace==+256
    dlspace==&$FF00
    MoveBlock(dlspace,dl,39)
    dlspace(4)=savmsc&$FF
    dlspace(5)=savmsc RSH 8
    dlspace(37)=dlspace&$FF
    dlspace(38)=dlspace RSH 8
    sdlstl=dlspace
    vdslst=Dli2
  RETURN

  PROC ShowFonts()

    BYTE cnt1, cnt2, cnt3
    CARD temp

    Graphics(0)
    ch=255
    sdmctl=0
    AllocMem()
    crsinh=1
    color2=$0C
    color1=2
    count=6
    DliSetup2()
    sdmctl=34
    temp=savmsc
    dindex=0
    IF selfiles>0 THEN
      FOR cnt1=0 TO selfiles-1
      DO
        rowcrs=0
        colcrs=0
        FOR cnt2=0 TO 7
        DO
          Put(fnames((cnt1 LSH 3)+cnt2))
        OD
        PutE()
        Print("  ABCDEFGHIJKLMNOPQRST")
        PrintE("UVWXYZ0123456789")
        PrintE("  abcdefghijklmnopqrst
uvwxyz")
        savmsc==+160
      OD
    FI
    savmsc=temp+1120
    rowcrs=0
    colcrs=0
    PrintE(" SPACEBAR to select more
fonts to view ")
    savmsc=temp+1200
    rowcrs=0
    colcrs=0
    Print("          START /Trigger to
 Quit          ")
    DO
    UNTIL consol=START OR strig0=0
          OR ch=33
    OD
    IF ch=33 THEN
      ch=255
      finish=1
    ELSE
      finish=0
    FI
  RETURN


  PROC Main()
    TitleScreen()
```

```
  finish=0
  DO
    FileScreen()
    ShowFonts()
  UNTIL finish=0
  OD
  Graphics(0)
RETURN
```

LISTING 3 : ACTION!

```
; IMPORTANT NOTE: AN "*" AT THE END
; OF A LINE MEANS THAT THE LINE WRAPS
; AROUND TO THE NEXT LINE OF THE
; LISTING.  THE TWO LINES SHOULD BE
; TYPED AS ONE, WITHOUT THE "*".

;      CHECKSUM DATA
;[4E 02 49 20 E4 FD 38 BB
; 66 E9 7E 37 FC 93 EC 8D
; 8F A6 32 BE FD DC 1B E6
; B2 6D ]

PROC dli()

  [PHA]
  chbase=$E0      ;go back to ROM set
  [PLA RTI]
RETURN


PROC DliSetup()

BYTE ARRAY temp

  nmien=192       ;enable interrupts
  temp=sdlstl     ;switch sets at
  temp(14)==+128 ;text window
  sdmctl=0
  vdslst=dli      ;call interrupt
  sdmctl=34
RETURN


PROC TitleScreen()

  Graphics(2)    ;all this work to get
  DliSetup()     ;upper and lower case
  color2=0       ;in Graphics 2!!
  crsinh=1
  lmargn=0
  RamSet=(memtop-$400)&$FC00
  chbas=RamSet RSH 8
  memtop=RamSet
  MoveBlock(RamSet,$E200,512)
  MoveBlock(RamSet,$E000,8)
  MoveBlock(RamSet+8,$E100+24,8)
  MoveBlock(RamSet+40,$E100+32,8)
  MoveBlock(RamSet+24,$E100+152,8)
  MoveBlock(RamSet+32,$E100+168,8)
  rowcrs=3
  colcrs=5
  PrintDE(6,"!HARACTER")
  PrintDE(6,"      #ET")
  PrintDE(6,"       %ISPLAY")
  PrintDE(6,"        $TILITY")
  txtrow=0
  txtcol=0
  PrintE("            by Dave Arlingt*
on")
  PrintE("     (c) 1989 for ANALOG C*
omputing")
  PrintE("")
  Print("  Insert Disk with Fonts, P*
ress START")
  DO
  UNTIL consol=START OR strig0=0
  OD
RETURN


PROC StashFile(BYTE filenum, col, row)

  BYTE index,cnt,temp

  index=(filenum-1)*8
  FOR cnt=index TO index+7
  DO
    temp=Locate(col,row)&$7F
    IF temp#32 THEN
      fnames(cnt)=temp
    FI
    col==+1
  OD
RETURN

PROC MarkFile(BYTE col, row)

  BYTE mark,letter

  mark=Locate(col,row)
  IF mark=32 THEN
    letter=Locate(col+1,row)
    IF letter#160 THEN
      selfiles==+1
      IF selfiles<=7 THEN
        color=20
        Plot(col,row)
      ELSE
        selfiles=7
      FI
    FI
  ELSEIF mark=20 THEN
    selfiles==-1
    color=32
    Plot(col,row)
  FI
RETURN


PROC ClearScreen()

  BYTE crow, ccol, cnt

  color=32
  FOR crow=1 TO 20
  DO
    FOR cnt=0 TO 3
    DO
      FOR ccol=cnt*10 TO 8+cnt*10
      DO
        Plot(ccol,crow)
      OD
    OD
  OD
RETURN


PROC pauz(CARD tim)
```

```
  CARD cnt

  FOR cnt=1 TO tim DO OD
RETURN


PROC FlipFont(BYTE col,row)

  BYTE cnt, letter, let

  let=Locate(col,row)
  FOR cnt=0 TO 7
  DO
    IF let#32 THEN
      letter=Locate(col+cnt,row)
      IF letter#32 THEN
        colcrs=col+cnt
        Put(letter!$80)
        colcrs=col+cnt+1
      FI
    ELSE
      colcrs=col+cnt
      Put(let!$80)
      colcrs=col+cnt+1
    FI
  OD
RETURN


PROC SelectFiles()

  BYTE oldrow, oldcol, mark
  INT  row, col, delay

  ch=255
  selfiles=0
  rowcrs=22
  colcrs=1
  SetBlock(fnames,56,32)
  Print("SPACE to view fonts ")
  Print("/ RETURN New Disk")
  row=1
  col=1
  FlipFont(row,col)
  DO
    IF consol=SELECT OR (stick0&2)=0
 THEN
      pauz(2500)
      oldrow=row
      oldcol=col
      row==+1
      IF row>20 THEN
        row=1
        col==+10
        IF col>31 THEN
          col=1
        FI
      FI
      FlipFont(oldcol,oldrow)
      FlipFont(col,row)
    FI
    IF consol=OPTION OR (stick0&1)=0
 THEN
      pauz(2500)
      oldrow=row
      oldcol=col
      row==-1
      IF row<1 THEN
        row=20
        col==-10
        IF col<1 THEN
          col=31
        FI
      FI
      FlipFont(oldcol,oldrow)
      FlipFont(col,row)
    FI
    IF (stick0&4)=0 THEN
      pauz(2500)
      oldrow=row
      oldcol=col
      col==-10
      IF col<1 THEN
        col=31
      FI
      FlipFont(oldcol,oldrow)
      FlipFont(col,row)
    FI
    IF (stick0&8)=0 THEN
      pauz(2500)
      oldrow=row
      oldcol=col
      col==+10
      IF col>31 THEN
        col=1
      FI
      FlipFont(oldcol,oldrow)
      FlipFont(col,row)
    FI
    IF consol=START OR strig0=0 THEN
      pauz(10000)
      MarkFile(col-1,row)
    FI
  UNTIL ch=12 OR ch=33
  OD
  IF ch=12 OR selfiles=0 THEN
    ClearScreen()
    rowcrs=22
    colcrs=1
    Print(" Insert Disk with Fonts,")
    Print(" Press START ")
    DO
    UNTIL consol=START
    OD
    ch=255
  ELSE
    selfiles=0
    FOR row=1 TO 20
    DO
      FOR col=0 TO 30 STEP 10
      DO
        mark=Locate(col,row)
        IF mark=20 THEN
          selfiles==+1
          StashFile(selfiles,col+1,row)
        FI
      OD
    OD
  FI
RETURN


PROC FileScreen()

  BYTE ARRAY name(20)
  BYTE cnt, row, col, numfiles

  Graphics(0)
  crsinh=1
  color2=$52
  color1=12
  Print("
```

continued on page 57

AUTORUN.SYS

I've been programming in BASIC now for over three years and have been frustrated by the fact that all I could do with an AUTORUN.SYS is use my SETUP.COM program on my Atari DOS 2.5 disk to make one of my BASIC programs load and run when my system was booted. I envied assembly language programmers with the power they had over the machine; so about six months ago I started to teach myself assembler with the help of my Assembler/Editor cartridge and a few good books.

It is everything the assembly language program is, with the advantage that you don't need an assembler.

Secrets in BASIC

It was about the time that I was starting to understand the assembler that I received my November '88 issue of ANALOG. To my delight it contained an article and program by LeRoy Baxter titled "AUTORUN.SYS Secrets." I quickly studied it and typed it in. It is a powerful and flexible tool, allowing me to load, merge and run programs in any way I need, along with displaying messages, etc. I haven't even started to utilize it to its fullest.

In my excitement I stopped and realized that if I were still programming strictly in BASIC, I wouldn't have an assembler and would have been unable to utilize this great tool. So for all of you who are strictly BASIC programmers, I have written AUTOSYS.BAS, a BASIC version of Mr. Baxter's program.

Type in Listing 1 (check your work with BASIC Editor II, elsewhere in this issue) and save a copy. You'll use it many times. Now run the program. It will ask you to enter your command line. You can now enter up to 119 characters of BASIC code (three lines minus one character). Type it in as if you were going to execute it in immediate mode (i.e. no line numbers). Make sure the code is legal BASIC code, as there is no syntax error checking. When you are finished, hit Return. The program will now ask for the disk on which to write the AUTORUN.SYS file. Insert the disk and hit Start. If all goes well you will see the READY prompt, and your AUTORUN.SYS file is ready for the next time you boot that disk.

You can write messages on the screen, automatically run a BASIC program, two-stage LOAD and ENTER and even create an AUTORUN.SYS for a language other than BASIC. It is everything the assembly language program is, with the advantage that you don't need an assembler, and you can use all the special characters (such as the screen clear and bell) and the quote mark without having to specify them in their ATASCII values as in the original version. For more information on what the program can do and how it works, read LeRoy Baxter's "AUTORUN.SYS Secrets" in Issue 66 of ANALOG.

This is how the BASIC modification works. First it asks for your command line. After entering your BASIC code into A$ (Line 130) Lines 220-240 calculate the length and ending address of the machine-language file to be written. It then opens the AUTORUN.SYS file on the disk and writes the two-byte identifier code and starting address (Line 270 and DATA statement 1000) and the ending address (Line 280) to the file. Next it writes Mr. Baxter's program byte for byte to the file (Line 290 and DATA statements 1010-1060). Next your command line is written from A$ (Line 300). Finally, a carriage-return character, required by the program, and the starting address of the program, required for an autoboot file, are written to the file (Line 310 and DATA statement 1070). The file is then closed, and you're ready for action.

David Schoch is 41 years old, happily married and the proud father of an 18-year-old daughter. He has been having a love affair with his Atari computer for three years, and his wife has accepted this mistress and considers her a "good friend."

Secrets in BASIC

AUTORUN.SYS

LISTING 1: BASIC

```
EI 1 REM ********************************
RH 2 REM *      AUTORUN.SYS MAKER       *
EY 3 REM *       by David Schoch        *
TB 4 REM *                              *
VR 5 REM *       COPYRIGHT 1989         *
LV 6 REM *     BY ANALOG COMPUTING      *
EO 7 REM ********************************
NN 8 REM
ZN 100 DIM A$(119):COUNT=0
OP 110 POKE 82,0:? CHR$(125):POSITION 0,2
LP 120 ? "            AUTORUN.SYS Creator":
   ? :? "*Input Your Command Line -119 Ch
   ar Max.*";:?
ZE 130 INPUT #16,A$
WO 140 COUNT=LEN(A$):IF COUNT=0 THEN 410
IO 150 POKE 82,2
OU 160 ? CHR$(125):POSITION 2,6
XB 170 ? "Insert Disk on which you wish t
   o"
NT 180 ? "write new AUTORUN.SYS into Driv
   e #1."
MO 190 ? :? "Press START when ready...";
RU 200 IF PEEK(53279)<>6 THEN 200
JK 210 ? :? :? "Creating...AUTORUN.SYS...
   ";
QU 220 LENGHT=123+COUNT+16384
MT 230 MSB=INT(LENGHT/256)
UI 240 LSB=LENGHT-(MSB*256)
OF 250 TRAP 400:OPEN #1,8,0,"D:AUTORUN.SY
   S"
UW 260 RESTORE 1000
UY 270 FOR I=1 TO 4:READ N:PUT #1,N:NEXT
   I
XP 280 PUT #1,LSB:PUT #1,MSB
TA 290 FOR I=1 TO 123:READ N:PUT #1,N:NEX
   T I
DU 300 FOR I=1 TO COUNT:N=ASC(A$(I,I)):PU
   T #1,N:NEXT I
WJ 310 FOR I=1 TO 7:READ N:PUT #1,N:NEXT
   I
LD 320 CLOSE #1
DN 330 ? CHR$(125):END
VD 400 TRAP 40000:? :? "ERROR-";PEEK(195)
   :END
LV 410 ? :? "ERROR- NO COMMANDS ENTERED":
   END
DL 1000 DATA 255,255,0,64
VJ 1010 DATA 162,36,189,26,3,201,69,240,5
   ,202,202,202,16,244,232,142,83,64,189,
   26,3
MB 1020 DATA 133,203,169,65,157,26,3,232,
   189,26,3,133,204,169,64,157,26,3,160,0
   ,140
IY 1030 DATA 82,64,177,203,153,65,64,200,
   192,16,144,246,169,83,141,69,64,169,64
   ,141,70
WP 1040 DATA 64,96,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0
UI 1050 DATA 172,82,64,185,123,64,201,155
   ,240,6,238,82,64,160,1,96,72,138,72,17
   4,83
HW 1060 DATA 64,165,203,157,26,3,232,165,
   204,157,26,3,104,170,104,160,1,96
HZ 1070 DATA 155,226,2,227,2,0,64
```

Character Set Display Utility

continued from page 53

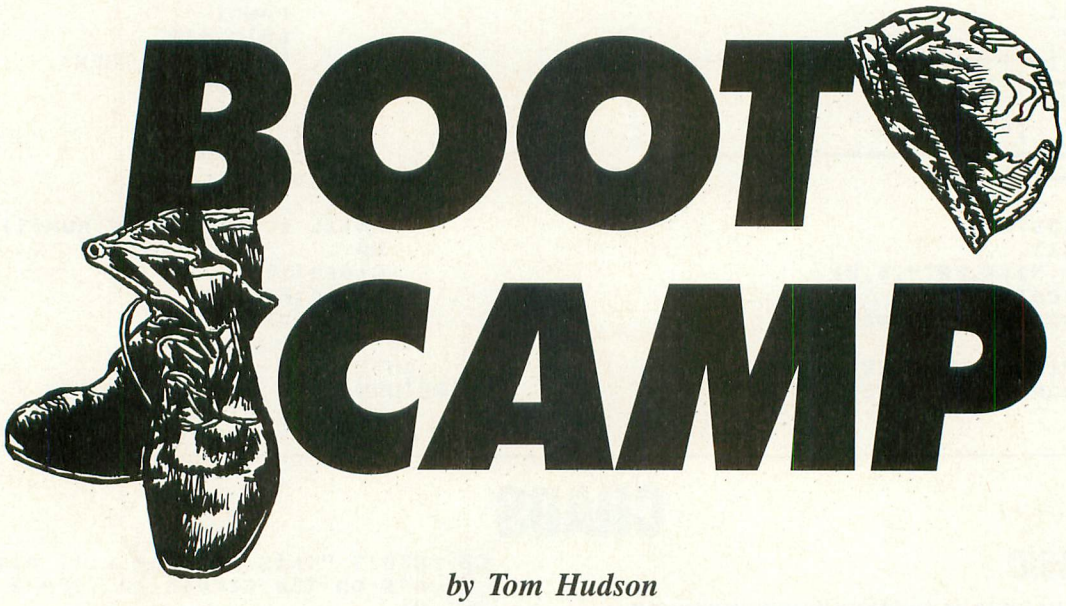
```
┌──────────┬──────────┐")
  FOR rowcrs=1 TO 20
  DO
    FOR colcrs=9 TO 29 STEP 9
    DO
      Put(124)
    OD
  OD
  rowcrs=21
  colcrs=0
  PrintE("├──────────┴──────────┴──────────●
└──────────┴──────────┘")
  PrintE("|                               ●
          |")
  Print("└──────────┴──────────┴──────────●
└──────────┴──────────┘")
  DO
    numfiles=0
    Close(1)
    Open(1,"D:*.FNT",6,0)
    row=1 col=1
    rowcrs=row colcrs=col
    DO
      InputMD(1,name,20)
      IF name(3)>64 THEN
        numfiles==+1
        FOR cnt=3 TO 10
        DO
          Put(name(cnt))
        OD
        row==+1
        IF row>20 THEN
          row=1
          col==+10
          IF col>31 THEN
            col=0
          FI
        FI
        rowcrs=row
        colcrs=col
      FI
    UNTIL icsta>$7F OR numfiles=80
    OD
    Close(1)
    SelectFiles()
  UNTIL ch=33
  OD
  ch=255
RETURN
```

CHAOS

continued from page 11

LISTING 1: BASIC

```
AE 10 REM ********************************
ZP 11 REM *            CHAOS             *
CO 12 REM *     by Alfredo L. Acosta     *
FY 13 REM *                              *
IP 14 REM *       COPYRIGHT 1989         *
YU 15 REM *     BY ANALOG COMPUTING      *
AQ 16 REM ********************************
BM 17 REM
ME 20 GOSUB 1000
KB 29 REM  PLOT TRIANGLE
SL 30 GRAPHICS 24:POKE 709,14:POKE 710,0:
   COLOR 1
LH 40 PLOT AX,AY:PLOT BX,BY:PLOT CX,CY
AC 49 REM  PLOT INITIAL POINT
ER 50 X=INT(RND(0)*310)+1:Y=INT(RND(0)*18
   5)+1:PLOT X,Y
DP 59 REM  MAIN ROUTINE
SS 60 Z=INT(RND(0)*3)+1
ZB 70 IF Z=1 THEN M=ABS(AX-X):N=ABS(AY-Y)
   :D=AX:E=AY:GOTO 100
FA 80 IF Z=2 THEN M=ABS(BX-X):N=ABS(BY-Y)
   :D=BX:E=BY:GOTO 100
PI 90 M=ABS(CX-X):N=ABS(CY-Y):D=CX:E=CY
KM 100 IF X>D THEN X=D+M/2:GOTO 120
ZJ 110 IF X<D THEN X=D-M/2
PX 120 IF Y>E THEN Y=E+N/2:GOTO 140
CK 130 IF Y<E THEN Y=E-N/2
TA 140 PLOT X,Y
SG 150 IF PEEK(53279)<>5 THEN 60
KC 160 GOTO 2000
QF 999 REM  INITIALIZATION & TEXT
UW 1000 DIM A$(193):POKE 712,148:POKE 752
   ,1:? CHR$(125)
NL 1010 AX=160:AY=176:BX=64:BY=16:CX=256:
   CY=BY
OK 1020 POSITION 13,2:? "*** CHAOS ***":P
   OSITION 5,6
CZ 1030 ? "This program will place three
   dots on the screen to form a triangle.
    ";
BA 1040 ? "It  will then place a dot rand
   omly on the screen. From there it will
    randomly   choose a direction ";
WB 1050 ? "toward one of the  three start
   ing dots and place a new   spot 1/2 th
   e distance ";
MX 1060 ? "towards it. Thislast step will
    be continually repeatedfrom the last
   new spot. ";
UU 1070 ? "Instead of    total coverage o
   f the area within the triangle, surpri
   sing patterns emerge!"
GZ 1080 ? :? "Press SELECT to dump to pri
   nter."
NQ 1090 ? :? "Press START to begin."
NK 1100 IF PEEK(53279)<>6 THEN 1100
AE 1110 RETURN
LK 1999 REM  SCREEN DUMP
DT 2000 RESTORE 3000:LPRINT CHR$(27);CHR$
   (64)
RN 2010 FOR T=1 TO 61:READ Q:POKE 1535+T,
   Q:NEXT T:DM=PEEK(88)+PEEK(89)*256:DM=D
   M+40*191
RQ 2020 LPRINT CHR$(27);"3";CHR$(20):FOR
   V=DM TO DM+39
MV 2030 A$=CHR$(0):A$(192)=CHR$(0):A$(2)=
   A$
KN 2040 A=USR(1536,V,ADR(A$)):LPRINT CHR$
   (27);"K";CHR$(192);CHR$(0);A$
XF 2050 NEXT V:END
CO 3000 DATA 104,104,141,21,6,104,141,20,
   6,104,141,27,6,104,141,26,6,160,193,17
   3,255,255,136,240,35,141,255,255,238
QI 3010 DATA 26,6,240,21,173,20,6,56,233,
   40,141,20,6,144,4,24,76,19,6,206,21,6,
   76,19,6,238,27,6,76,33,6,96
```



by Tom Hudson

As promised last issue, we're going to cover more 6502 instructions this time, and begin exploring the world of simple mathematical operations.

Before we start with the math operations, let's look at an instruction that will help us during the testing of the programs we write in this column.

BREAKing Away

Remember the do-nothing program from last issue? When we executed it with the "G" (execute program) command with the assembler editor cartridge, it ran forever. This is hardly a good way to test programs. Imagine trying to stop the program at a specific instruction with the Break key when hundreds of thousands of operations are being executed each second. You can see that this would be nearly impossible.

Luckily for us, the 6502 has a handy instruction called BRK (or BREAK). This instruction does the same thing as the Break key on the keyboard when an assembly program is executing. The nice part is that it will stop the program exactly where we want to stop.

The short program below has a BRK instruction after the load accumulator (LDA) instruction. The accumulator will be loaded with $4F (79 decimal) and the computer will stop. Type the program into your computer and assemble it into memory with the ASM command.

```
10  *=   $0600    ; START ADDRESS
20  LDA  #$4F     ; LOAD ACCUMULATOR
30  BRK           ; AND STOP
40  .END
```

After the program is assembled, go to the DEBUG mode with the BUG command. To execute the short program, type:

G 600

The program will execute in a fraction of a second and the computer will return with a display similar to this

```
0602     A=4F X=00 Y=00 P=30 S=00
```

Note that the accumulator (A) equals $4F. The X, Y, processor status and stack registers are also displayed, but have no significance to us at this time, since we didn't change them.

Now you can see that the BRK instruction can be helpful in the debugging stage of a program. We will be using it to stop the computer when we want to check the results of certain operations.

Using Index Registers

Index registers were mentioned briefly last issue. As you may recall, there are two index registers in the 6502, the X and Y registers. These two registers are built into the 6502 microprocessor chip. Each is made up of eight bits, allowing a range of values from 0-255.

The first instructions we'll look at are the LDX (load X) and LDY (load Y) instructions. These instructions are similar to the LDA (load accumulator) instruction we examined last time. Their formats are:

```
LDX #n      (IMMEDIATE)
LDX nn      (ABSOLUTE)
LDX n       (ZERO PAGE)
LDX nn,Y    (INDEXED Y)
LDX n,Y     (ZERO PAGE INDEXED Y)
LDY #n      (IMMEDIATE)
LDY nn      (ABSOLUTE)
```

LDY n (ZERO PAGE)
LDY nn,X (INDEXED X)
LDY n,X (ZERO PAGE INDEXED X)

The LDX and LDY instructions place a specified value in the X or Y register, respectively. For example, the following instruction will load the X register with $3A (58 decimal):

LDX #3A

The following instruction will load the Y register with the contents of Memory Location $3F00:

LDY $3F00

The following instruction will load the X register from the page zero-Location $4D, which is the attract mode counter:

LDX $4D

Like the LDA instruction, both the LDX and LDY instructions set the sign and zero flags depending on the number loaded into the register.

Storing the contents of the X and Y registers is just as easy as loading them. The following addressing modes are available with the STX (store X) and STY (store Y) instructions:

STX nn (ABSOLUTE)
STX n (ZERO PAGE)
STX n,Y (ZERO PAGE INDEXED Y)
STY nn (ABSOLUTE)
STY n (ZERO PAGE)
STY n,X (ZERO PAGE INDEXED X)

Unfortunately for us, the designers of the 6502 decided to limit indexed store X and Y instructions to page zero, even though there is a non-zero-page load instruction. This is simply something assembly programmers must live with.

Like the STA instruction, the STX and STY instructions do not affect any status flags.

The STX and STY instructions are very easy to use. For example, to store the X register at location $4FFB, simply use the instruction:

STX $4FFB

In addition to the LDX/LDY and STX/STY instructions, the 6502 provides four more instructions which help the programmer with X/Y operations. These are the transfer instructions.

The transfer instructions allow quick movement of information from one register to another. They are TAX, TAY, TXA and TYA. Two other transfer instructions, TSX and TXS, are used in stack operations, and we'll look at them in a later article.

The TAX and TAY instructions transfer the contents of the accumulator (A) to the X or Y register, respectively. The A register is unchanged.

The code below illustrates how the TAX instruction works. Type this short program into your computer and assemble it into memory:

```
10  *=   $0600    ; START ADDRESS
20  LDA  #$0F     ; PUT $0F IN A
30  TAX           ; PUT IN X, TOO
40  LDA  #$6A     ; PUT $6A IN A
50  TAY           ; NOW PUT IN Y, TOO
60  BRK           ; AND STOP!
70  .END
```

Line 20 loads the accumulator with $OF, (15 decimal).

Line 30 transfers the contents of the accumulator to the X register. At this point both the accumulator and the X register will contain $OF.

Line 40 loads the accumulator with $6A (106 decimal).

Line 50 transfers the contents of the accumulator to the Y register. Now the accumulator and the Y register will contain $6A. The X register will be unchanged.

Line 60 will BREAK the execution of the program.

After the program is assembled into memory, go to DEBUG mode and execute it by typing:

G 600

After execution, the screen of your computer should look like this:

```
0606     A=6A X=0F Y=6A P=30 S=00
```

You can see that the X register contains $OF, and that the A and Y registers contain $6A. Try some different combinations and observe the results.

The two other transfer instructions we are concerned with here are the TXA and TYA instructions. As you may have guessed, these instructions do the opposite of the TAX and TAY instructions. That is, TXA will transfer the contents of the X register to the accumulator, and TYA will move the Y register's contents to the accumulator.

Here's a small problem for you to solve using the instructions we've covered so far. This is a simple data-manipulation operation using the A, X and Y registers and as many locations as necessary.

Problem: Write a program which starts with A=$03, X=$07 and Y=$14. Then write

the code necessary to change these registers so that when the program ends, the registers are A=$07, X=14 and Y=$03.

The code necessary to perform this change is only four lines long, and there are many ways to do it. Next issue I'll show several possible solutions.

This issue, we've only shown how to make the X and Y registers contain the values we want. In order to make the X and Y registers do some real work, we'll need to cover the branch-on-condition instructions. These will be discussed next issue, along with X and Y register indexing techniques.

It All ADDs Up

I'm sure that, by now, just about every person reading this column wants to start working with something more interesting than loading and storing bytes, right? Well, let's take a break from all that admittedly dull stuff and get on with something fun—actual addition.

We'll start out with some simple addition, working with values from 0-255. This is known as single-byte integer arithmetic, and is the simplest kind of math on the 6502.

Why only integers from 0-255? Remember that all arithmetic operations must be processed through the accumulator, or A register. The accumulator is made up of only eight bits, and can't hold any number greater than 255. The accumulator doesn't know what a decimal point is either, so we are limited to integers for the time being.

Binary or BCD?

The 6502 microprocessor has the option of performing arithmetic instructions in two different modes, binary and binary-coded decimal (BCD). Let's look at how both these systems work.

Binary arithmetic, as we have noted before, produces numbers from 0-255 in one byte. All eight bits are used for the number. These numbers can be considered either signed or unsigned by the programmer, but they are handled the same by the computer. Since all eight bits are used to represent the number, the value of a byte is simply the byte's decimal contents.

BCD arithmetic, on the other hand, is a more human approach to computer math, and easier to use in input/output operations.

In BCD math, the byte is split into two four-bit sections, or NYBBLES. Each nybble contains one decimal number, from 0-9. With this system, each byte contains two decimal numbers, allowing easy Base-10 number storage. Of course, the BCD numbering system requires more storage than binary, since the value of a byte can now only range from 0-99, rather than 0-255. The nice thing about BCD is that when looking at the hexadecimal representation of the byte, you see the decimal value of the byte. For example, $56 is 56 decimal.

We'll cover BCD math later in this series, when we get into screen I/O. For now we'll stick with binary math. Even though it may seem more difficult, binary math is much more important at this early stage.

Getting Into BINARY

The 6502 can handle two different types of math, so how does it know which one you want to use? The answer lies in a single-bit flag in the processor status register, called the decimal-mode flag.

The decimal-mode flag has two states. When set (1), the decimal mode is selected. When cleared (0), the binary mode is selected. This flag is *extremely* important! The following example illustrates this fact.

Let's say you want to add two binary numbers, $23 and $18. A normal binary add would give a result of $3B.

What if the decimal-mode flag was set by mistake? The add would give a result of $41, the sum of 23 and 18. If your program adds or subtracts numbers with the decimal mode incorrectly set, the results can be very confusing. Moral: always know the setting of the decimal-mode flag.

For our purposes, until further notice, we

will always clear the decimal mode with the CLD (clear decimal mode) instruction. The format of this instruction is:

CLD

This is a simple instruction, but easy to forget. If you have trouble remembering things (like myself), I suggest that you tape an appropriate message to your monitor, computer, forehead, etc. This will save an incredible amount of debugging time.

Important: When writing assembly subroutines for BASIC programs, you *must* clear the decimal mode if you're doing any arithmetic in the subroutine. BASIC uses the floating-point arithmetic package built into the computer, which sets the decimal mode. The first time I wrote a BASIC assembly subroutine with math, it took me two days to find the problem. Once again, write a note.

Now that I've warned you about the evils of decimal-mode ignorance, let's get on with some actual addition.

Add 'Em Up!

First we'll cover single-byte additions, the simplest kind. These types of additions are sufficient for general counters, changing color registers, or any operation in which the result will not exceed 255.

The 6502 has only one add instruction, ADC add with carry. This instruction has the following formats:

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

The ADC instruction adds the number at the memory location specified in the operand to the accumulator and places the result in the accumulator. Depending on the result, the 6502 will alter the sign, overflow, zero and carry flags.

Let's look at a simple single-byte addition operation, using the immediate format. We will add 23 and 14 decimal and place the result in a location called ANSWER. Here's the code needed to perform this operation.

```
10  LDA  #23      ; PUT 23 IN A
20  CLC           ; CLEAR CARRY FOR ADD
30  ADC  #14      ; AND ADD 14 TO IT!
40  STA  ANSWER   ; SAVE RESULT
```

The first line places the number 23 in the accumulator. Simple enough, right?

The second line introduces a new operation code, CLC (clear carry). The CLC instruction places a 0 in the 6502 carry flag. This is an important instruction to remember, and should always be present in single-byte addition operations.

Why is the CLC instruction so important? The answer lies in the structure of the 6502 ADC instruction. Remember, ADC means "add with carry." Whenever an addition is performed on the 6502, the result is set to ACCUMULATOR + OPERAND + CARRY.

Here's an example of what can go wrong when the programmer is not sure of the contents of the carry flag. Let's say the carry happens to be set to 1. Fred the careless programmer wants to add 1+1 to verify that the answer is indeed 2, so he writes the following code:

```
LDA #1
ADC #1
STA ANSWER
```

When Fred runs the program, he is astounded to find that 1 plus 1 is 3! If Fred had only inserted a simple CLC instruction, his life would have been much happier, as well as more accurate.

Suffice it to say that in any single byte addition operation, you should *always* clear the carry flag before the ADC instruction.

The third line adds 14 to the accumulator, giving a result of 37 ($25 hex), which is, of course, correct.

You can use any of the eight addressing modes with the ADC instruction. All produce the same results, they just get their data with different methods.

Flag-waving

Earlier I mentioned the flags altered by the ADC instruction. These are the sign, overflow, zero and carry flags.

The sign flag indicates the sign of the result. The contents of the accumulator's 7th bit are placed in this flag. If the flag is 0 after an add, the result is considered positive. A 1 in this flag indicates a negative result. See Issue # 72's *Boot Camp* for an in-depth discussion of the sign flag.

The overflow flag is set to the exclusive/or of Bits 6 and 7 of the result. The overflow flag is rarely used, but it's a good idea to know what happens to it during processing.

The zero flag is set to 1 if the result of the add was 0, and is set to 0 if the result was *not* 0.

The carry flag is set to 1 if the result of the add is greater than 255. This flag is important in multi-byte addition (for numbers greater than 255). We'll be examining multi-byte operations next issue.

All these flags are important in the computer's decision-making process. Depending on the result of an operation, the programmer can go to other parts of the program using comparison and branch instructions (similar to IF/THEN statements in BASIC). We will also cover these operations next issue.

Starting With Subtraction.

Now that we've covered simple addition, let's do a little subtraction. Subtraction is just as easy as addition, with a couple of simple differences. Shown below are the formats of the 6502 subtraction instruction, SBC (subtract with borrow). You will notice that the SBC has the same formats as the ADC instruction.

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

The SBC instruction subtracts the number at the memory location specified in the operand from the accumulator and places the result in the accumulator. Like the ADC instruction, the sign, overflow, zero and carry flags will be altered.

For the time being we'll work only with single-byte subtractions, since they're the easiest to understand. We will subtract 14 from 23 decimal and place the answer in a location called ANSWER. Here's the code needed for this operation:

```
10  LDA  #23      ; PUT 23 IN A
20  SEC           ; SET CARRY FOR SUB
30  SBC  #14      ; AND SUB 14 FROM IT!
40  STA  ANSWER   ; SAVE RESULT
```

The first line simply places the number 23 in the accumulator.

The second line introduces another new operation code, SEC (set carry). This instruction sets the carry flag to 1. Like the CLC instruction in single-byte additions, the SEC instruction is a must for all single-byte subtractions.

The SBC instruction is strange in that it subtracts the contents of the memory byte indicated in the operand and the complement of the carry flag from the accumulator, placing the result back in the accumulator. Here's an example. Let's say the accumulator contains 4 decimal, and we're subtracting 3 decimal from this. Assume the carry flag is clear (0). The computer will subtract 3 from 4, then subtract 1 from this (the complement of the carry flag), giving a result of 0.

By setting the carry to 1, we make sure that the subtraction of our two numbers is unaffected by the subtraction of the carry's complement, which in this case is 0. The carry flag is used as a borrow in subtraction and not necessary in single-byte operations.

The third line performs the subtraction. The result will be 23−14−0 or 9.

The last line of the program places the result in the location labeled ANSWER. The result will still be in the accumulator.

Like the ADC instruction, the SBC instruction works the same with all eight addressing modes available with the instruction. The SBC instruction affects the 6502 status flags in the same way as ADC.

Applying What We've Covered

We've now progressed to the point where we can write simple math programs using addition and subtraction. Let's write a program to solve the equation:

$$4+5+34-(8-7)=?$$

Unlike BASIC, we can't simply code this equation right into our computer. In assembly language, it's up to the programmer to figure out the procedure needed to obtain the result and code it.

Let's look at the equation shown above. In any mathematical equation, the expressions in parentheses must be solved before proceeding with the rest of the equation. If we simply solve the equation from left to right, we will get an incorrect answer:

$$4+5+34-8-7=28$$

In order to solve the equation correctly, we must solve it as follows:

$$(8-7) = 1$$
$$4+5+34-(1) = 42$$

Now that we know how to proceed, let's write a section of code to solve the equation:

```
10  *=   $0600
20  CLD            ; NO DECIMAL MODE!
30  LDA  #8        ; PUT 8 IN A
40  SEC            ; SET CARRY,
50  SBC  #7        ; SUBTRACT 7 FROM 8
60  STA  HOLD      ; AND SAVE RESULT
70  LDA  #4        ; NOW PUT 4 IN A
80  CLC            ; CLEAR CARRY,
90  ADC  #5        ; ADD 4 AND 5
0100  CLC           ; CLEAR CARRY AGAIN
0110  ADC  #34      ; ADD 34 TO LAST #
0120  SEC           ; SET CARRY
0130  SBC  HOLD     ; SUBTRACT EARLIER #
0140  STA  ANSWER   ; AND SAVE ANSWER!
0150  BRK           ; ALL DONE!
0155 ;
0160 HOLD   *=*+1 ; TEMP. HOLD AREA
0170 ANSWER *=*+1 ; FINAL RESULT
0180  .END
```

Line 10 tells the assembler to place the program at Location $0600, a safe location in computer memory.

Line 20 clears the decimal mode, to avoid any accidental BCD results.

Line 30 places the number 8 in the accumulator.

Line 40 sets the carry flag to get ready for a single-byte subtract.

Line 50 subtracts 7 from 8, leaving the result in the accumulator.

Line 60 stores the result of the expression in parentheses at a memory location called HOLD. This is done because we will need this number in a moment.

Line 70 places a 4 in the accumulator in order to start solving the first part of the equation.

Line 80 clears the carry flag to get ready for a single-byte add.

Line 90 adds 5 to the accumulator, leaving the result in the accumulator.

Line 100 clears the carry again for the next addition. In this case, the CLC is not necessary since we know the previous add did not exceed 255, but it's a good idea to get into the CLC habit.

Line 110 adds 34 to the accumulator, once again leaving the result in the accumulator.

Line 120 sets the carry flag for the next subtract operation.

Line 130 subtracts the result of the expression in parentheses (stored in HOLD) from the accumulator and gets the final result.

Line 140 places the final result in the memory location called ANSWER.

Line 150 breaks the program execution. At this point the accumulator should equal 42 decimal ($2A hex).

Line 160 and *170* set up the one-byte storage areas, HOLD and ANSWER. The assembler directive *=*+1 simply tells the assembler to reserve one byte for each label.

Line 180 tells the assembler that the end of the source code has been reached.

After this code is typed in and assembled into memory, execute the program from DEBUG mode with the command:

G 600

The program will execute very quickly and return with a screen similar to this:

```
0618     A=2A X=0F Y=6A P=31 S=00
```

Note that the accumulator contains $2A (42 decimal). This is the correct answer to our equation.

This example shows how you can perform simple add/subtraction operations in assembly language. Of course, we're limited to one-byte integers, but we'll soon exceed these limitations.

Until Next Time...

Try your own problems until you're proficient with the 6502 add-and-subtract operations. Try using the various addressing modes to see how they work. In order to learn assembly language (or any other language, for that matter), you'll have to roll up your sleeves and dig in.

Next issue will cover a lot of material, including the assembly equivalent of the BASIC IF/THEN statement, index register usage and multi-byte addition and subtraction.

continued from page 43

```
6120      BNE F.SEC    ;no.
6130      LDA DSBUF+125 ;get the link
6140      AND #3       ;hi
6150      STA SECH     ;get the link
6160      LDA DSBUF+126 ;lo
6170      STA SECL     ;at EOF?
6180      ORA SECH
6190      BNE C.READ   ;no.
6200 S.EOF LDY #136   ;signal error.
6210      RTS
6220 C.READ JSR UP.READ ;grab sector
6230      PHP          ;save status
6240      LDA DSBUF+127 ;get # data
6250      STA SECBYTES ;bytes.
6260      LDA DSBUF    ;get data byte
6270      PLP          ;pull status
6280      RTS          ;and leave.
6290 ;
6300 F.SEC INC SECL   ;sec=sec+1
6310      BNE FS.1
6320      INC SECH
6330 FS.1 JSR UP.READ ;grab sector
6340      PHP          ;save status
6350      LDA DSBUF    ;get data
6360      PLP          ;pull status
6370      RTS          ;and leave.
6380 ;
6390 UP.READ LDA SECCNT ;any sectors
6400      ORA SECCNT+1 ;left?
6410      BEQ S.EOF    ;no! Done.
6420      LDA SECCNT   ;subtract 1
6430      SEC          ;from the sector
6440      SBC #1       ;count.
6450      STA SECCNT
6460      LDA SECCNT+1
6470      SBC #0
6480      STA SECCNT+1
6490      JMP READIT   ;read sector.
6500 ;
6510 G_FILE LDA BYTCNT ;any bytes
6520      ORA BYTCNT+1 ;left?
6530      BEQ HEADERS ;no, get more
6540      LDA BYTCNT   ;subtract 1
6550      SEC          ;from the count
6560      SBC #1
6570      STA BYTCNT
6580      LDA BYTCNT+1
6590      SBC #0
6600      STA BYTCNT+1
6610      INC BYTES    ;up byte count
6620      JMP CIOGET   ;get the byte
6630 ;
6640 ;Send remaining instr bytes
6650 ;
6660 ANY_LEFT LDA BYTES ;any extra?
6670      BEQ N.LF     ;no.
6680      LDY #151     ;bad opcode.
6690      JSR HAVOP    ;format .BYTE
6700      JSR PRINT.IT ;send it out.
6710      LDA DTEMP    ;prepare next
6720      STA OP
6730      DEC BYTES    ;down count
6740      BNE ANY_LEFT ;and loop
6750 N.LF RTS          ;bye.
6760 H_ERR JMP INERR ;input error
6770 ;
6780 ;Pull file headers
6790 ;
6800 HEADERS JSR ANY_LEFT ;send extra
6810      JSR GET.TWO ;grab 2 more
6820      BMI H_ERR    ;oops.
6830      LDY #29      ;copy message
6840 C.ST LDA STEN,Y ;to buffer
6850      STA IBUF,Y
6860      DEY
6870      BPL C.ST
6880      JSR CHKFF    ;are they $FF's?
6890      BNE NOT.FF   ;no.
6900      JSR GET.TWO ;get 2 more
6910      BMI H_ERR    ;oops.
6920 NOT.FF LDA HXL   ;save the load
6930      STA ADDR     ;address and
6940      STA FR0      ;convert it to
6950      LDA HXH      ;hex for the
6960      STA ADDR+1   ;output file.
6970      STA FR0+1
6980      JSR BIN2HEX ;make hex.
6990      LDX #4       ;copy it in
7000      JSR ALL
7010      JSR GET.TWO ;grab 2 more
7020      BMI H_ERR    ;oops.
7030      LDA HXL      ;make this #
7040      STA FR0      ;into hex for
7050      SEC          ;output also.
7060      SBC ADDR     ;We also calc
7070      STA BYTCNT   ;the length of
7080      LDA HXH      ;the segment.
7090      STA FR0+1
7100      SBC ADDR+1
7110      STA BYTCNT+1
7120      INC BYTCNT   ;add 1 to the
7130      BNE N.OV     ;segment len.
7140      INC BYTCNT+1
7150 N.OV JSR BIN2HEX ;make hex
7160      LDX #16      ;pos 16
7170      JSR ALL      ;copy it
7180      JSR PUTCR    ;line feed
7190      PLA          ;pull return
7200      PLA
7210      JSR PRINT.IT ;print it
7220      JMP DISAMEM ;go again.
7230 ;
7240 ;Grab two file bytes
7250 ;
7260 GET.TWO JSR CIOGET
7270      BMI G2ERR
7280      STA HXL
7290      JSR CIOGET
7300      STA HXH
7310 G2ERR RTS
7320 ;
7330 ;Print The output
7340 ;
7350 PRINT.IT  PRINT  IBUF ;to E:
7360      LDY #0       ;wait 5 jiffies
7370      STY RTCLOK
7380 WT_LP LDA RTCLOK
7390      CMP #5
7400      BCC WT_LP
7410      LDA DSTFILE ;to E: only?
7420      CMP #'E
7430      BNE PR.LP    ;no.
7440      RTS          ;yes.
7450 PR.LP LDA IBUF,Y ;move iobuf to
7460      STA (BUFPNT),Y ;big buffer
7470      CMP #EOL
7480      BEQ PR.DN
7490      INY
7500      BNE PR.LP
7510 PR.DN INY         ;now we add Y+1
7520      TYA          ;to the pointer
7530      CLC          ;so we are ready
7540      ADC BUFPNT   ;next time
7550      STA BUFPNT
7560      LDA BUFPNT+1
7570      ADC #0
7580      STA BUFPNT+1
7590      INC LINES
7600      LDA LINES    ;get line count
7610      CMP #200     ;200 lines?
7620      BNE SND.LV   ;no.
7630 ;
7640 ;Send the buffer out
7650 ;
7660 SEND_OUT LDA LINES ;any lines?
7670      BNE SND.IT   ;yes.
7680 SND.LV RTS        ;so long.
7690 SND.IT LDA #11    ;put bytes
7700      LDX #$20
7710      STA ICCOM,X
7720      LDA # <BIGBUF ;from bigbuf
7730      STA ICBAL,X
7740      LDA # >BIGBUF
7750      STA ICBAH,X
7760      LDA BUFPNT   ;calc the len
7770      SEC          ;of the buffer
7780      SBC # <BIGBUF
7790      STA ICBLL,X
7800      LDA BUFPNT+1
7810      SBC # >BIGBUF
7820      STA ICBLH,X
7830      LDA # <BIGBUF ;reset the
7840      STA BUFPNT   ;pointers.
7850      STA LINES
7860      LDA # >BIGBUF
7870      STA BUFPNT+1
7880      JSR TEST_DRV ;test drives
7890      BNE SND.OUT ;not equal
7900       PRINT  INS_DST ;ask for
7910      JSR GET_RET ;disk swap.
7920      LDX #$20
7930 SND.OUT JSR CIOV ;write out
7940      BMI OUT_ERR ;oops.
7950      JSR TEST_DRV ;swap again?
7960      BNE SND.OV   ;no.
7970      LDA SFLAG    ;test flag
7980      BNE SND.OV   ;no swap!
7990       PRINT  INS_SRC ;prompt it
8000      JSR GET_RET
8010 SND.OV LDY #1    ;signal good.
8020      RTS
8030 ;
8040 ;Test srcdrv vs. dstdrv
8050 ;
8060 TEST_DRV LDA DESDRV ;get dest
8070      CMP SRCDRV   ;is = source?
8080      RTS          ;bye.
8090 ;
8100 ;Output error handler
8110 ;
8120 OUT_ERR JSR F_ERR ;format err#
8130       PRINT  OUTP ;print OUTPUT
8140      LDX #$20     ;close #2
8150      JSR CLOSE
8160       PRINT  IBUF ;print err#
8170       PRINT  ALL.DONE+10
8180      JSR GET_RET ;get return
8190      JMP INTRO    ;so long.
8200 ;
8210 ;Get a RETURN
8220 ;
8230 GET_RET JSR GETKEY ;get key
8240      CMP #EOL     ;is it EOL?
8250      BNE GET_RET ;no.
8260      LDX #$20
8270      RTS
8280 ;
8290 ;Check if HXL/HXH=$FF
8300 ;
8310 CHKFF LDA HXL    ;get byte 1
8320      CMP #$FF     ;is it $FF?
8330      BNE CHKLV    ;no.
8340      CMP HXH      ;is #2=#1?
8350 CHKLV RTS
```

LISTING 5: ASSEMBLY

```
0100 ;SAVE#D:DMPT4.M65
0110 ;-----------------------------
0120 ; Disk Master part 4
0130 ;
0140 ;These are the mnemonics for
0150 ;the disassembler. They are
0160 ;arranged here in alphabetical
0170 ;order.
0180 ;-----------------------------
0190 ;
0200 MNE .BYTE "ADCANDASLBCCBCS"
0210      .BYTE "BEQBITBMIBNEBPL"
0220      .BYTE "BRKBVCBVSCLCCLD"
0230      .BYTE "CLICLVCMPCPXCPY"
0240      .BYTE "DECDEXDEYEORINC"
0250      .BYTE "INXINYJMPJSRLDA"
0260      .BYTE "LDXLDYLSRNOPORA"
0270      .BYTE "PHAPHPPLAPLPROL"
0280      .BYTE "RORRTIRTSSBCSEC"
0290      .BYTE "SEDSEISTASTXSTY"
0300      .BYTE "TAXTAYTSXTXATXS"
0310      .BYTE "TYA"
0320 ;
0330 ;Opcodes
0340 ;
0350 OPCODE
0360      .BYTE $69,$65,$75,$6D ;ADC
0370      .BYTE $7D,$79,$61,$71
0380      .BYTE $29,$25,$35,$2D ;AND
0390      .BYTE $3D,$39,$21,$31
0400      .BYTE $0A,$06,$16,$0E ;ASL
0410      .BYTE $1E
0420      .BYTE $90    ;BCC
0430      .BYTE $B0    ;BCS
0440      .BYTE $F0    ;BEQ
0450      .BYTE $24,$2C ;BIT
0460      .BYTE $30    ;BMI
0470      .BYTE $D0    ;BNE
0480      .BYTE $10    ;BPL
0490      .BYTE $00    ;BRK
0500      .BYTE $50    ;BVC
0510      .BYTE $70    ;BVS
0520      .BYTE $18    ;CLC
0530      .BYTE $D8    ;CLD
0540      .BYTE $58    ;CLI
0550      .BYTE $B8    ;CLV
0560      .BYTE $C9,$C5,$D5,$CD ;CMP
0570      .BYTE $DD,$D9,$C1,$D1
0580      .BYTE $E0,$E4,$EC ;CPX
0590      .BYTE $C0,$C4,$CC ;CPY
0600      .BYTE $C6,$D6,$CE,$DE ;DEC
0610      .BYTE $CA    ;DEX
```

continued on page 82

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:BACK5P=126:CHK5
   UM=0:EDIT=0
GO 30 GOSUB 450:POSITION 10,6:? "Start or
   Continue? ";:GOSUB 500:? CHR$(A)
ZG 40 POSITION 10,8:? "FILENAME";:INPUT F
   $:POKE 752,1:? " "
FE 50 IF LEN(F$)<3 THEN POSITION 20,10:?
   " ":GOTO 40
MF 60 IF F$(1,2)<>"D:" THEN F1$="D:":F1$(
   3)=F$:GOTO 80
KL 70 F1$=F$
TM 80 IF CHR$(A)="S" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$:TRAP 110
HQ 100 FOR X=1 TO 16:GET #2,A:NEXT X:LINE
   =LINE+10:GOTO 100
WM 110 CLOSE #2:OPEN #2,9,0,F1$:GOTO 170
VT 120 TRAP 160:OPEN #2,4,0,F1$:GOSUB 440
   :POSITION 10,10:? "FILE ALREADY EXISTS
   !!":POKE 752,0
ZU 130 POSITION 10,12:? "ERASE IT? ";:GOS
   UB 500:POKE 752,1:? CHR$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
    CLOSE #2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
   HEN 130
BN 160 CLOSE #2:OPEN #2,8,0,F1$
IE 170 GOSUB 450:POSITION 10,1:? "NOW ON
   LINE: ";LINE:CHK5UM=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:CHK5UM=CHK5UM+BYTE*X:IF
    CHK5UM)9999 THEN CHK5UM=CHK5UM-10000
MS 230 NEXT X:CHK5UM=CHK5UM+LINE:IF CHK5U
   M)9999 THEN CHK5UM=CHK5UM-10000
IG 240 POSITION 12,X+2:POKE 752,0:? "CHEC
   K5UM: ";:L1=4:GOSUB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
SY 270 POSITION 22,X+2:? C;"   "
IL 280 IF C=CHK5UM THEN 300
DI 290 GOSUB 440:EDIT=1:CHK5UM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GOSUB 500:IF (A=ASC("Q") OR A=ASC(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACK5P AND (A(4
   8 OR A)57) THEN 320
DX 331 IF A=RETRN AND N$="" 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
   ACK5P) AND L=0 THEN 320
DH 350 IF A=RETRN THEN POKE 752,1:? " ":R
   ETURN
GG 360 IF A<>BACK5P THEN 400
SA 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
AS 380 N$=""
RE 390 ? CHR$(BACK5P);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WK 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHICS 0:END
YT 430 GOSUB 440:POSITION 10,10:? "NO SUC
   H FILE!":FOR X=1 TO 1000:NEXT X:CLOSE
   #2:GOTO 30
FD 440 POKE 710,48:SOUND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:SOUND 0,0,0,0:RETURN
MY 450 GRAPHICS 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 STEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 STEP 2:POKE DL+X,0
   :NEXT X
ZW 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
HZ 500 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1
   :RETURN
```

continued from page 25

```
5830 SHOH1 CMP JIFFY
5840      BEQ SHOH1
5850      STX BANKSEL
5860      LDX #PAGE3
5870      LDA JIFFY
5880 SHOH2 CMP JIFFY
5890      BEQ SHOH2
5900      STX BANKSEL
5910      LDX CH
5920      CPX #255
5930      BEQ SHOHALF
5940      LDX #MAIN
5950      STX BANKSEL
5960      JMP HOLDNOW
5970 * (S)how half, pages 2 & 4
5980 SHOSECHAF LDX #PAGE2
5990      LDA JIFFY
6000 SHOS1 CMP JIFFY
6010      BEQ SHOS1
6020      STX BANKSEL
6030      LDX #PAGE4
6040      LDA JIFFY
6050 SHOS2 CMP JIFFY
6060      BEQ SHOS2
6070      STX BANKSEL
6080      LDX CH
6090      CPX #255
6100      BEQ SHOSECHAF
6110      LDX #MAIN
6120      STX BANKSEL
6130      JMP HOLDNOW
6140 * Show all screens (F)astest
6150 FASTEST LDA #255
6160 FASTEST1 LDX #PAGE1
6170      STX BANKSEL
6180      LDX #PAGE2
6190      STX BANKSEL
6200      LDX #PAGE3
6210      STX BANKSEL
6220      LDX #PAGE4
6230      STX BANKSEL
6240      CMP CH
6250      BEQ FASTEST1
6260      LDX #MAIN
6270      STX BANKSEL
6280      JMP HOLDNOW
6290      *=  $02E0
6300      .WORD INIZ
```

LISTING 3: ASSEMBLY

```
0 ;SAVE#D:SYSEQU.M65
10        .PAGE "OSS SYSTEM EQUATES FOR
 ATARI"
20 ;
30 ;  FILE = #DN:SYSEQU.ASM
40 ;
50 ;
60 ; I/O CONTROL BLOCK EQUATES
70 ;
80 SAVEPC =  *        ; SAVE CURRENT OR
G
90 ;
0100      *=  $0340   ;START OF SYSTEM
IOCBS
0110 IOCB
0120 ;
0130 ICHID *= *+1     ;DEVICE HANDLER I
S (SET BY OS)
0140 ICDNO *= *+1     ;DEVICE NUMBER (S
ET BY OS)
0150 ICCOM *= *+1     ;I/O COMMAND
0160 ICSTA *= *+1     ;I/O STATUS
0170 ICBADR *= *+2    ;BUFFER ADDRESS
0180 ICPUT *= *+2     ;DH PUT ROUTINE (
ADR-1)
0190 ICBLEN *= *+2    ;BUFFER LENGTH
0200 ICAUX1 *= *+1    ;AUX 1
0210 ICAUX2 *= *+1    ;AUX 2
0220 ICAUX3 *= *+1    ;AUX 3
0230 ICAUX4 *= *+1    ;AUX 4
0240 ICAUX5 *= *+1    ;AUX 5
0250 ICAUX6 *= *+1    ;AUX 6
0260 ;
0270 IOCBLEN = *-IOCB ;LENGTH OF ONE I
OCB
0280 ;
0290 ; IOCB COMMAND VALUE EQUATES
0300 ;
0310 COPN =  3        ;OPEN
0320 CGBINR = 7       ;GET BINARY RECOR
D
0330 CGTXTR = 5       ;GET TEXT RECORD
0340 CPBINR = 11      ;PUT BINARY RECOR
D
0350 CPTXTR = 9       ;PUT TEXT RECORD
0360 CCLOSE = 12      ;CLOSE
0370 CSTAT = 13       ;GET STATUS
0380 ;
0390 ; DEVICE DEPENDENT COMMAND EQUATE
S FOR FILE MANAGER
0400 ;
0410 CREN =  32       ;RENAME
0420 CERA =  33       ;ERASE
0430 CPRO =  35       ;PROTECT
0440 CUNP =  36       ;UNPROTECT
0450 CPOINT = 37      ;POINT
0460 CNOTE = 38       ;NOTE
0470 ;
0480 ; AUX1 VALUES REQD FOR OPEN
0490 ;
0500 OPIN =  4        ;OPEN INPUT
0510 OPOUT = 8        ;OPEN OUTPUT
0520 OPUPD = 12       ;OPEN UPDATE
0530 OPAPND = 9       ;OPEN APPEND
0540 OPDIR = 6        ;OPEN DIRECTORY
0550 ;
0560      .PAGE
0570 ;
0580 ;    EXECUTE FLAG DEFINES
0590 ;
0600 EXCYES = $80     ; EXECUTE IN PROG
RESS
0610 EXCSCR = $40     ; ECHO EXCUTE INP
UT TO SCREEN
0620 EXCNEW = $10     ; EXECUTE START U
P MODE
0630 EXCSUP = $20     ; COLD START EXEC
 FLAG
0640 ;
0650 ; MISC ADDRESS EQUATES
0660 ;
0670 CPALOC = $0A     ; POINTER TO CP/A
0680 WARMST = $08     ; WAR, START (0=C
OLD)
0690 MEMLO = $02E7    ; AVAIL MEM (LOW)
 PTR
0700 MEMTOP = $02E5   ; AVAIL MEM (HIGH
) PTR
0710 APPMHI = $0E     ; UPPER LIMIT OF
APPLICATION MEMORY
0720 INITADR = $02E2 ; ATARI LOAD/INIT
 ADR
0730 GOADR = $02E0    ; ATARI LOAD/GO A
DR
0740 CARTLOC = $BFFA ; CARTRIDGE RUN L
OCATION
0750 CIO =    $E456   ;CIO ENTRY ADR
0760 EOL =    $9B     ; END OF LINE CHA
R
0770 ;
0780 ;  CP/A FUNCTION AND VALUE DISPLA
CEMSNT
0790 ;      (INDIRECT THROUGH CPALOC)
0800 ;            IE. (CPALOC),Y
0810 ;
0820 CPGNFN = 3       ; GET NEXT FILE N
AME
0830 CPDFDV = $07     ; DEFAULT DRIVE (
3 BYTES)
0840 CPBUFP = $0A     ; CMD BUFF NEXT C
HAR POINTR (1 BYTE)
0850 CPEXFL = $0B     ; EXECUTE FLAG
0860 CPEXFN = $0C     ; EXECUTE FILE NA
ME (16 BYTES)
0870 CPEXNP = $1C     ; EXECUTE NOTE/PO
INT VALUES
0880 CPFNAM = $21     ; FILENAME BUFFER
0890 RUNLOC = $3D     ; CP/A LOAD/RUN A
DR
0900 CPCMDB = $3F     ; COMMAND BUFFER
(60 BYTES)
0910 CPCMDGO = $F3
0920 ;
0930      *=  SAVEPC  ; RESTORE PC
0940 ;
0950 RANDOM = 53770
0960 SAVMSC = $58
0970 CONSOL = 53279
0980 COLOR0 = 708
0990 CRSINH = 752
1000 BOOT? = 9
1010 DINDEX = 87
1020 HATABS = 794
1030 CH   =   764
1040 STRIG0 = 644
1050 STICK0 = 632
1060 POKMSK = $10
```

continued on page 69

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. REALITY DUNGEON.	26.95
BEYOND CASTLE WOLF	14.95
BISMARK	26.95
BOP & WRESTLE	26.95
BORDINO:1812	26.95
BOULDERDASH CONSTR.SET	17.95
BRUCE LEE	17.95
CASTLE WOLFENSTEIN	14.95
DALLAS QUEST	7.95
D-BUG	7.95
F-15 STRIKE EAGLE	31.50
FIGHT NIGHT	17.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS	22.50
GUNSLINGER	26.95
HARD HAT MAC	7.95
JAWBREAKER	9.95
KARATEKA	13.50
KNICKERBOCKERS	13.50
KORONIS RIFT	13.50
LAST V-8	8.95
LEADERBOARD	13.50
MAIL ORDER MONSTERS	13.50
MICROLEAGUE BASEBALL	35.95
MONTEZUMA'S REVENGE	14.95
MOUSEQUEST	17.95
MOON SHUTTLE	7.95
NINJA	8.95
OIL'S WELL	9.95
O'RILEY'S MINE	9.95
PIRATES OF BARB. COAST	22.50
PREPPIE I & II	9.95
RESCUE ON FRACTALAS	13.50
SILENT SERVICE	31.50
SPEEDKING	8.95
SPIDERMAN	5.35
SPITFIRE 40	31.50
STARFLEET I	44.95
SPY VS. SPY III	17.95
STOCKMARKET	22.50
STRIP POKER	26.95
SUMMER GAMES	17.95
TAX DODGE	9.95
THE HULK	5.35
TOMAHAWK (64K)	26.95
TOP GUNNER	17.95
TOUCHDOWN FOOTBALL	13.50
TRAILBLAZER	26.95
UNIVERSE	44.95
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
DOS 2.5	7.95
DOS XE	10.00
DISK I/O	26.95
KYAN PASCAL	62.95
LIGHTSPEED C	35.95
LOGO	19.95
MAC/65	71.95
MAC/65 TOOLKIT	26.95
MACRO ASSEMBLER	22.50
PILOT	19.95
SPARTA DOS X	71.95

PRODUCTIVITY

ANIMATION STATION	89.95
ATARIWRITER	29.95
ATARIWRITER (CART ONLY)	19.95
ATARIWRITER+	39.95
ATARI BOOKKEEPER	24.95
ATARI MUSIC II	14.95
AWARDWARE (1050)	13.50
BANK STREET WRITER	14.95
BLAZING PADDLES	31.50
CELEBRITY COOKBOOK	26.95
COMPUTE YOUR ROOTS	35.95
DATAMANAGER	17.95
FAMILY FINANCE	6.95
GUITAR WIZARD	26.95
HOME ACCOUNTANT	19.95
HOME FILING MANAGER.	6.95
HOMEPAK	24.95
INVENTORY MASTER	80.95
LETTER WIZARD	29.95
MUSIC CONSTRUCTION SET	13.50
NEWSROOM (1050 - 64K)	44.95
NEWS STATION	26.95
NEWS STA. COMPANION.	26.95
PAGE DESIGNER	26.95
PRINT POWER (1050)	13.50
PRINTKIT (1050)	13.50
PRINTSHOP	34.95
P.S. COMPANION (64K)	24.95
P.S.GRAPHICS LIBRARY 1	17.95
P.S.GRAPHICS LIBRARY 2	17.95
P.S.GRAPHICS LIBRARY 3	17.95
PROOF READER	17.95
PUBLISHING PRO	35.95
RUBBER STAMP	26.95
SYNTREND (130XE)	35.95
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
MASTER TYPE	14.95
STATES AND CAPITALS	9.95
TOUCH TYPING	9.95
CBS (AGE 3-6):	
ASTROGROVER	8.95
BIG BIRD SPEC DELIVE	8.95
ERNIE'S MAGIC SHAPE.	8.95
DESIGNWARE:	
MATHMAZE (6-11)	35.95
MISSION ALGEBRA (13+)	35.95
SPELLICOPTER (6-11).	35.95
TINK TONK (AGE 4-6):	
ABC'S	8.95
COUNT AND ADD	8.95
SMART THINKER	8.95
SPELLING	8.95
SUBTRACTION	8.95
THINKING SKILLS	8.95
ALL 6 TINK TONKS	39.95
UNICORN:	
10 LITTLE ROBOTS (PRE-SCHOOL)	26.95
FUN BUNCH (6-ADULT)	26.95
RACECAR RITHMETIC (AGE 6+)	26.95
WEEKLY READER (PRE-SCHOOL):	
STICKY BEAR SHAPES	26.95
STICKY BEAR NUMBERS.	26.95
STICKY BEAR ABC'S	26.96
STICKY BEAR OPPOSITE	26.95
SB BASKET BOUNCE	26.95
STICKY BEAR BOP	26.95
RUN FOR IT	26.95
PIC BUILDER	26.95

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 (REQ. LIGHT GUN)	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
CRIME BUSTER (REQ. LIGHT GUN)	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 THE EAGLES NEST	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
POLE POSITION	19.95
POPEYE	14.95
Q-BERT	14.95
QIX	14.95
RESCUE ON FRACTALAS	19.95
RETURN OF THE JEDI	14.95
ROBOTRON:2084	19.95
SKY WRITER	14.95
SLIME (400/800)	9.95
SPACE INVADERS	14.95
STAR RAIDERS	5.00
STAR RAIDERS II	19.95
SUBMARINEE COMMANDER	14.95
SUMMER GAMES (XL/XE)	24.95
SUPER BREAKOUT	9.95
SUPER COBRA	14.95
THUNDERFOX	19.95
TRACK & FIELD	24.95
TURMOIL	9.95
WIZARD OF WOR	5.00

PRODUCTIVITY

ATARIWRITER	19.95
MICROFILER	22.50

EDUCATION

ATARILAB STARTER SET	29.95
FUN WITH ART	14.95
MATH ENCOUNTERS	9.95
FISHER PRICE (PRE SCHOOL):	
DANCE FANTASY	8.95
LINKING LOGIC	8.95
LOGIC LEVELS	8.95
MEMORY MANOR	8.95
SPINNAKER (AGE 3-10):	
ALF IN COLOR CAVES	9.95
ALPHABET ZOO	9.95
DELTA DRAWING	9.95
FACEMAKER	9.95
KIDS ON KEYS	9.95
KINDERCOMP	9.95
(AGE 7 - ADULT):	
ADV.CREATOR (400/800)	9.95
FRACTION FEVER	9.95

SUPER SPECIALS
RECONDITIONED ATARI MERCHANDISE
30 DAY WARRANTY

800 (48K) COMPUTER $79.95

400 (16K) COMPUTER $29.95

SPACE AGE JOYSTICK $5.00.

ATARI TRACKBALL $9.95

1030 MODEM WITH EXPRESS! $24.95

1010 PROGRAM RECORDER $29.95

1020 COLOR PRINTER/PLOTTER $19.95
40 COLUMNS WIDE
(new in box)

ATARI BOOKKEEPER $14.95 - NO BOX

ATARI NUMERIC KEYPAD $7.95

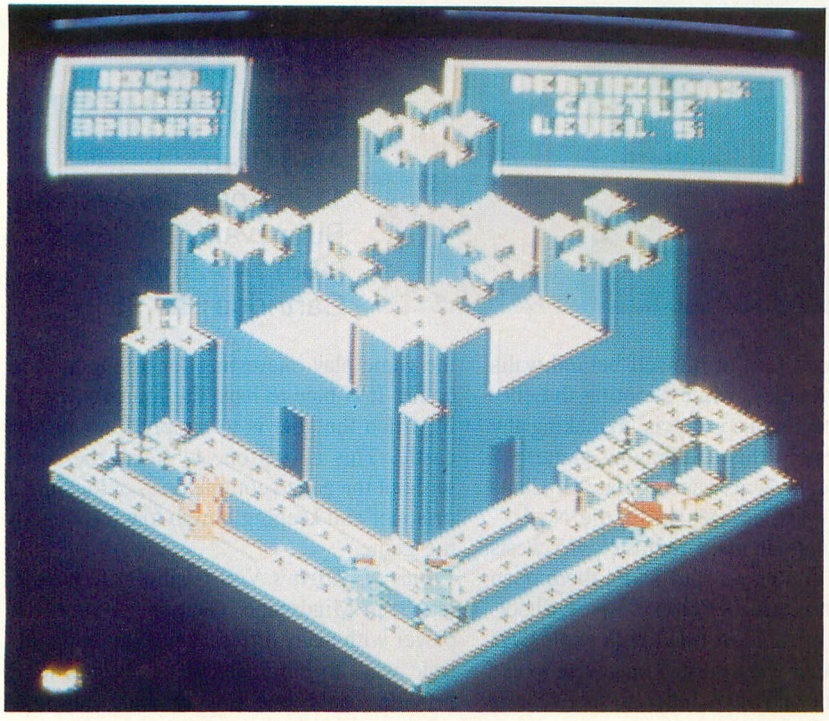
DISKETTES AS LOW AS 20 CENTS
10 FOR $4.00
100 FOR $29.95
1000 FOR $200
MOST ARE UNNOTCHED WITH OLD SOFTWARE

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

Phone orders accepted **TUESDAY THROUGH FRIDAY** from 10:00 am to 6:00 pm PST.
We carry a 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 #110 ON READER SERVICE CARD.



CRYSTAL CASTLES

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
Cartridge $29.95

Reviewed by Matthew J.W. Ratcliff

Crystal Castles is a challenging 3-D maze game that provides a lot of entertaining playtime for the novice and expert. The scenario is similar to the classic *Pac-Man*, but with a three-dimensional twist that adds a lot of depth to the game.

Bentley Bear must navigate, with the assistance of a joystick, an endless maze of alleys, stairways and elevators. These castles are viewed from a two-thirds overhead view, similar to the playfield of *Zaxxon* or *Desert Falcon*. However, the scenes in *Crystal Castles* are mazes that do not scroll. Each display is a unique image. Bentley can run and jump, picking up diamonds, rubies and pearls for points along the way.

Of course, there are the usual bad guys to overcome. Bentley must avoid the Crystal Balls, Nasty Trees, Gem Eaters, Swarms of Bees, Ghosts, Skeletons and the wicked Witch Berthilda. Most of the foes seem fairly passive. Evading the bees and Berthilda can be quite difficult, however. This certainly makes for a more sophisticated game than the relentless "ghosts" in *Pac-Man*. On some screens Bentley can pick up a magic hat, making him invincible for a short period of time. It allows Bentley to make short work of the remaining gems on a nearly completed screen.

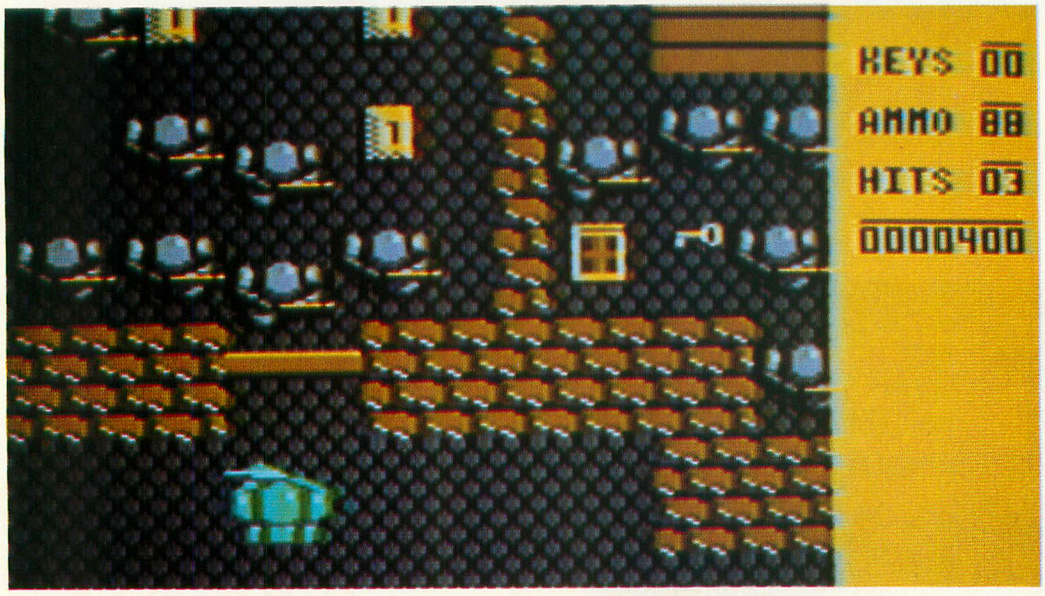
The graphics in *Crystal Castles* are well done, but the sound effects are not terribly sophisticated.

The player moves the joystick north, south, east and west to control Bentley Bear. The bear moves diagonally on the screen in response to these controls, since the mazes are oriented in this manner. It takes a bit of practice to get accustomed to this control, but is not difficult to master.

There are ten levels to this game, with four castles in each, except the last. On the tenth level there is only one castle to conquer. The game ends when Bentley loses all his lives, or picks up the last gem on the final level. This game can actually be completed by playing through all ten levels. But, with 37 different castles to overcome, it will take some serious playtime.

When the lower levels of *Crystal Castles* have been mastered, the player need not replay them over and over again. Unlike *Pac-Man*, where the user may select a starting level, *Crystal Castles* allows the player to use secret warps. This adds an element of mystery and surprise to the game. If Bentley makes just the right jump, at just the right location, he can warp to a higher level in the game. This warp earns the player extra lives and bonus points. Sometimes the magic hat is required to complete this maneuver. This eliminates the frustration of having to replay screens you already know by heart.

Crystal Castles is not a difficult game to play. My four-year-old son can play through the first two game screens. However, at higher levels the playscreens become increasingly more complicated. More stairways, elevators and tunnels make the going tough. *Crystal Castles* meets the needs of seasoned arcade-game players, while allowing the complete novice the opportunity to make good progress in the game as well. This is a new generation of *Pac-Man*, with plenty of entertaining challenges for all levels of game play.



INTO THE EAGLE'S NEST

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
Cartridge $29.95

Reviewed by Matthew J.W. Ratcliff

Into the Eagle's Nest is the finest game released by Atari in years. It provides some of the most exquisite graphics, superior sound effects and impressive playability on a game cartridge now available for the Atari 8-bit computers.

The setting is a World War II Nazi fortress called the Eagle's Nest. Three allied soldiers have preceded you, placing explosives on all four floors of the castle. Each level is huge and will take quite some time to conquer. Your allies have been captured and taken prisoner. At the outset, one of three rescue missions may be selected. The fourth and final objective is to blow up the castle.

This game is played from an overhead view. The main character, dressed in green battle fatigues, is controlled with a joystick. He can be moved in four basic directions. The screen scrolls smoothly as he battles his way through the rooms. He can be maneuvered to pick up keys to orange locked doors. The gray doors can be shot open. Each room and corridor presents more Nazis, dressed in drab gray, that must be shot twice to be obliterated.

The player accumulates "hit points" as he is shot at by the opposing soldiers. The player is dead after 50 hits and the game ends. Cold food may be found in some of the rooms. Moving over the food, the solder will eat it and "heal" some of his hit points. Sometimes, a first aid kit can be found to eliminate all current hits.

Only 99 rounds of ordnance can be carried at a time, but there is plenty of that to be found. Should the player run out of ammo, it is wise to pick up more quickly—there are always more Nazis around the next corner. The player shoots from his right shoulder, so he can "hide" part of himself behind walls or other obstacles, while shooting the enemies. This technique is crucial to making any real progress in this challenging game.

The player may also accumulate pendants and jewels—artifacts stolen by the Nazis—for more points. In many rooms closed chests are found. The player may shoot them open and find more treasures or explosives. Shooting into an open chest filled with explosives is always fatal. Generally, it is unwise to use valuable time and ammunition to shoot chests in hopes of accumulating more treasures, since these are not essential to winning the game.

The player must locate and activate the detonator by shooting it, to safely blast each level of the fortress, in the fourth scenario. Besides shooting doors and Nazi soldiers, the player may also shoot his way past boulders that block some rooms. Elevator passes are required to use the lifts to move on to a different floor of the castle.

In the rescue missions, the player must locate one of the captured prisoners and escort him to the safety of the storage room, where the game begins. The rescued prisoner is somewhat dazed by all the commotion and does not always cooperate. A shot in his general direction usually brings him to his senses and gets him back in gear. From the documentation it is apparent that the castle must be destroyed after the prisoner is rescued, but I haven't been able to make it this far through the game yet.

This is a superb action/graphics adventure game. It is extremely challenging, but not to the point of total frustration. It is one of the few games that can actually be played to completion. The four different scenarios will keep the game play interesting for a long time to come. It is nice to have the fourth mission, in which the goal is to simply blow up the castle. This allows one to finish the game without the added complexities of locating a prisoner and keeping him in tow throughout the game.

Into the Eagle's Nest is similar to *Gauntlet*, from Mindscape. Gauntlet presents the player with many different enemies and multiple ways to eliminate them. *Eagle's Nest* provides only one enemy and only one method of elimination. I don't find this a flaw however, since *Eagle's Nest* plays faster, and it is easier to master the game's controls. This allows the player to concentrate more on the adventure than the mechanics of manipulating the main character.

Into the Eagle's Nest has one particularly annoying "feature." At the end of each game, where a high score is achieved, the user is allowed to enter his name from joystick controls. This requires ten joystick-button presses, plus several more to start the next game. The names are not stored to disk, nor are they saved in battery-backed cartridge RAM, similar to *The Legend of Zelda* for the Nintendo Entertainment System. Therefore, this feature is of no use unless you leave your computer on for long periods of time or have a lot of game tournaments with friends. At the name entry screen, the Start key should send control directly to the next game; it doesn't.

Into the Eagle's Nest is a refreshing new offering that marks the beginning of the new gaming age from Atari. I think we can expect more original titles with top-notch graphics, sound and playability in the near future. *Into the Eagle's Nest* is a stellar game, one that I have found highly addictive.

continued from page 65

```
1070 RAMTOP = $6A
1080 SDMCTL = $022F
1090 STACK = $0100
1100 *
1110 IRQEN = $D20E
1120 NMIEN = $D40E
1130 SKCTL = $D20F
1140 DMACTL = $D400
1150 KBCODE = $D209
1160 POTGO = $D20B
1170 AUDCTL = $D208
1180 AUDF3 = $D204
1190 AUDC3 = $D205
1200 PACTL = $D302
1210 PORTA = $D300
1220 PORTB = $D301
1230 *
1240 * Timer counter:
1250 JIFFY = 20       ; 60Hz clock
1260 FILENAME = $0400 ; cassette buffe
r at $400
1270 ; will be used as FILENAME buffer
 for
1280 ; file i/o oriented operations.
1290 DLIST = $0230    ; display list po
inter
```

LISTING 4: ASSEMBLY

```
0 ;SAVE#D:GRAPHICS.M65
10 ;
20 ; GRAPHICS g
30 ;
40 ; ENTRY: A-REG CONTAINS GRAPHICS MO
DE 'G'
50 ;  EXIT: Y-REG HAS COMPLETION STATU
S
60 ;
70 SNAME .BYTE "S:",155,0
80 GRAFIX
90        PHA          ;SAVE 'G'
0100      LDX #6*$10   ;FILE 6
0110      LDA #CCLOSE
0120      STA ICCOM,X
0130      JSR CIO      ;FIRST WE MUST CL
OSE FILE #6
0140 ;WE IGNORE ANY ERRORS FROM THE CL
OSE
0150 ;
0160      LDX #6*$10   ;AGAIN, FILE 6
0170      LDA #COPN    ;WE WILL OPEN THI
S FILE
0180      STA ICCOM,X
0190      LDA # <SNAME
0200      STA ICBADR,X ;WE USE THE FILE
 NAME "S:"
0210      LDA # >SNAME
0220      STA ICBADR+1,X ;BY POINTING A
T IT
0230 ;
0240 ;ALL IS SET UP FOR OPEN, NOW
0250 ;WE TELL CIO (& S:) WHAT KIND OF
OPEN
0260 ;
0270      PLA          ;OUR SAVED 'G' GR
APHICS MODE
0280      STA ICAUX2,X ;IS GIVEN TO 'S:
'
0290 ; (NOTE THAT S: IGNORES UPPER BIT
S OF AUX2)
0300      AND #$F0     ;NOW WE GET THE U
PPER BITS
0310      EOR #$10     ;AND FLIP BIT 4
0320 ; (S: EXPECTS THIS BIT TO BE INVE
RTED
0330 ;FROM WHAT NORMAL BASIC USAGE IS)
0340      ORA #$0C     ;ALLOW READ & WRI
TE ACCESS FOR CIO
0350      STA ICAUX1,X ;FOR CIO AND S:
0360      JSR CIO      ;OPEN S:
0370      RTS          :GRAPHICS MODE 'G
' NOW OPEN
0380      .MACRO GRAPHICS
0390        .IF %0<>1
0400        .ERROR "GRAPHICS LEN"
0410        .ENDIF
0420        .IF %1>255
0430        LDA %1
0440        .ELSE
0450        LDA #%1
0460        .ENDIF
0470      JSR GRAFIX
0480      .ENDM
```

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

THE ANALOG #74 DISKETTE CONTAINS 21 MAGAZINE FILES. THEY ARE LISTED BELOW:

SIDE 1:

FILENAME.EXT	LANG.	LOAD	ARTICLE NAME
DISKMAST.OBJ	ML	(#3)	DISK MASTER
CSETDISP.OBJ	ML	(#3)	CHARACTER SET DISPLAY
AUTOMAKE.BAS	BASIC	LOAD	AUTORUN.SYS IN BASIC
CHAOS .BAS	BASIC	LOAD	CHAOS
DEGASVUF.OBJ	ML	(#3)	DEGAS VIEW
DEGASVUF.OBJ	MAC/65	LOAD	DEGAS VIEW SOURCE
GRAPHICS.M65	MAC/65	LOAD	DEGAS VIEW SOURCE
SYSEQU .M65	MAC/65	LOAD	DEGAS VIEW SOURCE
IOMAC .LIB	MAC/65	LOAD	DEGAS VIEW SOURCE
MAZERUNR.OBJ	ML	(#3)	MAZERUNNER
MAZERUNR.M65	MAC/65	LOAD	MAZERUNNER SOURCE
MLEDITOR.BAS	BASIC	LOAD	M/L EDITOR
EDITORII.LST	BASIC	ENTER	BASIC EDITOR II

SIDE 2:

FILENAME.EXT	LANG.	LOAD	ARTICLE NAME
DMPT1 .M65	MAC/65	LOAD	DISK MASTER SOURCE
DMPT2 .M65	MAC/65	LOAD	DISK MASTER SOURCE
DMPT3 .M65	MAC/65	LOAD	DISK MASTER SOURCE
DMPT4 .M65	MAC/65	LOAD	DISK MASTER SOURCE
DMPT5 .M65	MAC/65	LOAD	DISK MASTER SOURCE
SYSEQU .M65	MAC/65	LOAD	DISK MASTER SOURCE
CSETDISP.ACT	ACTION!	(#1)	CHAR. SET DISP. SOURCE
CSET2 .ACT	ACTION!	(#1)	CHAR. SET DISP. SOURCE

TO LOAD YOUR ANALOG DISK

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

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

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

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

LOADING NOTES

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

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

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,
127,36,597,230,256,182,430,168,5315
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
```

END USER

by Arthur Leyenberger

I have talked about it before, and I'm sure I will talk about it again. Technology! Our world is surrounded by, and couldn't exist without, the wonders that have been made possible by modern technology.

I remember reading a book in the late 1970s called *Computer Lib* by Ted Nelson. It contained all kinds of interesting information about computers, how they worked and how they were and would be used. I was fascinated by the future of computers. He was really talking about technology.

I recall one of the topics that was especially futuristic included the use of computers in banking. This was many years before the advent of electronic funds transfer (EFT) and automated teller machines. I could barely imagine what it would be like to have money transferred electronically from one computer to another.

At the time I thought I wouldn't want my bank account to use this new technology. I feared that one little computer mistake could wipe out my savings or cause unnecessary hassles trying to correct it. Now, of course, I take EFT for granted. Twice a month my paycheck is automatically transferred from the company's bank computer to my bank's computer. I rarely give it a second thought.

The same is true for automated tellers. The convenience and accessibility of these machines have changed my life. No longer do I have to make special arrangements to have access to my own money (i.e. banking hours). In fact, it seems so natural. However, I still refuse to deposit money into my account with one of these machines.

We are touched by technology in so many ways. Even the car I drive is affected. It would be impossible to have a car that had both good performance and good gas mileage without a computer-controlled engine.

The most obvious example of technology is the computer that we all use and (usually) love. It still provides plenty of power for all kinds of tasks. One of the reasons many Atari users refuse to upgrade to the ST is that their 8-bit machine can run hundreds of programs; most users have not exhausted the potential of their computer. That is as much a testament to the power of the 8-bit Atari as it is to the users.

New Disk Technology

Another improvement in technology is just around the corner and will dramatically affect the way we use computers. A new storage technology has been developed that will revolutionize the design of future computers. It uses a combination of magnetic and optical mediums to create a floppy disk that can store 20 megabytes of data.

The original Atari 810 disk drives could store only 90K of data on a 5¼-inch floppy disk. This was because they used only one side of the disk and a low-density data format. Over the years a number of new disk drives were introduced by Indus, Astra and others that used either twice the density or both sides of the disk to store data. Double-sided, double-density disks, like those used on IBM PCs and clones, can store 360K of data.

With the arrival of the AT and DOS 3.0, IBM introduced 1.2-megabyte floppy disks. Later, when 3½-inch disks became available, two storage capacities appeared: 720K and 1.44 megabytes. It wasn't until the introduction of CD-ROM (compact disc read-only

memory) that a significant increase in removable storage capacity was attained.

Most of us are now familiar with the new recording media used for music called the compact disc (CD). These laser "records" can contain up to 73 minutes of pop, click and static-free music with a fantastic signal-to-noise ratio and dynamic range. These same discs can also contain an incredible amount of data. Up to 550 megabytes of data can be stored on one side of a CD. That's equivalent to the storage capacity of over 6,000 of the original Atari 810 floppy disks.

Read-only memory (ROM) is not a new concept, either, and is one that is familiar to most computer users. The information in the ROMs is permanent and can only be "read" by the computer. All computers, including Atari 8-bit machines, incorporate ROMs that contain the necessary "startup" information the computer needs to perform its self-check and access disk drives.

The storage medium for digital audio discs and CD-ROMs is exactly the same. They use similar players in which a laser shines a beam under the disc and detects the presence or absence of microscopic pits on the disc surface. Using a binary code common to all computers, a pit is a 0 and the absence of a pit is a 1. In one case the data represents digitized music, and in the other it can represent programs, databases, dictionaries, encyclopedias or any other collection of information.

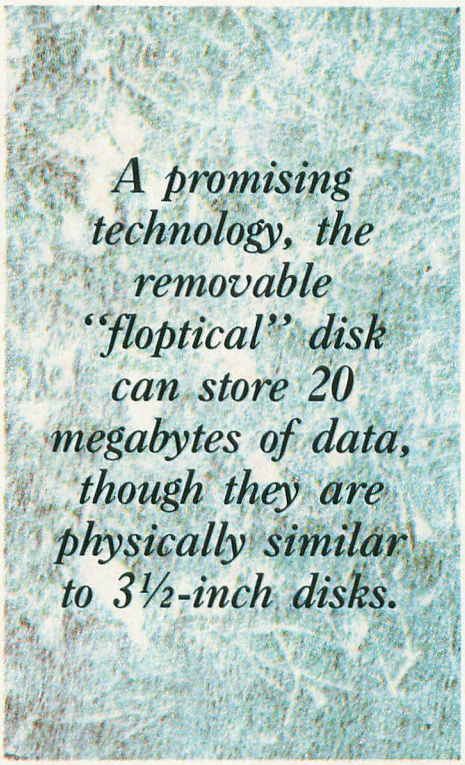
CD-ROMs are perfect for storing large amounts of information that will never change. However, a computer disk drive needs the ability to write as well as read. Two emerging technologies provide a solution to that need: WORM drives and floptical disks.

WORM means "write once, read many." Data is permanently written to a disk by means of a laser that burns small holes into the optical disk. However, this can happen only once and is not practical when the stored data may change. WORM drives have been around for several years but have not caught on as a replacement for floppy or hard disks.

A more promising technology has been created by Insite Peripherals under the guidance of Jim Shugart (one of the original designers of the 5¼-inch floppy disk). Each removable "floptical" disk can store 20 megabytes of data. Physically, they are similar to 3½-inch, high-density, 1.44-megabyte disks.

The difference, however, is in the method used to store data: a combination of optical servo and magnetic recording technologies. The 3½-inch magnetic disk is inscribed with concentric optical tracks at 20 micron intervals. A photo-detector (infrared LED) is used to guide the magnetic read/write head resulting in more precise storage of the data.

This arrangement allows a much higher track density compared with standard drives (1,250 vs. 135 tracks per inch). This means that more data can be packed onto the disk. The disk spins at 720 rpm, the servo-controlled head assembly is self-aligning, and the seek time is said to be 65 milliseconds. This access speed is typical for a 20-megabyte hard disk used on many PCs and compatibles.



Insite Peripherals will be selling their new optical-magnetic drives primarily to manufacturers who may then incorporate them into other products. Initially the drives will sell for about $250 and the special disks for under $10. You can probably expect to see some products containing floptical disk drives by the end of the year.

Imagine the possibilities! What if Atari were to use one of these new disk drives in the much talked-about portable ST? Instead of containing a heavy, expensive and power-hungry hard disk, the ST laptop would be an innovative computer and be much more usable. It wouldn't be the first time Atari introduced a powerful yet affordable computer. Let's hope they can do it again.

Rumor Mill

As of this writing, Atari may be preparing to shed the burden of its Federated Stores. As you probably know, Atari purchased the Federated Electronics chain a couple of years ago, hoping to improve their distribution channel for their computer and video game products. However, since the consumer electronics stores were bought, Atari has had nothing but financial and operational headaches with the stores.

The Federated group of stores has shown continued losses which have affected the bottom line of Atari financials. Atari claims that the stores were overvalued when they were purchased which led Atari to pay an inflated price for the stores. The outcome of this matter will be decided in the courts.

Another problem has stemmed from Atari allegedly playing favorites with their own stores. Some dealers claim that the Federated stores have been able to purchase products at lower prices than independent dealers. Further, when new products or upgraded products became available, the Federated stores allegedly were the first to receive them.

A specific example of this supposedly occurred when Atari began including double-sided drives in the 520STFM computer. Other dealers around the country apparently continued to get the older machines while Atari's own stores got the new models.

Whether these allegations are true or not, it does seem that the Federated stores have been, if nothing else, a thorn in Atari's corporate side. Perhaps it will be for the best if Atari decides to dump the stores. At least maybe then, Atari can get back to the business of making and selling computers. And with Atari's scarce resources, they can't afford to spread themselves too thin.

Arthur Leyenberger is a computer analyst and freelance writer who works out of his home in New Jersey. He can be reached on CompuServe at 71266,46 or on DELPHI as ARTL.

Mazeru

nner

by Matt Fruin

Left over from ancient days, the maze was always a great place to play and explore. At least it was until you and your friend, Velbert, got lost. With night coming on, you and Velbert grew more and more confused. Suddenly, the huge stone blocks comprising the maze rearranged themselves, shifting into new configurations. Far off in the distance you heard the scratching sounds of something big wandering the maze. Too late you remembered the legend of the monsters, the spirits of those who died in the maze, who, destined to wander it for all eternity, came out with the stars and hunted those who are unwise enough to remain in the maze at night....

Playing the Game

The object is to maneuver your player from the top left corner of the maze to the exit located at the lower right of the maze. The game can be paused by pressing any key, and can be aborted by pressing a console key (not System Reset), except when the game has been paused. Along the right side of the screen are choices to modify the game. To change any of the values, press Option until the larger cursor is opposite the category you want to change. Then press Select, either to move the smaller cursors, changing the visibility or game variation, or to change numbers, representing speeds or number of monsters.

Game Variations

There are three different game variations. These are capture, paralyze and kill. In capture, a player must touch the four monsters controlled by the computer before he or she can exit. The eight colored boxes on the far right signify which of the monsters (orange to the orange monster, blue to blue, etc.) have already been "captured." The top four correspond to the first player, the bottom four to the second player. In the paralyze option, the object is to exit without touching any of the computer-controlled monsters. If a player is touched, he or she is frozen for a few seconds and then slowly regains speed. In the kill variation, touching any of the monsters ends the game for that player. The other player can continue on, and try to get to the exit.

Mazerunner

Visibility

There are three different visibility levels. These are clear, blackout and flash. When the visibility is clear, the entire maze is visible at all times. When it's set on blackout, the maze is black and the walls are not visible. In the flash level, the maze is black except for periodic flashes of visibility, which last for a fraction of a second. In all three levels the players and monsters are always visible.

Enemy

Here the players can choose the number of computer-controlled enemies they will face in the kill and paralyze variations. The number ranges from 0 to 16. Whatever number is chosen, though, does not affect the capture variation, in which there are always four computer-controlled monsters.

The eight colored boxes on the far right signify which of the monsters (orange to the orange monster, blue to blue, etc.) have already been "captured."

Speed

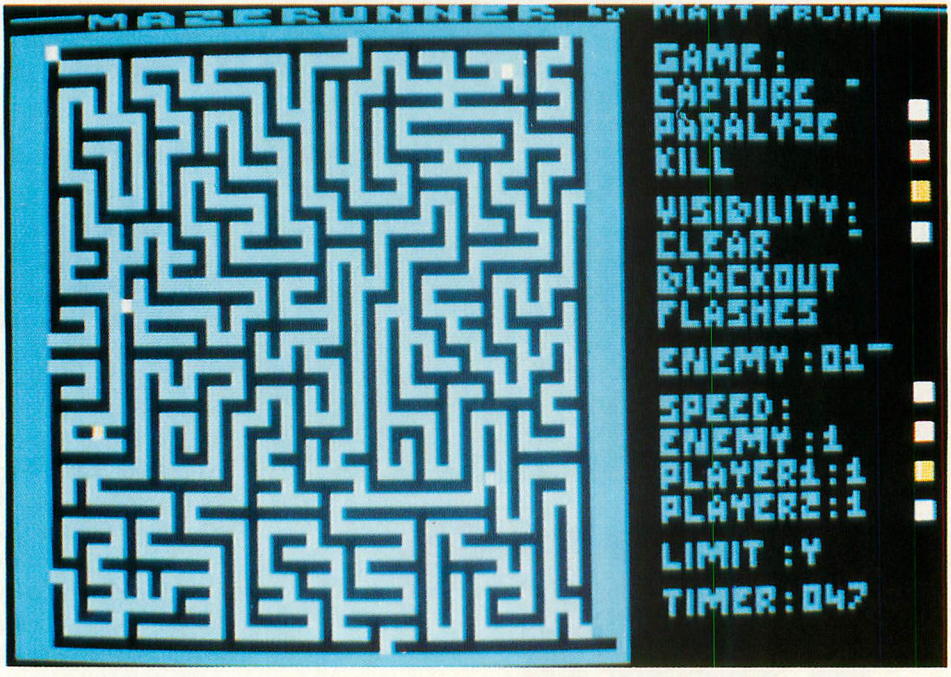
This allows the players to choose speeds for the monsters and each of the players individually. Speeds range from 1 to 4, 1 being fastest and 4 being slowest.

Limit

There are only two possible selections for this choice: Y for yes and N for no. During every game the timer counts from 000 to 999. If Y is chosen for the time limit, the game ends when the counter reaches 999. Otherwise, the counter just rolls over to 000 and continues counting. The time it takes for the counter to count from 000 to 999 is about 2 minutes, 45 seconds.

Good luck!

Matt Fruin has been programming his Atari, since 1982. Mazerunner is his first 100% machine-language game.



LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,96,251,96,169,12,1
33,12,169,96,133,13,169,1,4049
1010 DATA 133,9,32,64,108,169,64,133,2
04,169,0,133,203,162,0,160,7168
1020 DATA 0,145,203,200,208,251,230,20
4,232,224,16,208,242,169,83,133,4392
1030 DATA 203,162,96,169,3,157,66,3,16
9,0,157,69,3,169,203,157,6002
1040 DATA 68,3,169,12,157,74,3,169,0,1
57,75,3,32,86,228,169,4446
1050 DATA 0,133,88,169,64,133,89,169,0
,141,47,2,169,0,141,48,2380
1060 DATA 2,169,128,141,49,2,169,46,14
1,47,2,169,255,141,28,208,6982
1070 DATA 169,148,141,198,2,169,150,14
1,196,2,169,68,141,197,2,32,5544
1080 DATA 151,96,32,170,96,32,100,98,3
2,230,98,169,97,141,231,6,6845
1090 DATA 169,193,141,36,2,169,110,141
,37,2,76,171,100,160,0,132,4027
1100 DATA 203,169,64,133,204,185,74,11
2,145,203,200,192,160,208,246,96,3966
1110 DATA 169,0,133,208,133,209,169,16
0,133,206,169,64,133,207,162,5,9913
1120 DATA 32,239,96,202,16,250,162,87,
32,2,97,202,16,250,32,217,7595
1130 DATA 96,32,217,96,162,5,32,239,96
,202,16,250,76,42,97,160,6952
1140 DATA 0,32,32,97,200,169,0,145,206
,200,192,25,208,249,169,240,3693
1150 DATA 145,206,76,20,97,160,0,169,2
55,145,206,200,192,25,208,249,3173
1160 DATA 169,240,252,96,247,97,145,20
6,32,20,97,96,160,0,32,32,3208
1170 DATA 97,200,169,0,145,206,200,192
,24,208,249,32,32,97,24,169,7328
1180 DATA 40,101,206,144,2,230,207,133
,206,96,169,255,145,206,200,169,4725
1190 DATA 240,145,206,96,173,10,210,41
,63,133,206,56,233,26,176,244,217
1200 DATA 173,10,210,41,62,133,207,56,
233,48,176,244,169,0,133,203,73
1210 DATA 133,204,24,165,207,133,203,6
,203,6,203,101,203,133,203,162,1255
1220 DATA 4,6,203,38,204,202,208,249,2
4,165,203,101,206,144,2,230,1392
1230 DATA 204,133,203,24,165,203,105,1
60,144,2,230,204,133,203,24,165,328
1240 DATA 204,105,64,133,204,160,0,177
,203,41,64,208,167,169,1,133,7891
1250 DATA 207,173,10,210,41,3,106,144,
33,106,144,18,198,204,160,96,7313
1260 DATA 32,207,97,230,204,165,206,20
8,71,198,204,76,238,97,160,160,3220
1270 DATA 32,207,97,165,206,208,57,76,
3,98,106,144,12,160,1,32,2776
1280 DATA 207,97,165,206,240,92,76,230
,97,198,204,160,255,32,207,97,2561
1290 DATA 230,204,165,206,240,101,76,2
30,97,177,203,41,64,133,206,165,1217
1300 DATA 207,208,1,96,165,206,240,3,7
6,81,98,169,1,133,206,96,7176
1310 DATA 232,224,15,208,156,76,42,97,
169,80,160,216,145,203,160,176,1789
1320 DATA 145,203,248,97,243,98,160,13
6,145,203,160,96,145,203,76,61,104
1330 DATA 98,169,80,160,80,145,203,160
,120,145,203,160,200,145,203,160,3535
1340 DATA 160,145,203,76,61,98,160,0,1
77,203,9,85,145,203,160,40,7225
1350 DATA 145,203,169,80,160,41,145,20
3,160,1,145,203,76,61,98,160,7766
1360 DATA 39,169,85,145,203,160,255,19
8,204,145,203,24,230,208,208,2,2777
1370 DATA 230,209,165,208,201,228,208,
9,165,209,201,1,208,3,96,104,8894
1380 DATA 104,24,152,101,203,144,2,230
,204,133,203,162,0,134,207,76,9665
1390 DATA 135,97,162,1,134,205,202,189
,218,110,133,203,232,189,218,110,4688
1400 DATA 133,204,164,205,136,192,0,24
0,24,32,218,98,76,118,98,189,8249
1410 DATA 218,110,201,24,208,2,230,204
,24,101,203,144,2,230,204,133,462
1420 DATA 203,160,0,132,206,232,189,21
8,110,24,42,42,101,205,168,185,9819
1430 DATA 113,111,164,206,192,9,240,8,
145,203,200,132,206,76,151,98,533
1440 DATA 224,151,208,203,230,205,165,
205,201,5,240,5,162,0,76,105,8062
1450 DATA 98,162,31,189,45,113,157,176
,6,202,16,247,162,5,189,75,7556
1460 DATA 113,157,240,6,202,16,247,96,
165,203,24,105,40,144,2,230,7519
1470 DATA 204,133,203,96,160,2,185,80,
113,133,203,185,83,113,133,204,400
1480 DATA 185,86,244,98,239,99,113,145
,203,136,16,238,96,173,228,6,106
1490 DATA 208,35,206,229,6,16,30,169,4
,141,229,6,173,31,208,201,8046
1500 DATA 3,240,32,201,5,240,101,201,6
,240,16,173,132,2,240,11,6889
1510 DATA 173,133,2,240,6,32,128,109,7
6,95,228,173,214,6,208,248,1077
1520 DATA 76,154,106,162,1,189,192,6,2
01,0,240,5,169,0,76,65,4287
1530 DATA 99,169,150,141,196,2,173,176
,6,10,32,110,99,169,0,160,4631
1540 DATA 0,145,203,238,176,6,173,176,
6,201,7,208,5,169,0,141,6250
1550 DATA 176,6,24,10,32,110,99,169,25
5,160,0,145,203,76,35,99,6162
1560 DATA 170,189,178,6,133,203,189,17
9,6,133,204,96,173,176,6,170,9607
1570 DATA 56,233,2,176,57,32,163,99,16
9,0,145,203,254,192,6,189,9868
1580 DATA 192,6,201,3,208,5,169,0,157,
192,6,32,163,99,169,60,5435
1590 DATA 145,203,76,35,99,173,176,6,2
40,2,169,3,24,125,192,6,4435
1600 DATA 168,185,194,6,133,203,185,20
0,6,133,204,160,0,96,233,1,7925
1610 DATA 176,101,238,241,6,173,241,6,
201,10,208,25,238,240,6,169,214
1620 DATA 0,141,241,6,173,176,6,10,32,
110,99,198,203,198,203,173,769
1630 DATA 240,6,32,14,100,173,241,6,20
1,7,208,18,173,240,6,201,8605
1640 DATA 1,208,240,99,235,100,11,169,
0,141,240,6,141,241,6,76,7012
1650 DATA 210,99,173,176,6,10,32,110,9
9,198,203,173,241,6,32,14,5576
1660 DATA 100,76,35,99,10,10,105,109,1
70,160,0,189,113,111,145,203,7974
1670 DATA 152,24,105,40,168,232,192,16
0,208,241,96,233,1,176,32,173,1060
1680 DATA 176,6,10,32,110,99,198,203,1
98,203,238,242,6,173,242,6,1740
1690 DATA 201,5,208,5,169,1,141,242,6,
32,14,100,76,35,99,233,4738
1700 DATA 2,176,36,173,176,6,10,32,110
,99,198,203,173,176,6,41,6149
1710 DATA 1,170,254,243,6,189,243,6,20
```

Mazerunner

```
1,5,208,5,169,1,157,243,9359
1720 DATA 6,32,14,100,76,35,99,173,176
,6,10,32,110,99,198,203,6071
1730 DATA 198,203,198,203,173,231,6,73
,84,141,231,6,32,18,100,76,4780
1740 DATA 35,99,169,204,141,34,2,169,1
10,141,35,2,32,115,101,169,4625
1750 DATA 0,141,223,6,173,192,6,208,8,
238,223,6,169,2,141,227,9397
1760 DATA 6,169,4,141,229,6,169,251,14
1,34,2,169,98,141,35,2,4516
1770 DATA 169,0,141,228,6,169,150,141,
196,2,32,124,108,169,0,141,6146
1780 DATA 232,6,141,233,6,141,234,6,14
1,235,6,141,226,6,133,224,9919
1790 DATA 141,224,6,141,225,6,165,224,
240,252,141,228,6,169,160,141,3136
1800 DATA 1,210,236,100,231,101,32,52,
108,169,0,141,218,6,169,16,5793
1810 DATA 141,219,6,169,0,141,8,210,16
9,3,141,15,210,169,0,141,6299
1820 DATA 220,6,141,221,6,169,15,141,1
10,6,141,111,6,32,115,101,3436
1830 DATA 169,1,141,214,6,169,0,162,3,
157,192,2,202,16,250,169,8657
1840 DATA 150,141,196,2,32,170,96,32,1
75,103,169,0,133,209,32,160,7135
1850 DATA 108,169,255,141,30,208,169,0
,141,34,2,169,107,141,35,2,3787
1860 DATA 206,214,6,165,224,240,252,17
3,218,6,240,10,169,0,141,8,8102
1870 DATA 210,169,3,141,15,210,173,31,
208,201,7,240,3,76,140,100,7427
1880 DATA 173,232,6,240,16,173,233,6,2
40,11,76,153,100,173,31,208,8749
1890 DATA 201,7,208,249,96,162,1,189,1
02,6,201,132,208,23,189,210,690
1900 DATA 6,208,18,169,5,141,218,6,169
,15,141,220,6,169,30,141,6509
1910 DATA 221,6,76,153,100,202,16,223,
198,224,162,1,189,232,6,208,1530
1920 DATA 38,189,234,6,240,11,32,165,1
07,173,208,6,240,25,76,194,8639
1930 DATA 101,254,224,6,189,224,6,221,
243,6,208,11,169,0,142,108,8056
1940 DATA 6,157,224,6,32,232,103,202,1
6,210,238,226,6,173,226,6,405
1950 DATA 205,242,6,240,3,76,73,101,16
9,0,141,226,6,173,192,6,6667
1960 DATA 240,6,232,101,227,102,32,244
,101,76,73,101,32,141,105,76,6100
1970 DATA 73,101,238,80,6,174,80,6,224
,16,208,6,162,255,142,80,8583
1980 DATA 6,96,189,48,6,201,6,240,233,
189,0,6,133,203,189,16,7856
1990 DATA 6,133,204,62,64,6,62,64,6,17
6,16,32,218,102,165,203,6783
2000 DATA 157,0,6,165,204,157,16,6,76,
244,101,189,48,6,201,4,5247
2010 DATA 240,65,41,8,240,23,169,64,15
7,64,6,160,0,32,157,103,4058
2020 DATA 160,40,32,157,103,169,6,157,
48,6,76,244,101,198,204,160,9772
2030 DATA 216,32,86,106,230,204,160,0,
32,103,106,160,1,32,103,106,4592
2040 DATA 160,80,32,90,106,165,208,201
,1,208,8,189,48,6,73,2,3680
2050 DATA 76,204,102,173,10,210,41,3,1
33,207,189,48,6,201,4,240,7867
2060 DATA 11,165,207,93,48,6,240,4,41,
1,240,231,165,207,106,144,110
```

```
2070 DATA 31,106,144,16,160,216,198,20
4,32,86,106,230,204,165,208,240,4919
2080 DATA 210,76,202,102,160,80,32,86,
106,165,208,240,198,76,202,102,1720
2090 DATA 106,144,12,160,1,32,99,106,1
65,208,240,183,76,202,102,160,1205
2100 DATA 0,32,99,106,165,208,240,171,
165,207,157,48,6,169,1,157,8835
2110 DATA 64,6,32,218,102,76,32,102,18
9,48,6,106,144,33,106,144,4907
2120 DATA 15,160,228,102,223,103,40,32
,157,103,198,204,160,216,32,167,1141
2130 DATA 103,76,146,103,160,0,32,157,
103,160,80,32,167,103,160,40,5959
2140 DATA 76,146,103,160,0,32,157,103,
160,40,32,157,103,189,48,6,4449
2150 DATA 106,106,144,63,24,126,32,6,1
26,32,6,176,11,160,0,32,827
2160 DATA 167,103,160,40,32,167,103,96
,189,32,6,240,25,160,0,32,3868
2170 DATA 167,103,160,40,32,167,103,16
0,1,169,128,32,170,103,160,41,6545
2180 DATA 169,128,32,170,103,96,160,1,
32,146,103,169,160,157,32,6,5363
2190 DATA 76,27,103,62,32,6,176,14,62,
32,6,160,0,32,167,103,1882
2200 DATA 160,40,32,167,103,96,24,62,3
2,6,189,32,6,240,22,32,2050
2210 DATA 89,103,198,204,160,255,169,2
,32,170,103,230,204,169,2,160,933
2220 DATA 39,32,170,103,96,160,255,198
,204,32,146,103,169,10,157,32,8256
2230 DATA 6,76,89,103,152,24,101,203,1
44,2,230,204,133,203,96,189,1631
2240 DATA 32,6,73,255,49,203,145,203,9
6,189,32,6,17,203,145,203,9289
2250 DATA 96,169,120,141,7,212,24,105,
2,133,206,169,255,141,108,6,8570
2260 DATA 169,3,141,29,208,162,3,169,1
,157,8,208,202,16,250,169,9826
2270 DATA 202,141,192,2,169,138,141,19
3,2,96,238,108,6,173,108,6,6650
2280 DATA 201,2,224,103,219,104,208,6,
169,255,141,108,6,96,173,108,9455
2290 DATA 6,170,240,2,169,128,133,205,
189,96,6,133,203,189,98,6,8722
2300 DATA 133,204,62,106,6,62,106,6,17
6,15,32,226,104,173,192,6,6191
2310 DATA 208,6,32,54,109,174,108,6,96
,169,0,133,207,157,104,6,5930
2320 DATA 189,120,2,141,109,6,201,11,2
40,84,254,104,6,201,13,240,9364
2330 DATA 77,254,104,6,201,7,240,70,25
4,104,6,201,14,240,63,169,9533
2340 DATA 0,157,104,6,173,109,6,29,110
,6,201,15,240,40,230,207,8666
2350 DATA 173,109,6,73,15,93,110,6,76,
29,104,165,207,240,13,169,7550
2360 DATA 0,157,104,6,198,207,189,110,
6,76,29,104,173,218,6,208,8545
2370 DATA 5,169,1,141,218,6,169,64,157
,106,6,76,7,104,169,0,3634
2380 DATA 133,77,189,104,6,106,144,44,
106,144,24,160,216,198,204,32,9406
2390 DATA 86,106,230,204,165,208,240,1
95,222,98,6,160,96,32,21,105,8162
2400 DATA 76,209,104,160,80,32,86,106,
165,208,240,175,160,160,32,21,9329
2410 DATA 105,76,209,104,106,144,15,16
0,1,32,99,106,165,208,240,155,316
2420 DATA 32,21,105,76,209,104,160,0,3
2,99,106,165,208,240,140,222,2042
2430 DATA 98,6,160,255,32,21,105,173,1
09,6,157,110,6,169,1,157,5504
2440 DATA 106,6,220,104,215,105,32,226
,104,76,7,104,188,100,6,169,7030
2450 DATA 1,32,133,105,189,104,6,106,1
44,18,106,144,6,222,100,6,4945
2460 DATA 76,252,104,254,100,6,188,100
,6,76,119,105,106,144,6,254,7951
2470 DATA 102,6,76,14,105,222,102,6,18
9,102,6,157,0,208,96,152,6942
2480 DATA 24,125,96,6,144,3,254,98,6,1
57,96,6,96,165,209,73,6979
2490 DATA 2,133,209,230,206,170,169,0,
133,205,32,62,105,232,169,128,1077
2500 DATA 133,205,32,62,105,198,206,96
,138,73,2,170,189,120,6,168,8259
2510 DATA 169,0,145,205,32,133,105,145
,205,200,145,205,138,73,2,170,253
2520 DATA 41,1,168,189,124,6,153,2,208
,138,77,227,6,170,189,138,9890
2530 DATA 6,153,194,2,138,77,227,6,170
,189,120,6,168,32,119,105,7130
2540 DATA 96,169,0,145,205,169,1,32,13
3,105,169,0,145,205,96,200,9357
2550 DATA 145,205,200,145,205,200,96,2
38,136,6,173,136,6,201,4,208,185
2560 DATA 8,169,255,141,136,6,76,54,10
9,170,189,112,6,133,203,189,28
2570 DATA 116,6,133,204,62,132,6,62,13
2,6,176,6,32,112,106,76,3599
2580 DATA 141,105,198,204,160,216,32,8
6,106,230,204,160,0,32,103,106,8456
2590 DATA 160,1,32,103,106,160,80,32,9
0,106,165,208,201,1,208,10,7654
2600 DATA 189,128,216,105,211,106,6,73
,2,133,207,76,240,105,173,10,8312
2610 DATA 210,41,3,133,207,93,128,6,24
0,4,41,1,240,240,165,207,910
2620 DATA 106,144,44,106,144,24,160,21
6,198,204,32,86,106,230,204,165,2786
2630 DATA 208,240,219,160,96,222,116,6
,32,140,106,76,70,106,160,80,7127
2640 DATA 32,86,106,165,208,240,199,16
0,160,32,140,106,76,70,106,106,8801
2650 DATA 144,17,160,1,32,99,106,165,2
08,240,179,160,1,32,140,106,8546
2660 DATA 76,70,106,160,0,32,99,106,16
5,208,240,162,160,255,222,116,4552
2670 DATA 6,32,140,106,165,207,157,128
,6,169,1,157,132,6,32,112,5485
2680 DATA 106,76,141,105,169,0,133,208
,177,203,41,5,208,2,230,208,865
2690 DATA 96,169,0,133,208,177,203,41,
80,208,2,230,208,96,189,128,2020
2700 DATA 6,106,144,11,106,144,4,222,1
20,6,96,254,120,6,96,106,6616
2710 DATA 144,4,254,124,6,96,222,124,6
,96,152,24,125,112,6,144,5833
2720 DATA 3,254,116,6,157,112,6,96,169
,0,133,77,141,223,6,141,7079
2730 DATA 227,6,32,203,107,173,192,6,2
08,26,169,4,141,210,6,141,8000
2740 DATA 211,6,169,192,141,14,212,169
,23,141,0,2,169,108,141,1,5344
2750 DATA 2,76,234,106,169,64,141,14,2
12,169,0,141,210,6,141,211,9953
2760 DATA 6,173,212,106,207,107,240,6,
240,2,169,10,109,241,6,170,9337
2770 DATA 202,169,4,202,48,6,157,48,6,
76,225,106,169,1,133,224,8240
```

Mazerunner

```
2780 DATA 162,5,169,0,157,168,6,202,16
,250,169,255,141,252,2,76,1080
2790 DATA 95,228,173,252,2,201,255,240
,13,169,12,141,117,228,169,96,2204
2800 DATA 141,118,228,76,120,110,169,1
,133,224,173,193,6,240,40,173,1078
2810 DATA 222,6,240,6,206,222,6,76,69,
107,169,0,141,196,2,173,7725
2820 DATA 193,6,201,1,240,17,173,10,21
0,41,127,208,10,169,10,141,7320
2830 DATA 222,6,169,150,141,196,2,173,
192,6,208,6,32,35,105,76,5295
2840 DATA 154,107,162,0,189,4,208,240,
62,173,192,6,201,1,240,28,9186
2850 DATA 189,232,6,208,20,169,255,157
,232,6,169,2,141,218,6,169,218
2860 DATA 128,141,221,6,169,15,141,220
,6,76,149,107,189,234,6,201,415
2870 DATA 255,240,20,169,15,141,220,6,
169,224,141,221,6,169,4,141,9574
2880 DATA 218,6,169,255,157,234,6,232,
224,2,208,184,169,255,141,30,3618
2890 DATA 208,32,202,108,76,95,228,169
,0,141,208,6,254,236,6,189,1588
2900 DATA 236,6,221,234,6,208,21,94,23
4,6,169,0,157,236,6,189,9408
2910 DATA 234,6,201,1,208,3,222,234,6,
238,208,6,96,169,30,133,9233
2920 DATA 203,169,208,107,203,108,67,1
33,204,162,4,32,246,107,202,208,2647
2930 DATA 250,232,32,11,108,232,224,4,
208,248,32,246,107,202,208,250,6311
2940 DATA 162,7,189,234,112,157,160,6,
202,16,247,96,160,0,173,192,883
2950 DATA 6,208,2,169,168,41,168,145,2
03,160,40,145,203,160,80,145,1482
2960 DATA 203,24,169,240,101,203,144,2
,230,204,133,203,96,72,152,72,1122
2970 DATA 172,216,6,185,160,6,141,10,2
12,141,23,208,200,192,8,208,1038
2980 DATA 2,160,0,140,216,6,104,168,10
4,64,162,48,189,242,112,157,1021
2990 DATA 96,6,202,16,247,96,160,0,185
,32,113,153,0,128,200,192,9627
3000 DATA 13,208,245,152,153,0,128,200
,192,106,208,248,169,65,153,0,1487
3010 DATA 128,200,169,0,153,0,128,200,
169,63,153,0,128,160,21,162,7951
3020 DATA 4,169,141,153,0,128,153,41,1
28,152,24,105,6,168,202,208,9548
3030 DATA 240,96,162,15,169,104,157,0,
6,169,79,157,16,6,169,10,4060
3040 DATA 157,32,6,169,6,157,48,6,169,
64,157,64,6,169,0,157,4923
3050 DATA 80,6,202,16,223,96,160,0,169
,0,153,0,120,153,0,121,5465
3060 DATA 153,0,122,153,0,123,200,208,
241,169,1,141,20,122,141,21,7974
3070 DATA 122,141,148,122,141,149,122,
169,40,141,0,208,141,1,208,96,8980
3080 DATA 162,4,204,108,199,109,254,16
8,6,189,168,6,201,10,208,10,8962
3090 DATA 169,0,157,168,6,202,224,0,20
8,236,162,3,189,168,6,24,8545
3100 DATA 42,42,105,109,168,185,113,11
1,157,88,78,32,49,109,200,138,8512
3110 DATA 56,233,159,144,240,138,41,3,
170,202,224,0,208,222,162,3,1328
3120 DATA 189,168,6,201,9,208,33,202,2
08,246,173,231,6,201,97,208,4289
```

```
3130 DATA 23,169,1,141,232,6,141,233,6
,169,6,141,218,6,169,0,7060
3140 DATA 141,221,6,169,40,141,220,6,7
6,128,109,138,105,40,170,96,7881
3150 DATA 162,1,160,3,189,102,6,89,124
,6,41,254,208,53,189,100,8673
3160 DATA 6,89,120,6,41,254,208,43,140
,212,6,152,224,0,240,3,9087
3170 DATA 24,105,4,168,185,160,6,240,2
3,169,0,153,160,6,222,210,522
3180 DATA 6,169,15,141,220,6,169,0,141
,221,6,169,3,141,218,6,7404
3190 DATA 172,212,6,136,16,190,202,240
,185,96,173,218,6,240,57,201,3555
3200 DATA 2,240,54,201,3,240,83,201,4,
240,116,201,5,208,3,76,8654
3210 DATA 33,110,201,6,208,3,76,61,110
,174,219,6,202,142,219,6,9374
3220 DATA 169,240,141,0,210,138,24,105
,160,141,1,210,224,0,208,8,8719
3230 DATA 142,218,6,169,16,141,219,6,9
6,169,32,32,106,110,173,221,9348
3240 DATA 6,233,200,109,195,110,8,176,
11,174,220,6,202,48,9,142,7883
3250 DATA 220,6,169,128,141,221,6,96,1
69,0,141,218,6,96,169,160,9547
3260 DATA 32,106,110,173,221,6,24,105,
32,144,233,174,220,6,202,202,2260
3270 DATA 48,9,142,220,6,169,0,141,221
,6,96,169,160,141,1,210,9376
3280 DATA 76,218,109,169,64,32,106,110
,173,221,6,56,233,32,176,196,687
3290 DATA 174,220,6,202,48,194,142,220
,6,169,224,141,221,6,96,169,1889
3300 DATA 160,32,106,110,173,221,6,24,
105,32,144,11,174,220,6,202,8352
3310 DATA 48,166,142,220,6,138,10,141,
221,6,96,169,110,141,1,210,8964
3320 DATA 173,220,6,141,0,210,174,221,
6,232,224,60,208,22,173,220,3446
3330 DATA 6,201,60,208,8,169,160,141,1
,210,76,218,109,169,60,141,552
3340 DATA 220,6,162,0,142,221,6,96,24,
109,220,6,141,1,210,173,8467
3350 DATA 221,6,141,0,210,96,169,140,1
41,34,2,169,110,141,35,2,5555
3360 DATA 169,255,141,252,2,169,160,14
1,1,210,169,0,141,196,2,173,85
3370 DATA 192,6,208,3,32,35,105,173,25
2,2,201,255,240,30,169,255,4413
3380 DATA 141,252,2,169,0,141,34,2,169
,107,141,35,2,169,255,141,8842
3390 DATA 30,208,173,193,6,208,5,169,1
50,141,196,2,76,95,228,173,1238
3400 DATA 223,6,196,110,191,111,240,3,
32,35,105,76,98,228,173,192,826
3410 DATA 6,208,3,32,35,105,32,128,109
,76,95,228,211,65,44,45,6577
3420 DATA 46,47,48,37,26,26,26,200,4,1
,19,22,23,20,6,26,7524
3430 DATA 26,200,19,1,20,1,12,24,25,6,
26,200,11,51,52,53,9755
3440 DATA 26,26,26,26,26,24,38,39,40,4
1,42,43,22,24,37,200,677
3450 DATA 4,12,6,1,20,26,26,26,26,200,
2,3,1,4,11,18,6960
3460 DATA 23,22,26,200,7,12,1,21,8,6,2
1,26,26,24,6,13,6334
3470 DATA 14,15,16,17,37,27,28,24,21,1
9,6,6,5,37,26,26,6271
3480 DATA 26,200,6,13,14,15,16,17,37,2
8,26,200,19,12,1,24,8497
3490 DATA 6,20,28,37,28,200,19,12,1,24
,6,20,29,37,28,24,7591
3500 DATA 12,9,10,49,50,37,24,26,26,24
,22,9,10,6,20,37,6534
3510 DATA 27,27,27,0,48,204,252,204,48
,204,252,204,240,207,243,252,9919
3520 DATA 48,48,48,60,252,192,192,252,
240,204,204,240,252,240,192,252,2692
3530 DATA 252,240,192,192,204,252,204,
204,207,207,204,204,60,252,204,12,7383
3540 DATA 204,240,240,204,192,192,192,
252,243,243,207,207,63,60,48,63,3997
3550 DATA 60,63,51,48,243,243,48,48,48
,240,192,192,252,204,204,252,7946
3560 DATA 252,204,192,111,187,112,252,
192,252,204,240,204,252,240,12,252,391
3570 DATA 252,48,48,48,204,204,204,252
,204,252,48,48,252,60,192,252,6430
3580 DATA 0,0,0,0,252,204,204,252,48,2
40,48,252,252,60,192,252,6920
3590 DATA 252,60,12,252,192,204,252,12
,252,240,12,252,252,192,252,252,650
3600 DATA 252,12,48,192,252,204,252,25
2,252,204,252,12,0,48,0,48,9716
3610 DATA 204,204,252,48,207,207,192,2
07,204,12,204,204,240,207,243,252,790
3620 DATA 51,51,51,51,12,12,12,204,255
,192,207,255,12,51,63,51,8161
3630 DATA 60,63,51,48,243,243,51,51,24
0,192,0,240,207,195,195,195,6025
3640 DATA 192,0,0,0,204,204,204,207,12
,12,12,207,0,0,0,192,5076
3650 DATA 255,255,255,60,60,15,240,63,
252,63,252,63,240,60,60,60,240
3660 DATA 60,60,60,63,252,63,240,0,48,
0,0,243,195,15,207,192,9860
3670 DATA 252,252,195,51,204,255,255,2
55,0,0,0,63,252,60,60,0,7362
3680 DATA 252,63,240,60,60,60,60,63,60
,63,60,63,240,60,60,0,4048
3690 DATA 60,204,0,204,204,195,3,0,240
,204,195,51,204,0,0,0,6794
3700 DATA 0,0,0,51,204,63,252,63,0,60,
0,63,192,60,60,60,4122
3710 DATA 252,60,252,60,0,63,192,0,60,
48,0,204,207,195,3,0,5734
3720 DATA 192,240,188,112,88,113,195,5
1,60,0,0,0,0,0,0,48,9603
3730 DATA 12,60,60,63,252,63,252,60,60
,63,252,60,60,60,60,63,6366
3740 DATA 252,60,60,0,0,240,0,192,204,
195,3,0,192,204,60,51,8155
3750 DATA 60,0,0,0,104,70,22,118,104,7
0,22,118,145,145,65,65,5072
3760 DATA 19,19,40,40,0,0,64,64,0,0,0,
0,167,124,81,103,1827
3770 DATA 65,72,79,79,19,63,107,107,12
8,84,40,128,0,0,0,0,578
3780 DATA 64,64,64,64,0,0,106,72,24,12
0,112,112,112,78,0,64,3302
3790 DATA 14,14,14,0,77,64,65,0,0,220,
65,76,69,188,72,156,6030
3800 DATA 74,100,75,44,76,68,77,0,0,16
3,107,51,19,219,163,66,6035
3810 DATA 67,68,70,70,71,0,1,1,1,1,163
,18,218,66,70,65,2749
3820 DATA 60,60,255,226,2,227,2,0,96,0
,0,0,0,0,0,0,7919
```

continued from page 63

```
0620      .BYTE $88    ;DEY
0630      .BYTE $49,$45,$55,$4D ;EOR
0640      .BYTE $5D,$59,$41,$51
0650      .BYTE $E6,$F6,$EE,$FE ;INC
0660      .BYTE $E8    ;INX
0670      .BYTE $C8    ;INY
0680      .BYTE $4C,$6C ;JMP
0690      .BYTE $20    ;JSR
0700      .BYTE $A9,$A5,$B5,$AD ;LDA
0710      .BYTE $BD,$B9,$A1,$B1
0720      .BYTE $A2,$A6,$B6,$AE ;LDX
0730      .BYTE $BE
0740      .BYTE $A0,$A4,$B4,$AC ;LDY
0750      .BYTE $BC
0760      .BYTE $4A,$46,$56,$4E ;LSR
0770      .BYTE $5E
0780      .BYTE $EA    ;NOP
0790      .BYTE $09,$05,$15,$0D ;ORA
0800      .BYTE $1D,$19,$01,$11
0810      .BYTE $48    ;PHA
0820      .BYTE $08    ;PHP
0830      .BYTE $68    ;PLA
0840      .BYTE $28    ;PLP
0850      .BYTE $2A,$26,$36,$2E ;ROL
0860      .BYTE $3E
0870      .BYTE $6A,$66,$76,$6E ;ROR
0880      .BYTE $7E
0890      .BYTE $40    ;RTI
0900      .BYTE $60    ;RTS
0910      .BYTE $E9,$E5,$F5,$ED ;SBC
0920      .BYTE $FD,$F9,$E1,$F1
0930      .BYTE $38    ;SEC
0940      .BYTE $F8    ;SED
0950      .BYTE $78    ;SEI
0960      .BYTE $85,$95,$8D ;STA
0970      .BYTE $9D,$99,$81,$91
0980      .BYTE $86,$96,$8E ;STX
0990      .BYTE $84,$94,$8C ;STY
1000      .BYTE $AA    ;TAX
1010      .BYTE $A8    ;TAY
1020      .BYTE $BA    ;TSX
1030      .BYTE $8A    ;TXA
1040      .BYTE $9A    ;TXS
1050      .BYTE $98    ;TYA
1060 ;
1070 ;Offsets into Mnemonic table
1080 ;
1090 OFFSET
1100      .BYTE 0,0,0,0,0,0,0,0 ;ADC
1110      .BYTE 3,3,3,3,3,3,3,3 ;AND
1120      .BYTE 6,6,6,6,6 ;ASL
1130      .BYTE 9      ;BCC
1140      .BYTE 12     ;BCS
1150      .BYTE 15     ;BEQ
1160      .BYTE 18,18 ;BIT
1170      .BYTE 21     ;BMI
1180      .BYTE 24     ;BNE
1190      .BYTE 27     ;BPL
1200      .BYTE 30     ;BRK
1210      .BYTE 33     ;BVC
1220      .BYTE 36     ;BVS
1230      .BYTE 39     ;CLC
1240      .BYTE 42     ;CLD
1250      .BYTE 45     ;CLI
1260      .BYTE 48     ;CLV
1270      .BYTE 51,51,51,51 ;CMP
1280      .BYTE 51,51,51,51
1290      .BYTE 54,54,54 ;CPX
1300      .BYTE 57,57,57 ;CPY
1310      .BYTE 60,60,60,60 ;DEC
1320      .BYTE 63     ;DEX
1330      .BYTE 66     ;DEY
1340      .BYTE 69,69,69,69 ;EOR
1350      .BYTE 69,69,69,69
1360      .BYTE 72,72,72,72 ;INC
1370      .BYTE 75     ;INX
1380      .BYTE 78     ;INY
1390      .BYTE 81,81 ;JMP
1400      .BYTE 84     ;JSR
1410      .BYTE 87,87,87,87 ;LDA
1420      .BYTE 87,87,87,87
1430      .BYTE 90,90,90,90,90 ;LDX
1440      .BYTE 93,93,93,93,93 ;LDY
1450      .BYTE 96,96,96,96,96 ;LSR
1460      .BYTE 99     ;NOP
1470      .BYTE 102,102,102,102 ;ORA
1480      .BYTE 102,102,102,102
1490      .BYTE 105    ;PHA
1500      .BYTE 108    ;PHP
1510      .BYTE 111    ;PLA
1520      .BYTE 114    ;PLP
1530      .BYTE 117,117,117,117 ;ROL
1540      .BYTE 117
1550      .BYTE 120,120,120,120 ;ROR
1560      .BYTE 120
1570      .BYTE 123    ;RTI
1580      .BYTE 126    ;RTS
1590      .BYTE 129,129,129,129 ;SBC
1600      .BYTE 129,129,129,129
1610      .BYTE 132    ;SEC
1620      .BYTE 135    ;SED
1630      .BYTE 138    ;SEI
1640      .BYTE 141,141,141,141 ;STA
1650      .BYTE 141,141,141
1660      .BYTE 144,144,144 ;STX
1670      .BYTE 147,147,147 ;STY
1680      .BYTE 150    ;TAX
1690      .BYTE 153    ;TAY
1700      .BYTE 156    ;TSX
1710      .BYTE 159    ;TXA
1720      .BYTE 162    ;TXS
1730      .BYTE 165    ;TYA
1740 ;
1750 ;Instruction Types
1760 ;
1770 ; 0 = Immediate
1780 ; 1 = Zero page
1790 ; 2 = Zero page,X
1800 ; 3 = Absolute
1810 ; 4 = Absolute,X
1820 ; 5 = Absolute,Y
1830 ; 6 = Indirect,X
1840 ; 7 = Indirect,Y
1850 ; 8 = Accumulator
1860 ; 9 = Relative
1870 ;10 = Implied
1880 ;11 = Indirect
1890 ;12 = Zero page,Y
1900 ;
1910 ITYPE
1920      .BYTE 0,1,2,3,4,5,6,7 ;ADC
1930      .BYTE 0,1,2,3,4,5,6,7 ;AND
1940      .BYTE 8,1,2,3,4 ;ASL
1950      .BYTE 9      ;BCC
1960      .BYTE 9      ;BCS
1970      .BYTE 9      ;BEQ
1980      .BYTE 1,3    ;BIT
1990      .BYTE 9      ;BMI
2000      .BYTE 9      ;BNE
2010      .BYTE 9      ;BPL
2020      .BYTE 10     ;BRK
2030      .BYTE 9      ;BVC
2040      .BYTE 9      ;BVS
2050      .BYTE 10     ;CLC
2060      .BYTE 10     ;CLD
2070      .BYTE 10     ;CLI
2080      .BYTE 10     ;CLV
2090      .BYTE 0,1,2,3,4,5,6,7 ;CMP
2100      .BYTE 0,1,3 ;CPX
2110      .BYTE 0,1,3 ;CPY
2120      .BYTE 1,2,3,4 ;DEC
2130      .BYTE 10     ;DEX
2140      .BYTE 10     ;DEY
2150      .BYTE 0,1,2,3,4,5,6,7 ;EOR
2160      .BYTE 1,2,3,4 ;INC
2170      .BYTE 10     ;INX
2180      .BYTE 10     ;INY
2190      .BYTE 3,11   ;JMP
2200      .BYTE 3      ;JSR
2210      .BYTE 0,1,2,3,4,5,6,7 ;LDA
2220      .BYTE 0,1,12,3,5 ;LDX
2230      .BYTE 0,1,2,3,4 ;LDY
2240      .BYTE 8,1,2,3,4 ;LSR
2250      .BYTE 10     ;NOP
2260      .BYTE 0,1,2,3,4,5,6,7 ;ORA
2270      .BYTE 10     ;PHA
2280      .BYTE 10     ;PHP
2290      .BYTE 10     ;PLA
2300      .BYTE 10     ;PLP
2310      .BYTE 8,1,2,3,4 ;ROL
2320      .BYTE 8,1,2,3,4 ;ROR
2330      .BYTE 10     ;RTI
2340      .BYTE 10     ;RTS
2350      .BYTE 0,1,2,3,4,5,6,7 ;SBC
2360      .BYTE 10     ;SEC
2370      .BYTE 10     ;SED
2380      .BYTE 10     ;SEI
2390      .BYTE 1,2,3,4,5,6,7 ;STA
2400      .BYTE 1,12,3 ;STX
2410      .BYTE 1,2,3 ;STY
2420      .BYTE 10     ;TAX
2430      .BYTE 10     ;TAY
2440      .BYTE 10     ;TSX
2450      .BYTE 10     ;TXA
2460      .BYTE 10     ;TXS
2470      .BYTE 10     ;TYA
2480 ;
2490 ;Disassembler JMP Tables
2500 ;
2510 DCOMT .WORD IMMED ;2 bytes
2520      .WORD ZPAG  ;2 bytes
2530      .WORD ZPAGX ;2 bytes
2540      .WORD ABSOL ;3 bytes
2550      .WORD ABSOLX ;3 bytes
2560      .WORD ABSOLY ;3 bytes
2570      .WORD INDX  ;2 bytes
2580      .WORD INDY  ;2 bytes
2590      .WORD ACCUM ;1 bytes
2600      .WORD RELA  ;2 bytes
2610      .WORD IMP   ;1 bytes
2620      .WORD INDI  ;3 bytes
2630      .WORD ZPAGY ;2 bytes
2640 NBTAB .BYTE 2,2,2,3,3,3,2
2650      .BYTE 2,1,2,1,3,2
2660 ;
2670 COMMAX .BYTE ",X)"
2680 COMMAY .BYTE "Y,)"
2690 POINTB .BYTE " .BYTE $"
2700 STEN .BYTE " *=$      "
2710      .BYTE ";END=$     ",EOL,EOL,0
2720 ;
2730 INS_DST .BYTE EOL,"Insert des"
2740      .BYTE "t'n, RETURN",EOL,0
2750 ;
2760 INS_SRC .BYTE EOL,"Insert sou"
2770      .BYTE "rce, RETURN",EOL,0
2780 ;
2790 ALL.DONE .BYTE EOL,"Completed"
2800      .BYTE ", RETURN",EOL,0
2810 ;
2820 OUTP .BYTE EOL,"OUTPUT ",0
2830 IN_P .BYTE EOL,"INPUT ",0
```

LISTING 6: ASSEMBLY

```
0100 ;SAVE#D:DMPT5.M65
0110 ;-------------------------
0120 ;
0130 ;System Equates/Labels
0140 ;Consistent with MAPPING THE
0150 ;ATARI from Compute! books.
0160 ;
0170 ;Data storage technique by:
0180 ;Bryan Schappel
0190 ;
0200 ;-------------------------
0210 ;
0220 LAB .CBYTE $19,$00,$02,"CASINI"
0230      .CBYTE $09,$00,$06,"TRAMSZ"
0240      .CBYTE $09,$00,$08,"WARMST"
0250      .CBYTE $08,$00,$09,"BOOT?"
0260      .CBYTE $19,$00,$0A,"DOSVEC"
0270      .CBYTE $19,$00,$0C,"DOSINI"
0280      .CBYTE $09,$00,$10,"POKMSK"
0290      .CBYTE $09,$00,$11,"BRKKEY"
0300      .CBYTE $29,$00,$12,"RTCLOK"
0310      .CBYTE $09,$00,$20,"ICHIDZ"
0320      .CBYTE $09,$00,$21,"ICDNOZ"
0330      .CBYTE $09,$00,$22,"ICCOMZ"
0340      .CBYTE $09,$00,$23,"ICSTAZ"
0350      .CBYTE $09,$00,$24,"ICBALZ"
0360      .CBYTE $09,$00,$25,"ICBAHZ"
0370      .CBYTE $09,$00,$26,"ICPTLZ"
0380      .CBYTE $09,$00,$27,"ICPTHZ"
0390      .CBYTE $09,$00,$28,"ICBLLZ"
0400      .CBYTE $09,$00,$29,"ICBLHZ"
0410      .CBYTE $09,$00,$2A,"ICAX1Z"
0420      .CBYTE $09,$00,$2B,"ICAX2Z"
0430      .CBYTE $09,$00,$41,"SOUNDR"
0440      .CBYTE $09,$00,$42,"CRITIC"
0450      .CBYTE $09,$00,$4D,"ATRACT"
0460      .CBYTE $09,$00,$52,"LMARGN"
0470      .CBYTE $09,$00,$53,"RMARGN"
0480      .CBYTE $09,$00,$54,"ROWCRS"
0490      .CBYTE $19,$00,$55,"COLCRS"
0500      .CBYTE $19,$00,$58,"SAVMSC"
0510      .CBYTE $09,$00,$6A,"RAMTOP"
0520      .CBYTE $56,$00,$D4,"FR0"
0530      .CBYTE $56,$00,$DA,"FRE"
0540      .CBYTE $56,$00,$E0,"FR1"
0550      .CBYTE $56,$00,$E6,"FR2"
0560      .CBYTE $06,$00,$EC,"FRX"
0570      .CBYTE $07,$00,$ED,"EEXP"
0580      .CBYTE $08,$00,$EE,"NSIGN"
0590      .CBYTE $08,$00,$EF,"ESIGN"
0600      .CBYTE $0A,$00,$F0,"FCHRFLG"
0610      .CBYTE $08,$00,$F1,"DIGRT"
0620      .CBYTE $06,$00,$F2,"CIX"
0630      .CBYTE $19,$00,$F3,"INBUFF"
0640      .CBYTE $09,$00,$FB,"RADFLG"
0650      .CBYTE $18,$00,$FC,"FLPTR"
0660      .CBYTE $18,$00,$FE,"FPTR2"
0670      .CBYTE $19,$02,$00,"VDSLST"
0680      .CBYTE $19,$02,$06,"VBREAK"
0690      .CBYTE $19,$02,$08,"VKEYBD"
0700      .CBYTE $19,$02,$22,"VVBLKI"
0710      .CBYTE $19,$02,$24,"VVBLKD"
0720      .CBYTE $09,$02,$2B,"SRTIMR"
0730      .CBYTE $09,$02,$2F,"SDMCTL"
0740      .CBYTE $19,$02,$30,"SDLSTL"
0750      .CBYTE $09,$02,$32,"SSKCTL"
0760      .CBYTE $08,$02,$34,"LPENH"
0770      .CBYTE $08,$02,$35,"LPENV"
0780      .CBYTE $08,$02,$36,"BRKKY"
0790      .CBYTE $09,$02,$44,"COLDST"
0800      .CBYTE $09,$02,$6F,"GPRIOR"
0810      .CBYTE $79,$02,$70,"PADDL0"
0820      .CBYTE $39,$02,$78,"STICK0"
0830      .CBYTE $79,$02,$7C,"PTRIG0"
0840      .CBYTE $39,$02,$85,"STRIG0"
0850      .CBYTE $09,$02,$A2,"ESCFLG"
0860      .CBYTE $E9,$02,$A3,"TABMAP"
0870      .CBYTE $09,$02,$B6,"INVFLG"
0880      .CBYTE $09,$02,$BE,"SHFLOK"
0890      .CBYTE $09,$02,$BF,"BOTSCR"
0900      .CBYTE $09,$02,$C0,"PCOLR0"
0910      .CBYTE $09,$02,$C1,"PCOLR1"
0920      .CBYTE $09,$02,$C2,"PCOLR2"
0930      .CBYTE $09,$02,$C3,"PCOLR3"
0940      .CBYTE $09,$02,$C4,"COLOR0"
0950      .CBYTE $09,$02,$C5,"COLOR1"
0960      .CBYTE $09,$02,$C6,"COLOR2"
0970      .CBYTE $09,$02,$C7,"COLOR3"
0980      .CBYTE $09,$02,$C8,"COLOR4"
0990      .CBYTE $18,$02,$E0,"RUNAD"
1000      .CBYTE $19,$02,$E2,"INITAD"
1010      .CBYTE $09,$02,$E4,"RAMSIZ"
1020      .CBYTE $19,$02,$E5,"MEMTOP"
1030      .CBYTE $18,$02,$E7,"MEMLO"
1040      .CBYTE $39,$02,$EA,"DVSTAT"
1050      .CBYTE $09,$02,$F0,"CRSINH"
1060      .CBYTE $06,$02,$F2,"CH1"
1070      .CBYTE $08,$02,$F3,"CHACT"
1080      .CBYTE $08,$02,$F4,"CHBAS"
1090      .CBYTE $09,$02,$FB,"ATACHR"
1100      .CBYTE $05,$02,$FC,"CH"
1110      .CBYTE $09,$02,$FE,"DSPFLG"
1120      .CBYTE $09,$02,$FF,"SSFLAG"
1130      .CBYTE $09,$03,$00,"DDEVIC"
1140      .CBYTE $08,$03,$01,"DUNIT"
1150      .CBYTE $09,$03,$02,"DCOMND"
```

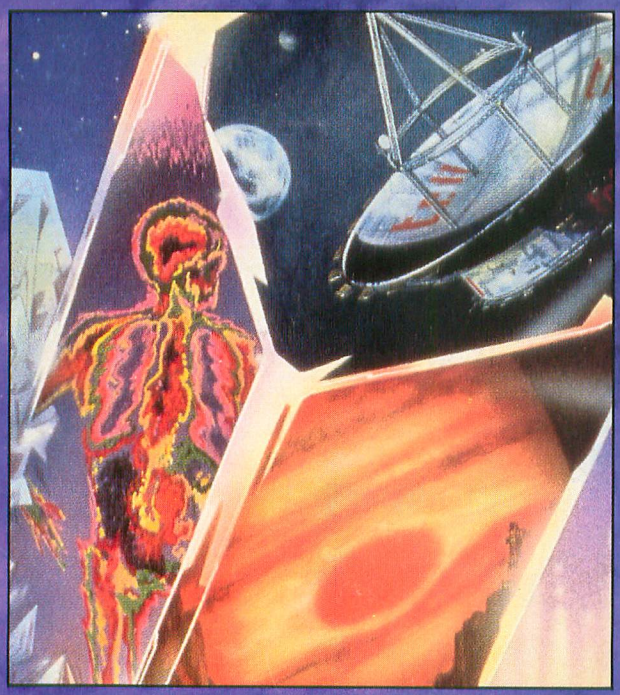
LISTING 7: ASSEMBLY

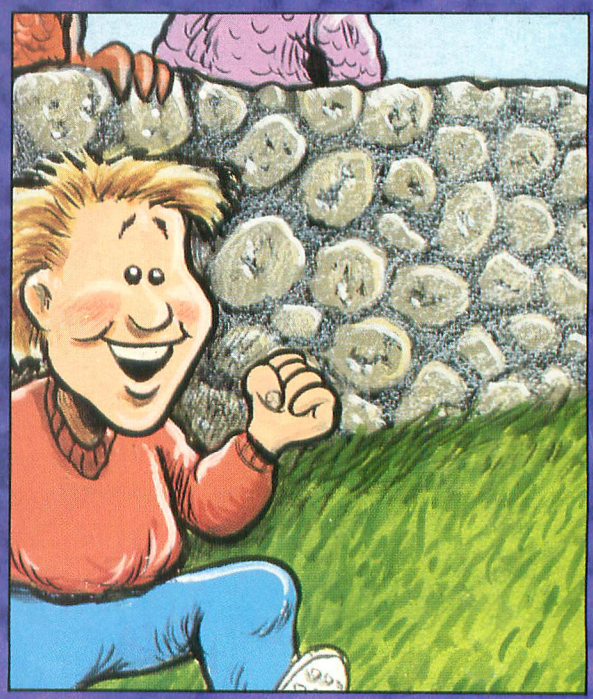
```
1160      .CBYTE $09,$03,$03,"DSTATS"
1170      .CBYTE $19,$03,$04,"DBUFLO"
1180      .CBYTE $09,$03,$06,"DTIMLO"
1190      .CBYTE $19,$03,$08,"DBYTLO"
1200      .CBYTE $18,$03,$0A,"DAUX1"
1210      .CBYTE $29,$03,$1A,"HATABS"
1220      .CBYTE $08,$03,$40,"ICHID"
1230      .CBYTE $08,$03,$41,"ICDNO"
1240      .CBYTE $08,$03,$42,"ICCOM"
1250      .CBYTE $08,$03,$43,"ICSTA"
1260      .CBYTE $08,$03,$44,"ICBAL"
1270      .CBYTE $08,$03,$45,"ICBAH"
1280      .CBYTE $08,$03,$46,"ICPTL"
1290      .CBYTE $08,$03,$47,"ICPTH"
1300      .CBYTE $08,$03,$48,"ICBLL"
1310      .CBYTE $08,$03,$49,"ICBLH"
1320      .CBYTE $08,$03,$4A,"ICAX1"
1330      .CBYTE $08,$03,$4B,"ICAX2"
1340      .CBYTE $09,$03,$C0,"PRNBUF"
1350      .CBYTE $09,$03,$FD,"CASBUF"
1360      .CBYTE $08,$05,$80,"LBUFF"
1370      .CBYTE $09,$D0,$00,"HPOSP0"
1380      .CBYTE $09,$D0,$01,"HPOSP1"
1390      .CBYTE $09,$D0,$02,"HPOSP2"
1400      .CBYTE $09,$D0,$03,"HPOSP3"
1410      .CBYTE $09,$D0,$04,"HPOSM0"
1420      .CBYTE $09,$D0,$05,"HPOSM1"
1430      .CBYTE $09,$D0,$06,"HPOSM2"
1440      .CBYTE $09,$D0,$07,"HPOSM3"
1450      .CBYTE $09,$D0,$08,"SIZEP0"
1460      .CBYTE $09,$D0,$09,"SIZEP1"
1470      .CBYTE $09,$D0,$0A,"SIZEP2"
1480      .CBYTE $09,$D0,$0B,"SIZEP3"
1490      .CBYTE $08,$D0,$0C,"SIZEM"
1500      .CBYTE $09,$D0,$12,"COLPM0"
1510      .CBYTE $09,$D0,$13,"COLPM1"
1520      .CBYTE $09,$D0,$14,"COLPM2"
1530      .CBYTE $09,$D0,$15,"COLPM3"
1540      .CBYTE $09,$D0,$16,"COLPF0"
1550      .CBYTE $09,$D0,$17,"COLPF1"
1560      .CBYTE $09,$D0,$18,"COLPF2"
1570      .CBYTE $09,$D0,$19,"COLPF3"
1580      .CBYTE $08,$D0,$1A,"COLBK"
1590      .CBYTE $08,$D0,$1B,"PRIOR"
1600      .CBYTE $09,$D0,$1D,"GRACTL"
1610      .CBYTE $09,$D0,$1E,"HITCLR"
1620      .CBYTE $09,$D0,$1F,"CONSOL"
1630      .CBYTE $08,$D2,$00,"AUDF1"
1640      .CBYTE $08,$D2,$01,"AUDC1"
1650      .CBYTE $08,$D2,$02,"AUDF2"
1660      .CBYTE $08,$D2,$03,"AUDC2"
1670      .CBYTE $08,$D2,$04,"AUDF3"
1680      .CBYTE $08,$D2,$05,"AUDC3"
1690      .CBYTE $08,$D2,$06,"AUDF4"
1700      .CBYTE $08,$D2,$07,"AUDC4"
1710      .CBYTE $09,$D2,$08,"AUDCTL"
1720      .CBYTE $09,$D2,$0A,"RANDOM"
1730      .CBYTE $08,$D2,$0E,"IRQEN"
1740      .CBYTE $08,$D2,$0F,"SKCTL"
1750      .CBYTE $08,$D3,$00,"PORTA"
1760      .CBYTE $08,$D3,$01,"PORTB"
1770      .CBYTE $08,$D3,$02,"PACTL"
1780      .CBYTE $08,$D3,$03,"PBCTL"
1790      .CBYTE $09,$D4,$00,"DMACTL"
1800      .CBYTE $09,$D4,$01,"CHACTL"
1810      .CBYTE $19,$D4,$02,"DLISTL"
1820      .CBYTE $09,$D4,$04,"HSCROL"
1830      .CBYTE $09,$D4,$05,"VSCROL"
1840      .CBYTE $09,$D4,$07,"PMBASE"
1850      .CBYTE $09,$D4,$09,"CHBASE"
1860      .CBYTE $08,$D4,$0A,"WSYNC"
1870      .CBYTE $09,$D4,$0B,"VCOUNT"
1880      .CBYTE $07,$D4,$0C,"PENH"
1890      .CBYTE $07,$D4,$0D,"PENV"
1900      .CBYTE $08,$D4,$0E,"NMIEN"
1910      .CBYTE $06,$D8,$00,"AFP"
1920      .CBYTE $07,$D8,$E6,"FASC"
1930      .CBYTE $06,$D9,$AA,"IFP"
1940      .CBYTE $06,$D9,$D2,"FPI"
1950      .CBYTE $07,$DA,$44,"ZFR0"
1960      .CBYTE $06,$DA,$46,"ZF1"
1970      .CBYTE $07,$DA,$60,"FSUB"
1980      .CBYTE $07,$DA,$66,"FADD"
1990      .CBYTE $07,$DA,$DB,"FMUL"
2000      .CBYTE $07,$DB,$28,"FDIV"
2010      .CBYTE $09,$DD,$40,"PLYEVL"
2020      .CBYTE $08,$DD,$89,"FLD0R"
2030      .CBYTE $08,$DD,$8D,"FLD0P"
2040      .CBYTE $08,$DD,$98,"FLD1R"
2050      .CBYTE $08,$DD,$9C,"FLD1P"
2060      .CBYTE $08,$DD,$A7,"FST0R"
2070      .CBYTE $08,$DD,$AB,"FST0P"
2080      .CBYTE $08,$DD,$B6,"FMOVE"
2090      .CBYTE $06,$DD,$C0,"EXP"
2100      .CBYTE $08,$DD,$CC,"EXP10"
2110      .CBYTE $06,$DE,$CD,"LOG"
2120      .CBYTE $08,$DE,$D1,"LOG10"
2130      .CBYTE $09,$E4,$00,"EDITRV"
2140      .CBYTE $09,$E4,$10,"SCRENV"
2150      .CBYTE $09,$E4,$20,"KEYBDV"
2160      .CBYTE $09,$E4,$30,"PRINTV"
2170      .CBYTE $09,$E4,$40,"CASETV"
2180      .CBYTE $09,$E5,$50,"DISKIV"
2190      .CBYTE $09,$E4,$53,"DSKINV"
2200      .CBYTE $07,$E4,$56,"CIOV"
2210      .CBYTE $07,$E4,$59,"SIOV"
2220      .CBYTE $09,$E4,$5C,"SETVBV"
2230      .CBYTE $09,$E4,$5F,"SYSVBV"
2240      .CBYTE $09,$E4,$62,"XITVBV"
2250      .CBYTE $09,$E4,$74,"WARMSV"
2260      .CBYTE $09,$E4,$77,"COLDSV"
2270      .BYTE $00
```

```
0100 ;-------------------------
0110 ;SYSTEM EQUATES FILE FOR
0120 ;ATARI HOME COMPUTERS
0130 ;Last Revised: 6/26/88
0140 ;-------------------------
0150 ;
0160 ;DISK I/O
0170 ;
0180 DSKINV = $E453
0190 SIOV =  $E459
0200 DDEVIC = $0300
0210 DUNIT = $0301   ;D: UNIT
0220 DCOMND = $0302  ;D: command
0230 DSTATS = $0303
0240 DBUFLO = $0304  ;sector buffer
0250 DBUFHI = $0305
0260 DTIMLO = $0306
0270 DBYTLO = $0308
0280 DBYTHI = $0309
0290 DSECLO = $030A  ;sector #
0300 DSECHI = $030B
0310 DAUX1 = $030A
0320 DAUX2 = $030B
0330 ;
0340 ;CIO ROUTINES
0350 ;
0360 CIOV =  $E456   ;CIO vector
0370 ICCOM = $0342   ;command
0380 ICBAL = $0344   ;buffer address
0390 ICBAH = $0345
0400 ICBLL = $0348   ;buffer len
0410 ICBLH = $0349
0420 AUX1 =  $034A
0430 AUX2 =  $034B
0440 RUNAD = $02E0
0450 INITAD = $02E2
0460 ;
0470 ;SCREEN HANDLER/INTERRUPTS
0480 ;
0490 VDSLST = $0200  ;DLI vector
0500 VBREAK = $0206  ;BRK vector
0510 VKEYBD = $0208  ;keyboard vec
0520 VVBLKD = $0224
0530 VVBLKI = $0222
0540 SRTIMR = $022B  ;key repeat
0550 SDMCTL = $022F  ;DMA control
0560 DMACTL = $D400
0570 SDLSTL = $0230  ;display list
0580 PCOLR0 = $02C0
0590 PCOLR1 = $02C1
0600 PCOLR2 = $02C2
0610 PCOLR3 = $02C3
0620 COLOR0 = $02C4
0630 COLOR1 = $02C5
0640 COLOR2 = $02C6
0650 COLOR3 = $02C7
0660 COLOR4 = $02C8
0670 CHBAS = $02F4   ;char set
0680 SETVBV = $E45C
0690 SYSVBV = $E45F
0700 XITVBV = $E462
0710 NMIEN = $D40E
0720 VCOUNT = $D40B
0730 CHBASE = $D409
0740 WSYNC = $D40A
0750 CHACT = $02F3
0760 CHACTL = $D401
0770 CRSINH = $02F0
0780 CHRSET = $E000  ;ROM characters
0790 HSCROL = $D404
0800 VSCROL = $D405
0810 IRQEN = $D20E
0820 ;
0830 ;MISC.
0840 ;
0850 CONSOL = $D01F  ;console keys
0860 RTCLOK = $14
0870 CH  =   $02FC
0880 MEMLO = $02E7
0890 SHFLOK = $02BE
0900 INVFLG = $02B6
0910 WARMSV = $E474
0920 COLDSV = $E477
0930 LBUFF = $0580
0940 EOL =   $9B
0950 RANDOM = $D20A
0960 PORTA = $D300
0970 PORTB = $D301
0980 ;
0990 ;Device Vectors
1000 ;
1010 HATABS = $031A
1020 EDITRV = $E400
1030 SCRENV = $E410
1040 KEYBDV = $E420
1050 PRINTV = $E430
1060 CASETV = $E440
1070 ;
1080 ;PAGE ZERO LOCATIONS
1090 ;
1100 CASINI = $02
1110 TRAMSZ = $06
1120 WARMST = $08
1130 BOOT? = $09
1140 DOSVEC = $0A
1150 DOSINI = $0C
1160 POKMSK = $10
1170 BRKKEY = $11
1180 SOUNDR = $41
1190 CRITIC = $42
1200 ATRACT = $4D
1210 KEYDEF = $79
```

```
1220 LMARGN = $52
1230 RMARGN = $53
1240 ROWCRS = $54
1250 COLCRS = $55
1260 SAVMSC = $58
1270 RAMTOP = $6A
1280 ;
1290 ;FLOATING POINT
1300 ;
1310 FR0 =   $D4
1320 FRE =   $DA
1330 FR1 =   $E0
1340 FR2 =   $E6
1350 FRX =   $EC
1360 EEXP =  $ED
1370 NSIGN = $EE
1380 ESIGN = $EF
1390 FCHRFLG = $F0
1400 DIGRT = $F1
1410 CIX =   $F2
1420 INBUFF = $F3
1430 RADFLG = $FB
1440 AFP =   $D800
1450 FASC =  $D8E6
1460 IFP =   $D9AA
1470 FPI =   $D9D2
1480 ZFR0 =  $DA44
1490 ZF1 =   $DA46
1500 FSUB =  $DA60
1510 FADD =  $DA66
1520 FMUL =  $DADB
1530 FDIV =  $DB28
1540 PLYEVL = $DD40
1550 FMOVE = $DDB6
1560 EXP =   $DDC0
1570 EXP10 = $DDCC
1580 LOG =   $DECD
1590 LOG10 = $DED1
1600 ;
1610 ;STICK, PADDLES, TRIGGERS
1620 ;
1630 PADDL0 = $0270
1640 PADDL1 = $0271
1650 PADDL2 = $0272
1660 PADDL3 = $0273
1670 STICK0 = $0278
1680 STICK1 = $0279
1690 PTRIG0 = $027C
1700 PTRIG1 = $027D
1710 PTRIG2 = $027E
1720 PTRIG3 = $027F
1730 STRIG0 = $0284
1740 STRIG1 = $0285
1750 ;
1760 ;AUDIO CONTROL
1770 ;
1780 AUDF1 = $D200
1790 AUDC1 = $D201
1800 AUDF2 = $D202
1810 AUDC2 = $D203
1820 AUDF3 = $D204
1830 AUDC3 = $D205
1840 AUDF4 = $D206
1850 AUDC4 = $D207
1860 AUDCTL = $D208
1870 SKCTL = $D20F
1880 ;
1890 ;P/M GRAPHICS
1900 ;
1910 PMBASE = $D407
1920 GPRIOR = $026F
1930 HPOSP0 = $D000
1940 HPOSP1 = $D001
1950 HPOSP2 = $D002
1960 HPOSP3 = $D003
1970 M0PF =  $D000
1980 M1PF =  $D001
1990 M2PF =  $D002
2000 M3PF =  $D003
2010 HPOSM0 = $D004
2020 HPOSM1 = $D005
2030 HPOSM2 = $D006
2040 HPOSM3 = $D007
2050 P0PF =  $D004
2060 P1PF =  $D005
2070 P2PF =  $D006
2080 P3PF =  $D007
2090 SIZEP0 = $D008
2100 SIZEP1 = $D009
2110 SIZEP2 = $D00A
2120 SIZEP3 = $D00B
2130 M0PL =  $D008
2140 M1PL =  $D009
2150 M2PL =  $D00A
2160 M3PL =  $D00B
2170 SIZEM = $D00C   ;missile size
2180 P0PL =  $D00C   ;player/pf
2190 P1PL =  $D00D
2200 P2PL =  $D00E
2210 P3PL =  $D00F
2220 COLPM0 = $D012
2230 COLPM1 = $D013
2240 COLPM2 = $D014
2250 COLPM3 = $D015
2260 COLPF0 = $D016
2270 COLPF1 = $D017
2280 COLPF2 = $D018
2290 COLPF3 = $D019
2300 COLBK = $D01A
2310 GRACTL = $D01D
2320 HITCLR = $D01E
2330 GRAFP0 = $D00D
2340 GRAFP1 = $D00E
2350 GRAFP2 = $D00F
2360 GRAFP3 = $D010
2370 GRAFM = $D011
```

INSIDE THIS ISSUE:





MORE
BOOT CAMP
PLUS
END USER ST NOTES