THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS

ANALOG COMPUTING

T.M.

FDC 50075

DECEMBER 1988
ISSUE 67

P
LAD

U.S.A. $3.95
CANADA $4.95

DungeonLords

Arcade fun!

★ Master Memory Map

★ Game Design Workshop

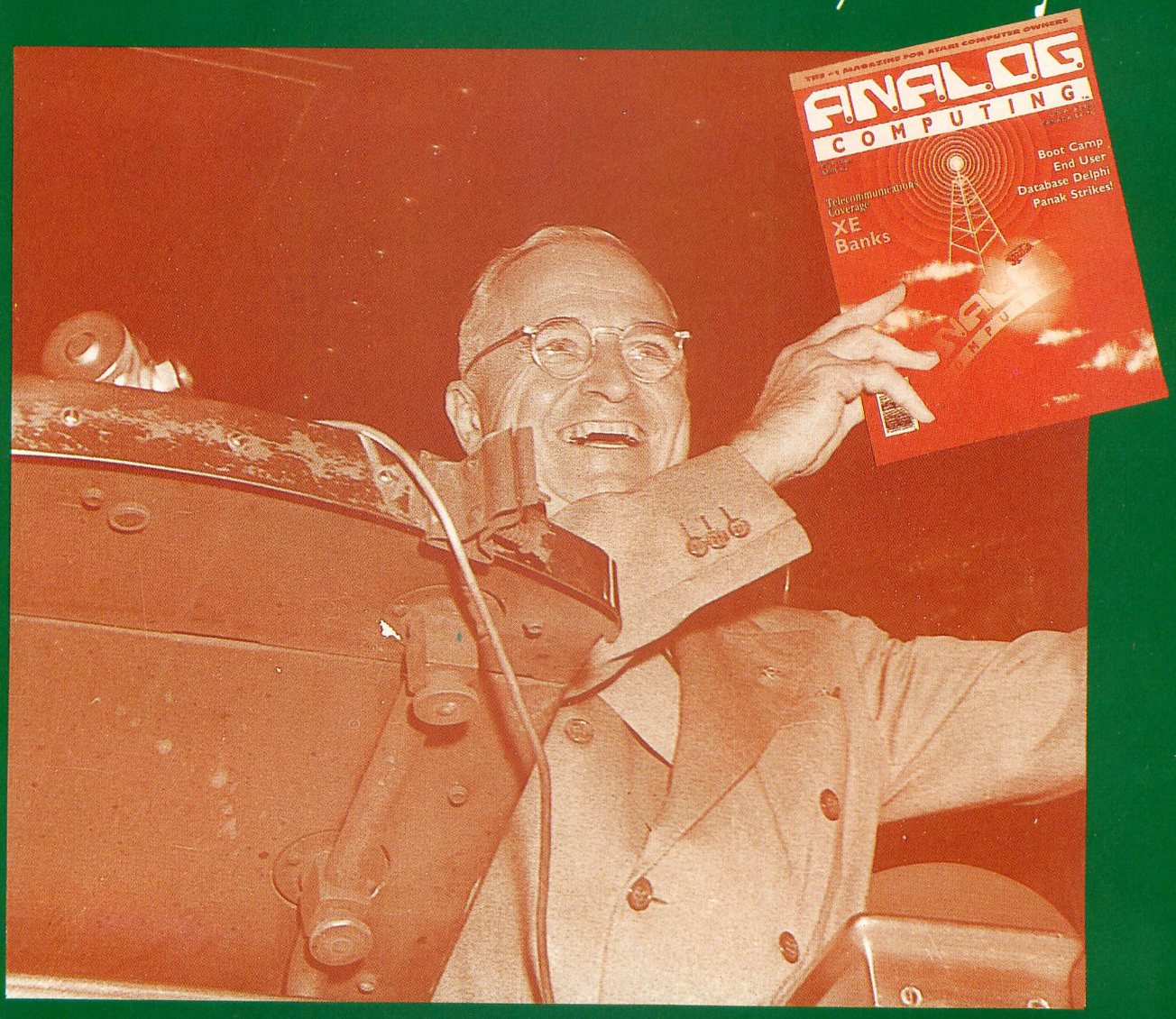
Reviews:

★ Sons of Liberty

★ Napoleon In Russia

0 74369 50075 4 12

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 $14 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 $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE!

Editorial

by Clayton Walnum

Here it is, Christmastime already. It seems only weeks ago that I dragged the tree ornaments up to the attic. Now I have to clomp back up there (Spiders, yuck!) and drag them all down again.

Is time moving faster? Or is it just me?

The reason I ask is it also seems like only yesterday that I bought my first Atari computer—while the calendar insists that that was almost eight years ago! Eight years! Back then there was no such thing as ANALOG Computing (Can you even imagine that?), though the original publishers, Lee Pappas and Michael DesChenes, were by then mulling over the idea of putting together an Atari newsletter to supplement their already thriving hardware and software store.

In the early 1980s the Atari 400s and 800s were thriving machines. Software packages were being released by the dozens every month, and it seemed as though the dream would last forever. Now as we face 1989, our Atari 8-bit machines—though they still look great to us—have become adrift in a sea of new technology, lost in the new wave as more and more software developers devote their attentions to the newer (and thus more profitable), more sophisticated machines.

If you're still out there doing Christmas shopping for someone on your Atari gift list, however, you'll find that there really is a silver lining in every cloud. All the programs that we original 8-bit owners paid $30 or more for a few years ago can now be found in many discount bins for a fraction of their original cost; sometimes for even less than $5!

It's old software, but it's still great software. If there's an Atari software dealer near you, you owe it to yourself (and those people on your Christmas list) to get down there and see what he may have to offer. In these days when new 8-bit software is as rare as the dodo bird, I can't think of anything that'll bring a smile to a new Atari 8-bit owner more than a box filled with classic software from days gone by. Can you?

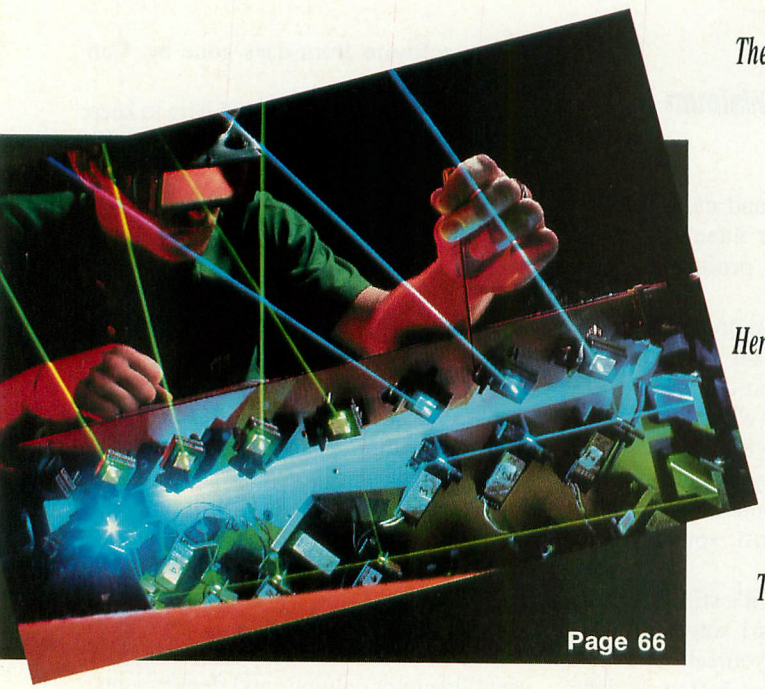
Of course, ANALOG is still here to keep all 8-bit owners supplied with helpful information and fine programs. Our files are bursting with great things for 1989, and this issue, though still bearing a 1988 cover date, is no exception.

Game lovers will delight to the fast action and the sensational graphics found in Brian Bradley's *DungeonLords*. Also, Robin Sherer's *Master Memory Map* reaches its halfway point with this issue. Nowhere within the pages of a magazine will you find a more complete guide to your Atari's internals. Action! programmers should not overlook (not that you would) Monty McCarty's *Action! Graphics Toolkit*, a series of Action! graphics routines that you can incorporate into your own programs. And, of course, you'll also find our usual lineup of columns and departments.

So with this issue we bid a farewell to 1988. But 1989 is right around the corner, and with a partnership like ours—you and ANALOG Computing—it can only be a great year.

Conte

FEATURES



Page 66

n t s

DECEMBER 1988
ISSUE 67

REVIEWS

COLUMNS

DEPARTMENTS

ANALOG COMPUTING STAFF

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 $7 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**

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 $7 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.

Attention Programmers!

ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing family of Atari computer owners.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. Typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG Computing**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number.

Send your programs and articles to:
ANALOG Computing
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

READER COMMENT

Kudos for the doctor

Lee Brilliant is! Brilliant, that is! His Atari Zucchini series may just be the answer to turning around the waning 8-bit support you referred to in Issue 62's editorial. Well, perhaps that is a bit of an overstatement, but. . .the article is exciting, and I await each edition eagerly. Keep Lee Brilliant writing these articles. Though I don't usually understand all the subtleties, I invariably learn a little from each one.

As to the use of our 8-bit, or perhaps I should say 8-bit*s* (we have two 800s, two 800XLs and a 130XE), we run the gamut. I manage a trust using AtariWriter for the correspondence, and VisiCalc and SynCalc for the book work.

I also do a fair amount of military writing using the Atari 8-bits. I agree with Arthur Leyenberger that my favorite peripheral is a Tandy 100 laptop. I've used it for trip reports, as well as correspondence.

I also used the Ataris to prepare a book of poetry for publication. It was written by my late grandmother in the last three years of her life. She lived to be 97!

Your magazine is super. I won't tell you it is the only Atari magazine I read, but the articles and ads always keep my interest. And the letters in "Reader Comment" are the first things I read after the "Table of Contents."

Good luck, and keep publishing for the good old Atari 8-bit. —Jim Cox
Colorado Springs, CO

A letter for Logo and Pilot users

In response to your recent editorial about Atari users: For the past two years, my kids and I have been programming with the two most underrated and unappreciated languages in the history of personal computing, namely Atari Logo and Atari Pilot. (Anybody remember them?) The Atari implementations of these two gems are easily superior to any other for an 8-bit machine. If Atari had been on the ball nine years ago, our schools would surely be full of Ataris rather than Apples. As it stands, the world has certainly bypassed these two languages, and it's a real shame.

I've often said to friends that, as amateurs, we could easily use up two lifetimes just to explore all the possibilities of our two cartridges and our $100 computer. While I doubt that any of us will ever become as adept or talented as the individuals who grace the pages of ANALOG, we nevertheless enjoy doing our projects and making our mistakes. We compute more for mental recreation than with the hopes of producing anything really useful. In fact, I sometimes think that the most interesting aspects of both Logo and Pilot are their inherent uselessness! One would never bother writing a fast-action game, much less a disk utility or a spreadsheet program, in either language—they are much too limited and slow. They exist for different purposes altogether, but what they do, they do very well indeed.

Pilot was written with the sole purpose of making it possible for an amateur to write useful educational programs. Period. Originally it was hoped that teachers would use Pilot to create specialized material for their students; but this never happened to any great extent for a variety of economic and social reasons. Pilot remains, however, a wonderful tool for the novice. I have had a great deal of enjoyment creating word and language games in Pilot for my kids and their friends, and I find that they are beginning to try their own hands at it, largely because they can easily (at six and eight years of age) begin to comprehend the logic behind the creation of a program. One needs a lot of simplicity at that age, and I feel that Pilot's clarity makes all the difference.

Logo can be a more demanding enterprise. I'm often enthralled by the range and the elegance of the language, much of which it derives from its daddy, LISP, the first language developed to explore artificial intelligence.

To program in Logo is to be hopelessly drawn into a world of logic and behavior, movement and vector geometry, visions of "flatland" and two-dimensional robotics. The project I have been hung up on lately is trying to simulate the behavior of a living organism within the confines of the Logo microworld. It's hard not to get philosophical when you start fooling around with this stuff!

The kids, however, regard the Logo turtle as a kind of family pet, residing in a strange, colorful universe, and have taught him (or her, depending on who is at the keyboard) to perform various feats of wonder and derring-do. And because the turtle lives in a world created largely by mathematical calculation, they get a good mental workout when they try to get the little guy to do anything.

So that's what one family is doing with its computer. I'm hoping that this letter may encourage other closet Logo and Pilot users to drop us a line and let us know what they are up to. Perhaps someone out there has managed to get a machine-language subroutine to run in Logo (heaven knows, I haven't) or created some other bit of virtuosity (or frivolity) that they would like to share. Perhaps we can swap disks and all get some new ideas! Jenny, Adam and I would love to hear from you.

—*Craig Rothfuss*
37 Broad St. #10
Freehold, NJ 07728

MIDI file standard accepted

A new standard file format for sequencers has been accepted by the International MIDI Association (IMA). After many months of proposals and discussions, the *MIDI Files* format was ratified at the June NAMM trade show. MIDI Files allow MIDI-compatible sequencers, music printing programs and composition software to easily share data with other programs. Through the use of MIDI Files, music can be written in one program, edited in another, transferred to a printing program for transcription and sent to yet another for playback. Passport, Hybrid Arts and Digidesign already have pledged their support of MIDI Files with new versions of their products to be released in the near future.

International MIDI Association
5316 W. 57th St.
Los Angeles, CA 90056
(213) 649-6434

CIRCLE #146 ON READER SERVICE CARD.

Panelologists unite!

Comicbooking Your Atari is a new XE/XL software package specially designed for panelologists (i.e., comic book collectors). At $19.95, the new system offers collectors an easy-to-use database to track and maintain information about comic book collections. The system focuses on appearance, title, issue number, cost, value and other data pertaining to a collector's library. The system works well for speculation; valuation of a comic book can be made at purchase or sale and to determine an asking price.

The 20-page user's manual reads like a beginning course in data management. The user is taken step-by-step through the system's many features and functions. "The program is so user-friendly," says A.L. Bue, the developer, "that most Atarians will have little need of the manual to get started."

Bueco
3900 Hampton Dr.
Anchorage, AK 99504

CIRCLE #147 ON READER SERVICE CARD.

Printing in color

Users of *The Print Shop*, Broderbund Software's popular desktop publishing program for the XE/XL, can now add special color graphics to their Print Shop documents with the *Color Print Graphics Disk* from WJA Software. WJA is offering two-color graphics diskettes: Disk 1 contains 100 graphics of a general nature that create 50 two-color graphics. Disk 2 is of a holiday and special occasion theme. Finished two-color graphics are printed using color ribbons, passing the paper through your printer twice. The disks sell for $6.95 each or $11.95 for both.

WJA Software
26 Hunters Ln.
Hendersonville, NC 28739

CIRCLE #143 ON READER SERVICE CARD.

GEO meets with Atari

Discussions between Merrill Ward Associates, the designers of the new *GEO* desktop operating system for the XE/XL, are continuing. Both Atari and Merrill Ward hope that GEO will become the accepted new standard operating system of the XE/XL computer. *GEO* does for the XE/XL what *GEOS* did for the Commodore 64, giving an 8-bit microcomputer an operating system similar to the GEM system on the 16-bit ST computer.

Although no agreement has been reached, Atari hopes to distribute GEO to the XE/XL community. An upgrade path is being considered for existing owners.

Merrill Ward & Associates
255 North El Cielo Rd., Ste. 222
Palm Springs, CA 92262

CIRCLE #142 ON READER SERVICE CARD.

RAMdisk protection

What do you do when your XE/XL computer crashes while you are using a RAMdisk? Pressing the reset button does nothing; so your only other recourse is to power down your computer: You lose the documents, data, and other important information stored on the RAMdisk. *The Ramdisk Protector*, a hardware/software combination, tricks your XE/XL computer into thinking that the power has been turned off, then back on. It is valuable even without a RAMdisk. The software enables DOS to recognize and provide immediate access to a RAMdisk. Both functions are entirely transparent.

The Protector installs without tools for push-button control. Included are DOS enhancements for support of Atari DOS 2.5 and SpartaDOS. The Ramdisk Protector has a list price of $22.95.

Logic One
P.O. Box 18123
Cleveland, OH 44118-0123

CIRCLE #144 ON READER SERVICE CARD.

Icon conversion utility

Users of *Print Shop*, *Newsroom*, *Awardware* and *Printpower* can share graphics and icons using *The Converter*, a new graphics utility program from No Frills Software. Icons and graphics can be converted from their original formats to any of the other formats needed by the popular desktop-publishing programs on the XE/XL. The Converter also allows materials to be added to Print Shop icons, so the "larger format" picture programs will have better looking graphics.

The Converter includes a graphics editor that can be used to create your own graphics from scratch. The Converter is now available for $22.95. The program comes with an XE/XL diskette and instructions on its usage.

No Frills Software
800 East 23rd St.
Kearney, NE 68847
(308) 234-6250

CIRCLE #145 ON READER SERVICE CARD.

8-Bit NEWS

New NERDS

A new company in the 8-bit Atari industry, the National Educational Report Drawing Services (or NERDS as they like to be called) has released the second two of a series of graphic data disks for use with *The Print Shop*. The graphic disks contain 115 biological illustrations for use in educational applications. Disk One contains icons of human and animal circulatory, digestive, endocrine, respiration and reproductive systems. Disk Two contains illustrations of microbiology, insects, basic biochemistry and plants.

NERDS has packaged the graphic sets in easily modifiable form, so *Print Shop* users will be able to accent the illustrations to their own needs. Each disk is priced at $15 and is available directly from NERDS. Site licensing for user groups and Lab Packs containing multiple copies for school are also available at special pricing.

NERDS also produces a two-disk blank-map set, periodic table of elements and *Quick Pix* conversions of the map or biology sets for either *Atariwriter* or *Paper Clip* usage for only $12 each set.

NERDS
C/O D. Loeffler
18 Wendy Dr.
Farmingville, NY 11738

CIRCLE #148 ON READER SERVICE CARD.

Newfangled joystick

Wico has introduced a new joystick for Atari computers. The *Ergostick* is a ergonomically designed joystick that is soft, pliable and form-fits to the human hand. The new joystick fits comfortably in your left or right hand and uses high-tolerance microswitches for high-speed responses. The Ergostick prevents such painful maladies as "Millipede blisters" and "PacMan wrist."

The lower portion of the joystick is made of a soft plastic that can be easily gripped. Your forefinger is used to push a fire button that is imbedded into the bottom of the joystick. At $19.95, the Ergostick is an affordable addition to your XE/XL system.

Wico Corporation
6400 W. Gross Point Rd.
Niles, IL 60648
(312) 647-7500

CIRCLE #141 ON READER SERVICE CARD.

Full-featured bulletin-board system

Carina II is an expandable, full-featured, modifiable bulletin-board system (BBS) for the XE/XL computer. Carina II uses Special Interest Groups (SIGs) to allow users to post messages, receive electronic mail, upload/download files and even play interactive games.

The system was written in Atari BASIC; so it is easily modified by BBS operators with a little programming experience. Though Carina II was written in BASIC, its operations are of the machine-language level. The system supports XMODEM file transfers and even the new YMODEM transfer protocol.

Networking will soon be implemented for BBS operators with more than one computer. Carina Software will offer the networking software at no additional charge. Carina II requires *SpartaDOS 2.3* or greater and a 500K storage device.

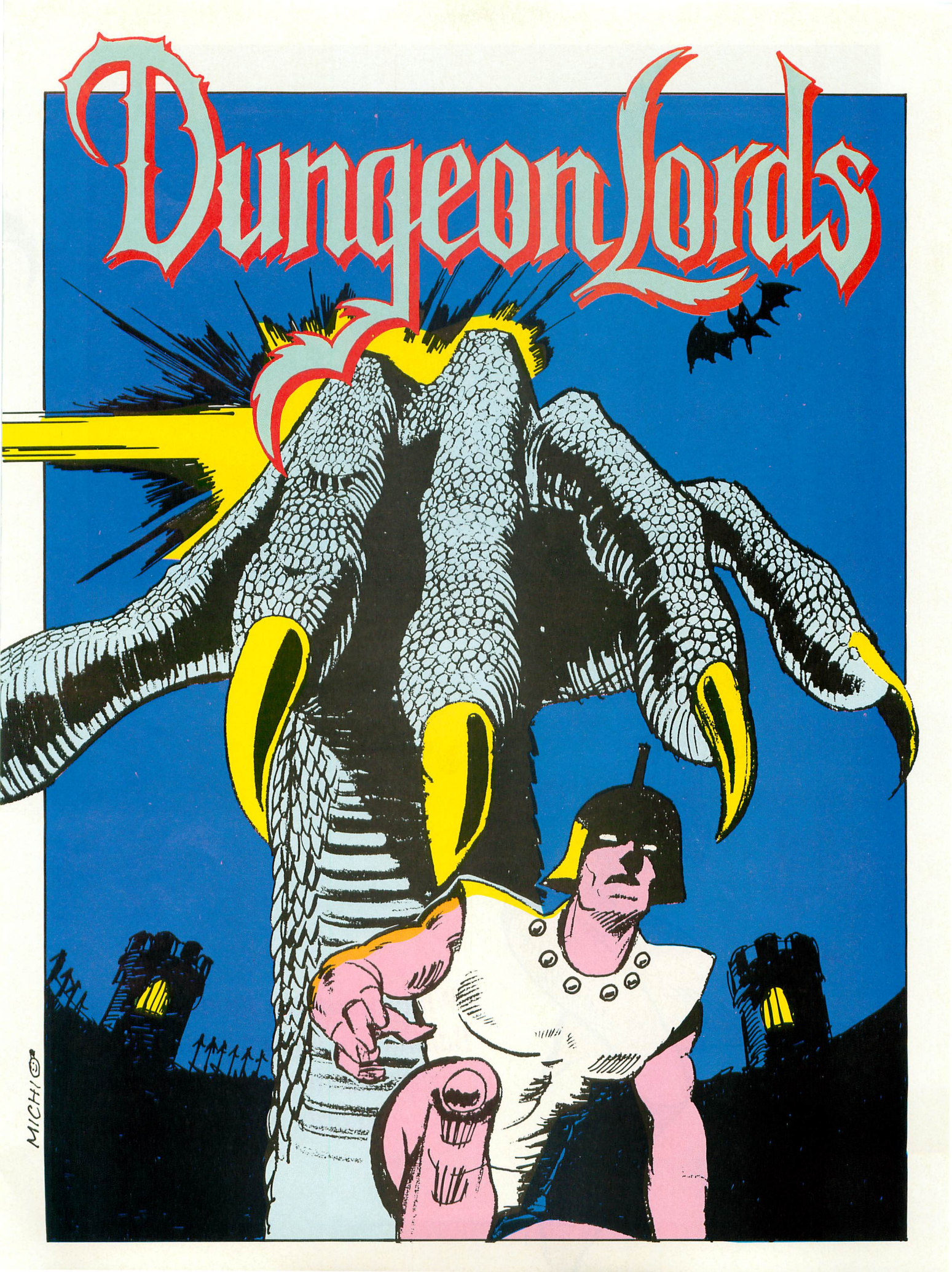
Carina II is available now for only $65.00 (including shipping and handling). A multi-user online game, *Robowar II*, is available for Carina II. This is a five-module game that costs only $15 additionally when ordering the Carina II system.

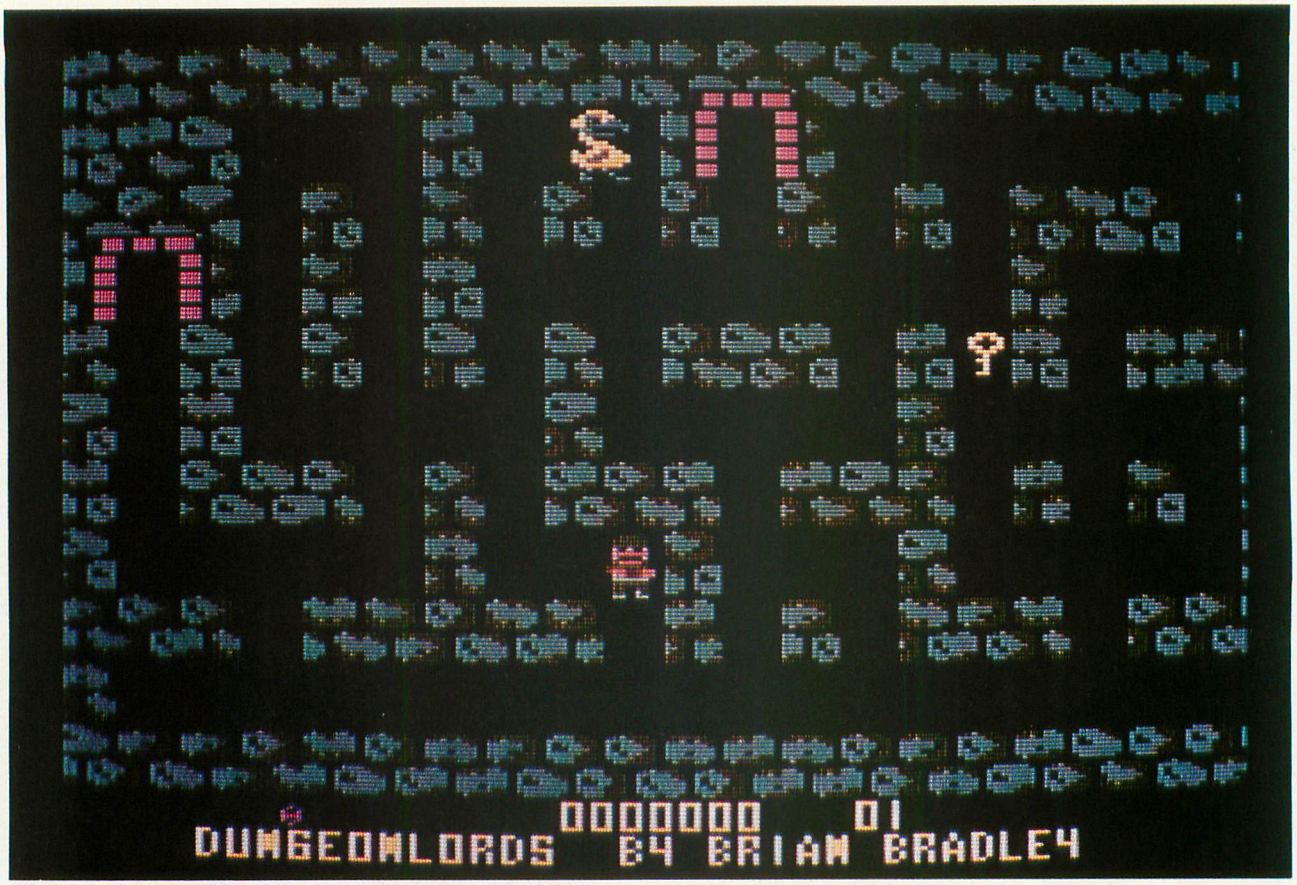
Carina Software Systems
322 Natchez Ct.
Jupiter, FL 33477
(407) 747-9195 (voice)
(407) 747-9196 (BBS demonstration)

CIRCLE #140 ON READER SERVICE CARD.

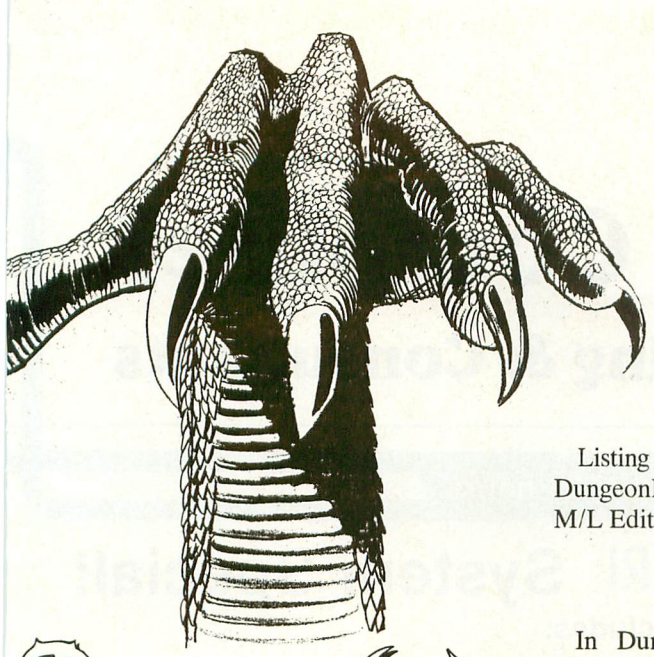
by Brian Bradley

Your name is Jon Russel. You're a freelance soldier, not to mention a rogue and a scoundrel to boot. That's okay, though, your friends are the same way. So what are you doing here, sweat dripping from your brow, disrupter pistol at the ready? Well, it seems that during your travels through Space-time with your friends, you have discovered the world of the *DungeonLords*. Furthermore, while you were hunting a Bantha for dinner, your friends were taken prisoner by the DungeonLords and locked away in the dungeons. So what does a true, blue-blooded freelance rogue like yourself do? What else can do you? Traveling through Space-time by yourself can be lonely. So here you are and before you lie the dungeons: evil, dank and smelly. Do you have your pistol ready?


Dungeon Lords
MICHI



While you were hunting for a Bantha for dinner, your friends were taken prisoner by the DungeonLords and locked away.



DungeonLords

Typing it in

Listing 1 is the BASIC data used to create DungeonLords. You should refer to the M/L Editor article for typing instructions.

How to play

In DungeonLords, each dungeon is created randomly so that you should never encounter the same dungeon twice. The object is to rescue as many of the prisoners as possible. Therefore you will want to get to the next level as quickly as possible. To do this, find all the keys around the maze and use them to unlock the doors guarding the exit. Remember, only one key can be carried at a time.

Every fourth level is the Prison level. To exit this level, you must first rescue the prisoner by touching him. When this feat is accomplished, the exit will appear at the other end of the maze.

There are three types of monsters: snakes, birds and horned demons. Killing any one of these will get you one point. Exiting a dungeon gives you 1,000 points, as does rescuing a prisoner. You may also collect 100 points for each treasure collected around the maze. Every 10,000 points gives you an extra life. The score is displayed at the bottom of the screen. To the right of the score is the level, and to the left are the remaining lives.

Monsters do not shoot, but are deadly to the touch. Each monster will enter the maze via a transporter. These are glowing doorways scattered around the maze. However, monsters will only appear in a transporter when it is glowing red. When it is pulsating blue, it is fairly safe to be around. Once the monsters enter the maze they will wander around the corridors in search of you. If one enters the same corridor you are in and he doesn't have his back to you, he will certainly spot you. If this happens, the monster that sees you will raise an inaudible alarm telling his comrades where you are. Then all of his friends will converge upon the spot you were seen last. Remember, it's not where you are, but where you were seen last! Use this to your advantage. If you wish to lure the monsters away from an area, allow one of them to see you and follow you away from that area; then kill all of the monsters that saw you and escape. The remaining monsters will rush to where they heard you were last. When they discover that you are no longer there, they will start searching for you starting from that point. Hopefully by that time, you will have made it to where you were going.

To control your on-screen character, use a joystick in Port 1. Simply push this joystick in the desired direction, and Jon Russel will move that way. He will continue moving in that direction until he runs into something, or you change direction. To shoot you have an option; you can use the same joystick for movement as for shooting. To do this, push the fire button down (keep it down), and push the joystick in the direction that you wish to fire. Remember, though, when the button is pressed the joystick controls the firing direction not the movement. The player will continue moving in the direction that he was moving in before the button was pressed. This allows you to move in one direction and fire in another. The alternative for firing is to use the joystick in Port 1 for movement and the joystick in Port 2 for firing. If you do this, you do not need to push the fire button on the second joystick. Simply push it in whatever direction, and that will be the direction in which you are shooting. The pistol that you are using is a unifield disrupter pistol that works on the principle of a concentrated disruption in four-space. This means that you can only have one shot in the air at a time, which leaves you defenseless until the disruption hits something. Therefore, shorter corridors are safer than longer ones, because you can shoot faster.

To make play easier, you can have a friend join you. One person can handle movement on the first joystick, and the other can take control of firing on the second joystick.

Okay, are we all ready to enter the dungeons? Good! Just press Start, and we are off!

continued on page 48

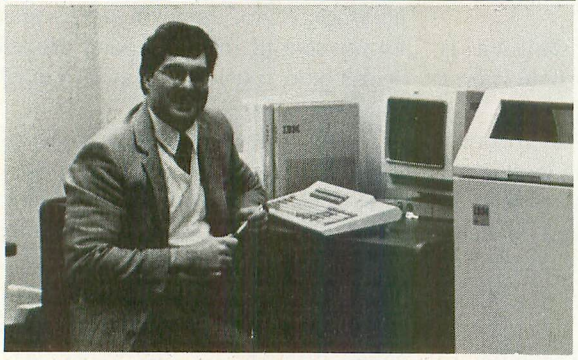
Lyco Computer

Marketing & Consultants

Since 1981

Air orders processed within 24 hours.

Lyco Means Total Service.



Mark "Mac" Bowser, Sales Manager

I would personally like to thank all of our past customers for helping to make Lyco Computer one of the largest mail order companies and a leader in the industry. Also, I would like to extend my personal invitation to all computer enthusiasts who have not experienced the services that we provide. Please call our trained sales staff at our toll-free number to inquire about our diverse product line and weekly specials.

First and foremost our philosophy is to keep abreast of the changing market so that we can provide you with not only factory-fresh merchandise but also the newest models offered by the manufacturers at the absolute best possible prices. We offer the widest selection of computer hardware, software and accessories.

Feel free to call Lyco if you want to know more about a particular item. I can't stress enough that our toll-free number is not just for orders. Many companies have a toll-free number for ordering, but if you just want to ask a question about a product, you have to make a toll call. Not at Lyco. Our trained sales staff is knowledgable about all the products we stock and is happy to answer any questions you may have. We will do our best to make sure that the product you select will fit your application. We also have Saturday hours - one more reason to call us for all your computer needs.

Once you've placed your order with Lyco, we don't forget about you. Our friendly, professional customer service representatives will find answers to your questions about the status of an order, warranties, product availability, or prices.

Lyco Computer stocks a multimillion dollar inventory of factory-fresh merchandise. Chances are we have exactly what you want right in our warehouse. And that means you'll get it fast. In fact, orders are normally shipped within 24 hours. Free shipping on prepaid orders over $50, and there is no deposit required on C.O.D. orders. Air freight or UPS Blue/Red Label shipping is available, too. And all products carry the full manufacturer's warranties.

I can't see why anyone would shop anywhere else. Selection from our huge inventory, best price, service that can't be beat — we've got it all here at Lyco Computer.

TO ORDER, CALL TOLL-FREE: 1-800-233-8760
New PA Wats: 1-800-233-8760
Outside Continental US Call: 1-717-494-1030

Hours: 9AM to 8PM, Mon. - Thurs.
9AM to 6PM, Friday — 10AM to 6PM, Saturday

For Customer Service, call 1-717-494-1670,
9AM to 5PM, Mon. - Fri.
Or write: Lyco Computer, Inc.
P.O. Box 5088, Jersey Shore, PA 17740

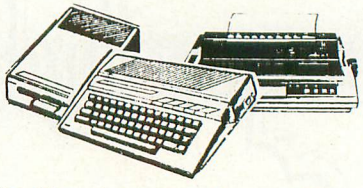
MasterCard VISA

PLEASE NOTE: • full manufacturers' warranties • no sales tax outside PA • prices show 4% cash discount; add 4% for credit cards • APO, FPO, international; add $5 plus 3% for priority • 4-week clearance on non-certified checks • we check for credit card theft • sorry, compatibility not guaranteed • return authorization required • due to new product guarantee, return restrictions apply • price/availability subject to change • prepaid orders under $50 in Continental US, add $3.00.

ATARI System Special!

System Includes:

- 130 XE Computer
- 551 Drive
- Seikosha SP-180 AI Printer

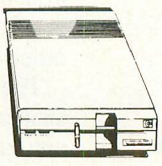


$429.95

Call For More ATARI Hardware Information.

INDUS GTS-100

- Atari ST Drive
- 3.5" DSDD



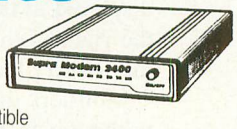
$195.95

ATARI Cartridge Software

3 for $9.95

CALL FOR TITLES!

SupraModem 2400



- fully compatible with industry-standard, intelligent "AT" commands
- Compact size
- 1-year warranty

$129.95

EPYX 500 XJ

$13.95

HEADSTART COLOR SYSTEM

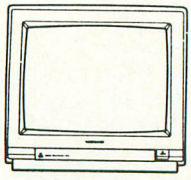
- plug in and use immediately
- IBM-XT compatible
- 2-360K Drives
- Free 1-year limited warranty

$969.95

• Hi Res color monitor included!

MAGNAVOX CM-8502

- Composite Color
- Green Text Switch
- Speaker
- Suggested Use 130 XE

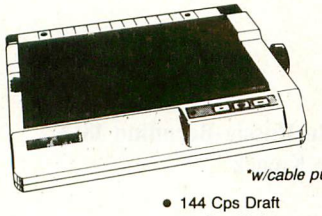


$179.95

1-800-233-8760

CIRCLE #103 ON READER SERVICE CARD.

star micronics NX1000



*w/cable purchase

- 144 Cps Draft
- 36 Cps NLQ
- EZ Soft Touch Selection
- Paper Parking
- Epson Std. & IBM Proprinter II Compatible

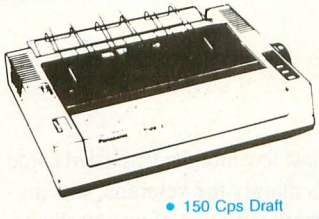
$169.95

NX-1000	$169.95*
NX-1000C	$169.95
NX-1000 Color	$225.95
NX-1000C Color	$229.95
NX-15	$289.95
NR-10	$319.95
NR-15	$419.95
NB-15 24 Pin	$669.95
NX-2400	$309.95
ND-15	$349.95

*w/cable purchase

Panasonic Office Automation

1080i Model II



- 150 Cps Draft
- Friction & Tractor Feed Std.
- Bidirectional & Logic Seeking
- NLQ in all Pitches

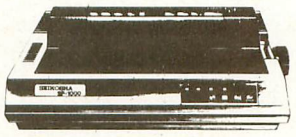
$149.95*

*quantities limited

1080i Model II	$149.95*
1091i Model II	$195.95
1092i	$309.95
1592	$375.95
1595	$439.95

*quantities limited

SEIKOSHA Sp180Ai



- 100 Cps Draft
- 24 Cps NLQ
- Tractor & Friction Feed
- Epson FX & IBM Graphic Compatible

$125.95*

*quantities limited

SP 180Ai	$125.95*
SP 180VC	$125.95*
SP 1000VC	$139.95
SP 1000AP	$159.95
SP 1200VC	$149.95
SP 1200Ai	$159.95
SP 1200AS RS232	$179.95
SL 80Ai	$329.95
MP5420FA	$999.95
SP Series Ribbon	$7.95
SK3000 Ai	$349.95
SK3005 Ai	$445.95
SPB 10	$CALL
SL 130Ai	$599.95
SP 1600Ai	$CALL

PRINTERS

Citizen

120 D	$144.95
180 D	$159.95
MSP-40	$279.95
MSP-15E	$309.95
MSP-50	$369.95
MSP-45	$349.95
MSP-55	$469.95

Epson

LX800	$184.95
FX850	$339.95
FX1050	$499.95
EX800	$434.95
LQ500	$339.95
GQ3500	$LOW
LQ850	$525.95
LQ1050	$749.95

Attention Educational Institutions:

If you are not currently using our educational service program, please call our representatives for details.

Okidata

Okimate 20	$129.95
Okimate 20 w/cart	$189.95
120	$189.95
180	$219.95
182	$209.95

Brother

M1109	$189.95
M1509	$335.95
M1709	$439.95
Twinwriter 6 Dot & Daisy	$899.95

INTERFACING AVAILABLE

ATARI

Access:
Triple Pack $11.95
Leader Board Double Pack $9.99

Activision:
Music Studio $19.95
Great American Road Race $9.99

Broderbund:
Print Shop $26.95
Graphic Lib. I, II, III $14.95

Electronic Arts:
Pinball Con. Set $8.95
One on One $8.95
Lords of Conquest $8.95
Super Boulderdash $8.95
Music Construction Set $8.95

Microleague:
Microleague Baseball .. $22.95
GM Disk $16.95

Microprose:
F-15 Strike Eagle $19.95
Silent Service $19.95

Mindscape:
Gauntlet $20.95

DEALER INQUIRIES INVITED

ATARI ST

Access:
Leader Board $22.95
10th Frame $22.95

Activision:
Music Studio $27.95

Broderbund:
Superbike Challenge ... $11.95

Electronic Arts:
Alien Fires $25.95
Hunt for Red October .. $32.95

Epyx:
Dive Bomber $22.95
Impossible Mission 2 ... $22.95
Winter Games $11.95

Firebird:
Universal Military Simulator $28.95
Carrier Command $25.95

Microleague:
Microleague Baseball .. $33.95
Micro. Wrestling $25.95

Microprose:
F-15 Strike Eagle $24.95
Gunship $28.95

Mindscape:
Balance of Power $28.95
Harrier Combat Simulator $28.95

ATARI ST

Origin:
Autoduel $24.95
Ultima IV $34.95

Strategic Simulations:
Questron II $32.95
Stellar Crusade $35.95

Sublogic:
Flight Simulator II $30.95
Jet $34.95

Timeworks:
Wordwriter ST $44.95
Partner ST $27.95

Unison World:
Art Gallery 1, 2 or 3 .... $14.95
Print Master $19.95

We stock over 3,000 software titles!

Video Tape

SKC T120 VHS Video Tape:

each	$3.99
3 pack	$10.95
10 pack	$35.95

Monitors

Magnavox:

BM7652	$79.95
BM7622	$79.95
7BM-613	$79.95
7BM-623	$79.95
CM8502	$179.95
9CM-053	$339.95
CM8762	$239.95
8CM-515	$CALL
8CM-873	$CALL
9CM-082	$439.95

Thomson:

4120 CGA	$199.95*
GB 100	$119.95*
GB 200 Super Card	$169.95*

*quantities limited

Joysticks

Winner 909	$24.95
Wico IBM/AP	$29.95
Lipstick Plus	$14.95
Kraft KC III Ap/PC	$16.95
Kraft PC Joystick Card	$27.95
Kraft Maze Master	$8.95
I Controller	$13.95
Epyx 500 XJ	$13.95

Modems

Avatex:

1200e	$65.95
1200i PC Card	$65.95
1200p	$89.95
1200hc Modem	$89.95
2400	$149.95
2400i PC Card	$139.95

Hayes:

Smartmodem 300	$139.95
Smartmodem 1200	$279.95
Smartmodem 2400	$419.95

US Robotics:

Courier 1200	$169.95
Courier 2400	$299.95

Supra:

300	$119.95*
1200	$119.95*
2400	$129.95

*limited quantities

Disc Storage

QVS-10 5¼	$3.95
QVS-75 5¼	$10.95
QVS-40 3½	$9.95

Join the thousands who shop Lyco and Save

In case you haven't noticed, for the past couple of months we've been a little short on new software. Unfortunately, this has been almost exclusively an Atari 8-bit phenomenon, and one I'm at a loss to completely explain. In the first place, I have trouble understanding the vast consumer interest in the Sega and Nintendo game systems. I mean, they have great games, great graphics—but that's all there is to them. They play arcade games. They'd have a lot of trouble dealing with the more complex games available on any of a handful of computer systems. And don't even think of asking them to help you with that late term paper.

I can understand some shift of resources to the ST. It is representative of the machines of the future; machines that enable you to play games you could only dream about a few years ago. But machines like the ST also carry rather high price tags, putting them out of the reach of the vast majority of home users. At least some people consider them high. But such is not the case in the 8-bit category. Used machines abound, and a prudent shopper can set himself up for around $200. So what's the problem?

Evidently the manufacturers don't perceive that there is a market. Unfortunately, this sticks us Atari users in a catch-22 situation. In order to cast our consumer vote, we need products—products the publishers won't produce without a vote tally. So what's to be done? The answer is simple. We've got to cast our vote in an equally effective way. Write a letter.

When you see a game you think you'd like, and all you see is a Commodore 64 version, write the publisher and request it. If you've got a word processor, the form can be set up easily enough and used repeatedly. It will be recognized as a form letter, but the vote will count nonetheless. If you don't have a word processor, scrawl it out in your own handwriting. And remember to make it legible. If enough letters are received, action will be taken.

As for newcomers, you need not be concerned or frightened away from the Atari, as there exists a large catalog of great games from scores of publishers. In fact, sporadically over the last few months I've listed some of the best. It's mainly the veterans, like myself, who notice the drought of new products.

And while you're all sharpening your pencils, I'll tell you that at least one publisher is taking a chance, showing some sign of a semi-solid spine, by introducing a new war simulation. This is an intrusion directly into SSI's territory, and hence I'll pit this newest product against SSI's latest simulation. But before we begin the bloodbath, let's take a quick look at a utility for a program we looked at a couple of months ago.

Video Title Shop Graphics Companion II
Datasoft
19808 Nordhoff Pl.
Chatsworth, CA 91311
48K disk, $19.95

With *Video Title Shop*, a video camera, a VCR and, of course, your Atari, you are able to generate titles and graphics for your home movies and anything else you might record. When I tested it, I found this inexpensive program to be relatively easy to use and complete. And the release of *Graphics Companion II* makes this tool even more complete.

If you were tuned in earlier, you know all about the Title Shop, and if you weren't, then check out the October '88 issue of ANALOG Computing and become enlightened. Graphics Companion II is basically a data disk for use with the Shop. Thus it is not a stand-alone product. What it gives you is a number of additional fonts, borders and images which can be pasted onto screens. These latter two features are improvements not found in the original program.

That's really about the extent of this one. So, if you have Video Title Shop and want to go a little farther, pick up this companion disk. You might find yourself on your way to Hollywood before you know it.

Sons of Liberty
by David Landrey and Chuck Kroegel
SSI
1046 N. Rengstorff Ave.
Mountain View, CA 94043
48K disk, $39.95

Napoleon in Russia-Borodino 1812
by R. Steve Krenek
Datasoft
19808 Nordhoff Pl.
Chatsworth, CA 91311
48K disk, $24.95

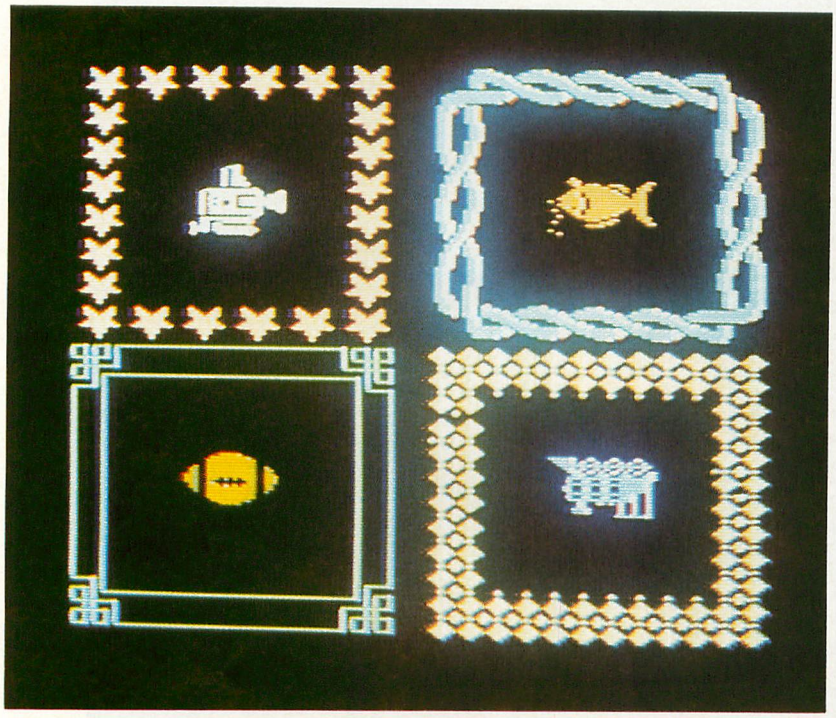
These two war simulations, while coming from two very different companies, are nonetheless very close together in their historical time frame. It is for this reason that I present them to you this month. SSI's latest game, *Sons of Liberty*, is its first to cover our own country's Revolutionary War. This package has scenarios covering the battles of Bunker Hill, Monmouth and Saratoga. Datasoft's contender, *Napoleon in Russia-Borodino 1812*, predictably enough allows you to be Napoleon, one of the greatest military tacticians of all time. Unfortunately, this battle is one which led to the Little Emperor's downfall, due to the fact that he let the retreating red menace escape rather than completely destroying them. This game asks the question: Can you do any better?

Sons of Liberty is an introductory to advanced level game, which means that both novices and experts can enjoy re-creating epic battles in our country's fight for independence. Games can take anywhere from two to ten hours to complete, and play progresses along the usual SSI format. Each player (either or both sides can be human or computer controlled) takes turns moving and attacking, with the computer resolving the various conflicts instantaneously. In the basic game the joystick is used to issue orders (the preferred method) while the more advanced options require keyboard input. While this complexity yields more realism, it is at a cost: regular and lengthy disk swaps, which frustrate and slow play. It seems that 48K is just too small to hold it all.

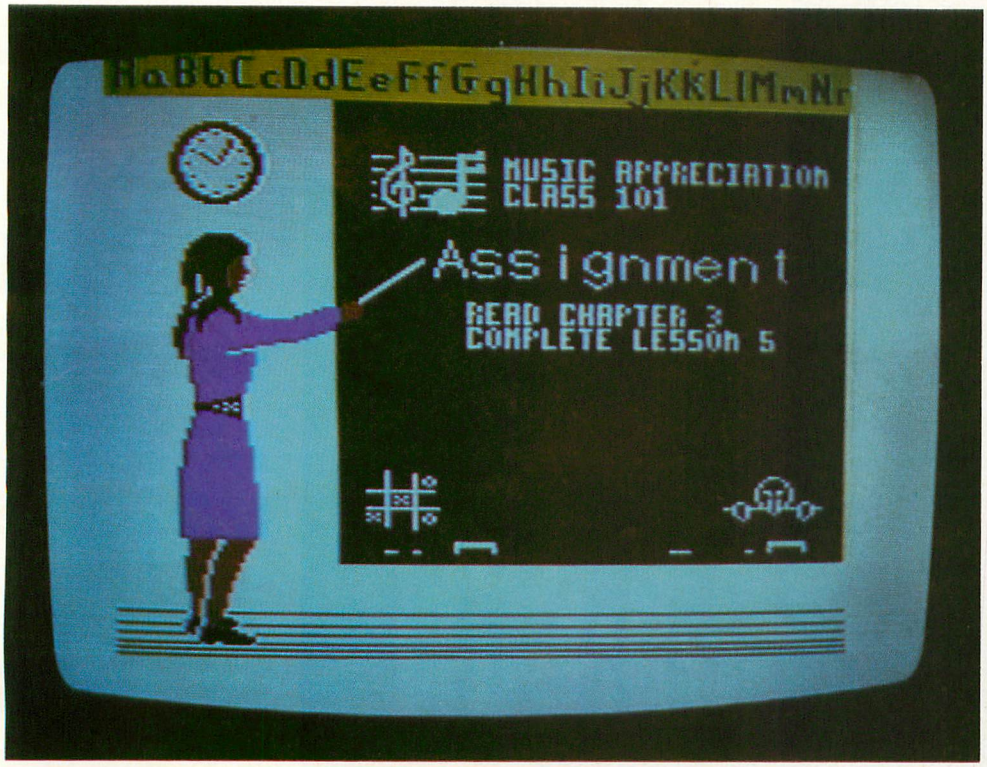
Napoleon, on the other hand, is much more accessible to the novice player. A joystick is used for most input, with the Option, Start and Reset keys used to a lesser degree. Especially different is the way time is handled in this game. A game minute can equal 60 seconds or one second, and time flows continuously, in contrast to SSI's game,

PANAK STRIKES

by Steve Panak



Video Title Shop Graphics Companion II



in which time (measured in turns) advances as moves are made. The practical difference is that Napoleon is more demanding; during your delays men are dying. Generally, though, the games play pretty much alike. In each you move units and attack the enemy. Unlike Liberty, in Napoleon the computer can control only Russia if you should fail to find a human opponent. Additional realism and difficulty are achieved by optionally taking into account troop fatigue and morale.

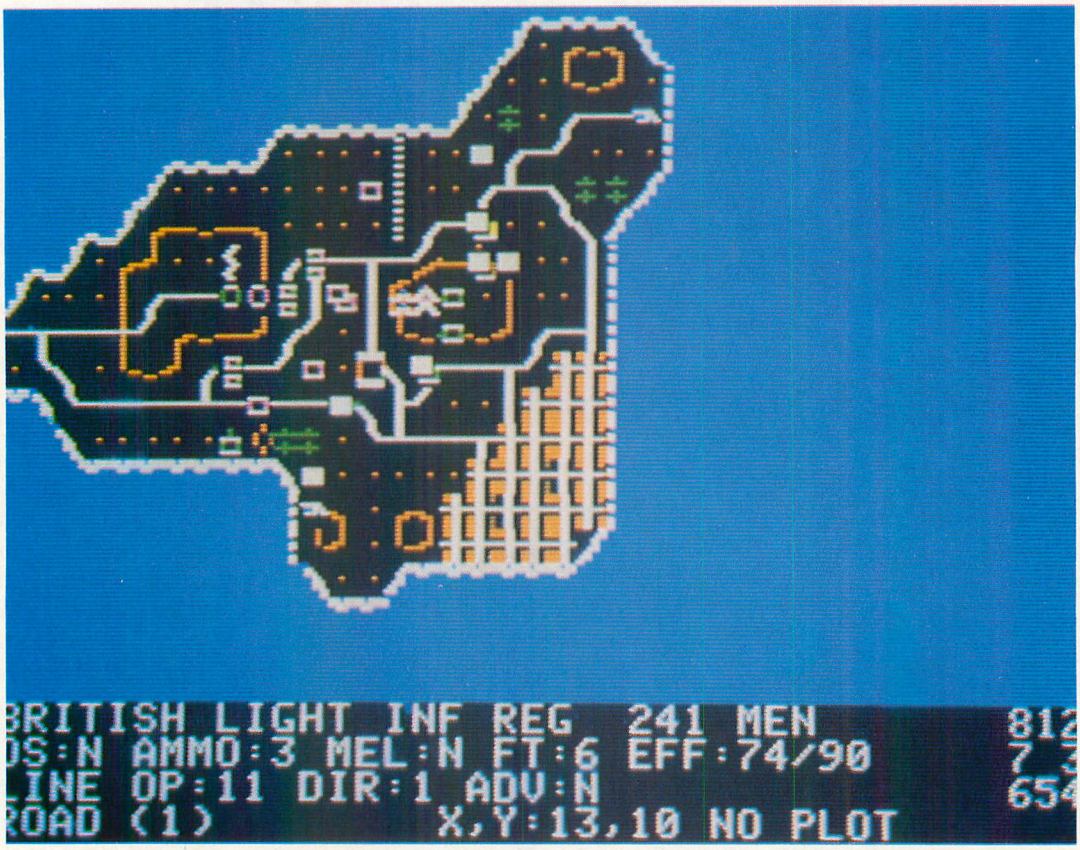
The graphics in Liberty, however, are superior to those of Napoleon. Slightly clearer and more detailed, their superiority is the result of the evolution of SSI's games for a number of years. There's nothing wrong with Napoleon's graphics, though, as this is strictly a judgement call. Noteworthy features of Liberty are the helpful on-screen prompts which continually remind the player of his options. Napoleon's screen is blank by comparison. Sound effects are about equal in both games.

The skirmish over the documentation has to be claimed by SSI. In Liberty it has produced one of its best packages, with two full-color, stiff cardboard maps and reference cards, a 51-page Historical Reference Guide and a 32-page rule book. The best feature of the latter is the seven-page tutorial section that starts even the computer novice on his way to a successful simulation.

Napoleon, on the other hand, is not as easy to learn. This is due to the lack of a tutorial portion of the manual, rather than inherent game design; because once the command structure is learned, the game is quickly mastered. Nonetheless, Napoleon does include substantial documentation. Its 37-page manual, while not as polished as SSI's, does present this complex game in a concise and more or less organized manner.

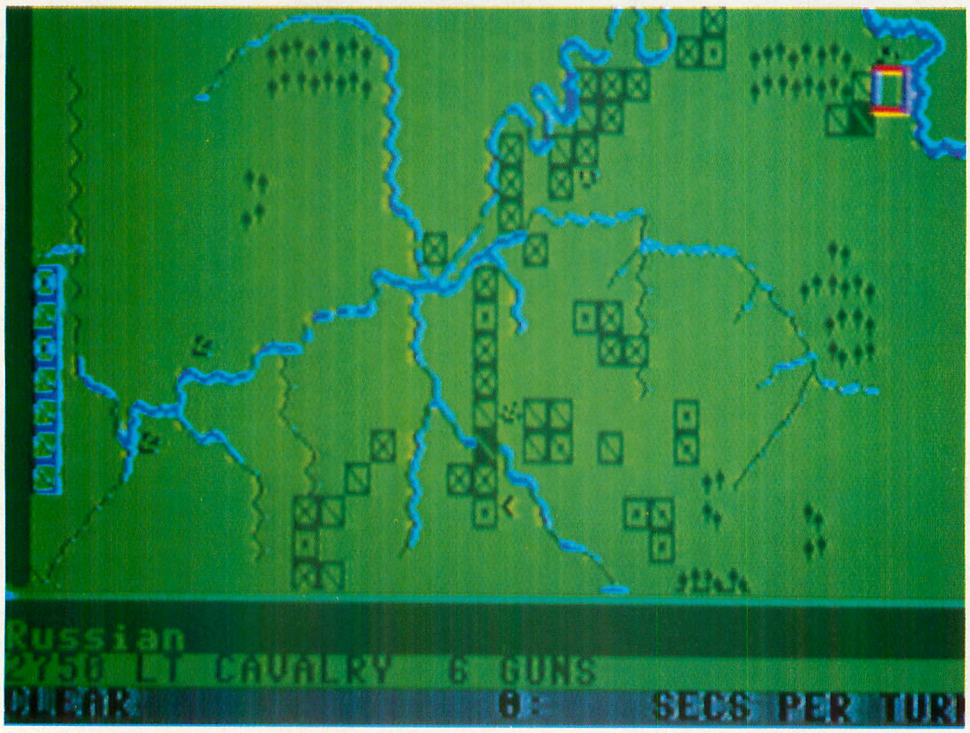
In conclusion, Napoleon in Russia is a fast-paced, simple simulation, ideally suited to the impatient novice. Sons of Liberty, on the other hand, has enough complexity to keep even the expert satiated for weeks, maybe months to come. For these reasons, each is recommended.

PANAK STRIKES



Sons of Liberty

Napoleon in Russia-Borodino 1812



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" in issue 47.

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
NF 60 IF F$(1,2)<>"D:" THEN F1$="D:":F1$(
   3)=F$:GOTO 80
KL 70 F1$=F$
TN 80 IF CHR$(A)="S" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$:TRAP 110
HQ 100 FOR X=1 TO 16:GET #2,A:NEXT X:LINE
   =LINE+10:GOTO 100
WM 110 CLOSE #2:OPEN #2,9,0,F1$:GOTO 170
VT 120 TRAP 160:OPEN #2,4,0,F1$:GOSUB 440
   :POSITION 10,10:? "FILE ALREADY EXISTS
   !!":POKE 752,0
ZU 130 POSITION 10,12:? "ERASE IT? ";:GOS
   UB 500:POKE 752,1:? CHR$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
    CLOSE #2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
   HEN 130
BH 160 CLOSE #2:OPEN #2,8,0,F1$
IE 170 GOSUB 450:POSITION 10,1:? "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
M5 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
DW 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
WX 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHICS 0:END
YT 430 GOSUB 440:POSITION 10,10:? "NO SUC
   H FILE!":FOR X=1 TO 1000:NEXT X:CLOSE
   #2:GOTO 30
FD 440 POKE 710,48:SOUND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:SOUND 0,0,0,0:RETURN
MY 450 GRAPHICS 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 STEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 STEP 2:POKE DL+X,0
   :NEXT X
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
WZ 500 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1
   :RETURN
```

BASIC to Binary

Once you have your BASIC program made into a binary load file, you can start appending data and M/L routines to your main program freely, as long as the data you append has binary load pointers.

by Matthew Arrington

BASIC to Binary is a utility program that will convert a file written and saved under Atari BASIC or BASIC XL into a binary load file. Binary load files are usually machine-language programs that are loaded from DOS or a quick menu, or they are sometimes named AUTORUN.SYS, and then automatically load when the computer is turned on.

BASIC to Binary will also let you determine what action to take if the reset key should be pressed while your program is running. You can have Reset do one of three things: reset and re-run the program, cold-start or simply reset the computer and return to BASIC's READY prompt. The break key can similarly be enabled or disabled. Disabling Break along with trapping Reset will, of course, make the program unlistable.

Another feature of BASIC to Binary is the ability to have a load title. If you choose to have one, a program title will be displayed while the program is loading in. Last, and certainly not least, is the ability to "relocate" your BASIC program to run anywhere (almost) in memory. This is accomplished by giving BASIC to Binary an address which will be stored at MEMLO (743, 744) during program initialization. This is an ideal way to reserve memory for machine-language subroutines, character sets, etc.

One thing BASIC to Binary won't do, unfortunately, is make your BASIC programs run faster. A compiler this ain't!

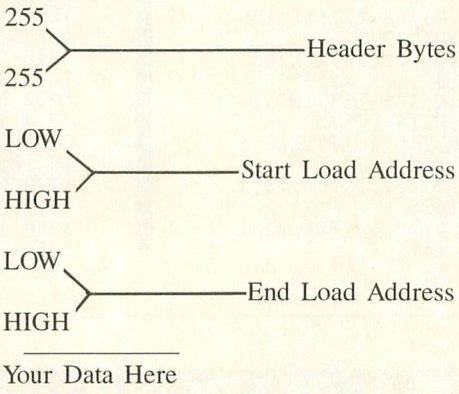
Listing 1 is the BASIC data used to create your copy of BASIC to Binary. You should type this data using the M/L Editor in this issue.

Listing 2 is the assembly source code for BASIC to Binary. You don't have to type it to use the program. It is included for those people who have an interest in assembly-language programming.

Why binary?

Atari's binary load process is great for getting things off the disk and into memory with minimum hassle. For example, let's say you're writing a BASIC program and you have a custom character set or some other large amount of data you need to get into memory. Normally your options would be to manually type the data into DATA statements (no thanks) or bring it in from a disk file. Using a disk file is fine while developing a program, but it's a lot nicer having everything in one file for the finished product. Once you have your BASIC program made into a binary load file, you can start appending data and M/L routines to your main program freely, as long as the data you append has binary load pointers.

A set of binary load pointers looks like this:



That's all there is to it!

Another plus to these hybrid files is that they can be loaded from one of the many three-sector quick menus. If you don't have a quick-menu program in your library, I'd suggest getting a hold of one. There is no faster or easier way to load a file from disk than with a quick menu.

Program operation

BASIC to Binary is written in machine language; so it will need to be loaded from the DOS menu. A quick menu won't work, since this program requires DOS for disk access.

After loading you should see the main menu. Options are: 1) Display/Set options, 2) Convert a file, 3) Disk Directory and 4) Return to DOS. Pressing *1* will take you to the "Control Pad." In the Control Pad, the select key is used to select a setting to change. The settings are: Reset, Break, Title and MEMLO. An inverse block will highlight the setting that will be affected when you press Option. Each time Option is pressed, you'll change the highlighted setting's function.

For example, press Select to choose the title setting, then press Option. You will be prompted for a load title. Enter the title, and hit Return. You will then see "ON" in front of the title setting. Note that when entering a title you don't have to worry about centering the title between the prompts. The program will automatically center the title on the screen for you. Once everything is set to your liking, press Start, and you'll be returned to the main menu. You're now ready to turn your BASIC code into binary.

From the main menu press *2* to convert your BASIC program. You'll see the "INPUT FILE -" prompt. Enter the filename of the program you wish to convert. If the program is on a drive other than Drive 1, you'll need to enter the device number too ("D2:MYPROG.BAS").

After you have entered the filename and pressed Return, the specified file will be opened and read in. This is the first pass, which consists of reading and counting bytes to find the exact size of your program. When completed, you will be asked for an output file. Use the same procedure to enter the output file as you did for the input file. The output file will be opened, the machine code required to run your BASIC program will be written (two or three sectors), and finally your BASIC program will be copied over. When the copy has completed, you'll be returned to the main menu. From here you can return to DOS and, if BASIC is in the machine, you can try out your newly converted program.

Things to watch out for

Disk swapping isn't supported. If you have only one disk drive, be sure you have enough free sectors on the disk for the conversion.

If you need to change the MEMLO address, be sure not to enter an address lower than the address originally shown. Let's say you need to reserve 1K, and from the Control Pad you see "MEMLO - 10000." Use the Select key to select MEMLO, and press Option to get the "New address" prompt. Then add 1024 (1K) to 10000 and enter "11024." Locations 10000 to 11023 will then be reserved for your use, and your program will run starting at location 11024.

BASIC to Binary

LISTING 1

```
1000 DATA 255,255,0,6,10,6,0,0,0,0,0,0
,0,0,0,0,1875
1010 DATA 0,0,37,115,37,169,0,141,198,
2,133,82,169,97,141,200,7027
1020 DATA 2,169,1,141,240,2,76,70,37,1
25,29,29,29,29,29,29,8163
1030 DATA 29,29,29,29,32,66,97,115,105
,99,32,116,111,32,66,105,1715
1040 DATA 110,97,114,121,32,98,121,58,
32,77,97,116,116,104,101,119,4129
1050 DATA 32,65,114,114,105,110,103,11
6,111,110,0,160,0,132,0,185,3671
1060 DATA 20,37,201,96,208,2,169,155,1
32,0,168,162,0,169,11,157,5639
1070 DATA 66,3,169,0,157,72,3,157,73,3
,152,32,86,228,164,0,3656
1080 DATA 164,0,200,185,20,37,208,213,
96,226,2,227,2,0,37,0,2517
1090 DATA 39,5,39,255,255,6,39,233,39,
8,39,3,40,32,32,32,8580
1100 DATA 32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,5452
1110 DATA 32,0,0,0,0,0,45,41,165,6,208
,102,169,1,141,240,5008
1120 DATA 2,169,0,133,82,76,93,39,125,
160,160,160,160,208,210,207,1680
1130 DATA 199,210,193,205,160,210,197,
209,213,201,210,197,211,160,194,193,78
41
1140 DATA 211,201,195,174,160,160,210,
197,194,207,207,212,160,160,160,160,57
57
1150 DATA 160,0,160,0,132,0,185,51,39,
201,96,208,2,169,155,132,6895
1160 DATA 0,168,162,0,169,11,157,66,3,
169,0,157,72,3,157,73,2622
1170 DATA 3,152,32,86,228,164,0,164,0,
200,185,51,39,208,213,76,7830
1180 DATA 138,39,173,31,39,240,87,162,
96,76,152,39,83,169,151,157,7565
1190 DATA 68,3,169,39,157,69,3,169,12,
157,74,3,169,1,157,75,2793
1200 DATA 3,169,3,157,66,3,32,86,228,1
69,10,133,84,169,0,133,4472
1210 DATA 85,160,0,132,0,185,8,39,201,
96,208,2,169,155,132,0,5049
1220 DATA 168,162,96,169,11,157,66,3,1
69,0,157,72,3,157,73,3,1651
1230 DATA 152,32,86,228,164,0,164,0,20
0,185,8,39,208,213,96,226,9532
1240 DATA 2,227,2,35,39,244,39,44,41,1
60,0,173,33,39,141,231,4957
1250 DATA 2,24,105,242,133,203,173,34,
39,4,40,255,40,141,232,2,5846
1260 DATA 105,0,133,204,173,29,39,240,
12,201,2,208,43,169,1,141,5644
1270 DATA 68,2,76,63,40,165,13,201,8,1
76,11,169,136,133,12,169,5206
1280 DATA 40,133,13,76,63,40,165,12,14
1,134,40,165,13,141,135,40,3572
```

```
1290 DATA 169,133,133,12,169,40,133,13
,173,231,2,24,113,203,153,128,7123
1300 DATA 0,200,173,232,2,113,203,153,
128,0,200,192,14,208,233,165,1365
1310 DATA 140,133,142,133,144,165,141,
133,143,133,145,169,0,133,146,133,8855
1320 DATA 202,162,255,154,216,173,34,2
,141,131,40,173,35,2,141,132,5404
1330 DATA 40,76,136,40,169,64,133,16,1
41,14,210,76,0,0,32,0,9489
1340 DATA 0,162,0,173,30,39,208,10,169
,123,141,34,2,169,40,141,4234
1350 DATA 35,2,189,26,3,201,69,240,5,2
32,232,232,208,244,232,142,5257
1360 DATA 25,41,189,26,3,133,203,169,2
7,157,26,3,232,189,26,3,3959
1370 DATA 133,204,169,41,157,26,3,160,
0,162,4,177,203,153,27,41,4454
1380 DATA 200,202,208,247,160,8,162,7,
177,203,153,27,41,200,202,208,955
1390 DATA 247,169,2,141,26,41,169,1,13
3,9,133,8,173,250,191,141,7828
1400 DATA 240,40,173,251,191,141,241,4
0,76,0,0,172,26,41,192,255,7671
1410 DATA 240,9,185,42,41,206,26,41,16
0,0,41,54,41,1,96,138,1076
1420 DATA 72,174,25,41,165,203,157,26,
3,232,165,204,157,26,3,104,6153
1430 DATA 170,169,155,160,1,96,0,0,0,0
,0,0,241,40,0,0,7317
1440 DATA 0,0,0,0,0,0,0,78,85,82,224,2
,225,2,244,39,3374
1450 DATA 0,0,0,0,55,41,50,42,112,112,
112,66,64,156,66,104,2479
1460 DATA 156,66,144,156,66,184,156,66
,224,156,66,8,157,66,48,157,6453
1470 DATA 66,88,157,66,128,157,66,168,
157,66,208,157,66,248,157,66,9821
1480 DATA 32,158,66,72,158,66,112,158,
66,152,158,66,192,158,66,232,9602
1490 DATA 158,66,16,159,66,56,159,66,9
6,159,66,136,159,66,176,159,7758
1500 DATA 66,216,159,65,55,41,125,96,9
6,96,96,96,32,32,32,32,787
1510 DATA 32,32,32,32,32,18,18,18,18,2
3,18,18,18,18,32,32,4652
1520 DATA 32,32,96,32,32,32,32,32,32,3
2,32,32,32,32,32,32,6064
1530 DATA 124,32,32,32,32,32,32,32,32,
96,32,32,32,32,32,17,6374
1540 DATA 18,18,18,18,18,18,18,18,24,18,1
8,18,18,18,18,18,18,5,3828
1550 DATA 96,32,32,32,32,32,124,194,22
5,243,233,227,160,244,239,160,6089
1560 DATA 194,233,238,225,242,249,124,
96,17,18,18,18,18,24,18,18,49
1570 DATA 18,18,18,18,18,18,18,18,18,1
8,18,18,18,24,18,18,4102
1580 DATA 18,18,5,96,124,32,32,32,32,3
2,32,32,32,32,32,32,6525
1590 DATA 32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,124,96,8346
1600 DATA 124,32,49,58,32,68,105,115,1
12,108,97,121,47,83,101,116,4141
1610 DATA 32,79,112,116,51,42,28,43,10
5,111,110,115,32,32,124,96,2552
1620 DATA 124,32,32,32,32,32,32,32,32,
32,32,32,32,32,32,32,32,6064
1630 DATA 32,32,32,32,32,32,32,32,32,3
2,124,96,124,32,50,58,9644
1640 DATA 32,67,111,110,118,101,114,11
6,32,66,97,115,105,99,32,102,3759
1650 DATA 105,108,101,32,32,32,124,96,
0,124,32,32,32,32,32,32,8222
1660 DATA 32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,6012
1670 DATA 32,32,32,124,96,124,32,51,58
,32,70,105,108,101,32,68,1472
1680 DATA 105,114,101,99,116,111,114,1
21,32,32,32,32,32,32,32,124,396
1690 DATA 96,124,32,32,32,32,32,32,32,
32,32,32,32,32,32,32,6290
1700 DATA 32,32,32,32,32,32,32,32,32,3
2,32,124,96,124,32,52,9596
1710 DATA 58,32,69,120,105,116,32,116,
111,32,68,79,83,32,32,32,426
1720 DATA 32,32,32,32,32,32,32,124,96,
124,32,32,32,32,32,32,8304
1730 DATA 32,32,32,32,32,32,32,32,32,3
2,32,32,32,32,32,32,6082
1740 DATA 32,32,32,124,96,26,18,18,18,
18,18,18,18,18,18,18,5134
1750 DATA 18,18,18,18,18,18,18,18,18,1
8,18,18,18,18,18,3,3958
1760 DATA 96,0,29,43,24,44,169,0,141,1
,6,169,10,141,24,2,9551
1770 DATA 173,252,2,201,255,240,54,76,
52,43,95,30,0,160,0,132,3613
1780 DATA 0,185,49,43,201,96,208,2,169
,155,132,0,168,162,0,169,7201
1790 DATA 11,157,66,3,169,0,157,72,3,1
57,73,3,152,32,86,228,4643
1800 DATA 164,0,164,0,200,185,49,43,20
8,213,76,228,43,173,24,2,6200
1810 DATA 208,190,173,2,6,240,59,76,11
6,43,95,30,0,160,0,132,2647
1820 DATA 0,185,113,43,201,96,208,2,16
9,155,132,0,168,162,0,169,7433
1830 DATA 11,157,66,3,169,0,157,72,3,1
57,73,3,152,32,86,228,4683
1840 DATA 164,0,164,0,200,185,113,43,2
08,213,169,0,141,2,6,76,4769
1850 DATA 34,43,76,175,43,32,30,0,160,
0,132,0,185,172,43,201,5081
1860 DATA 96,208,2,169,155,132,0,168,1
62,0,169,11,157,66,3,169,5128
1870 DATA 0,157,72,3,157,73,3,152,32,8
6,228,164,0,164,0,200,5992
1880 DATA 185,172,43,208,213,169,1,141
,2,6,76,34,43,162,48,169,4157
1890 DATA 7,157,66,3,169,0,157,72,3,15
7,73,3,32,86,228,201,5633
1900 DATA 126,208,11,174,1,6,208,3,76,
34,43,76,86,44,201,156,4346
1910 DATA 208,11,174,1,6,208,3,76,34,4
3,76,121,44,201,155,208,6636
1920 DATA 3,76,25,44,20,45,158,44,174,
116,65,208,9,41,127,201,5903
1930 DATA 97,144,3,56,233,32,174,1,6,2
36,0,6,208,3,76,34,2047
1940 DATA 43,157,126,5,232,142,1,6,132
,0,168,162,0,169,11,157,4785
1950 DATA 66,3,169,0,157,72,3,157,73,3
,152,32,86,228,164,0,4536
1960 DATA 76,34,43,32,193,44,169,126,1
32,0,168,162,0,169,11,157,5804
1970 DATA 66,3,169,0,157,72,3,157,73,3
,152,32,86,228,164,0,4556
1980 DATA 206,1,6,76,34,43,32,193,44,1
69,126,132,0,168,162,0,4544
1990 DATA 169,11,157,66,3,169,0,157,72
,3,157,73,3,152,32,86,2505
2000 DATA 228,164,0,206,1,6,208,225,76
,34,43,169,156,132,0,168,6766
2010 DATA 162,0,169,11,157,66,3,169,0,
157,72,3,157,73,3,152,3215
2020 DATA 32,86,228,164,0,174,1,6,169,
0,157,126,5,96,76,199,5156
2030 DATA 44,31,126,0,160,0,132,0,185,
196,44,201,96,208,2,169,7653
2040 DATA 155,132,0,168,162,0,169,11,1
57,66,3,169,0,157,72,3,2672
2050 DATA 157,73,3,152,32,86,228,164,0
,164,0,200,185,196,44,208,9731
2060 DATA 213,96,162,0,132,0,168,169,1
1,157,66,3,169,0,157,72,4274
2070 DATA 3,157,73,3,152,32,86,228,164
,0,96,160,0,185,117,65,5833
2080 DATA 153,100,21,45,94,45,65,200,1
92,3,208,245,173,127,5,201,9775
2090 DATA 58,208,5,160,2,76,56,45,160,
0,173,128,5,201,58,208,6393
2100 DATA 8,173,127,5,141,101,65,160,3
,162,3,185,126,5,157,100,5464
2110 DATA 65,240,4,200,232,208,244,96,
165,226,56,229,224,141,8,6,562
2120 DATA 165,227,229,225,141,9,6,238,
8,6,208,3,238,9,6,96,4333
2130 DATA 116,65,111,66,0,68,49,58,68,
48,58,42,46,42,162,48,806
2140 DATA 76,133,65,75,58,169,131,157,
68,3,169,65,157,69,3,169,5491
2150 DATA 4,157,74,3,169,0,157,75,3,16
9,3,157,66,3,32,86,1636
2160 DATA 228,169,55,141,48,2,169,41,1
41,49,2,169,64,133,88,169,5745
2170 DATA 156,133,89,169,1,141,240,2,1
69,31,133,83,169,0,141,47,5436
2180 DATA 2,141,198,2,169,97,141,200,2
,169,2,133,82,160,0,132,5824
```

```
2190 DATA 0,185,133,41,201,96,208,2,16
9,155,132,0,168,162,0,169,7855
2200 DATA 11,157,66,3,169,0,157,72,3,1
57,73,3,152,32,86,228,5053
2210 DATA 164,0,164,0,200,185,133,41,2
08,213,160,0,132,0,185,116,8344
2220 DATA 42,201,96,208,2,169,155,132,
0,168,162,0,169,11,157,66,6173
2230 DATA 3,169,0,157,72,3,157,73,3,15
2,32,86,228,164,0,164,6075
2240 DATA 0,200,185,116,42,208,213,169
,33,141,47,2,169,69,141,4,5550
2250 DATA 6,169,90,141,5,6,160,11,169,
0,162,3,157,16,157,153,5104
2260 DATA 16,157,172,4,6,173,5,6,140,5
,6,141,4,6,162,4,9971
2270 DATA 160,10,189,16,157,141,3,6,17
3,4,6,157,16,157,185,16,3765
2280 DATA 157,133,203,173,5,6,153,16,1
57,169,0,133,20,165,20,201,6049
2290 DATA 112,66,107,67,3,208,250,169,
0,32,147,66,157,16,157,169,7541
2300 DATA 1,32,147,66,153,16,157,173,2
52,2,201,255,208,92,232,136,3621
2310 DATA 192,2,208,194,76,54,66,201,1
,240,6,173,3,6,76,159,5038
2320 DATA 66,165,203,201,90,208,3,76,1
97,66,201,69,208,3,76,197,8966
2330 DATA 66,201,67,208,3,76,197,66,20
1,81,208,3,76,197,66,201,9104
2340 DATA 82,240,3,169,82,96,169,0,96,
173,4,6,201,90,208,13,5667
2350 DATA 169,67,141,4,6,169,81,141,5,
6,76,227,66,169,69,141,6011
2360 DATA 4,6,169,90,141,5,6,169,87,96
,169,0,133,82,162,48,5049
2370 DATA 169,7,157,66,3,169,0,157,72,
3,157,73,3,32,86,228,4279
2380 DATA 41,127,201,97,144,3,56,233,3
2,201,50,208,6,32,49,67,4337
2390 DATA 76,188,65,201,49,208,6,32,79
,69,76,188,65,201,51,208,7877
2400 DATA 6,32,118,76,76,126,65,201,52
,240,3,76,188,65,169,39,6653
2410 DATA 133,83,108,10,0,169,125,32,2
45,44,169,16,141,0,6,169,4541
2420 DATA 10,133,84,169,0,133,85,76,84
,67,73,78,80,85,84,32,2792
2430 DATA 70,73,76,69,32,45,32,0,160,0
,132,0,185,70,67,201,4302
2440 DATA 96,208,2,169,155,132,0,168,1
62,0,169,11,157,66,3,169,5708
2450 DATA 108,67,103,68,0,157,72,3,157
,73,3,152,32,86,228,164,6407
2460 DATA 0,164,0,200,185,70,67,208,21
3,32,29,43,173,1,6,208,5819
2470 DATA 1,96,32,14,45,162,16,169,12,
157,66,3,32,86,228,162,5548
2480 DATA 16,169,100,157,68,3,169,65,1
57,69,3,169,4,157,74,3,3395
2490 DATA 169,0,157,75,3,169,3,157,66,
3,32,86,228,48,140,169,6184
2500 DATA 0,141,6,39,141,7,39,162,16,1
69,7,157,66,3,169,0,2502
2510 DATA 157,72,3,157,73,3,32,86,228,
192,136,240,10,238,6,39,7267
2520 DATA 208,229,238,7,39,208,224,162
,16,169,12,157,66,3,32,86,4841
2530 DATA 228,162,16,169,100,157,68,3,
169,65,157,69,3,169,4,157,5451
2540 DATA 74,3,169,0,157,75,3,169,3,15
7,66,3,32,86,228,76,4350
2550 DATA 25,68,79,85,84,80,85,84,32,7
0,73,76,69,32,45,32,690
2560 DATA 0,160,0,132,0,185,10,68,201,
96,208,2,169,155,132,0,6560
2570 DATA 168,162,0,169,11,157,66,3,16
9,0,157,72,3,157,73,3,2713
2580 DATA 152,32,86,228,164,0,164,0,20
0,185,10,68,208,213,32,29,7140
2590 DATA 43,173,1,6,208,1,96,32,14,45
,162,32,169,12,157,66,3498
2600 DATA 3,32,86,228,162,32,169,100,1
57,68,3,169,65,157,69,3,5102
2610 DATA 104,68,99,69,169,8,157,74,3,
169,0,157,75,3,169,3,3208
2620 DATA 157,66,3,32,86,228,48,139,16
9,0,133,224,169,39,133,225,302
2630 DATA 169,54,133,226,169,41,133,22
7,32,71,45,173,33,39,24,105,4632
2640 DATA 242,133,203,173,34,39,105,0,
133,204,165,203,141,51,41,24,6622
2650 DATA 109,6,39,141,53,41,165,204,1
41,52,41,109,7,39,141,54,3914
2660 DATA 41,173,53,41,56,233,1,141,53
,41,173,54,41,233,0,141,5672
2670 DATA 54,41,162,32,169,11,157,66,3
,169,0,157,68,3,169,39,3644
2680 DATA 157,69,3,173,8,6,157,72,3,17
3,9,6,157,73,3,32,975
2690 DATA 86,228,162,16,169,7,157,66,3
,169,95,157,68,3,169,45,5123
2700 DATA 157,69,3,169,5,157,72,3,169,
20,157,73,3,32,86,228,4924
2710 DATA 189,72,3,141,8,6,189,73,3,14
1,9,6,140,10,6,162,1849
2720 DATA 32,169,11,157,66,3,169,95,15
7,68,3,169,45,157,69,3,4062
2730 DATA 173,8,6,157,72,3,173,9,6,157
,73,3,32,86,228,172,5481
2740 DATA 10,6,192,136,208,172,162,16,
169,12,157,66,3,32,86,228,6801
2750 DATA 162,32,169,12,157,66,3,32,86
,228,96,169,125,32,245,44,7579
2760 DATA 169,0,133,82,76,122,69,32,32
,66,97,115,105,99,32,116,3989
2770 DATA 100,69,95,70,111,32,66,105,1
10,97,114,121,32,67,111,110,5067
2780 DATA 116,114,111,108,32,80,97,100
,96,0,160,0,132,0,185,91,4579
2790 DATA 69,201,96,208,2,169,155,132,
0,168,162,0,169,11,157,66,6770
2800 DATA 3,169,0,157,72,3,157,73,3,15
2,32,86,228,164,0,164,6645
2810 DATA 0,200,185,91,69,208,213,76,2
01,69,32,32,13,13,13,13,1810
2820 DATA 13,13,13,13,13,13,13,13,13,1
3,13,13,13,13,13,13,4588
2830 DATA 13,13,13,13,13,13,13,96,0,16
0,0,132,0,185,170,69,3390
2840 DATA 201,96,208,2,169,155,132,0,1
68,162,0,169,11,157,66,3,5103
2850 DATA 169,0,157,72,3,157,73,3,152,
32,86,228,164,0,164,0,5232
2860 DATA 200,185,170,69,208,213,76,20
,70,32,32,211,197,204,197,195,2552
2870 DATA 212,32,58,32,73,116,101,109,
32,116,111,32,99,104,97,110,5099
2880 DATA 103,101,96,0,160,0,132,0,185
,249,69,201,96,208,2,169,9417
2890 DATA 155,132,0,168,162,0,169,11,1
57,66,3,169,0,157,72,3,3522
2900 DATA 157,73,3,152,32,86,228,164,0
,164,0,200,185,249,69,208,1698
2910 DATA 213,76,93,70,32,32,207,208,2
12,201,207,206,32,58,32,67,8746
2920 DATA 104,97,110,103,101,115,32,73
,116,101,109,96,0,160,0,132,4720
2930 DATA 96,70,91,71,0,185,68,70,201,
96,208,2,169,155,132,0,7297
2940 DATA 168,162,0,169,11,157,66,3,16
9,0,157,72,3,157,73,3,3083
2950 DATA 152,32,86,228,164,0,164,0,20
0,185,68,70,208,213,76,162,960
2960 DATA 70,32,32,211,212,193,210,212
,160,32,58,32,69,120,105,116,8208
2970 DATA 115,96,96,96,96,0,160,0,132,
0,185,141,70,201,96,208,8956
2980 DATA 2,169,155,132,0,168,162,0,16
9,11,157,66,3,169,0,157,5522
2990 DATA 72,3,157,73,3,152,32,86,228,
164,0,164,0,200,185,141,9161
3000 DATA 70,208,213,76,245,70,210,197
,211,197,212,32,45,96,96,66,130
3010 DATA 82,69,65,75,32,45,96,96,84,7
3,84,76,69,32,45,96,2473
3020 DATA 96,77,69,77,76,79,32,45,0,16
0,0,132,0,185,210,70,5267
3030 DATA 201,96,208,2,169,155,132,0,1
68,162,0,169,11,157,66,3,5293
3040 DATA 169,0,157,72,3,157,73,3,152,
32,86,228,164,0,164,0,5422
3050 DATA 200,185,210,70,208,213,169,0
,141,3,6,141,6,6,170,169,6504
3060 DATA 8,141,7,6,32,220,71,174,3,6,
232,142,3,6,224,4,4654
3070 DATA 208,242,169,8,141,31,208,173
,31,208,201,7,176,244,168,169,3614
3080 DATA 0,141,31,208,141,10,212,169,
8,141,31,208,173,31,208,201,1226
```

```
3090 DATA 92,71,87,72,7,208,244,169,10
,141,30,2,173,30,2,208,6097
3100 DATA 251,152,201,5,240,8,201,3,20
8,3,76,94,76,96,165,88,7038
3110 DATA 133,203,165,89,133,204,173,7
,6,170,165,203,24,105,40,133,8171
3120 DATA 203,165,204,105,0,133,204,20
2,208,240,160,0,177,203,240,7,3414
3130 DATA 41,127,145,203,200,208,245,1
73,6,6,201,3,208,28,169,0,8011
3140 DATA 141,6,6,169,8,141,7,6,165,20
3,56,233,240,133,203,165,2564
3150 DATA 204,233,0,133,204,160,0,76,2
05,71,238,6,6,238,7,6,5796
3160 DATA 238,7,6,160,80,177,203,240,8
,24,105,128,145,203,200,208,2931
3170 DATA 244,76,62,71,169,0,133,203,1
33,204,189,29,39,168,202,224,2573
3180 DATA 255,240,10,165,203,24,105,6,
133,203,76,230,71,136,192,255,3157
3190 DATA 240,10,165,204,24,105,2,133,
204,76,245,71,165,204,24,101,9709
3200 DATA 203,170,189,70,76,141,23,72,
189,71,76,141,24,72,76,0,3952
3210 DATA 0,169,8,133,84,169,8,133,85,
76,58,72,84,114,97,112,5620
3220 DATA 32,38,32,82,101,45,82,117,11
0,32,80,114,111,103,114,97,5742
3230 DATA 109,0,160,0,132,0,185,36,72,
201,96,208,2,169,155,132,9101
3240 DATA 0,168,162,0,169,11,157,66,3,
169,0,157,72,3,157,73,4702
3250 DATA 88,72,83,73,3,152,32,86,228,
164,0,164,0,200,185,36,7673
3260 DATA 72,208,213,96,169,8,133,84,1
69,8,133,85,76,137,72,82,6649
3270 DATA 101,115,101,116,32,78,111,11
4,109,97,108,108,121,32,32,32,4133
3280 DATA 32,32,32,32,0,160,0,132,0,18
5,115,72,201,96,208,2,6704
3290 DATA 169,155,132,0,168,162,0,169,
11,157,66,3,169,0,157,72,5464
3300 DATA 3,157,73,3,152,32,86,228,164
,0,164,0,200,185,115,72,8573
3310 DATA 208,213,96,169,8,133,84,169,
8,133,85,76,216,72,67,111,7532
3320 DATA 108,100,32,83,116,97,114,116
,32,32,32,32,32,32,32,144
3330 DATA 32,32,32,0,160,0,132,0,185,1
94,72,201,96,208,2,169,8949
3340 DATA 155,132,0,168,162,0,169,11,1
57,66,3,169,0,157,72,3,3972
3350 DATA 157,73,3,152,32,86,228,164,0
,164,0,200,185,194,72,208,1423
3360 DATA 213,96,169,10,133,84,169,8,1
33,85,76,26,73,68,105,115,5239
3370 DATA 97,98,108,101,100,0,160,0,13
2,0,185,17,73,201,96,208,7969
3380 DATA 2,169,155,132,0,168,162,0,16
9,11,157,66,3,169,0,157,5922
3390 DATA 72,3,157,73,3,152,32,86,228,
164,0,164,0,200,185,17,7577
3400 DATA 73,208,213,96,169,10,133,84,
169,8,133,85,76,92,73,69,5979
3410 DATA 84,73,79,74,110,97,98,108,10
1,100,32,0,160,0,132,0,3176
3420 DATA 185,83,73,201,96,208,2,169,1
55,132,0,168,162,0,169,11,7436
3430 DATA 157,66,3,169,0,157,72,3,157,
73,3,152,32,86,228,164,7538
3440 DATA 0,164,0,200,185,83,73,208,21
3,96,169,12,133,84,169,8,8614
3450 DATA 133,85,76,153,73,79,102,102,
0,160,0,132,0,185,149,73,6139
3460 DATA 201,96,208,2,169,155,132,0,1
68,162,0,169,11,157,66,3,5723
3470 DATA 169,0,157,72,3,157,73,3,152,
32,86,228,164,0,164,0,5852
3480 DATA 200,185,149,73,208,213,96,16
9,12,133,84,169,8,133,85,76,7978
3490 DATA 214,73,79,110,32,0,160,0,132
,0,185,210,73,201,96,208,81
3500 DATA 2,169,155,132,0,168,162,0,16
9,11,157,66,3,169,0,157,6042
3510 DATA 72,3,157,73,3,152,32,86,228,
164,0,164,0,200,185,210,785
3520 DATA 73,208,213,238,116,65,169,16
,133,84,169,0,133,85,76,32,6348
3530 DATA 74,69,78,84,69,82,32,84,73,8
4,76,69,58,96,96,0,2744
3540 DATA 160,0,132,0,185,17,74,201,96
,208,2,169,155,132,0,168,8794
3550 DATA 162,0,169,11,157,66,3,169,0,
157,72,3,157,73,3,152,4755
3560 DATA 32,86,228,164,0,164,0,200,18
5,17,74,208,213,76,124,74,9710
3570 DATA 80,74,75,75,91,32,32,32,32,3
2,32,32,32,32,32,32,8650
3580 DATA 32,32,32,32,32,32,32,32,32,9
3,30,30,30,30,30,30,8380
3590 DATA 30,30,30,30,30,30,30,30,30,3
0,30,30,30,30,30,0,7190
3600 DATA 160,0,132,0,185,80,74,201,96
,208,2,169,155,132,0,168,9232
3610 DATA 162,0,169,11,157,66,3,169,0,
157,72,3,157,73,3,152,4815
3620 DATA 32,86,228,164,0,164,0,200,18
5,80,74,208,213,169,19,141,1199
3630 DATA 0,6,32,29,43,169,16,133,84,1
69,0,133,85,76,190,74,6504
3640 DATA 156,0,160,0,132,0,185,188,74
,201,96,208,2,169,155,132,792
3650 DATA 0,168,162,0,169,11,157,66,3,
169,0,157,72,3,157,73,5112
3660 DATA 3,152,32,86,228,164,0,164,0,
200,185,188,74,208,213,173,3971
3670 DATA 1,6,208,6,206,116,65,76,59,7
6,160,19,169,32,153,8,5467
3680 DATA 39,136,192,255,208,248,162,1
0,173,1,6,202,56,233,2,176,222
3690 DATA 250,224,11,144,2,162,0,160,0
,185,126,5,240,8,157,8,6270
3700 DATA 39,200,232,76,21,75,206,116,
65,96,169,14,133,84,169,8,7204
3710 DATA 133,85,76,55,75,32,32,32,32,
32,0,160,0,132,0,185,2844
3720 DATA 49,75,201,96,208,2,169,155,1
32,0,168,162,0,169,11,157,8404
3730 DATA 76,75,71,76,66,3,169,0,157,7
2,3,157,73,3,152,32,3837
3740 DATA 86,228,164,0,164,0,200,185,4
9,75,208,213,169,14,133,84,241
3750 DATA 169,8,133,85,173,33,39,133,2
12,173,34,39,133,213,32,170,9465
3760 DATA 217,169,212,133,243,169,0,13
3,244,133,242,32,230,216,160,255,7842
3770 DATA 200,185,128,5,41,127,153,128
,5,201,46,208,243,169,0,153,836
3780 DATA 128,5,160,0,132,0,177,243,20
1,96,208,2,169,155,132,0,9669
3790 DATA 168,162,0,169,11,157,66,3,16
9,0,157,72,3,157,73,3,3933
3800 DATA 152,32,86,228,164,0,164,0,20
0,177,243,208,215,96,169,16,2823
3810 DATA 133,84,169,0,133,85,76,229,7
5,78,69,87,32,77,69,77,5176
3820 DATA 76,79,32,86,65,76,85,69,32,7
3,83,58,0,160,0,132,3401
3830 DATA 0,185,209,75,201,96,208,2,16
9,155,132,0,168,162,0,169,9859
3840 DATA 11,157,66,3,169,0,157,72,3,1
57,73,3,152,32,86,228,6693
3850 DATA 164,0,164,0,200,185,209,75,2
08,213,169,5,141,0,6,32,7035
3860 DATA 29,43,173,1,6,240,28,169,126
,133,243,169,5,133,244,169,2972
3870 DATA 0,133,242,32,0,216,32,210,21
7,165,212,141,33,39,165,213,2675
3880 DATA 141,34,39,174,6,6,169,0,157,
29,39,76,220,71,25,72,4576
3890 DATA 72,76,67,77,104,72,183,72,6,
73,72,73,59,76,138,73,4953
3900 DATA 199,73,59,76,38,75,198,75,59
,76,174,6,6,254,29,39,5322
3910 DATA 189,29,39,201,3,208,5,169,0,
157,29,39,32,220,71,76,5862
3920 DATA 62,71,169,125,32,245,44,169,
0,141,121,65,76,151,76,96,7720
3930 DATA 96,96,87,104,105,99,104,32,1
00,114,105,118,101,63,32,177,7116
3940 DATA 45,184,0,160,0,132,0,185,131
,76,201,96,208,2,169,155,314
3950 DATA 132,0,168,162,0,169,11,157,6
6,3,169,0,157,72,3,157,5670
3960 DATA 73,3,152,32,86,228,164,0,164
,0,200,185,131,76,208,213,2760
3970 DATA 162,48,169,7,157,66,3,169,0,
157,72,3,157,73,3,32,3335
3980 DATA 86,228,141,121,65,162,16,169
,12,157,66,3,32,86,228,162,8262
```

(to page 58)

Database DELPHI

by Michael A. Banks

One of the pleasant ironies of working with a magazine is that you're always thinking four to six months in the future. Last month's column, written in early July, found me fighting water shortages and record high temperatures. Now it's mid-August, we've had some rain, and I've had to mow the lawn twice (well, my son Mike has). But, despite the rain and relatively cooler temperatures, we'll probably be mowing the lawn in November. I really believe we're in for a continuation of warm weather; so those of you who live north of the Mason-Dixon Line shouldn't be surprised if you find yourself making holiday visits in shirtsleeves and jackets rather than heavy coats and boots. (New England excepted, of course—without cold weather, it wouldn't be New England.)

Speaking of holiday visits, I'm sure most of you have picked up the holiday spirit by now, and with that in mind I'd like to draw your attention to some appropriate software in the Atari Users' Group databases. Search the Games & Entertainment, Sight & Sound and Recent Arrivals database topics using the keywords HOLIDAY and CHRISTMAS. You'll find some nice surprises. And, in case I've piqued your interest in weather with all this talk of heat waves, you'll find at least one program dealing with the subject in the Education database.

Looking around

In addition to browsing the databases for seasonal software, you might be interested in looking around the Atari Users' Group and trying out some features that are new to you. And there are probably several of those: main menu selections that you've not tried, as well as some features that are not on any menu. Beginning this month, I'm going to show you some of these lesser-known features.

This time out, we'll take a look at some commands that let you examine your online environment, as it were. I'll cover commands that tell you the following:

- who's currently in the group,
- when a specific member was last in the group,
- who's been in recently,
- one-line greeting exchanges,
- the time you've spent online.

Who's on first?

You may have noticed a selection on the Main Menu labeled "Who's Here." This selection (actually a command) displays a list of the usernames of the members currently in the area. Type WHO, like this:

```
ANALOG> WHO

ANALOG's ATARI SIG Members online:

(KZIN)      BACHAND      (ANALOG4)      RAH
LAZARUS
```

WHO works at the Main Menu or Forum prompt. At database or workspace prompts, or in Conference, the command must be preceded by a slash, like this: /WHO.

In the example above, you'll notice that two membernames are enclosed by parentheses. This indicates that they are in Conference. (You won't be able to see their conference group until you enter the Conference area, however.)

The entry log

If you want to know when a member was last in, use the "Entry Log." Type EN at the Main Menu. You'll be prompted to enter a username, after which DELPHI will tell you when that member was last on. Or, you can simply type EN followed by the membername, like this:

```
ANALOG> EN MANUAL

Manny O'Kelly (MANUAL) was last on at
20-DEC-1988 17:10
```

Note that the member's real name (as entered by the member) is displayed, along with the date and time he was last in the group. This command works at the Main Menu, and in Conference.(When used in Conference, it must be preceded by a slash, like this: /EN MANUAL.)

At the Main Menu, you can also see a list of the last ten members to enter the SIG by substituting for the membername, like this:

```
ANALOG> EN *

20-DEC 20:18  MIKE BANKS (KZIN)
20-DEC 20:17  Charles Bachand (BACHAND)
20-DEC 20:03  Clayton Walnum (ANALOG4)
20-DEC 19:59  ROBERT ANSON (RAH)
20-DEC 19:51  L. Long (LAZARUS)
20-DEC 18:14  Andy Eddy (ANALOG2)
20-DEC 18:14  Debbie Jones (HOBBY)
20-DEC 18:09  MAT*RAT (MATRAT)
20-DEC 17:42  Bob Retelle (BOBRETELLE)
20-DEC 17:10  Manny O'Kelly (MANUAL)
```

All Entry Log times are Eastern Standard Time or Eastern Daylight Time, depending on the time of year.

One-liners

You can send one-liners to anyone currently in the group with you by typing /SEND followed by the member's username and a brief message (up to 80 characters). For instance, if you typed WHO and saw KZIN in the group and wanted to say "Hi," you would type:

```
/SEND KZIN Hi!
```

This works at all prompts except Mail and More? prompts.

If someone sends you a one-liner, you'll see the message, preceded by the member's username, like this:

```
KZIN>> Hello.  What's new?
```

If you don't wish to receive one-liners (say, you're in Mail and don't want to be disturbed), type /GAG or /BUSY. This will disable the messages. DELPHI will tell the member sending to you that you are busy. To reverse the "gag" and re-enable one-liners, type /NOGAG or /NOBUSY.

Incidentally, if the member to whom you send a one-liner is in the middle of a binary file transfer or certain other procedures, you will be told that he is busy but your message will be delivered later, if possible. If someone sends a one-liner to you while you're thus occupied, you'll be told that you have message(s) waiting when the procedure is completed and prompted to type *Y* to see them.

Who's who?

In addition to seeing who's currently in the group, and checking who's been in and when, there are several ways to find out more about members, which I'll discuss next month.

Checking the time

Need to check the time but can't find your watch? You can find out the time by typing /TIME at most any prompt in the Atari Users' Group. This command displays the current time and date, and—of more immediate interest to most users—the amount of time (hours, minutes and seconds) you've been online, like this:

```
ANALOG> /TIME
20-DEC-1988 20:18:15
44 minutes (0h 44m 14s)
```

As with the Entry Log, the time displayed is Eastern time.

What's new on DELPHI

DELPHI recently brought two new games online: *Scramble* and *TQ*. Scramble is an interactive game, played in conference areas. Players have 90 seconds to come up with as many words as possible from a Scramble list of 16 letters. The longer the word, the more points a player gets. Players compete with one another and can chat between rounds, or even during a game, to give hints or to distract opponents. Members who just want to practice can ignore everyone else and work on their personal scores.

TQ is a real-time, multiplayer trivia tournament. Each game lasts approximately 30 minutes. Games are held at 9 p.m. and 12 midnight EDT every Sunday and Wednesday evening, in DELPHI's main Conference area (type CONFERENCE at the DELPHI Main Menu).

Atari Users' Group updates

Scramble and another fascinating word game, *FlipIt!*, are available in the Atari Users' Group Conference area, as well. To try out the games, enter the Atari Conference area, join or create a conference group, then type /PLAY to see the Games menu.

A complete tutorial on the Atari Users' Group is now available in the General Interests database topic. At the Atari Group Main Menu, type DA GEN, then type READ ATARI SIG TUTORIAL. You'll find information on all aspects of using the wealth of features the group has to offer and a few pleasant surprises.

Reminders

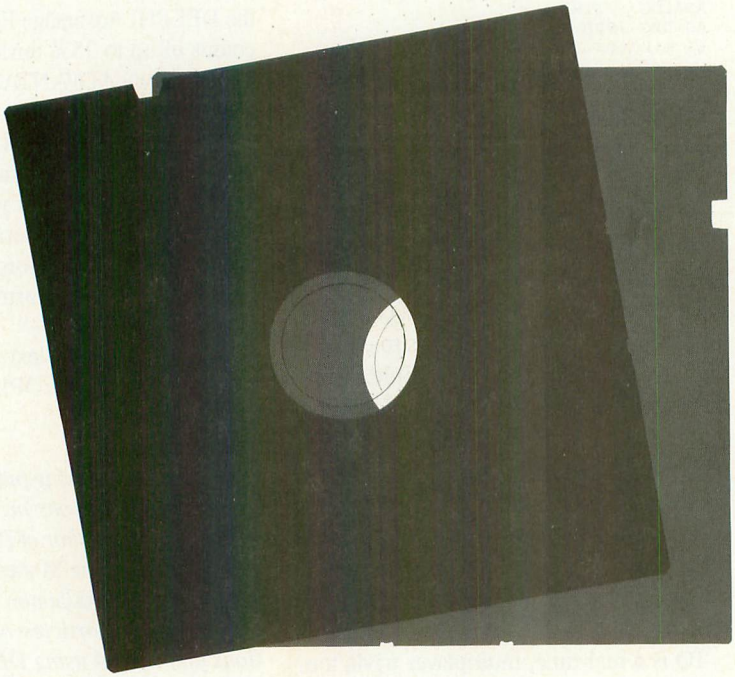
The popular Atari real-time conference is held in the Atari ST Users' Group every Tuesday at 10 p.m., EST. To join, type CO at the SIG menu, then type WHO at the conference menu. You'll see a conference group name, with a list of the members participating. The name will be preceded by a number. To join, simply type JOIN followed by the number, and you're in! Type to talk. If you get stuck, ask those in the conference group for help, or type /HELP.

If you spend more than three or four hours per month online, you'll want to investigate the DELPHI Advantage Plan. It provides discounts of up to 25% during nonprime time. Type USING ADVANTAGE at the DELPHI Main Menu for details.

Have a question about the SIG? Leave a message in the Atari Forum, or send E-mail to the group manager. If you have a question about DELPHI in general, send E-mail to me, KZIN. And don't forget: You can always type HELP to get information on any area of DELPHI.

That's it for now. Next month: The Member Directory and PEOPLE. Until then, see you online!

In addition to having published science fiction novels and books on rocketry, Michael A. Banks is the author of DELPHI: The Official Guide *and* The Modem Reference*—both from Brady Books/Simon & Schuster. Look for his general articles on telecommunications and tips on using DELPHI in the Atari Users' Group databases. You can contact Banks to exchange weather reports and other information on DELPHI by sending E-mail to membername KZIN.*



Master Memory Map Part V

by Robin Sherer

How to read the Memory Map

Beginning users: Read the text that is printed in bold type only. These memory locations will be the easiest for you to use and usually don't involve assembly language.

Advanced users: Read everything! Many areas of memory are not of any practical use, but you can learn a lot about how a computer works by reading the boring parts.

SDMCTL
559 022F

This location is amazing. So many things can be done here that you'll just flip! Maybe not. Anyway, SDMCTL controls something called Direct Memory Access (DMA). Simply put, DMA is the process by which ANTIC, the Atari graphics chip, gets the information from memory it needs to fill in the screen (information for the playfield and for player/missile graphics). Now this process obviously takes time and slows down the 6502 because of that. So what happens if we turn DMA off? Things will run faster. Try it; POKE 559,0 to turn DMA off. Uh-oh, what happened? The screen went blank. But, of course, with no DMA, ANTIC isn't getting the information for the screen; so there's no picture. What good does a computer do without a picture? Well, sometimes you don't need one. For example, if you're doing a lot of calculations, it's more important to get them done quickly than to have an "I'm thinking" message on the screen, and turning off DMA will speed things up by as much as 30% percent! Of course, with a blank screen the user may think that the computer just up and croaked on him; so be sure you give a warning before the lights go off. SDMCTL is a shadow register for DMACTL (54272).

By the way, in case you're still sitting there after the POKE 559,0 with a life-

less computer, just press System Reset or type in POKE 559,34. You can't see the POKE written on the screen, but it will still work when you press Return.

Okay, so we can turn off the screen. Big deal, right? Right, but it's what we can do when we turn the screen back on that counts. SDMCTL lets you make the playfield (the blue screen you PLOT and PRINT onto) wider, narrower or nonexistent. It also lets you turn players and missiles on and off, define how tall you want the players to be and, of course, turn ANTIC on and off.

Let's take a look at a breakdown of SDMCTL and see what does what (Figure 1). To get players and missiles going, see GRACTL [53277] as well.

So, to use SDMCTL, pick the options you want, add their values and POKE away. Note that ANTIC must be turned on if you want a display, and you can only have one type of playfield at a time. While we're on the subject of playfields, you'll probably want to know what "narrow" and "wide" playfields are.

The width of the playfield is measured by something called a "color clock." A color clock is twice as wide as a pixel in graphics mode 8. That means that a character in graphics mode 0 is four color clocks wide. We'll use graphics mode as an example since you can see the playfield is 160 color clocks wide (40 characters times four color clocks per character). A narrow playfield is only 128 color clocks (32 characters) wide, while a wide playfield has 192 color clocks (48 characters). The television set draws 228 color clocks total (including the black border), but not all of these can be seen. As a matter of fact, not all of the 192 in a wide playfield can be seen either, which makes it good for horizontal scrolling.

Is having a wide or narrow playfield as easy as it sounds? Well, yes and no: getting it on the screen's easy (try POKE 559,35 right now), using it properly isn't. Unfortunately, telling SDMCTL that you want a different size playfield doesn't tell the OS that anything's different. To see what I mean, try POKE 559,35 from graphics mode zero. Now you have 48 characters per line, but the OS still thinks you have 40. Try typing stuff in, and you'll see the problem. There is no way around this problem, which means that you have to set up the screen memory yourself if you want to take advantage of this feature. Sorry.

Double-height players, in case you were wondering, have dots the height of those in graphics mode 7, while regular-height players are the height of graphics mode 8. Despite the way I named the two, double-height players are given to you unless you specify otherwise by using SDMCTL.

SDLSTL
560,561 0230,0231

Another important location, SDLSTL holds the address of the display list. Let's talk display lists.

We already know about screen memory: the memory locations that hold the information that is to be displayed on the screen (see SAVMSC at 88,89). How does the computer know how to interpret this memory though? As we learned in SDMCTL, there is a special chip called ANTIC that takes care of the graphics. ANTIC has a list of commands that tells it how to display the screen memory. Oddly enough, this list is called the "display list." Since the display list is made up of commands, it's actually like a little program. And, since the screen memory has to be redisplayed 60 times a second, this program is a continuous loop, running over and over again. Why does the screen memory have to be redisplayed? The TV set draws a picture by making different parts of the screen glow at different brightnesses. The screen, however, will only glow for a very short period of time. Therefore, in order to get a picture to stay on the screen, the TV has to draw it 60 times a second.

For now, let's pretend that a display list is written just like a BASIC program, only with special commands. Let's see what such a display list would look like:

```
100 DRAW 8 BLANK SCAN LINES
110 DRAW 8 BLANK SCAN LINES
120 DRAW 8 BLANK SCAN LINES
130 CHARACTER MODE 0 LINE...
140 ...WITH SCREEN MEMORY
    STARTING AT [address]
150 CHARACTER MODE 0 LINE
160 CHARACTER MODE 0 LINE
•
•
•
370 CHARACTER MODE 0 LINE
380 GOTO 100 AND WAIT FOR
    VBLANK
```

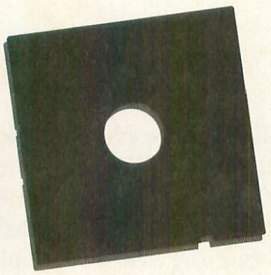
BIT(S)	PATTERN	VALUE	RESULT
0,1	------00	0	No playfield
	------01	1	Narrow playfield
	------10	2	Regular playfield
	------11	3	Wide playfield
2	-----0--	0	Missiles off
	-----1--	4	Missiles on
3	----0---	0	Players off
	----1---	8	Players on
4	---0----	0	Double height players
	---1----	16	Regular height players
5	--0-----	0	ANTIC off
	--1-----	32	ANTIC on
6,7	--------		Not used

FIGURE 1: SDMCTL chart

We start by telling ANTIC to leave the first 24 scan lines blank. Scan lines are the height of a graphics mode 8 line, start before the left edge of the screen and go all the way past the right edge. If you look closely at the screen, you can even see them. We leave 24 lines blank so that we can be sure that all of our picture will be on everyone's screen. TV and monitor screens all act slightly differently; so the blank lines will create a frame that can cover the edges of the screen. These blank lines make up the top of the black border that you can see in graphics mode 0.

Next we want to start our mode 0 lines, so we have a mode 0 line command. ANTIC has to know where the

Master Memory Map Part V



screen memory is before it can start drawing; so we make the first mode 0 command a special one that tells ANTIC that the address of the screen memory will come next. Then, after the screen memory address, we have 23 regular mode 0 commands. Finally, we tell ANTIC to go back to the beginning and start all over again after VBLANK. Remember that VBLANK is the time during which the TV is getting ready to start drawing the picture again. We want to make sure it's ready before ANTIC starts again; so we tell ANTIC to wait until VBLANK is over.

Now that we have a basic understanding, let's look at the specifics. First of all, ANTIC doesn't use line numbers. In a real display list, the line numbers would be memory locations. Secondly, ANTIC has abbreviations for all the commands. And thirdly, there is no thirdly. Let's therefore look at the proper way to write the preceding display list. We'll have it start at location 1000 (decimal), although it would usually be much higher in memory:

```
1000 BLK 8
1001 BLK 8
1002 BLK 8
1003 CHR 2 LMS
1004 <screen memory low byte>
1005 <screen memory high byte>
1006 CHR 2
1007 CHR 2
        •
        •
        •
1028 CHR 2
1029 JVB 1000
```

As you can see, this isn't that much different from our original. LMS, by the way, stands for Load Memory Scan and tells ANTIC that the next two bytes will be the address of the beginning of screen memory. Now, the final step is to convert these commands into numbers that we can POKE into memory. There is a unique number assigned to each command, and Figures 2 and 3 give you those numbers. Before you look at that chart, however, let me explain the other commands that you'll see there:

MAP is the same as CHR, except it's used to indicate a graphics mode rather than a character (text) mode.

JMP is like JVB, except it doesn't tell ANTIC to wait for the end of VBLANK. It's needed because of a quirk in ANTIC that says a display list can't cross a 1K boundary. What's a 1K boundary? It's a memory location that's a multiple of 1024. With display lists created by graphics commands, this is no problem. If you're designing your own, however, and you have to cross such a boundary, JMP over it. While we're on the topic of boundaries, you should also be aware that screen memory is not allowed to cross a 4K boundary. If it does, you have to have a second LMS instruction to get past the boundary. Under normal circumstances, however, this only happens in graphics modes 8 through 11, and the OS will take care of it for you.

HSC, like LMS, is not really a command but rather a modification to a command. It tells ANTIC that this mode line is to have the capability of fine horizontal scrolling. See HSCROL at 54276.

VSC, you guessed it, is another modification that specifies a fine vertical-scrolling capability. See VSCROL (54277).

DLI is the fourth modification, telling ANTIC that there is to be a display list interrupt at the end of this mode line. Pay particular attention to the "end." See VDSLST (512,513) for more details on DLIs.

Now that you'll be able to understand the chart, why don't you go take a quick peek at it.

Run the following program to take a look at the actual graphics mode 0 display list as it is stored in memory:

```
100 GRAPHICS 0
110 DLIST=PEEK(560)+PEEK(561)*256
120 PRINT PEEK(DLIST)
130 IF PEEK(DLIST)<>65 THEN DLIST=D
LIST+1:GOTO 120
140 PRINT PEEK(DLIST+1)
150 PRINT PEEK(DLIST+2)
160 END
```

Use CTRL-1 to pause the display list. Notice that the last two numbers PRINTed (the address for the JVB) are the same as the values in 560 and 561. If you can't figure out why, drink a couple of cups of coffee and read this whole description all over again. *Here is a hint*: They tell the computer to go back and use the same display list all over again. If you change the numbers here, the computer will use another display list at the new address, which means you could use several display lists at once.

We've pretty much covered the standard graphics display lists, but what about custom ones? It should have occurred to you by now that you can write your own display lists, mixing different graphics modes on the screen.

A few words before we move on. What can you use LMS for other than to tell AN-

Load Memory Scan (LMS) is a powerful tool that has no steadfast rules about its usage; use your creativity.

TIC where the screen memory is? Nothing! You can, however, tell ANTIC where the screen memory is more than once in the same display list. Why would you want to do that? Well, you have to if you're fine scrolling (see HSCROL and VSCROL). You could also do it to repeat the same line over and over again without wasting memory. LMS is another powerful tool that has no steadfast rules about what to use it for; use your creativity. Here's an example of repeating text. After the program has run, clear the screen and try typing in a line of text.

```
100 GRAPHICS 0
110 DLIST=PEEK(561)*256+67:POKE 560
,67
120 LOW=PEEK(88)
130 HIGH=PEEK(89)+2
136 POKE 89,HIGH
137 POKE DLIST,112
138 POKE DLIST+1,112
139 POKE DLIST+2,112
140 DLIST=DLIST+3
150 FOR ROW=0 TO 23
160 POKE DLIST,66
170 POKE DLIST+1,LOW
180 POKE DLIST+2,HIGH
190 DLIST=DLIST+3
200 NEXT ROW
210 POKE DLIST,65
220 POKE DLIST+1,PEEK(560)
230 POKE DLIST+2, PEEK(561)
```

What's going on here? Essentially we're rewriting a graphics mode 0 display list so that all the lines have LMSes that point to the same address. We also have to change SAVMSC (88,89), so that the OS knows where our new screen memory is. Why isn't the new screen memory in the same place as the old screen memory? Because our new display list overflows into the old screen memory, that's why.

If you press System Reset, a normal graphics mode 0 screen will appear.

SSKCTL
562 0232

SSKTL is used to control the serial port and is a shadow register for SKCTL (53775). As your state of confusion should indicate to you, it is not really a location for the inexperienced. Look at SKCTL if you *are* interested; look at the OS manual if you're *really* interested.

Noname
563 0233

This location is *currently* unused. Atari reserves the right to use it in future versions of the OS, so don't count on it being safe to use.

LPENH
564 0234

Ever hear of a light pen? The thing that looks like a pen, that you can draw on the screen with and so forth? Well, if you happen to be one of the lucky few who have one, this will tell you what horizontal position on the screen it's pointing to. Neat, huh?

LPENH is a shadow register for PENH (54284).

LPENV
565 0235

This is the vertical position of the light pen on the screen. It's a shadow register for PENV (54285).

Since light pens defy all reason, a few words about them are probably in order here. Firstly, LPENH and LPENV are set when the light pen is pressed to the screen. LPENV gets the value of VCOUNT (54283) when the pen was pressed (VCOUNT gets incremented by one every two scan lines). LPENH, on the other hand, gets a value based on the number of color clocks that have been drawn so far. See SDLSTL for an explanation of a "color clock." Now I'd be more than happy to give you the range of values that LPENH and LPENV will return as you move a light pen about the screen, but unfortunately the values depend on several things. Run the following program to see what the limits are for your computer. Also see STICKO-3 (632-635).

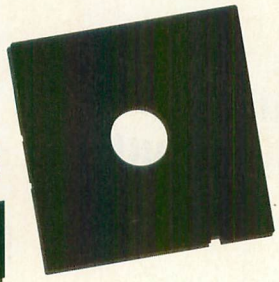
```
100 GRAPHICS 0
105 POKE 752,1
110 POSITION 2,11
120 PRINT "Light pen horizontal pos
ition = "
130 PRINT "          vertical posit
ion = "
140 POSITION 34,11
150 PRINT PEEK(564);"  ";
160 POSITION 32,12
170 PRINT PEEK(565);"  ";
180 GOTO 140
```

You should also note that light pens are not very precise. The values can vary slightly even if you hold it steadily at one point on the screen.

If you're writing a program that uses the light pen, be sure to allow for a little variation in the values. This can be done by "sampling" values. Basically, this means read ten or so values every time you need one, throw out the highest and lowest and average the rest to get a single value.

BRKKEY
566,567 0236,0237

Master Memory Map Part V



In the "B" version of the OS, this is the vector for the BREAK key interrupt. It is initialized to 59220, which means you will find 84 in location 566 and 231 in location 567. To disable the break key, POKE 143 into 566.

See BRKKEY at location 17 if you want to write your own BREAK key routine and you have the old OS. Also see locations 512 through 535 for more information on interrupt vectors.

Noname
568,569 0238,0239

More "unused" bytes. See the warning at location 563 about such bytes.

Okay, we're going back to I/O now. The next four bytes make up the Command Frame Buffer (CFB), a table that SIO uses when doing serial bus operations. Remember that the serial bus is what information travels back and forth on. It is *not* designed to be used by you; so you should be reading out of curiosity rather than necessity.

CDEVIC
570 023A

CDEVIC contains the device code.

CCOMND
571 023B

CCOMND contains the command code.

CAUX1
572 023C

CAUX1 contains the command auxiliary byte 1, which comes from DAUX1 at location 778.

CAUX2
573 023D

Finally, CAUX2 contains the command auxiliary byte 2, which SIO gets from DAUX2 at location 779.

TEMP
574 023E

SIO uses TEMP as a TEMPorary storage place (the people who name these things are *so* clever).

ERRFLG
575 023F

If any error occurred during device I/O, with the exception of a time-out, ERRFLG is set to 255. Otherwise, if everything is okay, it is set to 0.

Also see STATUS at location 48.

DFLAGS
576 0240

When a disk is booted (the computer is turned on with the disk drive on), the first disk "record"—a segment of information that has been recorded on the disk—is read from sector 1 into memory. A sector is a segment of the disk, shaped like a piece of pie. Some of the information from this record is used to continue the boot. The first byte in the record contains several useful flags. It gets stored in DFLAGS.

DBSECT
577 0241

The second byte tells how many sectors are used in the boot file. It gets stored in DBSECT.

BOOTAD
578,579 0242,0243

Finally, because we have to know where to put the file, the third and fourth bytes give the starting address. They get stored in BOOTAD. Once the OS knows BOOTAD, it moves the record it just read in over to the new address and starts loading in the rest of the file, putting each record after the previous one until the whole file is properly in memory.

BOOTAD gets transferred to RAMLO (4,5) which, because it is in page 0, is used to move the file from the sector buffer to its place in memory.

In most cases, the boot file is DOS, and BOOTAD will hold the address 1792.

COLDST
580 0244

COLDST is fun. The OS sets it to one during the power-up process, and then sets it to zero when everything has been properly initialized. If somebody presses System Reset, the OS looks at COLDST to see whether it was in the middle of power-up, when System Reset was pressed. If it was (COLDST equals one), then the turkey who did the pressing messed up the power-up, and the OS has to start all over again. Otherwise, the OS just treats it like a normal reset.

Fun? That's your idea of fun? Hey, let me finish. The OS isn't too smart; COLDST is the only way it knows whether or not it's in the middle of power-up when System Reset is pressed. That means you can set COLDST to one in your program, and if System Reset gets pressed, the computer will act as if somebody just turned the computer on. Your whole program will be erased rather than broken into (usually System Reset will cause the OS to jump to BASIC, where your program can be LISTed or SAVEd).

Use COLDST along with POKMSK (16) and STMCUR (138,139) to totally protect your BASIC program from being looked at or SAVEd. The disk or cassette they're

In the wonderful world of Atari graphics, there are two things that can appear on the screen: the playfield and player/missile graphics.

on could still be copied though. Another good use for this trick is to POKE 580,1 and press System Reset instead of using your On/Off switch when you want to load in another program. It saves wear and tear on the computer.

Noname
581 0245

Yet another unused byte for which the warning at location 563 applies.

DSKTIM
582 0246

The disk time-out register. We last saw time-outs at location PTIMOT (28). Well, they're back, this time for the disk drive (PTIMOT was for the printer). DSKTIM holds the time-out value for the FORMAT command. It is supposedly initialized to 160, but I have seen machines that initialize it to 120 and 224, which I suspect has something to do with different versions of the OS. Anyway, regardless of what it's initialized to, the value in DSKTIM is updated after every STATUS request with the value in DVSTAT+2 (748).

You should look at DTIMLO (774) for *lots* more information on disk time-out, including the exact use for DSKTIM.

LINBUF
583-622 0247-026E

Hey, remember buffers? This is a 40-byte-long buffer used by the screen editor. You see, the screen editor needs a place to temporarily store a line of text when it's moving stuff around on the screen. This is it.

ADDRESS (100,101) is used as a temporary zero-page pointer to LINBUF during the moving process.

GPRIOR
623 026F

GPRIOR is used to set priorities and to select GTIA modes.

In the wonderful world of Atari graphics, there are two kinds of things that can appear on the screen. First of all, there is the playfield. The playfield is what you get by PRINTing, PLOTing and DRAWTOing. The playfield is made up of as many as five colors, which are specified by the color registers (as in SETCOLOR color register, color, brightness). Each of these colors represents a different part of the playfield. The one with the same color as color register 0 is called playfield 0 and so forth. The one with the background color (color register 4) is called BAK.

The second type of thing that can appear on the screen is player/missile graphics. We'll be getting more into players and missiles in the CTIA/GTIA chip at location 53248, but for now just be aware that there are four players and four missiles that can appear on the screen at the same time as the playfield.

So, where is all of this getting us? Well, if you have player/missile graphics on the screen, and you also have a playfield, which should be seen when the two are in the same place? In other words, which should have priority over the other? GPRIOR tells ANTIC, the chip that draws the picture, who has the highest priority (i.e., who gets to be seen). Look at the chart in Figure 4.

"PF" means PlayField, "P" means Player. Missiles have the same priority as the player with the same number. Keep in mind that something with a higher priority will appear to move in front of something with a lower priority. Similarly, of course, something with a lower priority will appear to move behind something with a higher priority.

As you probably noticed, only the last four bits of GPRIOR are used to set the priority (make sure only one of those four bits is on). What about the other four? If you set bit 4, then all the missiles will have the same color as playfield 3. That lets you move the missiles together and use them as a fifth player (P4). If you set bit 5, then overlapping player 0 and player 1 will produce a third color in the overlap area. This goes for player 2 and player 3 as well. For machine languagers, or just the curious, the third color is produced by ORing the colors of the two players together.

As long as we're on the subject of overlap colors, if you do set more than one of the last four bits, then in a case where two overlapping objects have the same priority, the overlap area will be black.

In graphics modes 0 and 8, only the color of the text or pixels will be changed if a player or missile flies over them. The brightness will not change.

So now we're left with the seventh and eighth bits. They don't have anything to do with priorities or player/missile graphics. Instead they indicate whether or not a GTIA mode is being used, and if so, which one. They work as shown in Figure 5.

You only need to set these bits if you're writing your own display list. Otherwise the OS will take care of them when you use BASIC to call "GR.9, 10 or 11."

GPRIOR is a shadow register for PRIOR (53275).

GPRIOR	···· 0001 (0)	···· 0010 (2)	···· 0100 (4)	···· 1000 (8)	BIT PATTERN (VALUE)
HIGHER PRIORITY	P0	P0	PF0	PF0	
	P1	P1	PF1	PF1	
	P2	PF0	PF2	P0	
	P3	PF1	PF3+P4	P1	
	PF0	PF2	P0	P2	
	PF1	PF3+P4	P1	P3	
	PF2	P2	P2	PF2	
LOWER PRIORITY	PF3+P4	P3	P3	PF3+P4	
	BAK	BAK	BAK	BAK	

FIGURE 4: GPRIOR chart

Master Memory Map Part V

The next 24 locations (624-647) hold information about the joysticks, paddles and light pens.

PADDL0
624 0270

PADDL0 holds the current value of paddle 0 (the left paddle in the leftmost plug in the front of the computer). Paddles can have a value ranging from 0 to 228; the further you turn the paddle clockwise, the higher the value.

Paddles are actually little more than a "potentiometer." Let's look at a potentiometer as though it were a bathroom faucet. The computer sends a value 255 worth of water into the faucet, but the amount that comes out depends on how far open the faucet is. If it's all the way open (the paddle is turned clockwise as far as it will go), then almost all of the water will flow through (228 worth). If it's all the way closed (the paddle is turned counterclockwise as far as it will go), then none of the value will flow through. And that's exactly how a potentiometer works, except the computer is sending electricity into it rather than water (paddles aren't waterproof).

If you design a program that uses the paddles, be careful. Most of the time you won't want them to have a range of 0 to 228. For example, if you're using the paddles to move a player, the player will probably move off the screen. So what can you do to get the range that you want? Let's suppose you want to go from LOW to HIGH. First, do the following:

```
RANGE=HIGH-LOW+1
EACH=RANGE/228
```

These two lines should come somewhere in your program before you start using the paddles. Then, every time you read the paddle, do the following:

```
MYVAL=INT(OLDVAL*EACH+.5)+LOW
```

where OLDVAL is the value of the paddle, and MYVAL is the corresponding number in the range you wanted.

PADDL0 is a shadow register for POT0 at location 53760. Note that the value for the button (trigger) on the paddle can be found at PTRIG0 (636).

The following paddle locations work the same as paddle 0, but you change the number of the paddle (of course). Paddles 0 and 1 plug into the leftmost port (hold) numbered 1 on the front of the Atari computer. Likewise 6 and 7 plug into the rightmost port numbered 4 on the computer.

PADDL1
625 0271

The value for paddle 1 and a shadow register for POT1 at 53761.

PADDL2
626 0272

The value for paddle 2 and a shadow register for POT2 at 53762.

PADDL3
627 0273

I'll leave this and the next four for you to figure out. *Hint:* see the last three locations.

PADDL4
628 0274

PADDL5
629 0275

PADDL6
630 0276

PADDL7
631 0277

STICK0
632 0278

Let's see. PADDL0 was paddle 0, so I wonder what STICK0 is? Could it possibly be joystick 0? Despite all the odds against it, it is. Joystick 0 is the one plugged into the leftmost plug in the front of the computer.

Unlike paddle values, joystick values don't appear at first (or second) to make much sense. Let's take a look at those values (Figure 6).

00--------	(0)	No GTIA mode
01--------	(64)	GRAPHICS 9
10--------	(128)	GRAPHICS 10
11--------	(192)	GRAPHICS 11

FIGURE 5: GTIA chart

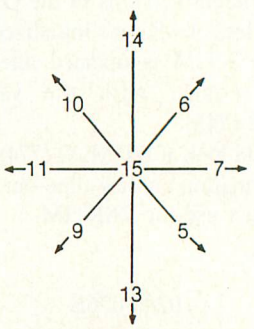


FIGURE 6: Joystick values in decimal

This figure represents the nine possible positions the joystick can be in, along with the value corresponding to each. If you move joystick 0 up, for example, STICK0

If you design a program that uses the paddles, be careful. Most of the time you won't want them to have a range of 0 to 228.

will have a value of 14. Now, unless you have some brilliant power of observation that I don't, these values don't seem to make any sense. I mean, does "14" mean "up" to you? Not to me. They must make some kind of sense to the computer, however, so let's take a look at them again (Figure 7), this time in binary, the way the computer sees them.

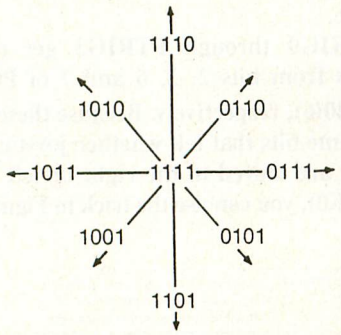


FIGURE 7: Joystick values in binary

It may not be immediately obvious, but now things make sense. Notice how the first bit (the digit on the right) of each value is only equal to zero when the joystick is up (straight up or diagonally up)? And the second bit is only zero when it's down, the third when it's left and the fourth when it's right. So we get Figure 8.

And *that's* why the numbers don't make sense when you first look at them.

---0	means "up"
---1	means "not up"
--0-	means "down"
--1-	means "not down"
-0--	means "left"
-1--	means "not left"
0---	means "right"
1---	means "not right"

FIGURE 8: Joystick bit chart

Here are a couple of machine-language routines to help you make a little more sense out of the joystick values. One looks for vertical movement and will return a zero for up, one for center and two for down. The other looks for horizontal movement and returns a zero for left, one for center and two for right. As you'll see from the following example, these values can prove to be very practical:

```
100 DIM STICKV$(19):DIM STICKH$(22)
110 FOR CHAR=1 TO 19
120 READ CODE
130 STICKV$(CHAR,CHAR)=CHR$(CODE)
140 NEXT CHAR
150 FOR CHAR=1 TO 22
160 READ CODE
170 STICKH$(CHAR,CHAR)=CHR$(CODE)
180 NEXT CHAR
200 GRAPHICS 0:POKE 752,1
210 PRINT :PRINT "Machine language
joystick example"
220 POSITION 10,4:PRINT "VERTICAL V
ALUE:"
230 POSITION 8,6:PRINT "HORIZONTAL
VALUE:"
240 VERT=USR(ADR(STICKV$),0)-1
250 HORZ=USR(ADR(STICKH$),0)-1
260 POSITION 26,4:PRINT VERT;"
 "
270 POSITION 26,6:PRINT HORZ;"
 "
280 GOTO 240
1000 DATA 104,104,133,213,104,170,1
89,120,2,41,3
1010 DATA 201,2,240,1,74,133,212,96
2000 DATA 104,104,133,213,104,170,1
89,120,2,74,74
2010 DATA 73,2,201,3,208,2,169,2,13
3,212,96
```

If you wanted to read joystick 1 instead of joystick 0, you'd use USR(ADR(STICKV$),1) and USR(ADR(STICKH$),1).

Here's the assembly code that's stored in the DATA statements:

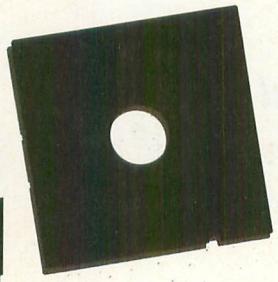
```
68      STICKV  PLA
68              PLA
85D5            STA  $D5
68              PLA
AA              TAX
BD7802          LDA  STICK0,X
2903            AND  #$03
C902            CMP  #$02
F001            BEQ  DONE
4A              LSR  A
85D4    DONE    STA  $D4
60              RTS
68      STICKH  PLA
68              PLA
85D5            STA  $D5
68              PLA
AA              TAX
BD7802          LDA  STICK0,X
4A              LSR  A
4A              LSR  A
4902            EOR  #$02
C903            CMP  #$03
D002            BNE  DONE1
A902            LDA  #$02
85D4    DONE1   STA  $D4
60              RTS
```

STICK0 is a shadow register for the last four bits (the leftmost four) of PORTA at location 54016. It is set to a value other than 15 when a light pen in the leftmost controller jack is pressed on the screen.

STICK1
633 0279

Same as STICK0 except it's a shadow register for the first four bits of PORTA rather than the last two. It's also the value for joystick 1 rather than joystick 0.

Master Memory Map Part V



			HSC	VSC	HSC VSC	LMS	HSC LMS	VSC LMS	HSC VSC LMS	DLI	HSC DLI	VSC DLI	HSC VSC DLI	LMS DLI	HSC LMS DLI	VSC LMS DLI	HSC VSC LMS DLI
BLK	1	00								80							
BLK	2	10								90							
BLK	3	20								A0							
BLK	4	30								B0							
BLK	5	40								C0							
BLK	6	50								D0							
BLK	7	60								E0							
BLK	8	70								F0							
JMP		01								81							
JVB		41								C1							
CHR	2	02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
CHR	3	03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
CHR	4	04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
CHR	5	05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
CHR	6	06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
CHR	7	07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
MAP	8	08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
MAP	9	09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
MAP	A	0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
MAP	B	0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
MAP	C	0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
MAP	D	0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
MAP	E	0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
MAP	F	0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF

FIGURE 2: Display list command chart, hex version

			HSC	VSC	HSC VSC	LMS	HSC LMS	VSC LMS	HSC VSC LMS	DLI	HSC DLI	VSC DLI	HSC VSC DLI	LMS DLI	HSC LMS DLI	VSC LMS DLI	VSC LMS DLI
BLK	1	0								128							
BLK	2	16								144							
BLK	3	32								160							
BLK	4	48								176							
BLK	5	64								192							
BLK	6	80								208							
BLK	7	96								224							
BLK	8	112								240							
JMP		1								129							
JVB		65								193							
CHR	2	2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
CHR	3	3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
CHR	4	4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
CHR	5	5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
CHR	6	6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
CHR	7	7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
MAP	8	8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
MAP	9	9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
MAP	10	10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
MAP	11	11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
MAP	12	12	26	44	60	76	92	108	124	140	156	172	188	204	220	236	252
MAP	13	13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
MAP	14	14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
MAP	15	15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

FIGURE 3: Display list command chart

STICK2
634 027A

Same as STICK1 except it's for joystick 2, and it's also a shadow register for the last four bits of PORTB (54017).

STICK3
635 027B

Joystick 3 (the rightmost one) value and a shadow register for the last four bits of PORTB.

PTRIG0
636 027C

If you press the trigger on paddle zero, PTRIG0 will have a value of zero. If you don't press it, PTRIG0 will have a value of one.

PTRIG0 through PTRIG3 get their values from bits 2, 3, 6 and 7 of PORTA(54016), respectively. Because these are the same bits that tell whether joysticks 1 and 2 are moved to the right or left (see STICK0), you can use the trick in Figure 9.

```
PTRIG(1) - PTRIG(0) = -1 if joystick zero is moved to the left
                    =0 if joystick zero is in the center
                    =1 if joystick zero is moved to the right
```

FIGURE 9: PTRIG chart

The same holds true for PTRIG(3)-PTRIG(2) and joystick 1. You can use this trick to make horizontal movement easier to program. Just add the value of the PTRIG difference to your old horizontal position. This saves trying to figure out the joysticks. For the same ease in vertical movement, use the routine given for STICK0.

PTRIG1
637 027D

Same as PTRIG0 but for paddle 1.

PTRIG2
638 027E

Trigger value for paddle 2.

PTRIG3
639 027F

Trigger value for paddle 3.

PTRIG4
640 0280

Trigger value for paddle 4.

PTRIG4 through PTRIG7 get their values from bits 2, 3, 6 and 7 of PORTB (54017), respectively. The same trick for horizontal movement that was described under PTRIG0 can be applied to joystick 2 and joystick 3 using PTRIG4 through PTRIG7.

PTRIG5
641 0281

Trigger value for paddle 5.

PTRIG6
642 0282

Trigger value for paddle 6.

PTRIG7
643 0283

Guess.

STRIG0
644 0284

Well, here we are again at another new, different and challenging name for a location. For the next three as well, actually. All three STRIG locations hold the values for the joystick button, and work exactly the same way as PTRIG (zero means pressed).

The STRIGS are shadow registers for the TRIGS (53264 to 53267).

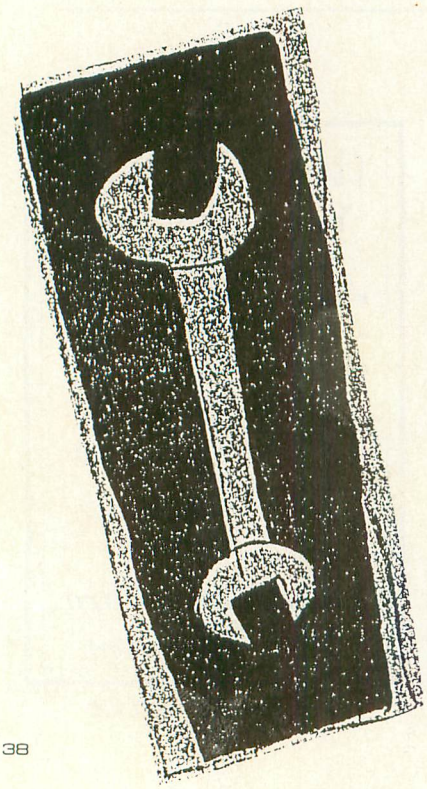
STRIG1
645 0285

STRIG2
646 0286

STRIG3
647 0287

Action! Graphics Toolkit

by Monty McCarty



Action! is a terrific language. It is fast, flexible and surprisingly easy to program in. Unfortunately, as you might have noticed with most other Atari programming languages, it comes up a little short on graphics routines other than the usual plot, DRAWTO and occasional player/missile commands. In an attempt to remedy this, and hopefully to get the ball rolling for more (and better) types of these routines to appear for all languages, here's the Action! Graphics Toolkit.

These routines reflect a bias of mine for working in high resolution, with speed being all important. To achieve this I used Clint Parker's fast graphics-eight plot routine from ANALOG #18. Also, for speed's sake, error trapping is at a bare minimum, so be careful.

Here are some general rules concerning the use of these routines. First, to see the demo program, just type in Listing 1 with your Action! cartridge (check your typing with D:CHECK in Action!) and save a copy before you compile and run it. To use the routines in your own programs, remove the procedure DEMO( ), since it is not needed. The variables declared at the beginning of the program should always be included. The screen buffer takes a big chunk of memory; so though there is enough memory for a fairly complex program, you will probably have to compile from disk.

In the descriptions below, the <plot>, <memory> and <text> after each command name tells you the command's type. The range of numbers for anything dealing with plot-type commands are 0-319 for X values and 1-192 for Y. The CX and CY parameters in a command such as BOX must always be larger than the X and Y parameters. The X and Y in such a command will always represent the upper left corner, and CX and CY will be the lower right corner.

The range for text and memory-type commands are 0-39 for X values and 0-191 for Y. Finally, use the COLOR variable normally to select plot colors 0 and 1.

Now, a short description of the commands and how to use them:

SCREEN(X,Y,CX,XY) — <plot>: This command defines the rectangular area of clipping for all commands which use the PLOT command. Nothing can be plotted outside of the area defined. This command does not work for any command dealing with text or moving blocks of memory.

UNCLIP( ) — <plot>: Simply returns the screen limits to the default values.

PLOT(X,Y) — <plot>: A very fast plot command slowed down somewhat by the addition of boundary error checking. You can now run a circle off the screen and not fear the dreaded Error 141, cursor out of range.

VLINE(Y,CY,X) — <plot>: A faster way to draw a straight vertical line than using DRAWTO. Y must be a smaller number than CY. X is the axis along which the line will be drawn.

HLINE(X,CX,Y) — <plot>: The horizontal version of the above, with similar restrictions.

DRAWTO(CX,CY) — <plot>: This will draw a line from the last point plotted to the point specified, slightly faster than the built-in DRAWTO. You can chain your DRAWTO's without PLOTs in between to create a series of lines.

CIRCLE(X,Y,R) — <plot>: Yes, this draws a circle. X and Y will be the center and R the radius. Remember, you can run circles off the screen or, using SCREEN, show only portions of circles, perhaps to create arcs.

BOX(X,Y,CX,CY) — <plot>: Draws unfilled rectangles.

FRAME( ) — <plot>: If you want to see the boundaries of the area you SCREENed off, FRAME will draw a border around it.

MOVE(T,F,L,M) — <memory>: The same as Action!'s MOVEBLOCK procedure except with the addition of being able to manipulate how the source memory is merged with the destination memory. As in MOVEBLOCK, T is for the destination address, F is for the origin address and L is for the number of bytes to move. If M equals one then the move works just like MOVEBLOCK. If it is two, then the source is ORed to the destination. A three means it will be ANDed and a four means it will be exclusive ORed. Check the Action! manual to find out what these mean, or better yet, try changing the options in the demo program and see for yourself.

CUT(X,Y,CX,CY) — <memory>: This will allow you to grab a block of screen memory and store it into a buffer for later use. In the demo program, the buffer, array CS, holds 1K of data. You can change it to any size your available memory can support. Remember, the screen is 40 bytes across by 192 lines down which equals 7680 bytes.

BLOCK(X,Y,CX,CY,F) — <memory>: Makes a solid block. Useful for erasing areas quickly, special effects, etc. Since this is a memory-related command, its horizontal resolution is only 40, limiting its usefulness somewhat. The F (0-255) parameter, however, allows you to define the pattern made by the block, like colors and vertical patterns.

PASTE(X,Y,M) — <memory>: This puts whatever is in the CS buffer back into screen memory, and using the M parameter allows you the same options as MOVE to alter the result. Only the upper left (X,Y) corner need be provided. The rest is done for you.

PRINT8(ST,X,Y,SZ,M) — <text>: Writes text on the high-resolution screen in 24 (1-24) vertical sizes with the same M options as MOVE and PASTE. ST can be any byte, char or card array. X and Y is where the text will start and SZ is the vertical height—one is normal. In this case, M can be use to remove the white space surrounding the text or to create an inverse video effect. Check out the demo.

OPENW(X,Y,LN,LINES,SZ,N) — <text>: Opens a window into which a text message is written. X and Y determines where the first line of text will start. LN is the length of the longest line of text in the window. LINES is the number of lines of text in the window. SZ is the size of the text and N determines if a frame will be made (1) or not (any other number). To set up a text window, first place your text in the card array, TW. The first line of your text should be in the zero element (TW(0)="line one") of the array, the second line in the first element and so on, only one line per element. Determine your values for the above parameters, and that is pretty much all there is. The nice thing is that nothing written over by the text window will be harmed. Simply close the text window and it disappears, leaving your screen in its original state. You may have only one window open at a time and preserve the screen.

CLOSEW(D) — <text>: Closes the text window with the option to wait a predetermined amount of time. CLOSEW() is enough to just close the window. Since Action! is so fast, if you want a delay of any length, provide a fairly large number for D.

GWINDOW(X,Y,CX,CY,N) — <plot>: Will create a solid-filled high-resolution box with or without a frame. If N equals one, a frame is drawn around the box, any other number means it won't.

Some suggested uses

Here are some ideas to get you started. If for example, you are writing a paint program and need an undo feature, try this: Before a command that will alter the screen is executed, save the screen to the screen buffer (array BF) with the following command MOVEBLOCK(BF,SC,7680); then to retrieve the screen to "undo" a mistake, MOVEBLOCK(SC,BF,7680) and the screen is back.

If you have a RAMdisk, you could replace the screen buffer by saving and retrieving the screen to the RAMdisk with bput and bget from the Action! toolkit. It is a little slower, but it works and saves 7680 bytes of program space. Bput and bget are also very useful for saving and loading screens to and from the disk drive.

It is possible to use CUT and PASTE to create simple motion or even animation. There is a simple example in the demo program using CUT, PASTE and MOVEBLOCK. I used MOVEBLOCK here since it is faster than MOVE.

If you have a Koalapad, Listing 2 gives an Action! routine for drawing with the Koalapad in graphics eight. Listing 3 is a BASIC program that creates data files necessary to run the Koalapad routine. When you run the routine, make sure these files are on a disk in Drive 1. I could have included the data in the program itself, but that would mean typing 456 numbers.

There are a lot more things you can try and many more routines to be written. I hope you find these routines useful.

Monty McCarty, who lives in Goldsboro, North Carolina, recently graduated from East Carolina University with a B.A. in commercial art and illustration. He has had an Atari for over five years, and his primary interest is computer graphics.

LISTING 1

```
; ACTION! GRAPHICS TOOLKIT
; by Monty McCarty
;
; Copyright 1988
; by ANALOG Computing
;
;      CHECKSUM DATA
;[33 BD 83 64 E9 A4 E3 92
; F5 59 85 A6 54 7D C5 8B
; B1 B8 16 2B F1 09 EC 8C
; F7 8F CE 4E FB D6 6A 57 ]

CARD SavMsc=88,OldCol=91,
     Xmin=[0],Xmax=[319],ET
CARD ARRAY Line(192),TW(20)
BYTE Color1=709,Color2=710,CPL,
     Color4=712,CharSet=57344,
     OldRow=90,Ymin=[1],Ymax=[192]
BYTE ARRAY D8(320),BF(7680),C5(1024),
     M1(8)=[128 64 32 16 8 4 2 1],
     M2(8)=[$7F $BF $DF $EF
            $F7 $FB $FD $FE]

INT FUNC Abs(INT N)
  IF N<0 THEN RETURN(-N) FI
RETURN(N)

PROC Screen(CARD X,BYTE Y,
            CARD CX,BYTE CY)
  Xmin=X Xmax=CX Ymin=Y Ymax=CY
RETURN

PROC Unclip()
  Xmin=0 Xmax=319 Ymin=1 Ymax=192
RETURN

PROC Plot(CARD X,BYTE Y)
  BYTE POINTER LOC
  OldCol=X OldRow=Y
  IF X<Xmin OR X>Xmax
    OR Y<Ymin OR Y>Ymax THEN
    RETURN
  FI LOC=Line(Y)+D8(X)
  IF COLOR#0 THEN
    LOC^==%M1(X&7)
    ELSE LOC^==&M2(X&7)
  FI
RETURN

PROC Vline(BYTE Y,CY,CARD X)
  DO Plot(X,Y) Y==+1 UNTIL Y>CY OD
RETURN

PROC Hline(CARD X,CX,BYTE Y)
  DO Plot(X,Y) X==+1 UNTIL X>CX OD
RETURN

PROC Drawto(CARD CX BYTE CY)
  BYTE Y,XF,YF,J,AY
  CARD X,I,AX
  INT A,B,T,DX,DY
  AX=OldCol AY=OldRow Plot(AX,AY)
  IF CX>AX THEN
    DX=CX-AX XF=0 ELSE DX=AX-CX XF=1
  FI
  IF CY>AY THEN
    DY=CY-AY YF=0 ELSE DY=AY-CY YF=1
  FI
  IF DX<2 AND DY<2 THEN
    RETURN
  FI X=AX Y=AY
  IF DX>DY THEN
    A=DY+DY T=A-DX B=T-DX
      FOR I=2 TO DX DO
       IF XF=0 THEN
        X==+1 ELSE X==-1
       FI
       IF T<0 THEN
        T==+A ELSE T==+B
        IF YF=0 THEN
         Y==+1 ELSE Y==-1
        FI
       FI Plot(X,Y)
```

```
  OD ELSE A=DX+DX T=A-DY B=T-DY
FOR J=2 TO DY DO
  IF YF=0 THEN
   Y==+1 ELSE Y==-1
  FI
  IF T<0 THEN
   T==+A ELSE T==+B
    IF XF=0 THEN
     X==+1 ELSE X==-1
    FI
  FI Plot(X,Y)
    OD
  FI Plot(CX,CY)
RETURN

PROC Circle(CARD X,BYTE Y,CARD R)
  INT CIR,CIRY,CIRXY,CX,CY
  CIR=0 CX=R CY=0
  DO
    CIRY=CIR+CY+CY+1
    CIRXY=CIRY-CX-CX+1
    Plot(X+CX,Y+CY) Plot(X-CX,Y+CY)
    Plot(X+CX,Y-CY) Plot(X-CX,Y-CY)
    Plot(X+CY,Y+CX) Plot(X-CY,Y+CX)
    Plot(X+CY,Y-CX) Plot(X-CY,Y-CX)
    CIR=CIRY CY==+1
     IF Abs(CIRXY)+0<Abs(CIRY) THEN
      CIR=CIRXY CX==-1
     FI UNTIL CY>CX
  OD
RETURN

PROC Box(CARD X,BYTE Y,
         CARD CX,BYTE CY)
  Hline(X,CX,Y) Vline(Y,CY,CX)
  Hline(X,CX,CY) Vline(Y,CY,X)
RETURN

PROC Frame()
  Box(Xmin,Ymin,Xmax,Ymax)
RETURN

PROC Move(BYTE POINTER T,F,
          CARD L,BYTE M)
  CARD C
  IF M<1 OR M>4 THEN
    M=1
  FI C=0
  DO
    IF M=1 THEN
      T^=F^
     ELSEIF M=2 THEN
      T^==%F^
     ELSEIF M=3 THEN
      T^==&F^
     ELSEIF M=4 THEN
      T^==!F^
    FI F==+1 T==+1 C==+1 UNTIL C=L
  OD
RETURN

PROC Cut(BYTE X,Y,CX,CY)
  CARD S,E,CT
  CT=CS S=SavMsc+(Y-1)*40+X CPL=CX-X
  E=SavMsc+CY*40+X ET=E-S
  DO
    Moveblock(CT,S,CPL)
    CT==+CPL S==+40
    UNTIL S=E OR S>=SavMsc+7680
  OD
RETURN

PROC Paste(BYTE X,Y,M)
  CARD S,E,CT
  CT=CS S=SavMsc+(Y-1)*40+X E=S+ET
  DO
    Move(S,CT,CPL,M) CT==+CPL S==+40
    UNTIL S=E OR S>=SavMsc+7680
  OD
RETURN

PROC Print8(BYTE ARRAY ST,
            BYTE X,Y,SZ,M)
  CARD S,E,CT
  BYTE A,B,C,D,LEN
  CT=SavMsc+Y*40+X B=1 S=CT LEN=ST(0)
  DO C=ST(B)
    IF C>127 THEN
     C==-126
    FI
    IF C>31 AND C<96 THEN
     C==-32
     ELSEIF C<32 THEN
      C==+64
    FI E=@CharSet+C*8 A=0
    DO D=0
     DO Move(S+40*D,E+A,1,M)
      D==+1 UNTIL D=SZ
     OD S==+40*SZ A==+1
     UNTIL A=8 OR S>=7680+SavMsc
    OD S=CT+B B==+1 UNTIL B=LEN+1
  OD
RETURN

PROC Block(BYTE X,Y,CX,CY,F)
  CARD S,E
  BYTE L
  S=SavMsc+(Y-1)*40+X
  L=CX-X E=SavMsc+CY*40+X
  DO
    Setblock(S,L,F) S==+40
    UNTIL S=E OR S>=SavMsc+7680
  OD
RETURN

PROC OpenW(BYTE X,Y,LN,LINES,SZ,N)
  BYTE W,H,C
  Moveblock(BF,SavMsc,7680)
  W=X+LN+2 H=LINES*(SZ LSH 3)+1
  Block(X,Y,W,Y+H,0)
  Screen(X LSH 3,Y,W LSH 3,Y+H)
  IF N=1 THEN
    Frame()
  FI Unclip()
  FOR C=0 TO LINES-1 DO
    Print8(TW(C),X+1,Y+(SZ*8)*C,SZ,1)
  OD
RETURN

PROC CloseW(CARD D)
  CARD A
  FOR A=0 TO D DO OD
  Moveblock(SavMsc,BF,7680)
RETURN

PROC Gwindow(CARD X,BYTE Y,
             CARD CX,BYTE CY,N)
  CARD XA,XB
  BYTE YA,YB
  Screen(X,Y,CX,CY)
  XA=(CX-X)RSH 1+X XB=XA
  DO
    IF XA>=X THEN
     Vline(Y,CY,XA) XA==-1
    FI
    IF XB<=CX THEN
     Vline(Y,CY,XB) XB==+1
    FI UNTIL XA<X
  OD
  IF N=1 THEN
    Frame()
  FI Unclip()
RETURN

PROC Init()
  CARD I
  BYTE POINTER LINELOC
  LINELOC=SavMsc
  FOR I=1 TO 192 DO
    Line(I)=LINELOC LINELOC==+40
  OD
  FOR I=0 TO 319 DO
    D8(I)=I RSH 3
  OD
RETURN

PROC Demo()
  CARD X,Y,C,D
  BYTE KEY=764
  BYTE ARRAY
   J1="___File___Edit___Text",
   J2="____________________"
  GRAPHICS(24) Init() COLOR=1
  Color1=0 Color2=12 Color4=12
  Print8(J1,0,0,1,1)
  Print8(J2,21,0,1,1)
```

```
  Gwindow(25,40,300,85,1)
  Block(5,50,10,75,128)
  Block(11,50,16,75,127)
  Block(17,50,23,75,229)
  Block(24,50,30,75,170)
  Block(31,50,36,75,85)
  Print8("TEXT SIZE 1",0,9,1,1)
  Print8("TEXT SIZE 2",4,16,2,1)
  Print8("TEXT SIZE 3",8,29,3,4)
  Screen(10,100,309,140) Frame()
  FOR X=0 TO 360 STEP 60 DO
    FOR D=1 TO 30 STEP 3 DO
     Circle(X,120,D)
    OD
  OD Unclip()
  FOR X=5 TO 310 STEP 15 DO
    Box(X,170,X+10,185)
  OD
  TW(0)="THIS IS"
  TW(1)="A TEST"
  OpenW(2,9,7,2,1,1) CloseW(50000)
  TW(0)="OF THE"
  TW(1)="POWER OF"
  TW(2)="THE AMAZING"
  OpenW(9,9,11,3,2,1) CloseW(50000)
  TW(0)="ACTION!"
  TW(1)="GRAPHICS"
  TW(2)="TOOLKIT"
  OpenW(16,9,8,3,3,1)
  Moveblock(BF,SavMsc,7680)
  Cut(0,1,10,50)
  FOR X=1 TO 99 DO
    Paste(X,130,X/20)
    Moveblock(SavMsc,BF,7680)
  OD
  TW(0)="PRESS ANY KEY TO EXIT"
  OpenW(8,84,21,1,4,1)
  DO
    UNTIL KEY#255
  OD KEY=255
RETURN
```

LISTING 2

```
;      CHECKSUM DATA
;[C0 AC 3F D7 02 ]

BYTE XDIR=624,YDIR=625,LEFTB=636,
     RIGHTB=637,FL=[0]
INT  OX,OY,X1,Y1
INT  ARRAY X(228),Y(228)

PROC FILLXY()
  BYTE A
  CARD B
  OPEN(1,"D:XARRAY.DAT",4)
  FOR A=1 TO 228 DO
   B=GETD(1)
    IF A>=183 THEN
     B==+255
    FI X(A)=B
  OD CLOSE(1) OPEN(1,"D:YARRAY.DAT",4)
  FOR A=1 TO 228 DO
   B=GETD(1) Y(A)=B
  OD CLOSE(1)
RETURN

PROC DRAW()
  IF FL=1 THEN
   IF OX-X1>11 OR OY-Y1>11 THEN
    RETURN ELSE DRAWTO(X1,Y1)
    OX=X1 OY=Y1 RETURN
    FI ELSE FL=1 PLOT(X1,Y1)
    OX=X1 OY=Y1
  FI
RETURN

PROC CHECKBUTTONS()
  DO
   DO X1=X(XDIR) Y1=Y(YDIR)
    IF XDIR<=4 AND YDIR<=4 THEN
     FL=1 EXIT
    FI
    IF LEFTB=0 THEN
     DRAW() ELSE FL=0
    FI
   OD
  OD
RETURN

PROC MAIN()
  BYTE COL1=709,COL2=710,COL3=712
  GRAPHICS(24) COL1=0 COL2=10
  COL3=10 COLOR=1
  FILLXY() CHECKBUTTONS()
RETURN
```

LISTING 3

```
CL 15 XDIV=320/228:YDIV=191/228
VN 20 OPEN #1,8,0,"D:XARRAY.DAT"
NH 25 FOR A=1 TO 228:B=INT(XDIV*A):PUT #1
   ,B:NEXT A:CLOSE #1
WH 30 OPEN #1,8,0,"D:YARRAY.DAT"
OH 35 FOR A=1 TO 228:B=INT(YDIV*A):PUT #1
   ,B:NEXT A:CLOSE #1
```

D:CHECK in Action!

by Steven Yates

Because of the nature of the Action! system, typing checkers like D:CHECK (Issue 16) cannot be implemented. A lack of line numbers leaves no way to communicate to the user where typos are. Moreover, the flexibility in the source program's form makes finding errors difficult—without requiring the reader to type the program *exactly* as it appears.

D:CHECK in Action! gives you a program (D:CHECK.ACT) which works with the Action! system to provide interactive checking and correcting of typing errors. Instead of printing a list of numbers which you compare to a similar list in the magazine, this Action! version finds the errors and puts you back into the editor at their approximate locations in your source.

This extra power takes some more typing on your part. If you look at Listing 1, you'll notice the first few lines contain a set of numbers headed "CHECKSUM DATA." All other Action! programs printed in ANALOG Computing will have similar lines at the start of their listings.

These lines give D:CHECK in Action! its power. The program generates numbers from what you've typed and compares them to the numbers here. If there are any discrepancies, it will then help you to find the problems so they can be corrected.

When typing in the listing, these lines must be typed in *exactly as printed.*

There are square brackets at the beginning and end of the checksum list, and one space between each value. All values are two-digit numbers and must be typed as such. After these lines are typed, the remainder of the program can be entered in any way you wish.

You may type it as listed, or you may decide not to include the blank lines inserted for easier reading. You may add *more* blank lines, if you like. You may combine short lines to form one line with a space or more between the originals; or you could break long lines into two or more lines, so everything fits on the screen.

The program ignores spaces; so you don't need to indent lines, and you can add spaces if it's more readable for you. D:CHECK in Action! also ignores comments, so you can leave them out and save some typing or add some of your own.

This flexibility of program form is basic to the Action! system which D:CHECK in Action! preserves. Remember, any modifications to the form of the program may make it more difficult to compare to the original if there's a problem—use your judgement. Anything between quotes must be typed ex-

D:CHECK in Action! gives you a program which works with the Action! system to provide interactive checking and correcting of typing errors.

actly as shown, including spaces and upper- and lowercase letters.

The latter presents another problem. Action! allows you to type in whatever case you want, but also offers the option of being case sensitive. This lets you have two or more variables of the same name, with different letters in uppercase to distinguish them.

D:CHECK in Action! offers the same option. If the article doesn't specify otherwise, you may use either case. If the program must be compiled with the case-sensitive option, the words *Case Sensitive* will appear with the checksum data. You must type all letters in the case in which they're listed. Because it must be able to handle other case-sensitive programs, D:CHECK in Action! must also be able to compile under this option; so remember to type each letter properly.

Using it

First, type in Listing 1 and save it. Because the program can't check itself until it compiles successfully, you must be careful keying it in. Don't forget that the program is case-sensitive.

Try compiling. If there are any compile errors, fix them as you normally would. Once D:CHECK in Action! compiles, run it. If the program says there are no problems, you're ready to use it for other programs printed in ANALOG Computing.

If it tells you there are problems, follow the directions below. The program cannot guarantee the locations it gives for errors in its own source file because typing errors may be causing problems with its error-locating routines. Be sure to save a final copy after you've fixed all the errors, as *D:CHECK.ACT* (cassette users may save it as *C:)*.

To check another program containing the checksum numbers, type it in and save a copy. With the newly typed program still in memory, go to the monitor. If the article says the program is case-sensitive, use the monitor option command to set it for this; otherwise set this option to "no."

When this is done, type R*"D:CHECK. ACT"* (cassette users, type R"C:"). The program will load from disk, compile, then run. It must be uncompiled when stored, because it needs to relocate itself for each program checked.

If the program being checked isn't too large (about 100 disk sectors), time can be saved by loading D:CHECK.ACT into the second editing window. It can then be compiled and run without accessing the disk each time.

To do this, type in the program to be checked and save it. Enter Window 2 by pressing CTRL-SHIFT-2, read D:CHECK.ACT, then enter the monitor and use *C* and *R* to compile and run.

To rerun D:CHECK.ACT after a problem's corrected, just enter Window 2 (CTRL-SHIFT-2) before returning to the monitor, then compile and run as before. If you don't move the cursor to Window 2 before entering the monitor, Action! will attempt to compile the program in Window 1, and D:CHECK.ACT won't be executed.

If you get an out-of-memory error when loading or compiling D:CHECK.ACT, make sure your source program is saved and use the monitor boot command to reset Action!. Then reload your program and run D:CHECK.ACT from disk or tape as explained previously.

If the D:CHECK.ACT program says there are no problems, you should be able to compile the program. If it does find a problem, it will display the checksum lines on-screen, with one number highlighted. It will ask if the sum was typed correctly.

Check the highlighted value against the magazine listing. If they don't match, press *N* in response to its question. Return to the editor. The cursor will be on the first digit of the incorrect sum. Retype the sum, return to the monitor and repeat the command to run D:CHECK.ACT.

If the highlighted sum matches that printed in the magazine, press *Y*. The program will tell you to return to the editor and check the line containing the cursor, plus a certain number of lines following it. The number of lines to check depends mostly on the line length, and blank lines aren't counted.

Find any mistakes you can on these lines, correct them, return to the monitor and rerun D:CHECK.ACT. Once you've found all errors and are given a clean check, save a final copy, then compile and run it, according to the directions in the related article.

It's possible that a program which checks out all right won't compile. If this happens, return to the editor and make sure you didn't insert or delete a space, which would confuse the compiler. Check the word the cursor's on and change it to look exactly as it does in the magazine. This should be the only thing to cause a problem in a properly checked program.

Now you should be able to enjoy printed Action! programs without being frustrated by cryptic compiler errors or having a program compile and not perform as expected. If you remember the difference D:CHECK made in the time it took to get a BASIC program running, you'll type in D:CHECK in Action! as soon as possible.

Because the program can't check itself until it compiles successfully, you must be careful keying it in. Don't forget it is case-sensitive.

D:CHECK in Action!

```
;  CHECK.ACT
; Steven Yates
;   12/02/85

;      CHECKSUM DATA
;[9E 43 B1 2D 74 DD 67 7C
; 13 30 E5 8F 7A CA C9 77
; AD AE 96 44 B0 F8 99 39
; EB ]


BYTE StartChar,CurChar,Count,X,
     Flag=[0],Character,Sum,ISum,
     Product,Key,Case=[0],Column=1152,
     Sensitive=[1],Lines,SumLine,
     WrongLine,Segment=[0],String=[1],
     SubString=[2],Space=[32]
CARD StartLine,Line=1160,
     FirstLine=1156
BYTE ARRAY Sums(256)
CARD ARRAY SumLines(32)
BYTE POINTER Length
CARD POINTER CurLine,NextLine,
             WrongSum

DEFINE is="=", not="<>", Done="Flag=1"


PROC End_Of_Line()

  DO
    CurLine=NextLine^
    CurChar=7
    Length=CurLine+6
    NextLine=CurLine+4
    IF CurLine=0 THEN
      EXIT
    FI
    UNTIL Length^>0
  OD

RETURN

PROC Quotes()

  IF Segment is String THEN
    Segment=0
  ELSE
    Segment=String
  FI

RETURN

PROC Ignore()

  X==-1
  Count==-1
  Character=0

RETURN

PROC Check_Line()

  Character=Peek(CurLine+CurChar)
  IF Segment is SubString THEN
    Segment=0
  ELSEIF Character='" THEN
    Quotes()
  ELSEIF Segment is String THEN
  ELSEIF Character='' THEN
    Segment is SubString
  ELSEIF Character is Space THEN
    Ignore()
  ELSEIF Character='; THEN
    Ignore()
    CurChar=Length^+6
  ELSEIF Case not Sensitive AND
         Character>96 AND
         Character<123 THEN
    Character==-32
  FI
  Product=(X*Character) RSH Case
  Sum==+Product
  CurChar==+1
  X==+1
  IF X=4 THEN
    X=1
  FI

RETURN

PROC Find_Sums()

  DO
    DO

  Character=Peek(CurLine+CurChar)
  IF Character not '; THEN
    End_Of_Line()
  ELSEIF Length^>1 THEN
    EXIT
  ELSE
    End_Of_Line()
  FI
  IF CurLine=0 THEN
    Done
    EXIT
  FI
OD
CurChar=8
Character=Peek(CurLine+CurChar)
IF Character is '[ THEN
  EXIT
FI
IF Done THEN
  PrintE("Listing does not")
  PrintE("contain checksums.")
```

D:CHECK in Action!

```
      PutE()
      PrintE("Cannot CHECK!")
      EXIT
    FI
    End_Of_Line()
  OD

RETURN

PROC Get_Sums()

 BYTE I
 BYTE ARRAY Hex(1)

  Find_Sums()
  SumLine=0
  DO
    IF Done THEN
      EXIT
    FI
    SumLines(SumLine)=CurLine
    CurChar=9
    FOR ISum=0 TO 7 DO
      FOR I=0 TO 1 DO
        Hex(I)=Peek(CurLine+CurChar+I)
        IF Hex(I)>='A THEN
          Hex(I)==-('A-'9-1)
        FI
        Hex(I)==-'0
      OD
      Sum=(Hex(0) LSH 4)+Hex(1)
      Sums(SumLine*8+ISum)=Sum
      IF Peek(CurLine+CurChar+3) is ']
       THEN
        Done
        EXIT
      FI
      CurChar==+3
    OD
    End_Of_Line()
    IF Done THEN
      Flag=0
      EXIT
    FI
    SumLine==+1
  OD

RETURN

PROC Mistyped()

  Line=StartLine
  Column=StartChar
  PrintE("Return to editor and check")
  Print("the ")
  PrintB(Lines)
  PrintE(" lines following the line")
  Print("the cursor is on for a ")
  PrintE("typo.")

RETURN

PROC Bad_Sum()

 BYTE I

  IF Case is Sensitive THEN
    Print("If article does not ")
    PrintE("specify")
    Print("case sensitive, use ")
    PrintE("option")
    Print("command to set this to ")
    PrintE("no.")
    PutE()
    PutE()
FI
WrongLine=ISum/8
Sum=ISum&7
WrongSum=SumLines(WrongLine)+9+Sum*3
WrongSum^==%$8080
FOR I=0 TO SumLine DO
  PrintE(SumLines(I)+6)
OD
WrongSum^==!$8080
PutE()
Print("Is highlighted sum correct?")
Key=GetD(1)
Key==%32
PutE()
  PutE()
  IF Key='y THEN
    Mistyped()
  ELSE
    Print("Return to editor and ")
    PrintE("correct")
    PrintE("mistyped sum.")
    Line=SumLines(WrongLine)
    Column=(Sum+1)*3
  FI

RETURN

PROC No_Problems()

  PrintE("Program CHECKs out fine.")
  PrintE("Save program and use")
  PrintE("according to directions")
  PrintE("in the article.")
  Done

RETURN

PROC Check_Sum()

  IF Sum<>Sums(ISum) THEN
    Bad_Sum()
    Done
  FI
  ISum==+1

RETURN

PROC Initialize()

  IF Peek(1226)=255 THEN
    Case is Sensitive
  FI
  X=1
  CurLine=FirstLine
  Length=CurLine+6
  NextLine=CurLine+4
  IF Length^=0 THEN
    End_Of_Line()
  FI
  CurChar=7
  Close(1)
  Open(1,"K:",4,0)

RETURN

PROC D_Check()

  Initialize()
  Get_Sums()
  ISum=0
  DO
    IF Done THEN
      EXIT
    FI
    Sum=0
    Lines=0
    FOR Count=0 TO 127 DO
      IF Count=0 THEN
        StartLine=CurLine
        StartChar=CurChar-6
      FI
      Check_Line()
      IF CurChar=Length^+7 THEN
        End_Of_Line()
        IF CurLine=0 THEN
          Check_Sum()
          IF Done THEN
            EXIT
          FI
          No_Problems()
          EXIT
        FI
        Lines==+1
      FI
    OD
    IF Done THEN
      EXIT
    FI
    Check_Sum()
  OD
  Close(1)

RETURN
```

B&C ComputerVisions

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

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

800/XL/XE SOFTWARE

ENTERTAINMENT

ACE OF ACES	13.50
ALIANTS	26.95
ALT. REALITY:	
THE CITY	26.95
THE DUNGEON	35.95
BEYOND CASTLE WOLF	17.95
BISMARK	26.95
CASTLE WOLFENSTEIN	17.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS	22.50
GUNSLINGER	26.95
KARATEKA	13.50
KICKSTART	8.95
LAST V-8	8.95
MOUSETRAP	17.95
NINJA	8.95
SARACEN	17.95
SPEEDKING	8.95
SPIDERMAN	7.95
SPITFIRE 40	31.50
STRIP POKER	26.95
THE HULK	7.95
TOMAHAWK (64K)	26.95
TOP GUNNER	22.50

SPORTS

COMPUTER BASEBALL	13.50
COMPUTER QUARTERBACK	13.50
LEADERBOARD	13.50
MICROLEAGUE BASEBALL	35.95
TRACK & FIELD	26.95

MUSIC

MUSIC COMPEMDIUM	35.95
MUSIC STUDIO	31.50
BANK ST. MUSIC WRITER	13.50
VIRTUOSO	44.95

PROGRAMMING

KYAN PASCAL	62.95
LIGHTSPEED C	35.95
LOGO	19.95
PILOT	19.95

PRODUCTIVITY

ATARIWRITER	30.00
CELEBRITY COOKBOOK	26.95
COMPUTE YOUR ROOTS	35.95
COMPUTER GOURMET	26.95
FAMILY FINANCE	12.95
HOME ACCOUNTANT	24.95
HOME FILING MANAGER	12.95
NEWSROOM (1050 - 64K)	44.95
NEWS STATION	26.95
NEWS STA. COMPANION	26.95
PAPERCLIP	29.95
POWER PRINT (1050)	13.50
PUBLISHING PRO	35.95
SYNCALC	31.50
TIMEWISE	12.95
VIDEO TITLESHOP (64K)	26.95
GRAPHICS COMPANION	17.95
VISICALC	24.95

ART

GRAPHICS MAGICIAN	22.50
BLAZING PADDLES	31.50

800/XL/XE CHILDRENS EDUCATIONAL SOFTWARE

CBS
(age 3-6 on disk)

ERNIES MAGIC SHAPES	8.95
ASTROGROVER	8.95
BIG BIRD SPEC. DELIVERY	8.95

FISHER PRIC
(pre-school on cartridge)

MEMORY MANOR	8.95
LINKING LOGIC	8.95
DANCE FANTASY	8.95
LOGIC LEVELS	8.95

SPINNAKER

ALF IN THE COLOR CAVES (age 3-6- cartridge)	9.95
ALPHABET ZOO (age 3-8 cartridge)	9.95
DELTA DRAWING (age 4-up cartridge)	9.95
FACE MAKER (age 3-8 cartridge)	9.95
HEY DIDDLE DIDDLE (age 3-10 disk)	9.95
KIDS ON KEYS (age 3-9 cartridge)	9.95
KINDERCOMP (age 3-8 cartridge)	9.95
STORY MACHINE (age 3-8 cartridge won't run on XL/XE)	9.95

TINK!TONK!
(age 4-8 on disk)

COUNT AND ADD	8.95
BEING A SMART THINKER	8.95
ABC's	8.95
SUBTRACTION	8.95
SPELLING	8.95
DEVELOP THINKING SKILLS	8.95

SAVE $13.75!!!
GET ALL 6 TINK! TONK!'S FOR
$39.95

WEEKLEY READER
(pre school - disk)

STICKY BEAR SHAPES	26.95
STICKY BEAR NUMBERS	26.95
STICKY BEAR ABC	26.95

HAYDEN
(age 4-10 disk)

MICRO ADD/SUBTRACT	9.95
MICRO SUBTRACTION	9.95
MICRO DIVISION	9.95

(pre-school disk)

SHAPE UP!	9.95
MATCH UP!	9.95

800/XL/XE CARTRIDGES

ALIEN AMBUSH	9.95
ARCHON	19.95
ASTEROIDS	15.95
ATARI TENNIS	9.95
BALL BLAZER	19.95
BARNYARD BLASTER (REQ. LIGHT GUN)	24.95
BATTLEZONE	19.95
CENTIPEDE	14.95
CLOUDBURST	9.95
DAVIDS MIDNIGHT MAGIC	19.95
DEFENDER	14.95
DIG DUG	19.95
DONKEY KONG	5.00
DONKEY KONG JR.	19.95
E.T. PHONE HOME	9.95
FIGHT NIGHT	19.95
FINAL LEGACY	19.95
FOOD FIGHT	19.95
FOOTBALL	14.95
FROGGER	14.95
FROGGER II	14.95
GALAXIAN	19.95
GATO	24.95
GYRUSS	14.95
HARDBALL	19.95
JOUST	19.95
JUNGLE HUNT	19.95
LODE RUNNER	24.95
MILLIPEDE	19.95
MISSILE COMMAND	5.00
MOON PATROL	19.95
MR. COOL	9.95
MS. PAC MAN	19.95
ONE ON ONE (XL/XE ONLY)	19.95
PAC MAN	5.00
PAST FINDER	24.95
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
SPACE INVADERS	14.95
STAR RAIDERS	5.00
STAR RAIDERS II	19.95
SUPER BREAKOUT	9.95
WIZARD OF WOR	5.00

ATARI XE GAME MACHINE

$139.95

Includes Missile Command, Flight Simulator II, Bug Hunt, light gun, joystick, BASIC programming language, and 64k of memory with a detachable keyboard. Add a disk drive and printer for a complete home computer system!

A GREAT CHRISTMAS GIFT!!

SUPER SPECIALS

RECONDITIONED ATARI MERCHANDISE
30 DAY WARRANTY

ATARI TRACKBALL **$9.95** SPICE UP THE ACTION IN YOUR ARCADE GAMES!!	**BOOKS ONLY** DE RE ATARI 10.00; ATARIWRITER 10.00; DOS 2.5 12.95; BASIC REF. 5.00; LOGO 10.00; BOOKKEEPER 10.00	**ATARI SPACE AGE JOYSTICK** **$5.00** GREAT STOCKING GIFTS!
400 (16K) COMPUTER **$29.95** 48K UPGRADE KIT $25.00	**600XL COMPUTER** **16K - $49.95** **64K - $59.95** INCLUDES BASIC	**800 (48K) COMPUTER** **$79.95** INCLUDES BASIC
1020 COLOR PRINTER/PLOTTER **$29.95** 40 COLUMNS WIDE INCLUDES PAPER AND COLOR PEN SET		**1030 MODEM WITH EXPRESS!** **$29.95** GET ONLINE TODAY!
ATARI NUMERIC KEYPAD **$7.95** INCLUDES HANDLER DISK -	**ATARI BOOKKEEPER** **$14.95 - NO BOX** (19.95 WITH KEYPAD) **$24.95 - IN BOX** (29.95 WITH KEYPAD)	**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. Calif. res. include 7% sales tax. Mastercard and Visa accepted if your telephone is listed in your local phone directory. Orders may be pre-paid with money order, cashier check, or personal check. Personal checks are held for three weeks before order is processed. C.O.D orders are shipped via UPS and must be paid with cash, cashier check or money order. International and APO orders must be pre-paid with cashier check or money order. $20.00 minimum on all orders. All sales are final - no refunds - prices are subject to change. Phone orders accepted TUESDAY THROUGH FRIDAY from 10:00 am to 6:00 pm PST.

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

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

CIRCLE #104 ON READER SERVICE CARD.

continued from page 13

Dungeon Lords

Listing 1.

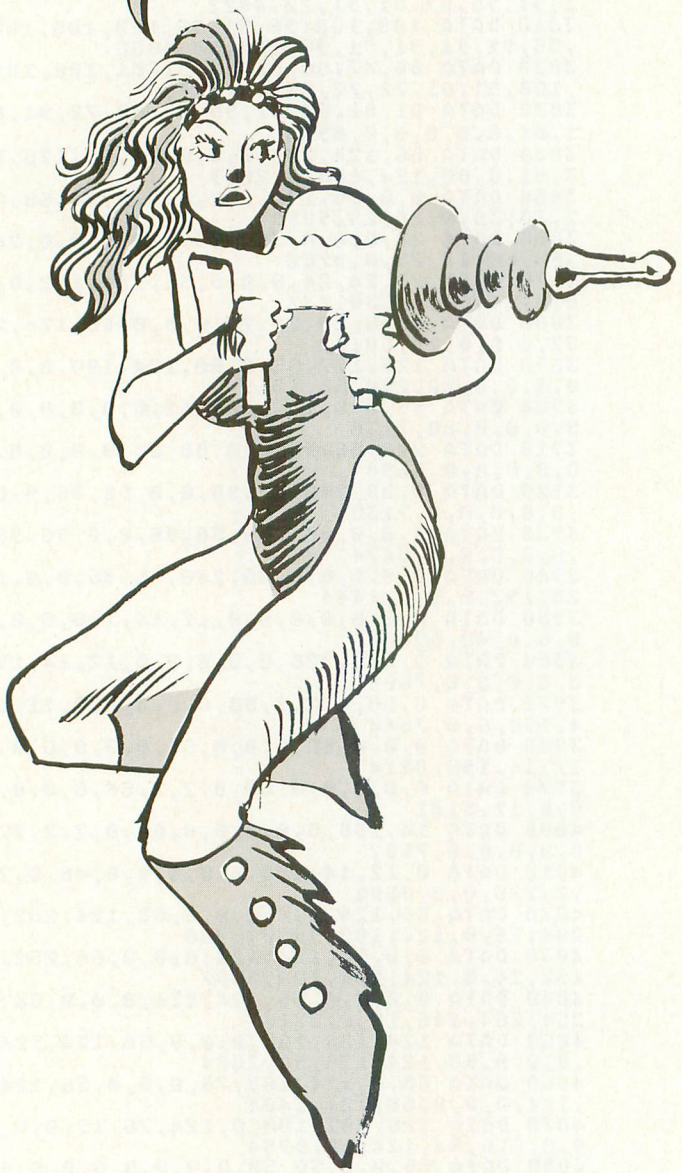
```
1000 DATA 255,255,71,34,0,57,112,112,1
12,112,112,71,102,34,70,122,3152
1010 DATA 34,112,112,71,162,34,70,142,
34,112,112,112,70,182,34,70,3618
1020 DATA 0,57,65,71,34,32,32,32,32,10
0,117,110,103,101,111,110,2528
1030 DATA 108,111,114,100,115,32,32,32
,32,32,32,98,121,32,98,114,800
1040 DATA 105,97,110,32,98,114,97,100,
108,101,121,32,32,32,112,114,2515
1050 DATA 101,115,115,32,115,116,97,11
4,116,32,116,111,32,112,108,97,3844
1060 DATA 121,32,32,32,32,32,32,231,22
5,237,229,32,239,246,229,242,5049
1070 DATA 32,32,32,32,32,32,32,32,32,3
2,32,32,32,243,227,239,4613
1080 DATA 242,229,32,32,32,32,32,32,32
,169,0,162,0,160,0,153,1446
1090 DATA 0,57,200,192,0,208,248,238,2
10,34,232,224,16,208,238,169,4324
1100 DATA 56,141,7,212,169,126,141,47,
2,169,3,141,29,208,141,15,4348
1110 DATA 210,169,0,141,200,2,160,0,15
3,128,0,200,192,128,208,248,787
1120 DATA 169,8,141,194,2,169,119,141,
0,208,141,1,208,141,2,208,7168
1130 DATA 32,171,39,169,212,141,192,2,
141,198,2,169,224,141,244,2,9440
1140 DATA 169,0,141,196,2,141,199,2,16
9,71,141,48,2,169,70,141,4837
1150 DATA 193,2,141,197,2,169,34,141,4
9,2,173,31,208,201,6,208,6620
1160 DATA 249,169,250,141,199,2,169,48
,141,244,2,169,180,141,48,2,6460
1170 DATA 169,47,141,49,2,169,1,133,17
7,32,172,46,76,28,35,32,921
1180 DATA 102,35,76,122,35,162,1,169,5
5,56,228,178,144,2,169,0,4708
1190 DATA 157,132,56,232,224,8,208,241
,96,162,0,169,0,133,174,157,8755
1200 DATA 32,6,189,64,6,201,0,240,14,1
81,184,133,143,181,216,133,540
1210 DATA 144,32,202,40,157,64,6,232,2
24,32,208,227,32,212,42,169,9317
1220 DATA 8,133,149,169,48,133,150,96,
32,96,35,165,177,56,233,1,5682
1230 DATA 74,74,133,169,230,169,10,10,
10,10,133,134,169,6,56,229,4887
1240 DATA 134,141,197,2,32,212,42,169,
0,133,175,133,139,133,136,173,8661
1250 DATA 10,210,41,7,133,140,173,10,2
10,41,7,24,105,8,133,141,3020
1260 DATA 162,0,134,143,165,136,133,14
4,32,41,38,169,0,145,128,232,6926
1270 DATA 224,64,208,238,230,136,169,2
1,197,136,208,228,162,0,169,0,9313
1280 DATA 133,144,134,143,32,247,37,16
9,20,133,144,32,247,37,232,232,327
1290 DATA 224,62,208,234,162,2,169,0,1
33,143,134,144,32,247,37,169,8165
1300 DATA 60,133,143,32,247,37,232,232
,224,20,208,234,169,56,133,143,1696
1310 DATA 162,2,134,144,32,247,37,232,
232,224,18,208,245,32,39,39,8075
1320 DATA 169,0,133,143,169,4,133,144,
32,108,39,169,2,133,143,133,5398
1330 DATA 144,32,247,37,169,4,133,143,
32,247,37,169,0,133,138,169,7200
1340 DATA 4,133,137,162,4,134,143,165,
137,133,144,169,0,133,138,32,6433
1350 DATA 41,38,177,128,201,0,240,3,76
,132,37,32,247,37,173,10,4498
1360 DATA 210,41,3,201,3,208,7,198,144
,198,144,76,70,37,201,2,5608
1370 DATA 208,7,230,144,230,144,76,70,
37,201,1,208,7,198,143,198,8990
1380 DATA 143,76,169,36,230,143,230,14
3,165,143,133,145,165,144,133,146,1698
1390 DATA 56,233,4,133,144,32,41,38,17
7,128,201,0,240,2,230,138,7849
1400 DATA 160,191,177,128,201,0,240,2,
230,138,160,194,177,128,201,0,332
1410 DATA 240,2,230,138,165,138,56,201
,2,144,3,76,132,37,165,145,5981
1420 DATA 133,143,165,146,133,144,32,4
1,38,177,128,201,0,240,3,76,5552
1430 DATA 132,37,165,139,230,139,197,1
40,208,5,169,175,76,24,37,197,8082
1440 DATA 141,208,5,169,175,76,24,37,1
73,10,210,41,63,56,197,177,6332
1450 DATA 144,3,76,64,37,169,167,32,30
,37,76,132,37,133,138,32,2693
1460 DATA 41,38,165,138,145,128,160,1,
230,138,165,138,145,128,160,96,9779
1470 DATA 230,138,165,138,145,128,160,
97,230,138,165,138,145,128,96,32,8962
1480 DATA 247,37,76,132,37,165,143,133
,145,198,143,198,143,165,144,133,1488
1490 DATA 146,198,144,198,144,32,41,38
,177,128,201,0,240,2,230,138,8649
1500 DATA 160,191,177,128,201,0,240,2,
230,138,160,128,230,129,177,128,2031
1510 DATA 201,0,240,2,230,138,165,138,
56,201,2,144,3,76,132,37,4615
1520 DATA 76,223,36,232,232,232,232,22
4,56,240,3,76,102,36,165,137,9392
1530 DATA 24,105,4,133,137,201,20,240,
3,76,100,36,169,58,133,143,5870
1540 DATA 169,6,133,144,169,179,32,30,
37,169,8,133,144,169,179,32,6221
1550 DATA 30,37,169,56,133,143,169,0,1
33,144,32,108,39,169,58,133,5247
1560 DATA 143,169,2,133,144,165,177,41
,3,201,0,208,5,169,232,76,7516
1570 DATA 213,37,169,244,32,30,37,165,
177,41,7,201,0,208,22,169,5697
1580 DATA 10,133,144,169,179,32,30,37,
169,54,133,143,169,2,133,144,6321
1590 DATA 169,175,32,30,37,96,32,41,38
,32,23,38,160,0,145,128,1312
1600 DATA 32,32,38,160,1,145,128,32,32
,38,160,96,145,128,32,23,2582
1610 DATA 38,160,97,145,128,96,173,10,
210,41,3,24,105,1,96,173,3554
1620 DATA 10,210,41,3,24,105,5,96,169,
0,133,134,133,135,133,129,6008
1630 DATA 165,144,133,128,160,0,165,12
9,10,133,129,165,128,10,144,3,4812
1640 DATA 230,129,24,133,128,200,192,5
,208,8,165,128,133,134,165,129,9403
1650 DATA 133,135,192,6,208,224,165,12
9,24,101,135,133,129,165,128,24,7822
1660 DATA 101,134,144,3,230,129,24,133
,128,165,129,24,105,64,133,129,6458
1670 DATA 165,128,24,101,143,144,2,230
,129,24,105,20,144,2,230,129,6210
1680 DATA 133,128,160,0,96,32,25,39,16
9,129,133,138,198,143,32,223,8262
1690 DATA 38,198,143,201,0,240,247,165
,143,133,139,32,25,39,169,1,5798
1700 DATA 133,138,230,143,32,223,38,23
0,143,201,0,240,247,165,143,133,2946
1710 DATA 140,32,25,39,169,224,133,138
,198,144,32,223,38,198,144,201,1261
1720 DATA 0,240,247,165,144,133,141,32
,25,39,169,96,133,138,230,144,9403
1730 DATA 32,223,38,230,144,201,0,240,
247,165,144,133,142,96,165,138,2014
1740 DATA 41,127,133,134,165,138,41,12
```

```
8,201,0,240,14,165,128,56,229,8992
1750 DATA 134,176,2,198,129,133,128,76
,6,39,165,128,24,101,134,144,5816
1760 DATA 2,230,129,133,128,160,0,177,
128,232,201,0,240,7,201,55,8953
1770 DATA 240,3,169,1,96,169,0,96,162,
0,165,147,133,143,165,148,8400
1780 DATA 133,144,32,41,38,96,173,10,2
10,41,3,24,105,3,10,10,8856
1790 DATA 133,143,24,105,2,133,157,173
,10,210,41,3,10,10,133,144,3238
1800 DATA 24,105,2,133,158,32,108,39,1
73,10,210,41,3,24,105,9,1175
1810 DATA 10,10,133,143,24,105,2,133,1
59,173,10,210,41,3,10,10,1315
1820 DATA 133,144,24,105,2,133,160,32,
108,39,96,169,17,32,30,37,1074
1830 DATA 230,144,230,144,169,9,32,30,
37,198,144,198,144,230,143,230,2167
1840 DATA 143,169,21,32,30,37,230,143,
230,143,169,25,32,30,37,230,6368
1850 DATA 144,230,144,169,13,32,30,37,
230,144,230,144,32,247,37,165,9162
1860 DATA 143,56,233,4,133,143,32,247,
37,96,160,197,162,39,169,6,7253
1870 DATA 32,92,228,162,0,169,157,157,
142,56,232,224,7,208,248,169,3292
1880 DATA 51,133,133,96,169,64,133,183
,165,149,74,74,41,254,133,182,277
1890 DATA 160,4,165,182,153,180,47,200
,165,183,153,180,47,165,182,24,248
1900 DATA 105,96,144,2,230,183,133,182
,200,200,192,70,208,228,165,149,4779
1910 DATA 41,7,73,7,141,4,212,76,95,22
8,32,254,39,133,136,41,6633
1920 DATA 12,201,0,208,3,76,24,40,201,
8,208,2,230,143,201,4,6397
1930 DATA 208,21,198,143,76,40,40,165,
136,41,3,201,2,208,2,230,6293
1940 DATA 144,201,1,208,2,198,144,165,
136,96,133,136,41,12,201,0,5842
1950 DATA 240,2,73,12,133,134,165,136,
41,3,201,0,240,2,73,3,3074
1960 DATA 24,101,134,96,169,8,133,171,
96,32,69,40,162,168,134,170,7775
1970 DATA 162,192,134,132,162,4,32,222
,40,169,171,32,30,37,169,171,6774
1980 DATA 96,165,175,201,1,208,1,96,32
,69,40,162,168,134,170,162,8327
1990 DATA 208,134,132,169,1,133,175,32
,202,40,169,0,96,165,175,201,9298
2000 DATA 0,240,21,32,69,40,162,168,13
4,170,162,200,134,132,169,0,8947
2010 DATA 133,175,32,202,40,169,0,96,1
69,171,96,32,69,40,162,168,6625
2020 DATA 134,170,162,224,134,132,32,2
02,40,162,3,32,222,40,169,0,5556
2030 DATA 133,143,169,2,133,144,32,108
,39,169,2,133,143,169,4,133,5653
2040 DATA 144,169,244,32,30,37,169,0,9
6,32,41,38,169,0,145,128,3448
2050 DATA 160,1,145,128,160,96,145,128
,160,97,145,128,96,169,1,133,7872
2060 DATA 134,165,134,201,1,208,45,224
,2,208,20,230,178,134,134,32,8880
2070 DATA 102,35,32,69,40,169,224,133,
132,169,168,133,170,166,134,169,2030
2080 DATA 0,133,134,189,142,56,24,105,
1,201,167,208,4,169,157,230,363
2090 DATA 134,157,142,56,202,224,255,2
08,200,96,169,146,133,134,165,179,4306
2100 DATA 56,201,128,144,4,169,68,133,
134,165,20,41,7,24,101,134,3746
2110 DATA 141,196,2,173,0,211,41,12,73
,12,201,0,240,5,133,153,5611
2120 DATA 76,77,41,173,0,211,41,3,73,3
,133,153,165,149,74,74,5253
2130 DATA 41,254,133,147,165,149,41,7,
201,0,240,3,76,17,42,165,4709
2140 DATA 150,74,74,74,41,254,133,148,
165,150,41,15,201,0,240,3,6677
2150 DATA 76,17,42,165,155,201,0,240,5
5,165,153,201,0,240,49,165,9922
2160 DATA 147,133,143,165,148,133,144,
165,153,32,251,39,32,41,38,177,6846
2170 DATA 128,201,167,208,3,32,74,40,2
01,175,208,3,32,98,40,201,6565
2180 DATA 0,240,9,56,201,52,144,8,201,
171,240,4,165,153,133,156,285
2190 DATA 165,147,133,143,165,148,133,
144,165,156,32,251,39,32,41,38,6003
2200 DATA 177,128,201,167,208,3,32,74,
40,201,175,208,3,32,98,40,5166
2210 DATA 201,179,208,3,32,126,40,201,
244,208,2,230,174,201,232,208,5151
2220 DATA 3,32,156,40,201,0,240,16,56,
201,52,144,7,201,171,240,9852
2230 DATA 3,76,249,41,169,0,133,156,16
5,149,133,143,165,150,133,144,1018
2240 DATA 165,156,32,254,39,165,143,13
3,149,165,144,133,150,76,48,42,7656
2250 DATA 165,155,201,0,240,226,165,15
3,201,0,208,3,76,249,41,165,125
2260 DATA 156,32,43,40,197,153,208,4,1
65,153,133,156,76,249,41,165,239
2270 DATA 156,201,0,240,37,165,176,41,
128,201,0,240,16,165,176,24,8107
2280 DATA 105,18,201,218,208,2,169,72,
133,176,76,91,42,165,176,56,7984
2290 DATA 233,18,201,238,208,2,169,128
,133,176,169,32,133,128,169,60,9589
2300 DATA 133,129,165,156,201,8,208,8,
169,96,133,130,169,55,133,131,8945
2310 DATA 201,4,208,8,169,0,133,130,16
9,52,133,131,201,2,208,8,6956
2320 DATA 169,32,133,130,169,53,133,13
1,201,1,208,8,169,64,133,130,7985
2330 DATA 169,54,133,131,201,0,208,3,7
6,129,42,165,176,41,127,24,5582
2340 DATA 101,130,144,2,230,131,133,13
0,165,128,24,101,150,144,2,230,8965
2350 DATA 129,133,128,162,0,160,0,177,
130,145,128,200,192,18,208,247,2401
2360 DATA 165,130,24,105,90,144,2,230,
131,133,130,230,129,232,224,3,1477
2370 DATA 208,227,96,169,32,133,128,16
9,60,133,129,162,0,142,30,208,8201
2380 DATA 160,0,169,0,145,128,200,192,
170,208,249,230,129,232,224,3,4918
2390 DATA 208,238,96,169,0,133,137,181
,184,56,197,151,144,12,56,229,9982
2400 DATA 151,133,134,169,4,133,164,76
,22,43,165,151,56,245,184,133,9770
2410 DATA 134,169,8,133,164,181,216,56
,197,152,144,15,56,229,152,133,703
2420 DATA 135,165,164,24,105,1,133,164
,76,58,43,165,152,56,245,216,9855
2430 DATA 133,135,165,164,24,105,2,133
,164,165,135,56,197,134,176,35,8732
2440 DATA 169,12,133,138,32,211,43,201
,1,208,3,76,138,43,169,3,4932
2450 DATA 133,138,32,211,43,201,1,208,
3,76,138,43,165,164,32,43,5321
2460 DATA 40,133,164,169,3,133,138,32,
211,43,201,1,208,3,76,138,6615
2470 DATA 43,169,12,133,138,32,211,43,
201,1,208,3,76,138,43,165,6470
2480 DATA 164,32,43,40,133,164,76,65,4
3,181,184,133,143,181,216,133,1276
2490 DATA 144,189,0,6,32,254,39,165,14
3,149,184,165,144,149,216,96,1828
2500 DATA 169,0,133,137,173,10,210,41,
3,201,0,208,2,160,5,201,6429
2510 DATA 1,208,2,160,9,201,2,208,2,16
0,6,201,3,208,2,160,6139
2520 DATA 10,132,164,173,10,210,41,1,2
01,0,240,3,76,65,43,76,3827
2530 DATA 100,43,169,0,133,165,230,137
,169,4,56,197,137,144,16,189,9186
2540 DATA 0,6,32,43,40,133,136,165,164
,37,138,197,136,240,55,181,667
2550 DATA 184,133,143,181,216,133,144,
165,164,37,138,32,251,39,32,41,7052
2560 DATA 38,177,128,201,0,240,20,56,2
01,166,144,5,56,201,193,144,22
2570 DATA 21,56,201,55,144,16,56,201,2
31,176,11,169,1,133,165,165,9320
2580 DATA 164,37,138,157,0,6,165,165,9
```


```
6,169,0,133,166,166,167,189,532
2590 DATA 64,6,201,0,208,65,173,10,210
,56,197,177,144,3,76,56,6681
2600 DATA 45,165,179,56,201,128,176,3,
76,56,45,173,10,210,41,1,4281
2610 DATA 201,1,240,24,165,157,149,184
,165,158,149,216,165,177,45,10,665
2620 DATA 210,41,3,24,105,1,157,64,6,7
6,120,44,165,159,149,184,7371
2630 DATA 165,160,149,216,76,93,44,181
,216,133,144,181,184,133,143,32,1061
2640 DATA 202,40,181,184,41,1,201,0,24
0,3,76,144,45,181,216,41,7588
2650 DATA 1,201,0,240,3,76,144,45,181,
184,197,147,208,57,181,216,2925
2660 DATA 56,197,141,144,50,56,197,142
,176,45,165,148,56,213,216,144,2089
2670 DATA 10,189,0,6,201,1,240,31,76,1
95,44,189,0,6,201,2,4538
2680 DATA 240,21,169,1,133,172,133,173
,160,0,169,1,153,32,6,200,6523
2690 DATA 192,32,208,248,76,5,45,181,2
16,197,148,208,39,181,184,56,1470
2700 DATA 197,139,144,32,56,197,140,17
6,27,165,147,56,213,184,144,10,9432
2710 DATA 189,0,6,201,4,240,13,76,195,
44,189,0,6,201,8,240,7006
2720 DATA 3,76,195,44,181,184,197,151,
208,11,181,216,197,152,208,5,2686
2730 DATA 169,0,157,32,6,165,172,201,0
,240,12,165,173,133,172,165,1173
2740 DATA 147,133,151,165,148,133,152,
189,32,6,201,1,208,6,32,244,8123
2750 DATA 42,76,56,45,32,161,43,169,0,
133,135,189,64,6,201,0,5085
2760 DATA 240,54,201,1,208,5,169,56,76
,87,45,201,2,208,5,169,6594
2770 DATA 200,76,87,45,169,216,133,134
,189,0,6,41,8,201,8,208,6332
2780 DATA 7,165,134,24,105,4,133,134,1
81,184,133,143,181,216,133,144,2491
2790 DATA 165,134,24,101,135,32,30,37,
232,224,32,208,4,162,0,134,6712
2800 DATA 173,230,166,169,8,197,166,24
0,3,76,48,44,134,167,96,32,6786
2810 DATA 138,43,169,8,133,135,76,60,4
5,173,0,211,74,74,74,74,5019
2820 DATA 133,134,41,12,73,12,201,0,24
0,5,133,154,76,211,45,165,8014
2830 DATA 134,41,3,73,3,201,0,240,5,13
3,154,76,211,45,165,153,8765
2840 DATA 133,154,173,16,208,133,155,2
01,0,240,7,165,163,201,0,208,1113
2850 DATA 40,96,165,163,201,0,240,3,76
,250,45,165,154,201,0,208,741
2860 DATA 1,96,165,147,133,161,165,148
,133,162,165,154,133,163,32,69,181
2870 DATA 40,169,192,133,132,169,136,1
33,170,165,161,133,143,165,162,133,332
0
2880 DATA 144,32,41.38,169,0,145,128,1
65,163,32,254,39,32,41,38,4940
2890 DATA 177,128,201,0,240,7,169,0,13
3,163,76,42,46,169,55,145,6627
2900 DATA 128,165,143,133,161,165,144,
133,162,160,0,165,143,133,145,165,1760
2910 DATA 144,133,146,165,145,133,143,
165,146,133,144,185,184,0,133,134,1241
2920 DATA 165,143,197,134,240,7,56,233
,1,197,134,208,63,185,216,0,594
2930 DATA 133,134,165,144,197,134,240,
7,56,233,1,197,134,208,45,185,1425
2940 DATA 64,6,201,0,240,38,169,0,153,
64,6,32,69,40,169,136,4865
2950 DATA 133,170,169,216,133,132,162,
6,32,222,40,185,184,0,133,143,9276
2960 DATA 185,216,0,133,144,132,138,32
,202,40,164,138,200,192,32,208,1617
2970 DATA 162,96,165,168,24,101,169,13
3,168,56,201,11,176,1,96,56,6517
2980 DATA 233,12,133,168,32,134,38,32,
42,44,96,32,180,39,32,169,4122
2990 DATA 35,169,157,141,152,56,141,15
3,56,169,3,133,178,162,11,32,6787
3000 DATA 222,40,32,96,35,169,0,141,30
,208,173,5,208,41,6,201,6996
3010 DATA 0,240,81,198,178,169,138,141
,1,210,133,20,165,20,141,0,6875
3020 DATA 210,201,255,208,247,169,0,14
1,1,210,197,178,208,51,162,0,866
3030 DATA 189,142,56,56,233,13,157,7,5
7,232,224,7,208,242,32,212,1638
3040 DATA 42,169,224,141,244,2,169,71,
141,48,2,169,34,141,49,2,4621
3050 DATA 169,0,141,196,2,133,19,141,1
97,2,165,19,201,3,208,250,106
3060 DATA 96,32,96,35,32,27,41,230,179
,169,0,133,77,141,30,208,7747
3070 DATA 165,20,56,201,8,144,26,169,0
,133,20,164,171,192,0,240,8954
3080 DATA 16,136,132,171,177,132,141,0
,210,166,170,202,134,170,142,1,1224
3090 DATA 210,32,154,45,32,147,46,32,1
47,46,162,32,160,32,136,196,7279
3100 DATA 169,208,251,202,228,169,208,
244,165,174,201,0,240,66,230,177,6570
3110 DATA 162,11,32,222,40,169,0,141,4
7,2,32,169,35,165,177,41,5519
3120 DATA 3,201,1,208,33,169,170,141,1
,210,169,232,133,132,160,0,586
3130 DATA 132,171,169,0,133,20,177,132
,141,0,210,165,20,201,10,208,9302
3140 DATA 250,200,192,21,208,236,169,1
26,141,47,2,162,3,32,222,40,7259
3150 DATA 76,203,46,112,112,112,84,252
,59,84,92,60,84,188,60,84,7125
3160 DATA 28,61,84,124,61,84,220,61,84
,60,62,84,156,62,84,252,8129
3170 DATA 62,84,92,63,84,188,63,84,28,
64,84,124,64,84,220,64,6225
3180 DATA 84,60,65,84,156,65,84,252,65
,84,92,66,84,188,66,84,6976
3190 DATA 28,67,84,124,67,84,220,67,68
,126,56,4,65,180,47,0,3621
3200 DATA 0,0,0,0,0,0,0,59,170,138,138
,162,170,43,0,63,3864
3210 DATA 251,170,170,168,170,47,0,42,
170,138,130,170,170,42,0,47,6381
3220 DATA 171,170,234,234,251,63,0,240
,252,188,172,188,252,240,0,240,7694
3230 DATA 252,188,140,60,252,240,0,160
,168,168,168,168,188,240,0,240,5198
3240 DATA 188,172,168,168,168,160,0,47
,170,170,170,170,170,47,0,21,7468
3250 DATA 21,21,0,21,21,21,0,240,188,1
72,168,168,168,160,0,21,7584
3260 DATA 21,21,0,21,21,21,0,84,84,84,
0,84,84,84,0,240,3022
3270 DATA 252,188,172,188,252,240,0,84
,84,84,0,84,84,84,0,63,4418
3280 DATA 250,170,170,170,171,47,0,59,
170,170,170,170,250,63,0,240,1781
3290 DATA 252,188,172,188,252,240,0,24
0,188,172,172,168,168,160,0,42,2222
3300 DATA 170,170,192,197,197,5,0,42,1
70,162,138,170,171,47,0,240,9954
3310 DATA 188,172,188,252,252,240,0,17
6,172,168,0,69,69,69,0,47,6193
3320 DATA 170,170,0,69,69,69,0,42,170,
```

```
170,234,250,254,63,0,160,749
3330 DATA 168,168,172,188,252,240,0,24
0,188,172,8,80,80,80,0,42,7014
3340 DATA 170,170,234,250,254,63,0,63,
51,51,51,51,51,63,0,12,1583
3350 DATA 12,12,12,12,12,12,0,63,51,3,
63,48,51,63,0,63,8417
3360 DATA 51,3,15,3,51,63,0,51,51,51,6
3,3,3,3,0,63,7302
3370 DATA 51,48,63,3,51,63,0,63,51,48,
63,51,51,63,0,63,9652
3380 DATA 51,3,15,12,12,12,0,63,51,51,
63,51,51,63,0,63,8993
3390 DATA 51,51,63,3,51,63,0,63,255,25
3,213,221,221,221,213,240,7703
3400 DATA 252,252,92,220,252,252,92,25
3,253,221,213,253,255,63,0,220,8335
3410 DATA 220,220,92,252,252,240,0,63,
255,252,192,192,192,192,192,240,9693
3420 DATA 252,252,12,204,252,252,12,25
2,252,204,192,252,255,63,0,12,3733
3430 DATA 12,12,12,252,252,240,0,0,0,3
,15,12,15,3,0,0,7786
3440 DATA 0,240,60,12,60,240,192,3,0,0
,3,0,0,0,0,192,361
3450 DATA 192,192,192,0,0,0,0,0,42,255
,240,195,138,195,240,0,634
3460 DATA 160,252,60,12,136,12,60,252,
240,252,252,240,42,0,0,252,2450
3470 DATA 252,252,252,252,160,0,0,0,0,
16,68,84,68,16,0,3,9862
3480 DATA 11,63,255,136,0,63,0,240,172
,47,255,63,255,255,252,15,3708
3490 DATA 63,252,240,240,252,63,15,240
,0,60,195,3,15,252,240,15,9744
3500 DATA 58,248,255,252,255,255,63,19
2,224,252,255,34,0,252,0,15,2126
3510 DATA 0,60,195,192,240,63,15,240,2
52,63,15,15,63,252,240,3,9824
3520 DATA 11,63,255,255,3,0,3,240,172,
47,255,255,255,255,252,15,6186
3530 DATA 63,252,240,240,252,63,15,240
,0,12,3,3,15,252,240,15,7192
3540 DATA 58,248,255,255,255,255,63,19
2,224,252,255,255,192,0,192,15,6678
3550 DATA 0,48,192,192,240,63,15,240,2
52,63,15,15,63,252,240,3,9831
3560 DATA 15,43,170,128,10,0,3,240,172
,47,175,40,172,60,240,15,8013
3570 DATA 62,254,255,63,2,8,160,192,24
3,191,175,252,32,8,40,15,8285
3580 DATA 58,248,250,40,58,60,15,192,2
40,232,170,2,160,0,192,3,8717
3590 DATA 207,254,250,63,8,32,40,240,1
88,191,255,252,128,32,10,3,9480
3600 DATA 15,43,170,138,0,0,3,240,172,
47,175,168,60,60,240,15,8123
3610 DATA 62,254,255,63,0,0,10,192,243
,191,175,252,128,128,128,15,1465
3620 DATA 58,248,250,42,60,60,15,192,2
40,232,170,162,0,0,192,3,8615
3630 DATA 207,254,250,63,2,2,2,240,188
,191,255,252,0,0,160,192,2206
3640 DATA 192,243,255,48,2,10,10,3,3,2
07,255,252,60,143,243,10,9952
3650 DATA 63,62,2,3,3,15,60,240,252,25
4,191,252,255,15,3,192,2875
3660 DATA 192,243,255,63,60,242,207,3,
3,207,255,12,128,160,160,15,170
3670 DATA 63,191,254,63,255,240,192,16
0,252,188,128,192,192,240,60,192,8156
3680 DATA 192,243,255,48,32,32,10,3,3,
207,255,252,60,143,243,10,274
3690 DATA 59,59,2,3,0,0,3,240,252,238,
175,188,240,240,240,192,7807
3700 DATA 192,243,255,63,60,242,207,3,
3,207,255,12,8,8,160,15,6522
3710 DATA 63,187,250,62,15,15,15,160,2
36,236,128,192,0,0,192,170,491
3720 DATA 133,149,140,191,171,143,129,
170,86,86,234,234,250,82,86,133,4143
3730 DATA 170,137,137,137,137,137,170,
86,170,102,102,98,98,98,170,51,9378
3740 DATA 51,51,51,51,51,63,0,63,48,48
,60,48,48,63,0,60,1
3750 DATA 51,51,60,51,51,60,0,12,12,51
,51,63,51,51,0,51,9126
3760 DATA 63,63,63,63,51,51,0,48,48,48
,48,48,48,63,0,60,9817
3770 DATA 51,51,51,51,51,60,0,63,51,51
,60,51,51,51,0,0,9017
3780 DATA 5,4,4,5,0,1,1,0,64,64,64,64,
0,16,16,1,7006
3790 DATA 1,0,0,0,0,0,0,16,80,0,84,16,
16,16,0,29,6651
3800 DATA 35,45,53,64,64,60,53,72,72,8
1,91,96,81,91,91,72,4422
3810 DATA 108,108,96,91,96,108,108,108
,96,91,91,91,91,96,108,60,6508
3820 DATA 60,47,60,81,60,81,81,108,108
,108,81,81,72,72,53,64,4373
3830 DATA 81,81,64,81,96,60,60,72,91,8
1,81,0,0,0,0,0,8945
3840 DATA 56,124,252,0,0,0,24,60,126,6
3,51,0,60,124,108,0,2009
3850 DATA 0,0,56,124,252,0,0,0,24,60,6
2,38,38,0,28,28,9090
3860 DATA 28,0,0,0,56,124,252,0,0,0,24
,60,60,28,28,0,9252
3870 DATA 28,24,24,0,0,0,56,124,252,0,
0,0,24,60,62,30,232
3880 DATA 30,0,28,28,28,0,0,0,56,124,2
52,0,0,0,24,60,82
3890 DATA 126,127,63,0,60,124,100,0,0,
0,0,0,0,80,240,56,1819
3900 DATA 96,0,0,192,204,12,0,0,0,0,0,
0,0,0,0,80,7136
3910 DATA 240,56,96,0,0,88,88,0,0,0,0,
0,0,0,0,0,5694
3920 DATA 0,80,240,56,96,0,0,96,96,0,0
,0,0,0,0,0,7136
3930 DATA 0,0,0,80,240,56,96,0,0,96,96
,0,0,0,0,0,8474
3940 DATA 0,0,0,0,0,0,80,240,56,96,0,0,1
28,192,0,0,0,1444
3950 DATA 0,0,0,0,0,0,0,0,12,14,198,6,0,
0,0,0,48,6986
3960 DATA 0,2,2,226,0,0,0,0,0,12,14,19
8,6,0,0,0,7602
3970 DATA 0,60,0,2,2,58,0,0,0,0,0,0,12,1
4,198,6,0,7644
3980 DATA 0,0,0,60,0,0,0,56,0,0,0,0,0,0,
12,14,198,8214
3990 DATA 6,0,0,0,0,60,0,2,2,58,0,0,0,
0,0,12,5162
4000 DATA 14,198,6,0,0,0,0,60,0,2,2,22
6,0,0,0,0,7662
4010 DATA 0,12,14,198,6,0,0,0,0,48,0,2
,2,226,0,0,8592
4020 DATA 56,124,124,0,0,0,68,124,252,
204,76,0,124,100,96,0,5760
4030 DATA 0,0,56,124,124,0,0,0,68,252,
152,24,0,124,108,100,5362
4040 DATA 0,0,0,0,56,124,124,0,0,0,68,
254,254,146,16,0,5314
4050 DATA 124,108,108,0,0,0,56,124,124
,0,0,0,68,124,126,50,2524
4060 DATA 48,0,124,108,76,0,0,0,56,124
,124,0,0,0,68,124,1404
4070 DATA 126,102,100,0,124,76,12,0,0,
0,0,0,0,84,124,68,9984
4080 DATA 56,0,0,50,50,0,0,0,0,0,0,0,0
,0,0,84,5930
4090 DATA 124,68,56,0,102,102,0,0,0,0,
0,0,0,0,0,0,5640
4100 DATA 0,84,124,68,56,0,0,108,108,0
,0,0,0,0,0,0,7028
4110 DATA 0,0,0,84,124,68,56,0,0,204,2
04,0,0,0,0,0,150
4120 DATA 0,0,0,0,0,84,124,68,56,0,0,1
52,152,0,0,0,340
4130 DATA 0,0,0,0,0,0,0,0,0,0,56,0,0,0,0
,0,124,6674
4140 DATA 0,8,12,96,0,0,0,0,0,0,0,0,0,
0,0,0,4576
4150 DATA 124,0,0,8,108,0,0,0,0,0,0,0,
0,0,0,0,4846
4160 DATA 0,0,0,124,0,0,0,108,0,0,0,0,
```

Dungeon Lords



```
0,0,0,0,5520
4170 DATA 0,0,0,0,0,0,124,0,0,32,108,0,0
,0,0,0,0,6282
4180 DATA 0,0,0,0,0,0,0,0,124,0,32,96,12
,0,0,0,0,6692
4190 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4190
4200 DATA 56,124,124,0,0,0,68,124,252,
252,124,0,124,96,96,0,6892
4210 DATA 0,0,56,124,124,0,0,0,68,124,
252,252,124,0,124,108,8342
4220 DATA 96,0,0,0,56,124,124,0,0,0,68
,124,254,254,124,0,7162
4230 DATA 124,108,108,0,0,0,56,124,124
,0,0,0,68,124,126,126,3920
4240 DATA 124,0,124,108,12,0,0,0,56,12
4,124,0,0,0,68,124,1340
4250 DATA 126,126,124,0,124,12,12,0,0,
0,0,0,0,0,0,0,5776
4260 DATA 0,0,0,2,2,0,0,0,0,0,0,0,0,0,
0,0,4278
4270 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4270
4280 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4280
4290 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4290
4300 DATA 0,0,0,0,0,0,0,0,0,0,0,0,128,12
8,0,0,0,7500
4310 DATA 0,0,0,0,0,0,0,0,124,124,124,56
,0,0,0,0,124,258
4320 DATA 0,12,12,96,0,0,0,0,0,0,124,124
,124,56,0,0,0,9584
4330 DATA 0,124,0,0,12,108,0,0,0,0,0,1
24,124,124,56,0,962
4340 DATA 0,0,0,124,0,0,0,108,0,0,0,0,
0,124,124,124,1280
4350 DATA 56,0,0,0,0,124,0,0,96,108,0,
0,0,0,0,124,9078
4360 DATA 124,124,56,0,0,0,0,124,0,96,
96,12,0,0,0,0,8052
4370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4370
4380 DATA 28,62,63,0,0,0,24,60,126,254
,252,0,60,62,38,0,4033
4390 DATA 0,0,28,62,63,0,0,0,24,60,124
,120,120,0,56,56,1953
4400 DATA 56,0,0,0,28,62,63,0,0,0,24,6
0,60,56,56,0,8797
4410 DATA 56,24,24,0,0,0,28,62,63,0,0,
0,24,60,124,100,457
4420 DATA 100,0,56,56,56,0,0,0,28,62,6
3,0,0,0,24,60,8077
4430 DATA 126,252,204,0,60,62,54,0,0,0
,0,0,0,10,15,28,7535
4440 DATA 6,0,0,1,3,0,0,0,0,0,0,0,0,0,
0,10,4625
4450 DATA 15,28,6,0,0,6,6,0,0,0,0,0,0,
0,0,0,4617
4460 DATA 0,10,15,28,6,0,0,6,6,0,0,0,0
,0,0,0,4769
4470 DATA 0,0,0,10,15,28,6,0,0,26,26,0
,0,0,0,0,5341
4480 DATA 0,0,0,0,0,10,15,28,6,0,0,3,5
1,48,0,0,6294
4490 DATA 0,0,0,0,0,0,0,48,112,99,96,0
,0,0,0,60,8888
4500 DATA 0,64,64,71,0,0,0,0,0,48,112,
99,96,0,0,0,9252
4510 DATA 0,60,0,64,64,92,0,0,0,0,0,48
,112,99,96,0,616
4520 DATA 0,0,0,60,0,0,0,28,0,0,0,0,0,
48,112,99,8920
4530 DATA 96,0,0,0,0,60,0,64,64,92,0,0
,0,0,0,48,7762
4540 DATA 112,99,96,0,0,0,0,12,0,64,64
,71,0,0,0,0,7430
4550 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4550
4560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4560
4570 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4570
4580 DATA 0,0,0,0,0,0,0,0,0,0,242,236,24
0,163,237,157,240,4108
4590 DATA 241,157,243,242,162,128,128,
238,161,128,238,243,158,239,240,128,53
1
4600 DATA 238,243,239,242,241,237,161,
0,0,0,0,0,0,0,0,0,763
4610 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4610
4620 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,224,
0,0,0,7532
4630 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4630
4640 DATA 226,2,227,2,202,34,0,0,0,0,0
,0,0,0,0,0,6773
```

Attention Programmers!

ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing family of Atari computer owners.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. Typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG Computing**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number.

Send your programs and articles to:
ANALOG Computing
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

BASIC Editor II

by Clayton Walnum

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

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

Typing in the Editor

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

Disk version:

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

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

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

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

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

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

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

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

Cassette version:

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

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

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

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

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

(6) Rewind the cassette.

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

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

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

Using the Editor

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

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

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

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

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

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

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

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

Leaving the Editor

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

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

Large listings

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

The post-processor

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

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

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

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

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

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

Murphy's Law

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

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

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

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

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

Listing 1.
BASIC listing.

```
32600 IF FL THEN 32616
32602 DIM L$(115),5V$(115),C2$(2),B$(1
15),M$(119),S$(98),E$(69),A$(1):FL=1:S
TMTAB=PEEK(136)+PEEK(137)*256
32604 GRAPHICS 0:POKE 710,0:P=0:ABR=0:
? "ALLOW ABBREVIATIONS";:INPUT A$:IF A
$="Y" OR A$="y" THEN ABR=1
32606 B$(1)=" ":B$(115)=" ":B$(2)=B$
32616 OPEN #17,4,0,"E:":L$=" ":GOSUB 3
2662:START=0
32618 POKE 766,1:POKE 83,39:POSITION 1
,3:IF LEN(L$)<39 THEN ? L$:GOTO 32624
32620 IF LEN(L$)<77 THEN ? L$(1,38):?
L$(39,LEN(L$)):GOTO 32624
32622 ? L$(1,38):? L$(39,76):? L$(77,L
EN(L$))
32624 POKE 752,0:POKE 766,0:POKE 559,3
4:POKE 82,1:POKE 83,38:POSITION 0,10:?
 " ";:INPUT #17;L$:POKE 766,1
32626 IF (L$="P" OR L$="p") AND START=
0 THEN P=1:L$=""
32628 IF L$="E" OR L$="e" THEN E=1:POS
ITION 1,10:? 5V$ GOTO 32624
32630 IF L$="Q" OR L$="q" THEN 32690
32632 IF L$="" AND P=1 THEN 32686
32634 IF L$="" THEN 32624
32636 IF L$="B" OR L$="b" THEN GRAPHIC
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$(2)):POSITI
ON 1,9:LIST EL:GOTO 32624
32640 5V$=L$:TRAP 32624:X=VAL(L$)
32642 START=1:IF P AND  NOT E THEN 326
52
32644 GOSUB 32674:IF  NOT ABR OR P THE
N 32652
32646 POKE 766,0:? CHR$(125):POSITION
0,3:L=VAL(L$):LIST L:? :? :? "CONT":L$
=B$
32648 POSITION 0,0:POKE 842,13:STOP
32650 POKE 842,12:A=USR(ADR(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:? "R";:POKE 82,1:DL=PEEK(560)+256*PEE
K(561)+4
32664 POKE DL-1,70:POKE DL+2,6:POKE DL
+3,112:POKE DL+4,112:POKE DL+5,112:POK
E DL+13,112:POKE DL+14,112
32666 POKE DL+22,112:POKE DL+23,112:PO
KE DL+24,65:POKE DL+25,PEEK(560):POKE
DL+26,PEEK(561):POKE 83,39
32668 POSITION 20,0:? "  basic editor
II  ":POSITION 0,7:? "                  TYP
ING WINDOW                ":POSITION 0,1:? "                  MESS
AGE WINDOW                ":POSITION 1,7
:? "CODE:";
32672 POKE 559,34:RETURN
32674 GRAPHICS 0:POKE 559,0:POKE 766,1
:POKE 82,0:POKE 83,39:POSITION 0,3:? L
$:? :? :? :? "CONT":POSITION 0,0
32676 POKE 842,13:STOP
32678 POKE 842,12:TRAP 32682:A=USR(ADR
(E$),VAL(L$)):IF A=4 THEN POP :GOTO 32
682
32680 RETURN
32682 GOSUB 32662:SOUND 0,75,10,8:FOR
X=1 TO 20:NEXT X:SOUND 0,0,0,0:POSITIO
N 1,3:? "SYNTAX ERROR!":POKE 766,1
32684 POKE 83,38:POSITION 1,10:? 5V$:G
OTO 32624
32686 LINE=PEEK(STMTAB)+PEEK(STMTAB+1)
*256:IF LINE>32599 THEN 32690
32688 OFS=PEEK(STMTAB+2):STMTAB=STMTAB
+OF5:POSITION 1,9:LIST LINE:GOTO 32624
32690 POKE 766,0:POSITION 1,10:? "READ
Y TO QUIT";:INPUT A$:IF A$<>"Y" THEN P
OSITION 1,10:? B$(1,38):GOTO 32624
32692 GRAPHICS 0:? :? :? :FOR X=32600
TO 32636 STEP 2:? X:NEXT X:? "CONT":PO
SITION 0,0:POKE 842,13:STOP
32694 POKE 842,12:GRAPHICS 0:? :? :? :
FOR X=32638 TO 32674 STEP 2:? X:NEXT X
:? :? "CONT":POSITION 0,0
32696 POKE 842,13:STOP
32698 POKE 842,12:GRAPHICS 0:? :? :? :
FOR X=32676 TO 32702 STEP 2:? X:NEXT X
:? :? "POKE 842,12":POSITION 0,0
32700 POKE 842,13:STOP
32702 POKE 16,112:POKE 53774,112:RETUR
N
```

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

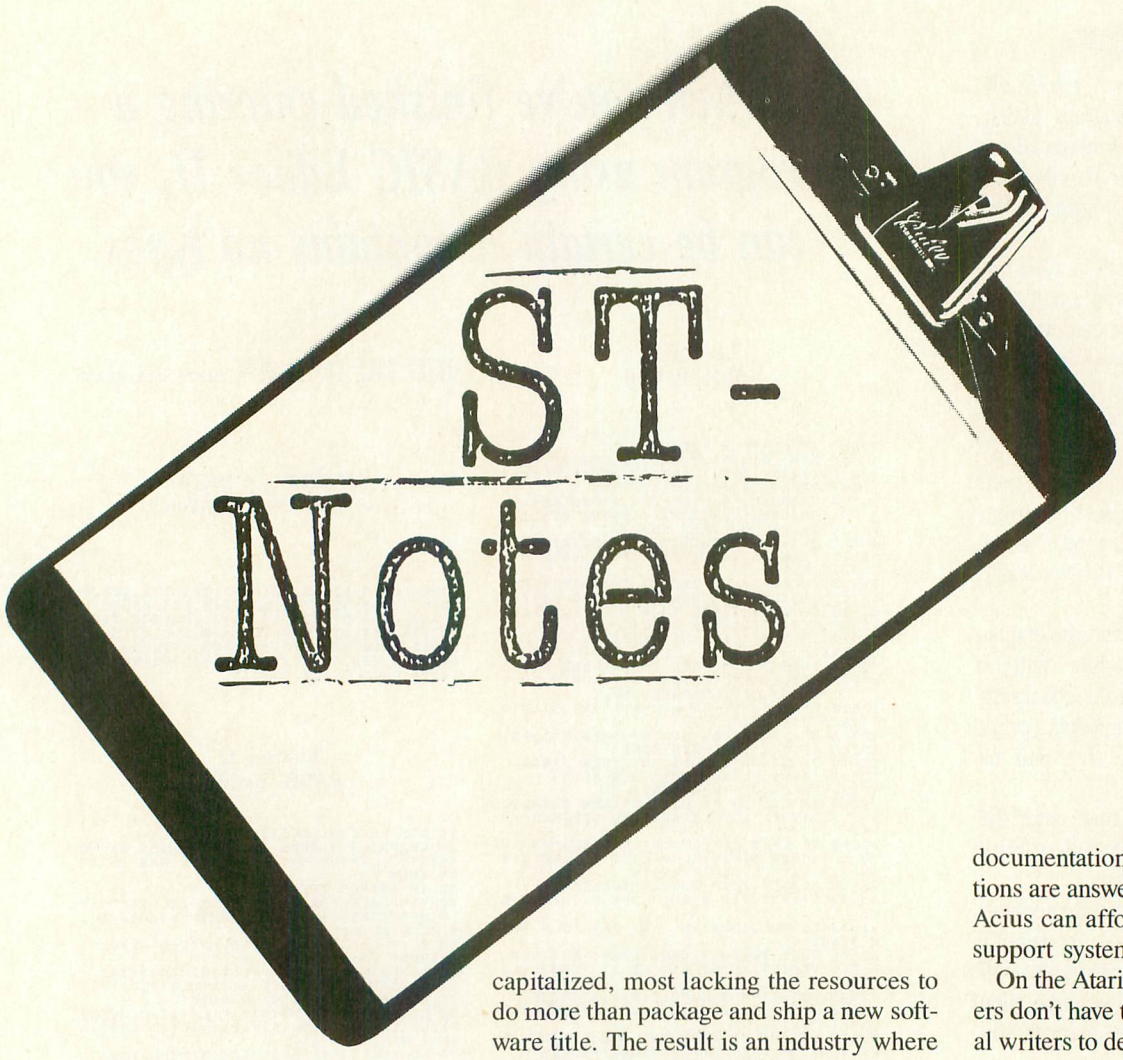
```
32600 DATA 6,665,923,757,809,171,225,8
98,532,499,910,267,912,144,735,8453
32638 DATA 97,358,230,693,706,878,317,
127,36,597,238,258,182,430,168,5315
32668 DATA 864,953,472,385,887,724,72,
687,908,736,625,612,672,184,891,9672
32698 DATA 8,856,85,949
```

Listing 2.
BASIC listing.

```
10 DIM L$(120),ML$(119),A$(1)
20 GRAPHICS 0:POKE 710,0:? "DISK OR CA
SSETTE";:INPUT A$:IF A$<>"C" AND A$<>"
D" THEN 20
30 IF A$="C" THEN 50
40 ? "PLACE FORMATTED DISK IN DRIVE":?
 "THEN PRESS RETURN":INPUT L$:OPEN #1,
8,0,"D:ML.DAT":GOTO 60
50 ? :? "READY CASSETTE, PRESS RETURN"
;:INPUT L$:OPEN #1,8,0,"C:"
60 L$="32608 M$(1)=":L$(13)=CHR$(34)
70 N=119:GOSUB 130:L$(14)=ML$(1,58):L$
(LEN(L$)+1)=CHR$(34):? #1;L$
80 L$(1)="32610 M$(59)=":L$(14)=CHR$(3
4):L$(15)=ML$(59):L$(LEN(L$)+1)=CHR$(3
4):? #1;L$
90 L$(1)="32612 S$=":L$(10)=CHR$(34)
100 ML$="":N=98:GOSUB 130:L$(11)=ML$:L
$(LEN(L$)+1)=CHR$(34):? #1;L$
110 L$(1)="32614 E$=":L$(10)=CHR$(34)
120 ML$="":N=69:GOSUB 130:L$(11)=ML$:L
$(LEN(L$)+1)=CHR$(34):? #1;L$:END
130 FOR X=1 TO N:READ A:ML$(X)=CHR$(A)
:NEXT X:RETURN
140 DATA 104,104,133,204,104,133,203,1
04,104,133,205,169,0,141,3,6,141,2,6,1
41,4,6,141,5,6
150 DATA 141,6,6,238,3,6,32,68,218,172
,2,6,177,203,133,212,32,170,217,32,182
,221,32,68,218
160 DATA 173,3,6,133,212,32,170,217,32
,219,218,32,210,217,165,212,141,0,6,16
5,213,141,1,6,24
170 DATA 173,0,6,109,4,6,141,4,6,173,1
,6,109,5,6,141,5,6,144,3,238,6,6,238,2
180 DATA 6,172,2,6,196,205,208,176,173
,4,6,133,212,173,5,6,133,213,96
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
```



The software crisis of 1989

The Atari ST computer didn't just happen overnight. It took the creative talents of a number of people to make the ST a useful machine. Most of the software available for the ST was developed by independent computer programmers and sold to ST users through the traditional distributor/dealer channels. Three years after the introduction of the ST, the software industry has reached a crisis.

Selling computer software to Atari ST users has often proved to be confusing, difficult, nerve-racking and futile. Since Atari's motto is "Power without the price," programmers have found that their motto is "Lots of work with little pay." Many programmers have found it not worth their whiles to spend a year of their time on a program that is released by a software publishing company.

Software publishers are usually undercapitalized, most lacking the resources to do more than package and ship a new software title. The result is an industry where dealers receive new products from distributors that know nothing about the products they are selling. The dealers rarely get any help from the software publisher because the publishers don't have the resources to teach each and every dealer how to use their products.

If a programmer has developed a product that is at all technical, the dealers don't understand it and leave it up to the product's packaging to convince a customer to plunk down some money. For the software consumer it is "Let the buyer beware." Once you take that software home, don't expect much support from your local dealer.

This criticism of dealers and distributors isn't local to the Atari industry; the same can be said about IBM-PC and Macintosh dealers. The difference between the dealers goes back to Atari's motto. When spending $695 for a copy of Acius *Fourth Dimension* 4GL database for the Mac, inside the large box you will find over 800 pages of documentation, quick reference cards and a program that has been thoroughly tested. With that much documentation, 99% of the end-user questions are answered. With the high list price, Acius can afford a decent advertising and support system for its dealers.

On the Atari side, most software publishers don't have the money to hire professional writers to develop well-written manuals. The most popular desktop-publishing program for the ST comes with documentation full of bad grammar, spelling mistakes and poor organization. Luckily, the program's good user interface makes up for the manual's problems.

The Atari software market is becoming more mature. Buyers have come to expect technological breakthroughs and are becoming more critical of consumer-level products. Three years after the release of the ST, we still haven't seen a desktop-publishing system of the caliber of *Adobe Illustrator* for the Macintosh. We still have not seen a word processor as advanced as *Microsoft Word 4.0* for MS-DOS. The software-buying audience is aware of this.

The software crisis of 1989 is the impending problem of supplying new software to an evermore technically mature, buying public. We have reached the point where the old methods of marketing Atari software no longer work. Atari dealers and distributors will have to become more advanced, or the public will outgrow the Atari ST.

The software crisis of 1989 is the impending problem of supplying new software to an evermore technologically mature, buying public.

The story of George

Turtle is a public-domain, hard-disk back-up utility that is available at no charge from DELPHI, CompuServe and GEnie. The program was written by George Woodside, who also authored several antivirus programs that can also be found in the public domain. George is a professional programmer working mostly on mainframe and minicomputer systems as a systems analyst and project leader.

In the years since the ST was released, George has often produced programs and utilities that make using an ST a lot of fun. George's programs are always of commercial quality, yet he has not yet charged for any of his work. *Turtle* is a good example of the ingenuity built into his programs: The program creates a temporary RAMdisk—the equivalent to a floppy diskette with files stored in your ST's memory. Files from your hard disk are copied into the RAMdisk. Once the RAMdisk is full, *Turtle* copies the entire RAMdisk contents onto a floppy diskette. The result is a floppy disk containing a copy of your hard-disk files, but with the use of the RAMdisk, backups are finished in just a few minutes. *Turtle* was written two years ago, and a better hard-disk, back-up program has yet to surface.

George is working on another ST application. This one will manage picture files from *DEGAS*, *Neochrome* and *Spectrum*. The picture manager will work much like the GEM Desktop; however, instead of managing files, the program will manage pictures created with any of the popular drawing programs for the ST.

George has become a data-compression expert while writing the picture manager; up to 12 different compression techniques will be used to reduce the amount of storage space needed to maintain a set of graphics. George has reported his compression techniques are 10% to 15% more efficient than using *Squeeze*, *ARC* or *Tiny* format.

The picture manager will also maintain an index of all the images in your library. More than 30 floppy disks of images can be indexed, with a powerful mechanism to locate a desired graphic. Simply describe a design, and the picture manager will tell you which floppy disk to insert. The graphic will appear on your ST's screen instantly, regardless of the format. A special animation editor will also be included, and users will be able to create slide shows that include color rotation and other special effects.

George is currently working on the final version of the picture manager, which should be released shortly. If you have a suggestion or interesting idea, he can be reached directly on CompuServe (76537,1342) and GEnie (G.WOODSIDE).

The new GDOS is G+Plus

Charles Johnson and John Eidsvoog have produced their own version of GDOS, the missing part of your Atari ST's GEM operating system, that allows programs to display and print graphic fonts and styles. Johnson and Eidsvoog began the project as curious hackers: They were interested in the inner workings of GDOS and the font system being used. What they found was unexpected.

GDOS was originally developed in C by Digital Research, the creators of the GEM operating system. During the development of the ST, Atari and DRI decided not to include GDOS in the ST's ROM operating system because of physical size limitations of the ROM and problems interfacing new device drivers to the ST version of GEM. Atari fell from grace with DRI shortly after the ST was released, which made hopes of GDOS being made available for the ST slight. Through some delicate maneuverings, Atari eventually bought the rights to produce an ST version of GDOS. First tests of GDOS were embarrassing. The ST Desktop slowed down to a crawl, and most ST programs bombed when run.

After almost a year of work, Atari released GDOS 1.1. The new version arrived amongst a flurry of controversy: Atari had decided to charge a high royalty for use of GDOS with commercial programs. Atari explained that they had spent a lot of money developing GDOS-compatible fonts that came with the licensed version of GDOS. ST software developers were not satisfied with that explanation, and eventually Atari backed off its stance and offered GDOS to developers for a one-time-only fee of $500.

Similar work on GDOS has been done by staff programmers of the most popular German ST magazine, *68000er Magazine*. A German programmer for the magazine wrote his own version of GDOS with the intentions of publishing the new software in the magazine as an article. A copy of the German GDOS was sent to Atari Senior Engineer Shiraz Shivji in the hopes that Atari would sanction the new GDOS and distribute it to ST users.

Johnson and Eidsvoog originally intended to seek Atari's official approval of the new GDOS, before they realized the delay in releasing the product to the market such a move would cause. Instead, Johnson and Eidsvoog created a partnership, Codehead Software, which is marketing *G+Plus* directly to ST users.

G+Plus is a more technologically superior software product than GDOS. When using GEM-compatible fonts, GDOS forces your ST to load all of the fonts into memory when you power-up your computer. This causes long boot-up periods and severely reduces available memory for your applications—try running a spreadsheet while GDOS is resident, and you will sometimes find less than half the worksheet size available. G+Plus solves this problem by allowing you to define which fonts will be loaded when an application is opened. When running your word processor, many fonts might be loaded. While running a graphics program, a different set of fonts could be loaded.

G+Plus offers many other advantages, which will be covered later in a full ST-LOG review. The package comes with a large instruction manual and has a list price of only $34.95.

(from page 25)

```
3990 DATA 16,169,120,157,68,3,169,65,1
57,69,3,169,6,157,74,3,4991
4000 DATA 169,0,157,75,3,169,3,157,66,
3,32,86,228,16,1,96,3993
4010 DATA 162,0,169,12,157,66,3,32,86,
228,162,0,76,20,77,69,4548
4020 DATA 169,19,157,68,3,169,77,157,6
9,3,169,12,157,74,3,169,6274
4030 DATA 0,157,75,3,169,3,157,66,3,32
,86,228,169,0,141,198,8580
4040 DATA 2,162,16,169,7,157,66,3,169,
0,157,72,3,157,73,3,4045
4050 DATA 68,77,192,77,32,86,228,48,26
,132,0,168,162,0,169,11,6199
4060 DATA 157,66,3,169,0,157,72,3,157,
73,3,152,32,86,228,164,8168
4070 DATA 0,16,210,162,16,169,12,157,6
6,3,32,86,228,76,129,77,7017
4080 DATA 96,96,84,121,112,101,32,97,1
10,121,32,107,101,121,46,96,6339
4090 DATA 0,160,0,132,0,185,112,77,201
,96,208,2,169,155,132,0,8876
4100 DATA 168,162,0,169,11,157,66,3,16
9,0,157,72,3,157,73,3,4243
4110 DATA 152,32,86,228,164,0,164,0,20
0,185,112,77,208,213,162,48,2154
4120 DATA 169,7,157,66,3,169,0,157,72,
3,157,73,3,32,86,228,6029
4130 DATA 96,226,2,227,2,126,65,0,0,0,
0,0,0,0,0,0,6813
```

LISTING 2

```
10 ;BASIC TO BINARY -10/16 ;
20 ;BY: MATTHEW ARRINGTON  ;
30 ;                       ;
40 ; OSS. MAC-65 SOURCE    ;
50 ;
60       .OPT NO LIST
70 ;
80 ;
90 ; MACROS
0100 ;
0110     .MACRO @PT
0120 ;
0130 ; PUT TO DEVICE
0140 ; " PUT IOCB#,BYTE "
0150 ;
0160     STY 0
0170     TAY
0180       .IF %0=1
0190       LDX #%1*16
0200       .ENDIF
0210     LDA #11
0220     STA 834,X   ; COMMAND PUT=11
0230     LDA #0
0240     STA 840,X
0250     STA 841,X
0260     TYA
0270     JSR 58454
0280     LDY 0
0290     .ENDM
0300 ;
0310 ;
0320 ;
0330     .MACRO @GT
0340 ;
0350 ; GET FROM DEVICE (INTERNAL)
0360 ; " GET IOCB# "
0370 ;
0380       .IF %0=1
0390       LDX #%1*16
0400       .ENDIF
0410     LDA #7
0420     STA 834,X   ; COMMAND GET=7
0430     LDA #0
0440     STA 840,X
0450     STA 841,X
0460     JSR 58454
0470     .ENDM
0480 ;
0490 ;  PUT BYTE
0500 ;
0510 ;
0520     .MACRO PUT
0530       .IF %0=0
0540        @PT
0550       .ELSE
0560        @PT  %1
0570       .ENDIF
0580     .ENDM
0590 ;
0600 ; GET  EX. "GET 0"
0610 ;
0620     .MACRO GET
0630       .IF %0=0
0640        @GT
0650       .ELSE
0660        @GT  %1
0670       .ENDIF
0680     .ENDM
0690 ;
0700 ; PRINT
0710 ;
0720     .MACRO PRINT
0730 ;
0740     JMP @OVER
0750 @TEXT .BYTE %$2,0
0760 @OVER LDY #0
0770 @LOOP STY 0
0780     LDA @TEXT,Y
0790     CMP #'♦
0800     BNE @CIO
0810     LDA #155
0820 @CIO  @PT  %1
0830     LDY 0
0840     INY
0850     LDA @TEXT,Y
0860     BNE @LOOP
0870     .ENDM
0880 ;
0890 ; PRINT BUFFER
0900     .MACRO PRINTB
0910     LDY #0
0920 @LO STY 0
0930 ;
0940       .IF %2<256
0950       LDA (%2),Y
0960       .ELSE
0970 ;
0980       LDA %2,Y
0990       .ENDIF
1000 ;
1010     CMP #'♦
1020     BNE @CO
1030     LDA #155
1040 @CO  @PT  %1
1050     LDY 0
1060     INY
1070       .IF %2<256
1080       LDA (%2),Y
1090       .ELSE
1100       LDA %2,Y
1110       .ENDIF
1120     BNE @LO
1130     .ENDM
1140 ;
1150 ; INPUT RECORD/ SAVE EOL
1160 ; EX. "INPUT IOCB#"
1170 ;
1180     .MACRO INPUT
1190 ;
1200     LDX #%1*16
1210     LDA #%2&255 ; INBUF LO
1220     STA 836,X
1230     LDA #%2/256 ; INBUF LO
1240     STA 837,X
1250     LDA #5      ; GET RECORD
1260     STA 834,X
1270     LDA #40     ; BUF LEN
1280     STA 840,X
1290     LDA #0
1300     STA 841,X
1310     JSR 58454
1320     LDY 840,X
1330     LDA #0
1340     STA %2,Y
1350     .ENDM
1360 ;
1370     .MACRO OPEN
1380 ;
1390     LDX #%1*16  ; IOCB #
1400       .IF %4<256
1410       JMP @EXC
1420 @OPEN .BYTE %$4
1430 @EXC  LDA #@OPEN&255 ;(S:,D:)
1440       STA 836,X
```

```
1450        LDA #@OPEN/256
1460        STA 837,X
1470        .ELSE
1480        LDA #%4&255
1490        STA 836,X
1500        LDA #%4/256
1510        STA 837,X
1520        .ENDIF
1530      LDA #%2
1540      STA 842,X    ; OPEN (8,12,4)
1550      LDA #%3
1560      STA 843,X    ; AUX1
1570      LDA #3       ; COMMAND OPEN
1580      STA 834,X
1590      JSR 58454
1600      .ENDM
1610 ;
1620      .MACRO RWSECT
1630 ;
1640 ; DUNIT,DCOM,DBUFF,SIZE,SECT
1650 ;
1660      LDA #49
1670      STA 768      ; DDEVIC
1680        .IF %1>4
1690        LDA %1
1700        .ELSE
1710        LDA #%1
1720        .ENDIF
1730      STA 769      ; DUNIT
1740      LDA #%2
1750      STA 770
1760      LDA #%6
1770      STA 771
1780      LDA #%3&255
1790      STA 772
1800      LDA #%3/256
1810      STA 773
1820      LDA #31
1830      STA 774
1840      LDA #%4&255
1850      STA 776
1860      LDA #%4/256
1870      STA 777
1880      LDA #%5&255
1890      STA 778
1900      LDA #%5/256
1910      STA 779
1920      JSR 58457
1930      .ENDM
1940 ;
1950 ; MACRO CLOSE IOCB
1960 ;
1970      .MACRO CLOSE
1980 ;
1990      LDX #%1*16
2000      LDA #12      ; COMMAND CLOSE
2010      STA 834,X
2020      JSR 58454    ; CIO
2030      .ENDM
2040 ;
2050      .MACRO XIO
2060 ;
2070      LDX #%2*16   ; IOCB
2080      LDA #%1      ; COMMAND
2090      STA 834,X
2100      LDA #%3
2110      STA 842,X    ;  AUX1
2120      LDA #%4
2130      STA 843,X    ; AUX2
2140      LDA #%5&255
2150      STA 836,X
2160      LDA #%5/256
2170      STA 837,X
2180      JSR 58454    ; CIOV
2190 ;
2200      .ENDM
2210 ;
2220      .MACRO POS
2230 ;.EX POS COL,ROW
2240 ;
2250      LDA #%1
2260      STA 84
2270      LDA #%2
2280      STA 85
2290 ;
2300      .ENDM
2310 ;
2320      .MACRO BGET
2330 ;
2340 ; EX. BGET 1,BUFFER,10000,LEN
2350 ;
2360      LDX #%1*16
2370      LDA #7
2380      STA 834,X    ; COMMAND
2390      LDA #%2&255
2400      STA 836,X
2410      LDA #%2/256
2420      STA 837,X
2430      LDA #%3&255
2440      STA 840,X
2450      LDA #%3/256
2460      STA 841,X
2470      JSR $E456    ;CIO
2480 ;
2490      LDA 840,X
2500      STA %4
2510      LDA 841,X
2520      STA %4+1
2530      .ENDM
2540 ;
2550 ; EX. BPUT 1,BUFFER,LEN
2560 ;
2570      .MACRO BPUT
2580      LDX #%1*16
2590      LDA #11
2600      STA 834,X    ; COMMAND
2610      LDA #%2&255
2620      STA 836,X
2630      LDA #%2/256
2640      STA 837,X
2650      LDA %3
2660      STA 840,X
2670      LDA %3+1
2680      STA 841,X
2690      JSR $E456    ;CIO
2700 ;
2710      .ENDM
2720 ;
2730 ;
2740 ; WORK BYTES
2750 ;
2760      *=  $0600
2770 ;
2780 MAXLINE .BYTE 0
2790 INLINE .BYTE 0
2800 FLASH .BYTE 0
2810 TEMP1 .BYTE 0
2820 CHR1 .BYTE 0
2830 CHR2 .BYTE 0
2840 OPTION .BYTE 0
2850 SCRN .BYTE 0
2860 LEN .BYTE 0,0
2870 TEMY .BYTE 0
2880 ;
2890 LBUFF = $057E
2900 HATABS = $031A
2910 ;
2920 ; ZERO PAGE
2930 ;
2940 TEMP2 = 203
2950 TEMP3 = 204
2960 PSTART = 224
2970 PEND =  226
2980 MEMBOT = 228
2990 AMOUNT = 230
3000 ;
3010      *=  $2700
3020 ;
3030 ; LOAD SCREEN
3040 ;
3050 INIT LDA #0      ; BLACK
3060      STA 710     ; SCREEN
3070      STA 82
3080      LDA #97     ; RED
3090      STA 712     ; BORDER
3100      LDA #1      ; NO CURSOR
3110      STA 752
3120       PRINT  0,"K↓↓↓↓↓↓↓↓↓↓ Basic
to Binary by: Matthew Arrington"
3130      RTS         ; CONTINUE LOAD
3140 ;
3150      *=  738
3160      .WORD INIT  ; FOR BIN LOAD
3170 ;
3180      *=  $2700
3190 ;
3200 ;
3210 ; EXECUTE & RUN BASIC PROGRAM
3220 ;
3230 ; THIS ROUTINE GETS SAVED ALONG
```

```
3240 ; WITH BASIC PROGRAM.
3250 ;
3260 ;
3270 BASRUN .BYTE "[illegible]" ; BIN LOAD HEAD
ER
3280     .WORD FILELEN ; LOAD START
3290     .WORD RN3    ; LOAD END
3300 ;
3310 ;
3320 FILELEN *= *+2  ; BASIC FILE LEN
3330 ;
3340 TLINE .BYTE "
♥" ; TITLE STORED HERE
3350 ;
3360 RESET .BYTE 0   ; OPTION FLAGS
3370 BREAK .BYTE 0
3380 TITLE2 .BYTE 0,0
3390 MEMLO .WORD LA12+3
3400 ;
3410 GO  LDA 6       ; CART PRESENT?
3420     BNE RN2     ; YES CONTINUE
3430 ;
3440     LDA #1      ; CURSOR OFF
3450     STA 752
3460     LDA #0      ; ZERO MARGIN
3470     STA 82
3480      PRINT  0,"[illegible] PROGRAM REQUI
RES BASIC.  REBOOT     "
3490 X2  JMP X2      ; LOOP FOREVER
3500 ;
3510 RN2 LDA TITLE2  ; DISPLAY TITLE?
3520     BEQ RN3     ; NO
3530 ;
3540      OPEN  6,12,1,"S" ; GR. 1
3550      POS  10,0
3560      PRINTB  6,TLINE
3570 RN3 RTS         ; COUNTINE LOAD
3580 ;
3590 ; MORE BIN POINTERS
3600 ;
3610     .WORD 738,739 ; INIT ADD.
3620     .WORD GO
3630 ;
3640     .WORD X1,LA12+2
3650 ;
3660 ;
3670 ; INIT MEMLO
3680 ;
3690 X1  LDY #0
3700     LDA MEMLO   ; SET REAL MEMLO
3710     STA 743
3720 ;
3730     CLC
3740     ADC #242
3750     STA TEMP2
3760 ;
3770     LDA MEMLO+1
3780     STA 744
3790 ;
3800     ADC #0
3810     STA TEMP3
3820 ;
3830 ; INIT RESET VECTORS
3840 ;
3850     LDA RESET   ; TRAP RESET?
3860     BEQ TRAP    ; BR. IF YES
3870 ;
3880     CMP #2      ; COLD START?
3890     BNE MOVEPOINT ;NO, RESET NORM
AL
3900     LDA #1      ; DIE ON RESET
3910     STA 580     ; COLST
3920     JMP MOVEPOINT
3930 ;
3940 TRAP LDA 13     ; TRAP RESET
3950     CMP #8      ; DOS PRESENT?
3960     BCS DODOS   ;BR. IF SO
3970     LDA #INITAL&255 ; SET RESET V
ECTOR
3980     STA 12
3990     LDA #INITAL/256
4000     STA 13
4010     JMP MOVEPOINT
4020 ;
4030 DODOS LDA 12    ; SAVE DOS INIT V
EC
4040     STA RESET1+1
4050     LDA 13
4060     STA RESET1+2
4070 ;
4080     LDA #RESET1&255 ; SET NEW RES
ET VEC
4090     STA 12
4100     LDA #RESET1/256
4110     STA 13
4120 ;
4130 ; ADD VALUE OF MEMLO TO BASIC'S
4140 ; PROGRAM POINTERS.
4150 ;
4160 MOVEPOINT LDA 743
4170     CLC
4180     ADC (TEMP2),Y
4190     STA 128,Y
4200     INY
4210 ;
4220     LDA 744
4230     ADC (TEMP2),Y
4240     STA 128,Y
4250     INY
4260     CPY #14
4270     BNE MOVEPOINT
4280 ;
4290     LDA 140     ; COPY STARP TO:
4300     STA 142     ; RUNSTAK
4310     STA 144     ; MEMTOP  (LOW)
4320 ;
4330     LDA 141     ; (HIGH)
4340     STA 143
4350     STA 145
4360 ;
4370 ;
4380 ;   SET UP TO RUN
4390 ;
4400     LDA #0
4410     STA $92     ; BASIC DOES THIS
4420     STA $CA     ; ON COLD START
4430     LDX #$FF
4440     TXS
4450     CLD
4460 ;
4470     LDA $0222   ; SAVE VBI VEC
4480     STA VBIX+1
4490     LDA $0223
4500     STA VBIX+2
4510     JMP INITAL
4520 ;
4530 ; IMMEDIATE VBI TO DISSABLE
4540 ; BREAK.
4550 ;
4560 BRK LDA #64
4570     STA 16
4580     STA 53774
4590 VBIX .BYTE $4C,0,0
4600 ;
4610 ;
4620 RESET1 .BYTE $20,0,0 ; JSR INIT D
OS
4630 INITAL LDX #0
4640     LDA BREAK   ; DISSABLE BREAK?
4650     BNE LA2     ; BR. IF NOT
4660 ;
4670     LDA #BRK&255 ; SET IMM. VBI
4680     STA $0222   ; VECTOR
4690     LDA #BRK/256
4700     STA $0223
4710 ;
4720 ; PATCH THE "E:" INPUT COMMAND
4730 ; WILL FORCE A "RUN" WHEN BASIC
4740 ; GOES TO THE "READY" PROMPT.
4750 ;
4760 LA2 LDA HATABS,X ; GET A BYTE
4770     CMP #'E     ; LOOK FOR E
4780     BEQ LA1     ; FOUND YA
4790 ;
4800     INX         ; NEXT HANDLER
4810     INX         ; ADDRESS
4820     INX
4830     BNE LA2     ; KNOW ITS THERE
4840 ;
4850 LA1 INX
4860     STX LA3     ; SAVE OFFSET
4870 ;
4880     LDA HATABS,X ; SAVE OLD
4890     STA TEMP2
4900     LDA #VTBLE&255 ; REPLACE WITH
 NEW
4910     STA HATABS,X ;$031A
4920 ;
4930     INX         ; NOW HI BYTES
4940 ;
```

```
4950     LDA HATABS,X ; SAVE OLD
4960     STA TEMP2+1
4970     LDA #VTBLE/256 ; REPLACE WITH
 NEW
4980     STA HATABS,X
4990 ;
5000     LDY #0       ; COPY VTABLE
5010     LDX #4
5020 LA7 LDA (TEMP2),Y
5030     STA VTBLE,Y
5040     INY
5050     DEX
5060     BNE LA7
5070     LDY #8
5080     LDX #7
5090 LA8 LDA (TEMP2),Y
5100     STA VTBLE,Y
5110     INY
5120     DEX
5130     BNE LA8
5140 ;
5150     LDA #2       ; COUNTER
5160     STA LA10
5170 ;
5180     LDA #1       ; BOOT SUCCESSFUL
5190     STA 9        ; BOOT
5200     STA 8        ; WARMST
5210 ;
5220     LDA 49146    ; CART RUN ADD.
5230     STA CART+1
5240     LDA 49147
5250     STA CART+2
5260 CART .BYTE $4C,0,0 ; JMP TO BASIC
5270 ;
5280 EINPUT LDY LA10 ; GET OFFSET
5290     CPY #255    ; END?
5300     BEQ LA11    ; YES
5310 ;
5320     LDA LA12,Y
5330     DEC LA10
5340     LDY #1
5350     RTS
5360 ;
5370 LA11 TXA
5380     PHA
5390     LDX LA3
5400     LDA TEMP2
5410     STA HATABS,X ;$031A
5420     INX
5430     LDA TEMP2+1
5440     STA HATABS,X ;$031A
5450     PLA
5460     TAX
5470     LDA #155    ; C/R
5480     LDY #1      ; SUCCESS
5490     RTS         ; DONE
5500 LA3 .BYTE 0
5510 LA10 .BYTE 0
5520 ;
5530 VTBLE .BYTE 0,0 ; CLOSE
5540     .BYTE 0,0   ; OPEN
5550     .WORD EINPUT-1 ; GET
5560     .BYTE 0,0   ; PUT
5570     .BYTE 0,0   ; STATUS
5580     .BYTE 0,0   ; SPECIAL
5590     .BYTE 0,0,0 ; JMP INIT
5600 ;
5610 LA12 .BYTE "NUR"
5620 ;
5630     .BYTE "[] a |" ;RUN POINTERS
5640     .WORD X1    ;RUN BAS RUN
5650 ;
5660 BASICSTART .BYTE 0,0
5670     .BYTE 0
5680 ENDRUN .BYTE 0
5690 ;
5700 ; DISPLAY LIST
5710 ;
5720 DLIST .BYTE 112,112,112,66,64,156
5730     .BYTE 66,104,156,66,144,156,6
6,184,156
5740     .BYTE 66,224,156,66,8,157,66,
48,157
5750     .BYTE 66,88,157,66,128,157,66
,168,157
5760     .BYTE 66,208,157,66,248,157,6
6,32,158
5770     .BYTE 66,72,158,66,112,158,66
,152,158
5780     .BYTE 66,192,158,66,232,158,6
6,16,159
5790     .BYTE 66,56,159,66,96,159,66,
136,159
5800     .BYTE 66,176,159,66,216,159,6
5
5810     .WORD DLIST
5820 MENU .BYTE "K♦♦♦♦♦"
5830     .BYTE "          ┌──────────
♦"
5840     .BYTE "          |
♦"
5850     .BYTE "     ┌────┴─────────
♦"
5860     .BYTE "     |Basic to Binary|
♦"
5870     .BYTE "┌────┘              └
──────┐♦"
5880     .BYTE "|
   |♦"
5890     .BYTE "| 1: Display/Set Optio
ns  |♦"
5900     .BYTE "|
   |♦"
5910     .BYTE "| 2: Convert Basic fil
e   |♦♦"
5920 MENU2 .BYTE "|
  |♦"
5930     .BYTE "| 3: File Directory
|♦"
5940     .BYTE "|
|♦"
5950     .BYTE "| 4: Exit to DOS
|♦"
5960     .BYTE "|
|♦"
5970     .BYTE "└──────────────────
──────┘♦♦"
5980 ;
5990 ;
6000 ;
6010 ; EVERYDAY TYPE SUBROUTINES
6020 ;
6030 ;
6040 ; INPUT A LINE
6050 ;
6060 GETLINE LDA #0  ; LINE INDEX &
6070     STA INLINE  ; LINE LEN=0
6080 ;
6090 GLOOP LDA #10   ; 1/6 SEC
6100     STA 536     ; SYS TIMER
6110 TIMLOOP LDA 764 ; STAT KEYBOARD
6120     CMP #255    ; KEY IN?
6130     BEQ CHECKTIME ; BR. IF NO
6140      PRINT  0,"_←" ; PRINT CURSOR
6150     JMP GCHR    ; GET KEY
6160 ;
6170 CHECKTIME LDA 536 ; SYS TIMER=0?
6180     BNE TIMLOOP ; NO, LOOP
6190     LDA FLASH   ; FLASH ON OR OFF
?
6200     BEQ OFF     ; TURN IT OFF
6210      PRINT  0,"_←" ; CURSOR ON
6220     LDA #0      ; FLASH OFF=0
6230     STA FLASH   ; FOR NEXT TIME
6240     JMP GLOOP   ; LOOP
6250 ;
6260 OFF  PRINT  0," ←" ; CUROSRO OFF
6270     LDA #1      ; FLASH ON=1
6280     STA FLASH   ; FOR NEXT TIME
6290     JMP GLOOP   ; LOOP
6300 ;
6310 GCHR  GET  3    ; GET A CHAR
6320 ;
6330     CMP #'{     ; BACKSPACE?
6340     BNE DL      ; BR. IF NOT
6350 ;
6360     LDX INLINE  ; SOMTHING TO BS?
6370     BNE OK1     ; BR. IF YES
6380     JMP GLOOP   ; NOTHING THERE
6390 OK1 JMP BACKSPACE ; DO BACKSPACE
6400 ;
6410 DL  CMP #'[]    ; DELETE LINE?
6420     BNE ENDL    ; BR. IF NOT
6430     LDX INLINE  ; SOMTHING TO DEL
?
6440     BNE OK2     ; BR. IF YES
6450     JMP GLOOP   ; ELSE IGNORE & L
OOP
6460 OK2 JMP DLINE   ; DO DELETE LINE
6470 ;
```

```
6480 ENDL CMP #155   ; END OF LINE?
6490      BNE NOCTRL  ; NO
6500      JMP EOL     ; DO ENDLINE
6510 NOCTRL LDX STRIP ; FILTER CHARS?
6520      BNE OK3     ; BR. IF NO
6530 ;
6540 ;
6550      AND #127    ; STRIP INVERSE
6560      CMP #97     ; LOWWER CASE??
6570      BCC OK3     ; BRANCH IF NOT
6580      SEC
6590      SBC #32     ; MAKE IT UPPER
6600 OK3 LDX INLINE   ; RESTORE X
6610      CPX MAXLINE ; LINE TOO LONG?
6620      BNE OK5     ; BR. AND SAVE
6630      JMP GLOOP   ; IGNORE KEY & LO
OP
6640 ;
6650 OK5 STA 1406,X   ; SAVE BYTE
6660      INX         ; FOR NEXT SAVE
6670      STX INLINE  ; SAVE X
6680       PUT  0     ; PRINT CHAR
6690      JMP GLOOP   ; GET NEXT BYTE
6700 ;
6710 ;
6720 BACKSPACE JSR ST
6730 ;
6740 DOBS LDA #'{     ; BS CHAR
6750       PUT  0     ; PRINT IT
6760      DEC INLINE  ; DEC LINE LEN
6770      JMP GLOOP   ; NEXT KEY
6780 ;
6790 ;
6800 DLINE JSR ST
6810 ;
6820 DODL LDA #'{     ; BS CHAR
6830       PUT  0     ; PRINT IT
6840      DEC INLINE  ; DEC LINE LEN
6850      BNE DODL    ; LOOP TILL GONE
6860      JMP GLOOP   ; START AGAIN
6870 ;
6880 ;
6890 EOL LDA #'}      ; DELETE PROMPT
6900       PUT  0
6910      LDX INLINE  ; GET LINE LEN
6920      LDA #0
6930      STA 1406,X  ; MARK EOL
6940      RTS         ; RETURN
6950 ;
6960 ;
6970 ST    PRINT  0,"→{" ; ERASE CURSOR
6980      RTS
6990 ;
7000 ;
7010 PUTBYTE LDX #0   ; IOCB #0
7020 PUTD  PUT        ; PUT A
7030      RTS         ; RETURN
7040 ;                      ;
7050 ;                      ;
7060 ; VALIDATE A FILENAME  ;
7070 ;                      ;
7080 SETNAME LDY #0   ; SET DEFUALT
7090 L8   LDA DEF,Y   ; TO DRIVE ONE
7100      STA FNAME,Y
7110      INY
7120      CPY #3
7130      BNE L8
7140 ;
7150      LDA LBUFF+1 ; "D:FNAME"?
7160      CMP #':
7170      BNE L5      ; BR. IF NOT
7180      LDY #2      ; SET OFFSET
7190      JMP MOVENAME ; COPY NAME OVER
7200 ;
7210 L5   LDY #0      ; "DX:FNAME"??
7220      LDA LBUFF+2
7230      CMP #':
7240      BNE MOVENAME ;NO, ASSUME "D1:
"
7250      LDA LBUFF+1 ; GET D# BYTE
7260      STA FNAME+1 ; SAVE IT
7270      LDY #3      ; OFFSET
7280 ;
7290 MOVENAME LDX #3 ; COPY FILE NAME
7300 L6   LDA LBUFF,Y ; FROM LBUFF.
7310      STA FNAME,X
7320      BEQ L7
7330      INY
7340      INX
7350      BNE L6
7360 ;
7370 L7   RTS
7380 ;
7390 ; CALCULATE THE DISTANCE BETWEEN
7400 ; TO ADRESS'S...
7410 ;
7420 SUBTRACT LDA PEND
7430      SEC
7440      SBC PSTART
7450      STA LEN
7460 ;
7470      LDA PEND+1
7480      SBC PSTART+1
7490      STA LEN+1
7500 ;
7510      INC LEN
7520      BNE L15
7530      INC LEN+1
7540 L15 RTS
7550 ;
7560 BUF *=  *+5125  ; 5K BUFFER
7570 ;
7580 FNAME *= *+16
7590 STRIP .BYTE 0
7600 DEF .BYTE "D1:" ; DEFUALT DRIVE
7610 DIR .BYTE "D0:*.*" ; DIRECTORY
7620 ;
7630 RUN   OPEN  3,4,0,"K:"
7640      LDA #DLIST&255 ; POINT TO
7650      STA 560      ; MY DIS. LIST
7660      LDA #DLIST/256
7670      STA 561
7680 ;
7690      LDA #64      ;  LET THE "E:"
7700      STA 88       ; IN ON THE CHANG
E.
7710      LDA #156
7720      STA 89
7730      LDA #1       ; CURSOR OFF
7740      STA 752
7750      LDA #31      ; RIGHT MARGIN
7760      STA 83
7770 ;
7780 ;
7790 MAINMENU LDA #0 ; SCREEN OFF
7800      STA 559
7810      STA 710      ; CHANGE COLOR
7820      LDA #97
7830      STA 712
7840      LDA #2       ; LEFT MARGIN
7850      STA 82
7860       PRINTB  0,MENU ; PRINT MENU
7870       PRINTB  0,MENU2
7880      LDA #33      ; SCREEN ON
7890      STA 559
7900 ;
7910 ; TWIRLING THING
7920 ;
7930      LDA #69
7940      STA CHR1
7950      LDA #90
7960      STA CHR2
7970 TITLE LDY #11
7980      LDA #0
7990      LDX #3
8000      STA 40208,X
8010      STA 40208,Y
8020      LDY CHR1
8030      LDA CHR2
8040      STY CHR2
8050      STA CHR1
8060      LDX #4
8070      LDY #10
8080 TLOOP LDA 40208,X
8090      STA TEMP1
8100      LDA CHR1
8110      STA 40208,X
8120      LDA 40208,Y
8130      STA TEMP2
8140      LDA CHR2
8150      STA 40208,Y
8160      LDA #0
8170      STA 20
8180 TTIME LDA 20
8190      CMP #3
8200      BNE TTIME
8210      LDA #0
8220      JSR TWIST
8230      STA 40208,X
8240      LDA #1
```

```
8250      JSR TWIST
8260      STA 40208,Y
8270 ;
8280      LDA 764
8290      CMP #255
8300      BNE GOPT
8310 ;
8320      INX
8330      DEY
8340      CPY #2
8350      BNE TLOOP
8360      JMP TITLE
8370 TWIST CMP #1
8380      BEQ OVER
8390      LDA TEMP1
8400      JMP SWAP
8410 OVER LDA TEMP2
8420 ;
8430 SWAP CMP #90
8440      BNE F1
8450      JMP SWITCH
8460 F1   CMP #69
8470      BNE F2
8480      JMP SWITCH
8490 F2   CMP #67
8500      BNE F3
8510      JMP SWITCH
8520 F3   CMP #81
8530      BNE F4
8540      JMP SWITCH
8550 F4   CMP #82
8560      BEQ ZERO
8570      LDA #82
8580      RTS
8590 ZERO LDA #0
8600      RTS
8610 SWITCH LDA CHR1
8620      CMP #90
8630      BNE OTHER
8640      LDA #67
8650      STA CHR1
8660      LDA #81
8670      STA CHR2
8680      JMP EXIT
8690 ;
8700 OTHER LDA #69
8710      STA CHR1
8720      LDA #90
8730      STA CHR2
8740 EXIT LDA #87
8750      RTS
8760 ;
8770 ; GET OPTION FROM KEYBOARD
8780 ;
8790 GOPT LDA #0       ; ZERO MARGIN
8800      STA 82
8810       GET  3      ; GET KEY
8820      AND #127     ; STRIP INVERSE
8830      CMP #97      ; LOWWER CASE?
8840      BCC A6       ; BR. IF NOT
8850      SEC          ; ELSE....
8860      SBC #32      ; MAKE IT UPPER
8870 ;
8880 A6   CMP #'2      ; OPTION 2?
8890      BNE A1       ; NO
8900      JSR CONVERT  ; GO CONVERT
8910      JMP MAINMENU ;
8920 ;
8930 A1   CMP #'1      ; OPTION 1?
8940      BNE A2       ; NO
8950      JSR PAD      ; CONTROL PAD
8960      JMP MAINMENU
8970 ;
8980 A2   CMP #'3      ; HOW ABOUT 3?
8990      BNE A4       ; NO
9000      JSR FILES    ; DIRECTORY
9010      JMP RUN      ; REINIT SCREEN
9020 ;
9030 A4   CMP #'4      ; OPTION 4?
9040      BEQ A5       ; YES...
9050      JMP MAINMENU ; BAD INPUT
9060 ;
9070 A5   LDA #39      ; RESET RIGHT
9080      STA 83       ; MARGIN
9090      JMP (10)     ; GO TO DOS
9100 ;
9110 ; CONVERT A FILE TO BINARY
9120 ;
9130 CONVERT LDA #125
9140      JSR PUTBYTE ; CLEAR SCREEN
9150      LDA #16
9160      STA MAXLINE ; INPUT LINE LEN.
9170 ;
9180       POS  10,0
9190 L11   PRINT  0,"INPUT FILE - "
9200      JSR GETLINE ; GET INPUT NAME
9210      LDA INLINE  ; NAME ENTERED?
9220      BNE L10     ; BR. IF YES
9230      RTS         ; NO, RETURN
9240 ;
9250 L10 JSR SETNAME ; VALIDATE NAME
9260       CLOSE  1
9270       OPEN  1,4,0,FNAME
9280      BMI L11     ; ERROR TRY AGAIN
9290 ;
9300 ; PASS 1, COUNT BYTES
9310 ;
9320 ; MUST KNOW EXACT LEN OF BASIC
9330 ; FILE.
9340 ;
9350      LDA #0      ; ZERO BASIC
9360      STA FILELEN ; FILE LENGTH
9370      STA FILELEN+1
9380 ;
9390 L13   GET  1     ; GET A BYTE
9400      CPY #136    ; CHECK FOR EOF
9410      BEQ PASS2   ; BR. EOF REACHED
9420 ;
9430      INC FILELEN ; COUNT BYTES.
9440      BNE L13
9450      INC FILELEN+1
9460      BNE L13
9470 ;
9480 ; PASS2 SAVE ROUTINE THAT RUNS
9490 ; THE BASIC. THEN COPY BASIC
9500 ; PROGRAM OVER TO THE NEW FILE.
9510 ;
9520 PASS2  CLOSE  1 ; REOPEN INPUT FI
LE
9530       OPEN  1,4,0,FNAME
9540 ;
9550 L3    PRINT  0,"OUTPUT FILE - "
9560      JSR GETLINE ; GET FILENAME
9570      LDA INLINE  ; NAME ENTERED?
9580      BNE L12     ; BR. IF SO
9590      RTS         ; RETURN - ABORT
9600 ;
9610 L12 JSR SETNAME ; VALIDATE NAME
9620       CLOSE  2
9630       OPEN  2,8,0,FNAME
9640      BMI L3
9650 ;
9660 ; CALCULATE LENGTH OF THE ROUTINE
9670 ; THAT'LL EXECUTE THE BASIC
9680 ; PROGRAM. SAVE RESULT IN "LEN"
9690 ;
9700      LDA #BASRUN&255
9710      STA PSTART
9720      LDA #BASRUN/256
9730      STA PSTART+1
9740 ;
9750      LDA #ENDRUN&255
9760      STA PEND
9770      LDA #ENDRUN/256
9780      STA PEND+1
9790      JSR SUBTRACT ; GET LENGTH
9800 ;
9810 ; MAKE BIN LOAD POINTERS
9820 ;
9830 ; ADD THE LEN. OF THE BAS PROGRAM
9840 ; TO THE ADDRESS WHERE IT FIRST
9850 ; LOADS IN MEMORY.
9860 ;
9870      LDA MEMLO   ; START ADDRESS
9880      CLC
9890      ADC #242    ; BAS. BUFFER OFF
SET
9900      STA TEMP2
9910      LDA MEMLO+1
9920      ADC #0
9930      STA TEMP3
9940 ;
9950 ; LOW BYTES
9960 ;
9970      LDA TEMP2   ; SAVE START POIN
T
9980      STA BASICSTART
9990      CLC         ; ADD IN FILE LEN
GTH
010000    ADC FILELEN
```

```
010010   STA ENDRUN-1 ;SAVE END POINT
010020 ;
010030 ; HIGH BYTES
010040 ;
010050   LDA TEMP3    ; SAVE START
010060   STA BASICSTART+1
010070   ADC FILELEN+1 ; ADD IN LEN
010080   STA ENDRUN   ; SAVE END
010090 ;
010100   LDA ENDRUN-1 ; SUBTRACT 1
010110   SEC          ; FROM END POINT
010120   SBC #1       ; (FUDGE)
010130   STA ENDRUN-1
010140   LDA ENDRUN
010150   SBC #0
010160   STA ENDRUN
010170 ;
010180 ; SAVE THE EXECUTE ROUTINE
010190 ; ALONG WITH ALL BIN POINTERS
010200 ;
010210    BPUT  2,BASRUN,LEN
010220 ;
010230 ; COPY BASIC FILE OVER
010240 ;
010250 L4  BGET  1,BUF,5125,LEN
010260   STY TEMY     ; SAVE ERROR FLAG
010270    BPUT  2,BUF,LEN
010280 ;
010290   LDY TEMY     ; GET ERROR FLAG
010300   CPY #136     ; EOF REACHED?
010310   BNE L4       ; BR. IF NOT
010320 ;
010330    CLOSE  1    ; CLEAN UP
010340    CLOSE  2
010350   RTS          ; ALL DONE
010360 ;
010370 ; CONTROL PAD?!
010380 ;
010390 PAD LDA #125   ; CLEAR SCREEN
010400   JSR PUTBYTE
010410   LDA #0       ; ZERO MARGIN
010420   STA 82
010430 ;
010440    PRINT  0,"  Basic to Binary
Control Pad♦"
010450    PRINT  0,"  ──────────────
──────────♦"
010460    PRINT  0,"  SELECT : Item to
 Change♦"
010470    PRINT  0,"  OPTION : Changes
 Item♦"
010480    PRINT  0,"  START  : Exits♦♦
♦♦"
010490    PRINT  0,"RESET -♦♦BREAK -♦♦
TITLE -♦♦MEMLO -"
010500 ;
010510 ;
010520   LDA #0       ; ZERO VARIABLES
010530   STA TEMP1
010540   STA OPTION
010550   TAX
010560 ;
010570   LDA #8
010580   STA SCRN
010590 PRINTITEM JSR PITEM ; PRINT OPT
ION
010600   LDX TEMP1    ; GET X
010610   INX          ; ADD 1
010620   STX TEMP1    ; SAVE IT
010630   CPX #4       ; ALL DONE?
010640   BNE PRINTITEM ; NO.
010650 ;
010660 STATCON LDA #8 ; CLEAR CONSOL
010670   STA 53279
010680   LDA 53279    ; STATUS CONSOL
010690   CMP #7       ; PRESSED?
010700   BCS STATCON ; BR. IF NOT
010710   TAY          ; SAVE A
010720   LDA #0       ;
010730   STA 53279    ; CLICK
010740   STA 54282    ; WAIT
010750 ;
010760 CN1 LDA #8     ; CLEAR CON
010770   STA 53279
010780   LDA 53279    ; STATUS CON
010790   CMP #7       ; HANDS OFF?
010800   BNE CN1      ; NO, PEOPLE ARE
SLOW
010810 ;
010820   LDA #10      ; 1/6 SEC DELAY
010830   STA 542
010840 CN2 LDA 542
010850   BNE CN2      ; LOOP TIL ZERO
010860 ;
010870   TYA          ; RESTORE A
010880 ;
010890   CMP #5       ; SELECT PRESSED?
010900   BEQ CON1     ; YES
010910 ;
010920   CMP #3       ; OPTION PRESSED?
010930   BNE EXITPAD ; NO, EXIT
010940 ;
010950   JMP OPT
010960 EXITPAD RTS
010970 ;
010980 CON1 LDA 88    ; GET SCREEN ADDR
010990   STA TEMP2
011000   LDA 89
011010   STA TEMP3
011020 ;
011030   LDA SCRN     ; CALC. SCREEN
011040   TAX          ; POSITON
011050 Z1 LDA TEMP2
011060   CLC
011070   ADC #40
011080   STA TEMP2
011090   LDA TEMP3
011100   ADC #0
011110   STA TEMP3
011120   DEX
011130   BNE Z1
011140 ;
011150   LDY #0
011160 Z2 LDA (TEMP2),Y ; GET BYTE
011170   BEQ Z7       ; END OF BLOCK
011180   AND #127     ;  INVERSE OFF
011190   STA (TEMP2),Y ; PUT IT BACK
011200   INY          ; NEXT BYTE
011210   BNE Z2       ; NO
011220 ;
011230 Z7 LDA OPTION ; LAST OPTION?
011240   CMP #3
011250   BNE Z3       ; NO
011260 ;
011270   LDA #0       ; YES LOOP TO FIR
ST
011280   STA OPTION
011290   LDA #8       ; RESET POSITION
011300   STA SCRN
011310   LDA TEMP2
011320   SEC
011330   SBC #240
011340   STA TEMP2
011350   LDA TEMP3
011360   SBC #0
011370   STA TEMP3
011380   LDY #0
011390   JMP Z4
011400 ;
011410 Z3 INC OPTION ; NEXT OPTION
011420   INC SCRN
011430   INC SCRN
011440   LDY #80
011450 ;
011460 Z4 LDA (TEMP2),Y ; GET BYTE
011470   BEQ Z6       ; END OF BLOCK
011480   CLC
011490   ADC #128     ; INVERSE ON
011500   STA (TEMP2),Y ; PUT IT BACK
011510   INY          ; NEXT BYTE
011520   BNE Z4       ; LOOP
011530 Z6 JMP STATCON ; DONE
011540 ;
011550 ; PRINT SELECTED ITEM.
011560 ;
011570 PITEM LDA #0   ; ZERO VARIABLES
011580   STA TEMP2
011590   STA TEMP3
011600 ;
011610   LDA RESET,X ; GET Y VALUE
011620   TAY
011630 ;
011640 ; (X*6)+(Y*2)=JMP ADDR
011650 ; X=OPTION Y=ITEM
011660 ;
011670 ; MULTIPLY X*6
011680 ;
011690 DOX DEX
011700   CPX #255
011710   BEQ DOY
```

```
011720   LDA TEMP2
011730   CLC
011740   ADC #6
011750   STA TEMP2
011760   JMP DOX
011770 ;
011780 ; MULTIPLY Y*2
011790 ;
011800 DOY DEY
011810   CPY #255
011820   BEQ ADDEM
011830   LDA TEMP3
011840   CLC
011850   ADC #2
011860   STA TEMP3
011870   JMP DOY
011880 ;
011890 ADDEM LDA TEMP3
011900   CLC
011910   ADC TEMP2
011920   TAX
011930 ;
011940   LDA TABLE1,X
011950   STA GOPRNT+1
011960   LDA TABLE1+1,X
011970   STA GOPRNT+2
011980 ;
011990 GOPRNT .BYTE $4C,0,0 ; JMP XX
012000 ;
012010 ; PRINT'S TO MAKE TABLE1
012020 ;
012030 00I0  POS  8,8
012040    PRINT  0,"Trap & Re-Run Prog
ram"
012050   RTS
012060 00I1  POS  8,8
012070    PRINT  0,"Reset Normally
  "
012080   RTS
012090 00I2  POS  8,8
012100    PRINT  0,"Cold Start
  "
012110   RTS
012120 ;
012130 ;
012140 01I0  POS  10,8
012150    PRINT  0,"Disabled"
012160   RTS
012170 01I1  POS  10,8
012180    PRINT  0,"Enabled "
012190   RTS
012200 ;
012210 02I0  POS  12,8
012220    PRINT  0,"Off"
012230   RTS
012240 02I1  POS  12,8
012250    PRINT  0,"On "
012260   INC STRIP   ; STRIP CHR OFF
012270    POS  16,0
012280    PRINT  0,"ENTER TITLE:♦♦"
012290    PRINT  0,"[
  ]←←←←←←←←←←←←←←←←←←←←"
012300   LDA #19
012310   STA MAXLINE ; 19 CHR LIMIT
012320   JSR GETLINE ; GET TITLE
012330    POS  16,0
012340    PRINT  0,"□" ; ERASE PROMPT
012350   LDA INLINE  ; TITLE ENTERED?
012360   BNE TI5     ; BR IF SO
012370   DEC STRIP   ; NO TITLE
012380   JMP NULL
012390 ;
012400 TI5 LDY #19   ; TITLE LEN-1
012410   LDA #32     ; SPACE CHR
012420 ;
012430 TI4 STA TLINE,Y ; ERASE OLD TIT
LE
012440   DEY
012450   CPY #255
012460   BNE TI4
012470 ;
012480 ; CENTER TITLE
012490 ;
012500   LDX #10     ; MAX TITLE LEN/2
012510   LDA INLINE  ; TITLE LEN
012520 ;
012530 TI1 DEX       ; DIVIDE TITLE
012540   SEC         ; LEN BY 2
012550   SBC #2
012560   BCS TI1
012570 ;
012580   CPX #11     ; X>10 ?
012590   BCC TI9     ; NO
012600 ;
012610   LDX #0      ; YES
012620 ;
012630 TI9 LDY #0    ; MOVE TITLE
012640 TI3 LDA LBUFF,Y ; GET BYTE
012650   BEQ TI2     ; BR IF END
012660   STA TLINE,X ; SAVE BYTE
012670   INY         ; NEXT BYTE
012680   INX
012690   JMP TI3     ; LOOP
012700 TI2 DEC STRIP ; STRIP CHR ON
012710   RTS         ; DONE
012720 ;
012730 03I0  POS  14,8
012740    PRINT  0,"     "
012750    POS  14,8
012760    PNUM  0,MEMLO
012770   RTS
012780 ;
012790 03I1  POS  16,0
012800    PRINT  0,"NEW MEMLO VALUE IS
:"
012810   LDA #5      ; 5 CHRS MAX
012820   STA MAXLINE
012830 ;
012840   JSR GETLINE ; GET NUMBER
012850   LDA INLINE  ; CHECK LEN
012860   BEQ NULL    ; NOTHING THERE
012870    ASCIINUM  LBUFF,MEMLO ; ASCI
I->FP
012880 ;
012890 NULL LDX OPTION ; NULL ITEM
012900   LDA #0      ; LOOP TO VALID
012910   STA RESET,X ; ITEM
012920   JMP PITEM
012930 ;
012940 TABLE1 .WORD 00I0,00I1,00I2
012950   .WORD 01I0,01I1,NULL
012960   .WORD 02I0,02I1,NULL
012970   .WORD 03I0,03I1,NULL
012980 ;
012990 ;
013000 ;
013010 OPT LDX OPTION ; GET OFFSET
013020   INC RESET,X ; NEXT ITEM
013030   LDA RESET,X
013040   CMP #3      ; INC TO FAR?
013050   BNE Z5      ; BR. IF NOT
013060   LDA #0      ; LOOP TO FIRST
013070   STA RESET,X ; ITEM.
013080 Z5 JSR PITEM  ; PRINT NEW ITEM
013090   JMP STATCON ; CHECK CONSOL
013100 ;
013110 ; FILES
013120 ;
013130 FILES LDA #125
013140   JSR PUTBYTE
013150 ;
013160   LDA #0      ; ZERO DRIVE BYTE
013170   STA DIR+1
013180 ;
013190    PRINT  0,"♦♦♦Which drive? 1-
8"
013200    GET  3     ; GET A KEY
013210   STA DIR+1   ; STORE AT  D#
013220    CLOSE  1
013230    OPEN  1,6,0,DIR
013240   BPL DD1     ; BR. IF SUCCESSF
UL
013250   RTS         ; ERROR, RETURN
013260 ;
013270 DD1  CLOSE  0 ; CLOSE EDITOR
013280    OPEN  0,12,0,"E" ; GR. 0
013290   LDA #0      ; CHANGE COLOR
013300   STA 710
013310 DD3  GET  1   ; GET A BYTE
013320   BMI DD2     ; BR. ON ERROR
013330    PUT  0     ; PRINT BYTE
013340   BPL DD3     ; LOOP
013350 ;
013360 DD2  CLOSE  1
013370    PRINT  0,"♦♦Type any key.♦"
013380    GET  3     ; GET KEY
013390   RTS
013400 ;
013410   *=  736     ; BIN LOAD RUN
013420   .WORD RUN
```

Game Design Workshop

by Craig Patchett

The display list

You probably already know at this point that the display list is something that describes how the screen will be set up. For example, the display list for a graphics mode zero screen tells the computer that there are to be 24 graphics mode zero lines on the screen. In graphics mode one, it tells the computer that there are to be ten graphics mode one lines and four graphics mode zero lines (remember that graphics mode one is a split-screen mode). The unique thing about the display list, however, is that you can set it up any way you choose. That means you can design a custom screen that is a mix of whichever graphics modes you choose, and the ability to do this comes in very handy when you're trying to design a game with a special look.

Let's begin by taking a look at a sample display list, in this case for graphics mode 0:

Display list for GRAPHICS 0 (small text)

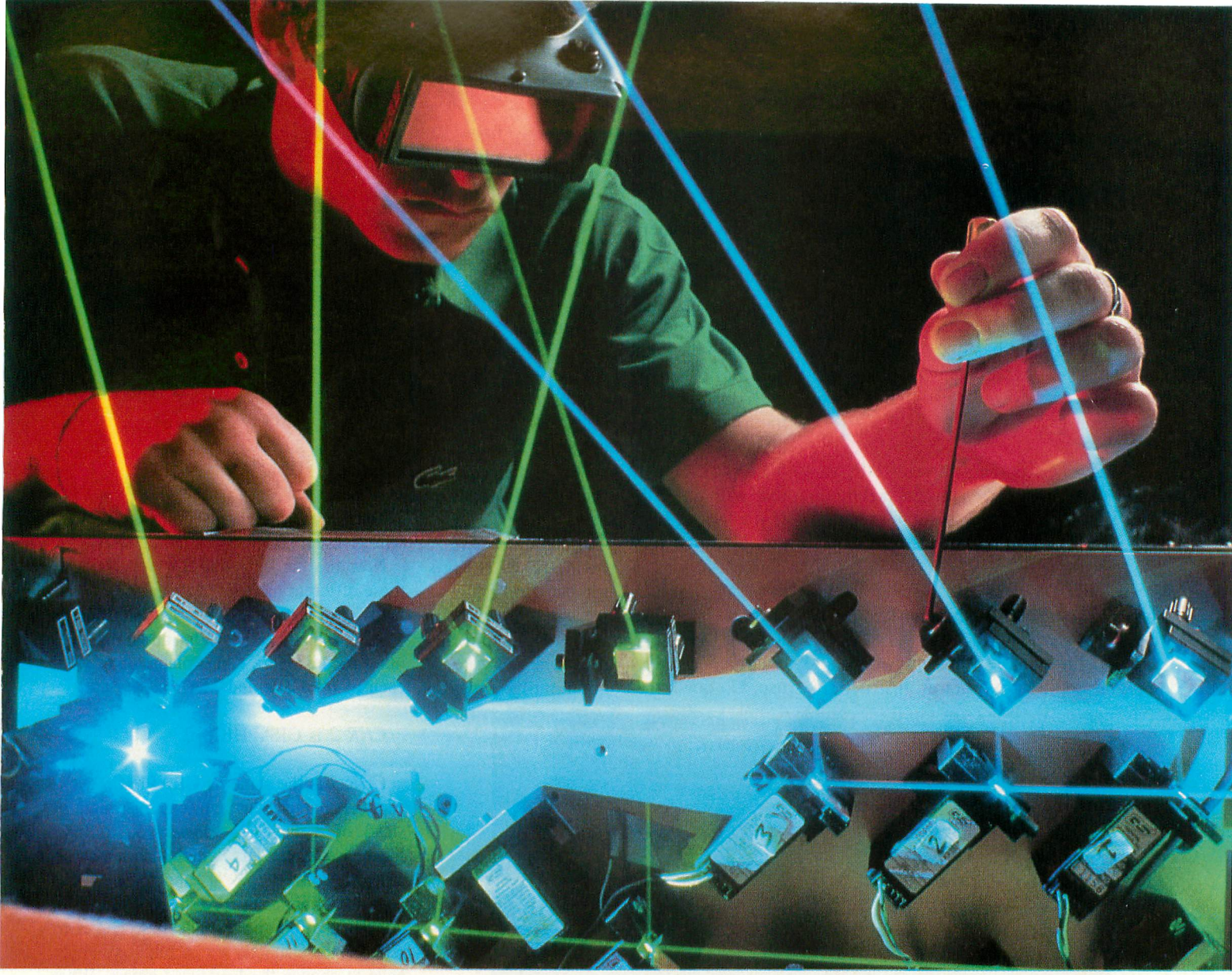
```
BLK 8 —leave eight scan
       lines blank
BLK 8 —leave eight scan
       lines blank
BLK 8 —leave eight scan
       lines blank

CHR 2 LMS — GRAPHICS 0 lines
            with screen memory
            address
LMS LOW NUMBER
LMS HIGH NUMBER
CHR2 ──┐
■      │
■      │  23 instructions total
■      │  =23 GRAPHICS 0 lines
CHR2 ──┘
JVB — go back to the beginning of the
      display list, wait until the end
      of VBLANK, and do it all over
      again!
JVB LOW
JVB HIGH
```

FIGURE 1

If this was the display list that the computer was using, then it would leave the first 24 scan lines blank (because they're not completely visible on the screen) and then put 24 graphics mode zero lines on the screen. Once it was done with that, it would run into the JVB command, which tells it to go back to the beginning of the display list and wait for the end of vertical blank. Why? You'll recall from our previous discussion on the television screen that the screen has to be redrawn every $^1/_{60}$ of a second, and that vertical blank is the time between screens. So by making the display list into a loop that gets executed after every vertical blank, we make sure that screen gets redrawn properly.

You probably noticed that I forgot to explain the LMS after the first CHR2 instruction. LMS stands for "Load Memory Scan," which is just a fancy way of saying "Here's where the screen memory is." In the last column we discussed the need for this feature, and you should make sure that you use an LMS whenever you want to change the address of screen memory. The first line on the screen must have an LMS.



If the next one doesn't, then the computer will just assume that the screen memory for that line comes right after the screen memory for the previous line. In the case of graphics mode 0, each line needs 40 bytes, so the screen memory for the second line will begin 40 bytes after the screen memory begins for the first line.

So, what are the steps to creating a display list? First of all, there are a few simple rules that must be followed. If you want to make sure that the whole screen is visible, the display list should begin with three BLK 8 instructions, like in Figure 1. In some cases, like when you're doing vertical scrolling, you may not need them. Assuming you use them, then the main part of the screen should use a total of 192 scan lines if you want to make sure that the bottom of the screen is completely visible. Again, there is no reason why you can't use more. If you use too many, however, the screen will jump. If this happens, just get rid of a few until it stops jumping. The final rule, before we get going, is that no matter what you do to the display list, no matter how badly it gets screwed up, no matter how bizarre the resulting screen looks, you cannot screw up the computer or your program. If something does go wrong, just press System Reset, and you'll go back to a graphics mode 0 screen with your program intact. With these basic rules in mind, let's now take a look at the display list's instruction set.

BLK n: This instruction is used to leave blank scan lines on the screen. We've already seen one possible use—to leave the top of the screen blank so that the rest will be visible. Blank scan lines are also useful when, for some reason, you're not using part of the screen. The more BLK instructions you use, the faster (just a little) your program will run. Unfortunately, there are very few times when you don't use the entire screen. Personally, I hardly ever use them except at the beginning of the screen.

JMP: This is a jump instruction, the equivalent of BASIC's GOTO. The only reason it's needed is because the display list is not allowed to cross a 1K boundary. What's a 1K boundary? It's a memory location that is a multiple of 1024. When the GRAPHICS instruction is used, the display list is automatically positioned so that it does not cross such a boundary. When you design your own display lists, however, you may run into this problem (apparently not very often, however, since I have yet to do it). If you do, then you should put a JMP instruction right before the boundary. It's a three-byte instruction (one for the instruction and two for the address to jump to), so it should begin at the boundary address minus three, and then should jump to the boundary address. If this sounds kind of silly, it's only because it is.

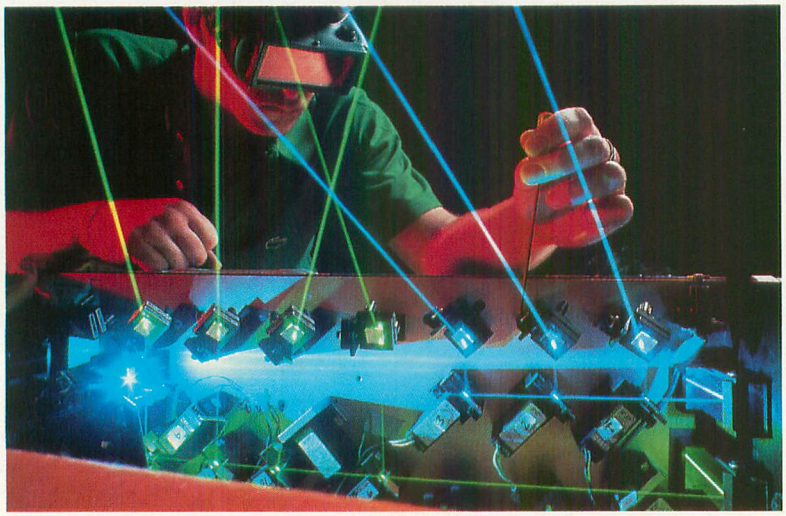
JVB: This is the other jump instruction that we ran across in our example display list above. It tells the computer to go back

to the beginning of the display list and wait until the end of vertical blank before drawing the screen again. Like the JMP instruction, it is also three bytes, with the second two specifying the address of the beginning of the display list. You'll see later exactly how to do this.

CHR: This specifies one of the character modes. For each character mode line on the screen you would use one of these instructions.

MAP n: This is the same as CHR, except it's used to specify a bit-mapped mode (PLOT and DRAWTO) instead of a character mode.

So much for the basic display-list instruction set. You may have noticed a few things that were missing, like the LMS instruction, for example. Well, the LMS is actually a modification, not an instruction.



What this means is that the LMS, along with three other modifications, is added on to the above instructions. So you would have a CHR 2 LMS instruction, for example, or a CHR 6 LMS HSC instruction, which would be a graphics mode 1 line with LMS and horizontal fine scrolling. Here are the four modifications possible:

HSC: This is used to specify a horizontally fine-scrolling line.

VSC: This is used to specify a vertically fine-scrolling line.

LMS: We already know that this is used to specify the address of screen memory. An instruction with the LMS modification must be followed by a two-byte address.

DLI: This is used to specify a line with a display list interrupt at the end of it. We'll get into display list interrupts in next month's column.

You can add any or all of these modifications to the CHR and MAP instructions, but, for obvious reasons, the JMP, JVB and BLK instructions can only have the DLI modification. By this stage, you're probably wondering just how to get from CHR 6 LMS HSC, or whatever, to the number that will get POKEd into the display list. If you look at Figure 2 you will find a chart that gives the values for each of the possible instructions. To use the chart, first go down the left-hand side and find the row with the instruction you want to use. Then look at the top of the chart and find the column with the modifications you want to use. Go down the column until you get to the row you want, and you'll find the correct value. If charts bother you, then there is another way. Each instruction and each modification has its own value. If you take the values of the instruction and the modifications you want and add them all together, you'll get the correct value for the whole thing also. Here are the individual values:

BLK 1:	0
BLK 2:	16
BLK 3:	32
BLK 4:	48
BLK 5:	64
BLK 6:	80
BLK 7:	96
BLK 8:	112

JMP:	1
JVB:	65
CHR n:	n
MAP n:	n
HSC:	16
VSC:	32
LMS:	64
DLI:	128

You can check for yourself that this method will give the same values as the chart.

Now the big question is, "What are all these different modes?" After all, we've already seen that CHR 2 is not graphics mode 2 but graphics mode 0. What about the other other modes? Here's an explanation of what each of the CHR and MAP modes are:

CHR 2 is GRAPHICS 0.

CHR 3 is the same as GRAPHICS 0 except the characters are ten scan lines high instead of eight. This allows you to have lowercase descenders, which means that the tails on "g," "j," "p," "q" and "y" can drop below the line, as they're supposed to. How do you use this mode? The first step is to redefine the character set. Actually, you only have to change the lowercase letters. What will happen is the computer will take the first two bytes of the character description and stick them on the end of the character. It will then make the first two scan lines of the character blank. For non-lowercase characters, it will leave the bytes in order and make the last two scan lines blank.

You should note that because each character is now ten scan lines high, you can only have 19 rows on the screen (192/10). Make sure of this when you change the display list.

CHR 4 lets you use multicolored characters. We mentioned this mode briefly in the column on character sets. CHR 4 characters are the same size as graphics mode 0 (CHR 2) characters. The difference, however, is in the size of the pixels that make up the characters. The pixels are the same height in both modes, but in CHR 4, they are twice as wide as in graphics mode 0. Why? In order to have four colors per pixel, there has to be two bits per pixel rather than one. This means that each character will be four pix-

Although it's true that you have to know machine language to push the display list to its limits, there's more than enough that can be handled from BASIC.

els wide and eight high. While this isn't much use for designing letters (unless you put two characters side by side), it's great for graphics. Regardless of what you end up using this mode for, here's how the computer interprets a character description byte in this mode:

BITS	7 6	5 4	3 2	1 0
USE	PIXEL 1	PIXEL 2	PIXEL 3	PIXEL 4

FIGURE 3

PIXEL VALUE	COLOR REGISTER
00	COLOR4 (background)
01	COLOR0
10	COLOR1
11	COLOR2

Now, there is a way to add one more color to get all five colors on the screen at the same time. If you print a CHR 4 character in inverse video (i.e., the normal character value plus 128), then the pixels with a value of 11 binary will get their color from COLOR3 instead of COLOR2.

Finally, in case it isn't obvious, you should note that the procedure for designing a character set for this mode is exactly the same as that for graphics mode 0, with the exception, of course, that you will now be using two bits per pixel instead of one.

CHR 5 is the same as CHR 4 except the characters are twice as high (the same height as graphics mode two).

CHR 6 is GRAPHICS 1.

CHR 7 is GRAPHICS 2.

MAP 8 is GRAPHICS 3.

MAP 9 is GRAPHICS 4.

MAP 10 is GRAPHICS 5.

MAP 11 is GRAPHICS 6.

MAP 12 is the same as MAP 11 except the pixels are one scan line high instead of two.

MAP 13 is GRAPHICS 7.

MAP 14 is the same as MAP 13 except the pixels are one scan line high instead of two. This mode is sometimes called GRAPHICS 7.5 because it's halfway between GRAPHICS 7 and GRAPHICS 8. We'll be using it in our game, so you'll be able to see it in action.

MAP 15 is GRAPHICS 8.

Whew! Well, that's all the modes that the Atari makes available for you, with the exception of the GTIA modes or GRAPHICS 9, 10 and 11. We'll be covering those modes later in a special column just for them.

Okay, now we know almost everything that we need to know to actually design a display list, so let's add a few small details and then actually do some designing. First of all, you should know that locations 560 and 561 hold the address of the beginning of the display list. (PRINT PEEK(560)+PEEK(561)*256 will give you the decimal address.) Let's see, what else? Oh, screen memory is not allowed to cross a 4K boundary (a memory location that is a multiple of 4096). In the GRAPHICS modes, the only time this happens is in graphics modes 8 through 11, since in these modes screen memory takes up a total of 7680 bytes, well over 4096. In these modes, an extra LMS is added to the line where the offensive boundary is crossed. This is also what you should do if you need screen memory to cross a 4K boundary. Simply add LMS to the line that will cross the boundary and have it point to the first address after the boundary.

I realize that I have presented a lot of information so far without any concrete examples. This is mainly due to the fact that the display list is probably one of the most powerful graphics techniques that the Atari has to offer. Although it's true that you have to know machine language to push the display list to its limits (which is true with all the other techniques as well), there's more than enough that can be handled from BASIC. You've already seen the powers of fine scrolling in the last chapter, and we're now going to see how to go about mixing modes. Before we do, however, you may want to quickly go back to the last month's column and look again at the sections that involved the display list. Now that you have a better understanding of how the display list works, those sections may make a little more sense to you.

We're now ready to get our screen in the format that we decided on earlier in another column.

The first step is to get an initial display list and screen memory. There's no reason why we can't start off from scratch, but as long as there's a way for BASIC to help us out, we may as well take advantage of it by using the GRAPHICS command, since it automatically sets up a display list and screen memory that we can fool around with. The question is, which graphics mode do we use? In the past, most

Display list interrupts (DLIs) are important and powerful features of your Atari computer.

people pick the mode that has the biggest screen memory out of the ones they want to mix. That way, they make sure that there will be enough screen memory for the custom screen. It also means that some of the display list won't need to be changed. This method is fine, but it tends to waste memory. For example, in the screen that we're designing, GRAPHICS 7.5 uses the most screen memory at 7680 bytes. But we're not using that many GRAPHICS 7.5 lines.

We must figure out the total amount of memory that we need for screen memory. We see that it only comes out to 1360 bytes (16*20 +24*40+3*20+1*20), a far cry from 7680. That gives us two options. We can either set up for GRAPHICS 8, which is the same screen memory as GRAPHICS 7.5) and waste the extra memory, which is fine if you've got it to spare, or we can set up for GRAPHICS 6, which uses 1920 bytes for screen memory, much closer to our 1360. The choice is entirely up to you, but I would recommend wasting as little memory as possible, since the less memory your game requires, the more people that will be able to use it. So this will be the first part of our display list redefinition:

```
4000 GRAPHICS 22:POKE 55
9,0:POKE 756,CB+2
```

Remember, this is a change to the program that we've been developing throughout the previous columns. Our next step is to find the display list so we can go about changing it. We already did that in last month's column, but here's the line again, just to refresh your memory:

```
5010 DLIST=PEEK(560)+PEE
K(561)*256
```

Now we're all set to go in and change the display list. Here are the program lines to do:

```
5020 POKE DLIST+3,86
5030 L=PEEK(DLIST+4)+44:
POKE DLIST+5,PEEK(DLIST+
5)+(L>255):POKE DLIST+4,
L-256*(L>255)
5040 FOR X=6 TO 20:POKE
DLIST+X,22:NEXT X:FOR X=
24 TO 50:POKE DLIST+X,14
:NEXT X
5050 MEM7=PEEK(88)+PEEK(
89)*256+600
5060 POKE DLIST+21,78:PO
KE DLIST+23,INT(MEM7/256
):POKE DLIST+22,MEM7-INT
(MEM7/256)*256
5070 POKE DLIST+31,78:PO
KE DLIST+33,INT((MEM7+32
0)/256):POKE DLIST+32,ME
M7+320-PEEK(DLIST+33)*25
6
5080 POKE DLIST+41,78:PO
KE DLIST+43,INT((MEM7+64
0)/256):POKE DLIST+42,ME
M7+640-PEEK(DLIST+43)*25
6
5090 POKE DLIST+51,22:PO
KE DLIST+52,22
5100 POKE DLIST+53,22
5110 POKE DLIST+54,6:POK
E DLIST+55,70:POKE DLIST
+56,PEEK(88):POKE DLIST+
57,PEEK(89)
5120 POKE DLIST+58,65:PO
KE DLIST+59,PEEK(560):PO
KE DLIST+60,PEEK(561)
```

And here is the line-by-line explanation:

5020: This makes the first line a CHR 6 HSC LMS. Remember that the first 24 scan lines are left blank, which requires three BLK 8 instructions. That's why the first CHR 6 line is at DLIST+3 not DLIST.

5030: This is from a previous column, where we decided that we were going to skip over the first 44 bytes of screen memory, using them later to hold the score. If you don't remember why we did this, go back and double-check. In any case, this line sets up the LMS address for the first line on the screen.

5040: Here we set up 15 more CHR 6 HSC lines and 27 MAP 14 lines. Wait a minute, though. Don't we only want 24 MAP 14 lines? And what happened to DLIST+21 through DLIST+23? The reason that things look a little funny at this point is because we want to set up the MAP 14 area in three sections. Remember that the invaders will eventually move far enough down the screen so that they run into the barriers, which are in the MAP 14 section. When they do this, we want to switch the part of the barriers that they run into to a CHR 6 line. We can do that quite simply, but we have to be careful about screen memory, since one CHR 6 line (20 bytes of screen memory) will be replacing eight MAP 14 lines (320 bytes of screen memory). To avoid a problem, we'll put an LMS at the beginning of each group of eight MAP 14 lines (three groups altogether). You'll see this being done in the next few program lines, and you will then understand why we did things in a funny way here.

5050: To make sure that the invaders' screen memory is completely separate from the barriers' screen memory, we'll start the barriers' screen memory 600 bytes past the beginning of the invaders' screen memory. Why 600? Why not! Actually, it could have been anything greater than 444 (20 CHR 6 lines times 20 bytes apiece, plus the initial 44 bytes we skipped over). I just chose 600 to make sure there would be no conflicts.

5060: Now we put the first MAP 14 LMS into the display list. Notice that it fills in the gap we were worrying about earlier.

5070-5080: Here are the other two MAP 24 LMSes. These two go in the middle of the 27 MAP 14s we put in before. The two LMS addresses replace four of the MAP 14s, leaving 23 intact. Add these to the one we set up in Line 5060, and that gives us the 24 that we wanted initially.

Since the DLI has to do whatever it does in an extremely short time, a DLI routine has to be written in machine language.

5090-5100: Here we add the last three CHR 6 HSCs. Did you notice that we didn't use an LMS here? That means that screen memory for these three lines will come right after screen memory for the barriers. Is that what we want? No, but the invaders won't appear in these lines until after the MAP 14s have been replaced with CHR 6 HSCs, like I mentioned earlier. Once this happens, the three CHR 6 HSCs here will be attached to the invader screen memory, as they should be.

5110: We're not done yet. Here we put in the CHR 6 and the CHR 6 LMS for the score line (remember that the address for this LMS is the beginning of the original screen memory).

5120: Now we finish the display list with a JVB back to the beginning. Note that unless you're doing fancy stuff with alternating display lists, this is the way to end any display list.

Okay, now our custom display list is in place and ready to go. Unfortunately, we now run into a few problems. First of all, the computer thinks we're in graphics mode 6. This is no problem at the moment, but it will be later when we try to PRINT the score, and when we try and PLOT and DRAWTO in graphics mode 7.5. (There's no problem with the invaders, since we're moving them directly into screen memory.) How do we tell the computer which graphics mode we want to use? Luckily for us, location 87 exists just to tell the computer what graphics mode is being used. All we have to do is POKE location 87 with the graphics mode we want to use (for graphics mode 7.5, we POKE 87,7 since GRAPHICS 7 is the closest thing to it from BASIC). So much for this "problem."

Our next problem is with screen memory. For example, let's suppose that we hadn't shifted screen memory around and that the score line was at the end of screen memory. What if we wanted to PRINT the score? We could try POSITION 0,23: PRINT #6; "SCORE." Would this work? No, because the computer will print the score 460 (23*20) bytes into screen memory, which would be somewhere in our graphics mode 7.5 area. Because the computer only expects there to be 480 bytes of screen memory in graphics mode one, it will not let you get at the score line. So how do we get around this problem? Locations 88 and 89 point to where the computer thinks screen memory is (there is actually a separate chip called ANTIC that interprets the display list and draws the screen. That's why the computer needs locations 87, 88 and 89). So to print the score, we would change locations 88 and 89 to point to the beginning of the screen memory that the score line uses. Then we can POSITION 0,0 and PRINT #6; "SCORE." Add these lines to our program to see what I mean:

```
5180 MEM1=PEEK(DLIST+4)+
PEEK(DLIST+5)*256:SCRL=P
EEK(DLIST+56):SCRH=PEEK(
DLIST+57)
5190 POKE 87,1:POKE 88,5
CRL:POKE 89,SCRH:POSITIO
N 0,0:? #6;" SCORE:
0"
```

You'll see another example of this next when we draw the barriers.

Display list interrupts

I know, I know, this section was supposed to be about drawing the barriers. So I lied. But I did mention display list interrupts (DLIs) in the last section, and I promised to explain them; so this will be a relatively quick and painless explanation.

Everybody says that the DLI is the most powerful feature of Atari computers. I don't know about "most," but it certainly is powerful. DLIs let you change the screen colors partway down the screen or have more than one character set on the screen at the same time. They let you scroll part of the screen in one direction and the other part in another direction. They let you reposition players partway down the screen, so that one player appears to be two, three and more. Anything else they can do is up to your creativity and imagination.

You already know that a DLI is a modification to a display list instruction. You'll also recall that ANTIC is the chip that is responsible for drawing the screen. Anyway, when ANTIC gets to a display list instruction with a DLI, it first draws that line and then interrupts the 6502 or main chip. The 6502 executes the DLI routine and then goes back to whatever it was doing before the interruption. And that's all there is to a DLI.

Unfortunately, since the DLI has to do whatever it does in an extremely short amount of time, a DLI routine has to be written in machine language. Luckily though, it's usually very simple machine language. For example, here's a DLI routine to change the color of the screen:

```
PHA
LSA #212
STA WSYNC
STA COLBK
PLA
RTI
```

What this does is save the accumulator value, load the new color value into the accumulator, and then store it into WSYNC at location 54282. When any value is stored into WSYNC, the 6502 refrains from doing anything until the next HBLANK (the time between scan lines). Anyway, then the color value gets stored into the background color register at location 53274, the ac-

cumulator is restored to its original value, and we return from the interrupt.

"But," you may be wondering, "isn't the background color register at location 712?" Actually, only sort of. Location 712 is something called a shadow register, which more or less acts like a messenger to the real color register at location 53274. Does this sound complicated? It is a little. You see, during VBLANK, the value in location 712 is transferred to location 53274. Why? Take another look at the above DLI routine. We'll see it in action a little later on, but for now take my word that it will change the background color, so that part of the screen is one color and the other part is another. Think about this. The DLI changes the color partway down the screen, but how does it know to change it back at the beginning of the next screen? It knows from the shadow register. If it wasn't for the shadow register, you would have to have a DLI at the top of the screen as well. In any case, just remember to change the hardware registers in a DLI, not the shadow registers.

How does the computer know where the DLI routine is? Locations 512 and 513 hold the address of the DLI routine. So to get a DLI going, you would change the display list, store the routine somewhere safe in memory (somewhere that doesn't shift around, which means you can't store it in a string like other routines), set locations 512 and 513, and then POKE 54286, 192 to turn the DLI on (POKE 54286, 64 will turn it off again). That's all there is to it. Actually, this would be a good time to add our sample routine above to our invaders program and prove to you that it isn't that difficult; so why don't you try adding the following lines to our program:

```
3100 FOR BYTE=0 TO 10:RE
AD DAT:POKE 1536+BYTE,DA
T:NEXT BYTE
3110 DATA 72,169,212,141
,10,212,141,26,208,104,6
4
5100 POKE DLIST+53,150:P
OKE 512,0:POKE 513,6:POK
E 54286,192
```

Here's an explanation of what you just did:

3100: This just reads in the data for the routine and stores it in page 6, where it will be safe.

3110: Here's the data. We'll be taking a closer look at this in a minute, showing you how you can change it for your own use.

5100: This adds a DLI modification to the display list, sets up the DLI routine address, and turns on the DLI.

That wasn't too tough now, was it? But, of course, all it does is change the color. What if we wanted more than one DLI? Can we do that? Well, yes, but it starts to get a little more complicated. The problem with having more than one DLI is that we have to update locations 512 and 513 each time we change routines. For example, the first routine would have to update the locations to point to the second routine, the second to point to the third, and so on up to the last routine, which would have to update the locations to point back to the first routine.

Not too difficult, right? (Assuming you know some machine language.) Right, but this limits what you can do during the DLIs since there is not much time available. Actually you really can't do much more than change a few locations before you run out of time. If you decide to get adventurous and design your own DLIs, and if you try to do a lot of stuff during them, and if the screen looks kind of funny once they're working, then you have probably used up too much time. In other words, don't expect a DLI to do too much. Oh, and DLIs are best suited for making changes to the screen. Save everything else for your regular program.

Okay, we've now gotten all of the annoying little details out of the way; so let's take a quick look at how you would make changes to the DLI routine I've given you, assuming you want it to do something different. First of all, here's how the numbers in the Line 3110 correspond to the assembly language:

```
      72  PHA
  169 212  LDA #212
141  10 212  STA WSYNC
141  26 208  STA COLBK
     104  PLA
      64  RTI
```

What good does this do you? Well, if you wanted to change to a different color, you could just change the first 212 in Line 3110 to something else. Try it. Also, suppose you wanted to change character sets instead of colors. You would change the first 212 to the page address of the second character set, and then you would change COLBK (26, 208) to CHBASE (9, 212). As you can see, this simple DLI routine can be changed by you in a number of ways to get the specific result you want; so don't be afraid to play. I should warn you, however, to save the program before trying a new DLI. If you did something wrong by mistake, there is a chance that you lock up the machine. If you do, and System Reset

			HSC VSC LHS	HSC VSC LHS	HSC VSC LHS	HSC VSC LHS		HSC VSC LHS	HSC VSC LHS	HSC VSC LHS	HSC VSC LHS		HSC VSC LHS	HSC VSC LHS	HSC VSC LHS
								DLI	DLI	DLI	DLI	DLI	DLI	DLI	DLI
BLK	1	0						128							
BLK	2	16						144							
BLK	3	32						160							
BLK	4	48						176							
BLK	5	64						192							
BLK	6	80						208							
BLK	7	96						224							
BLK	8	112						240							
JMP		1						129							
JVB		65						193							
CHR	2	2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
CHR	3	3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
CHR	4	4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
CHR	5	5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245
CHR	6	6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
CHR	7	7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
MAP	8	8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
MAP	9	9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
MAP	10	10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
MAP	11	11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
MAP	12	12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
MAP	13	13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
MAP	14	14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
MAP	15	15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

FIGURE 2

doesn't help, just turn the computer off and back on again, and load back the program so you can figure out what went wrong.

Bit-mapping (Part 1)

This is probably going to be one of the easiest sections in this column, because bit-mapping is a relatively simple technique—at least it is when you're using it for stationary objects, as we are with the barriers. You'll recall that bit-mapping can also be used for simple animation, in which case things can get a little more complicated and also extremely slow. For BASIC games, you're better off only using bit-mapping for non-animated objects.

The first step for bit-mapping is the same as that for character sets and for player/missile graphics, deciding what the object is going to look like. Figure 2 is our barrier shape.

Drawing the barriers

The next question is: How do we draw this quickly and easily? Actually, it should be how do we draw these, since there are supposed to be four barriers, not just one. Well, let's look at the methods available to us. We could PLOT each of the points in the barriers, but that would obviously take too long in this case. We could use PLOT and DRAWTO and draw them line by line. This is better, but let's see what else there is. We could use PLOT, DRAWTO and X10 18 (FILL). This is faster still, but doesn't look quite as neat. Finally, we could store the data for the barriers in a string and then use MOVMEM to transfer this data directly into screen memory. This is by far the quickest method, but we're talking here about 40*24=960 bytes of screen memory, and therefore 960 bytes of string space. Keeping memory requirements down is important, so we'll stay away from this method. What's our final decision then? Well, it comes down to PLOT and DRAWTO or PLOT and DRAWTO with FILL. I'd like to try for a technique that gives the effect of building the barriers, so, as I explained earlier, I personally want to use the visual effect that PLOT and DRAWTO give. There's no reason, however, why FILL could not be added if that's your own personal preference.

Our next step is to figure out the easiest way to use our chosen technique. We're obviously going to want to use some FOR/NEXT loops, since there is a lot of repetition in the barrier shape (and there are also four identical barriers). Let's break the barrier up into groups of horizontal rectangles, since a PLOT and DRAWTO FOR/NEXT loop is good for drawing rectangles. Figure 3 shows our barrier shape broken up this way.

Now we have what we need to start programming. What we'll do is set up three nested FOR/NEXT loops. The outside loop will count off the ten rectangles, the next will count off the horizontal lines in each rectangle, and the inside loop will count off the four barriers. We'll store the vital information about each rectangle in a DATA statement so that the loops can get to it easily.

If I told you right now to try and write the loops I described by yourself, how could you do it? Would you jump right in and start work on the first loop? If you would, then you forgot all about what we discussed at the end of the previous section. (Don't feel too badly, I'm writing this paragraph because I forgot also!) Now do you remember? Because we're using a custom display list, we have to tell the computer where our screen memory is and what mode we're using. The following line will do that for us:

```
5210 POKE 87,7:POKE 89,I
NT(MEM7/256):POKE 88,MEM
7-PEEK(89)*256:COLOR 3
```

Remember that location 87 tells the computer the BASIC mode number, and locations 88 and 89 tell it the location of screen memory. If you're using a graphics mode that doesn't have a BASIC equivalent (as we are here), then choose the BASIC mode that is closest to it.

Okay, with that out of the way, we can now go ahead and write our loops. Here are the lines to add to our program:

```
5210 POKE 87,7:POKE 89,I
NT(MEM7/256):POKE 88,MEM
7-PEEK(89)*256:COLOR 3
5230 RESTORE 5260
5240 FOR X=1 TO 10:READ
N,XS,Y,XE:FOR T=N-1 TO 0
 STEP -1:FOR Z=0 TO 3:PL
OT Z*40+9+XS,Y+T:DRAWTO
Z*40+9+XE,Y+T
5250 NEXT Z:NEXT T:NEXT
X
```

```
5260 DATA 4,16,20,20,4,1
,20,5,2,15,18,20,2,1,18,
6,2,14,16,20,2,1,16,7,10
,1,6,20,2,2,4,19,2,3,2,1
8,2,4,0,17
```

And, of course, the explanation:

5230: Any time you have a program with more than one set of DATA statements, it's a good idea to RESTORE the first line of the DATA you want to use. Why? When BASIC encounters a READ statement, it will look for the DATA that comes after whatever it read last. It will not look for the DATA closest to the READ statement. So, since you'll be reading data out of order often, the RESTORE becomes a necessity.

5420-5250: This is our loop. X keeps track of which rectangle we're drawing. N is the number of lines in the current rectangle, XS is the starting X position for the rectangle, Y is the starting Y position, and XE is the ending X position. T keeps track of which line we're drawing within the current rectangle, and Z keeps track of which barrier we're working on. Keeping all this in mind, you should be able to figue out what's going on here.

5260: This is the data for the rectangles. There are four numbers for each rectangle (N, XS, Y and XE), the meanings for which were discussed above.

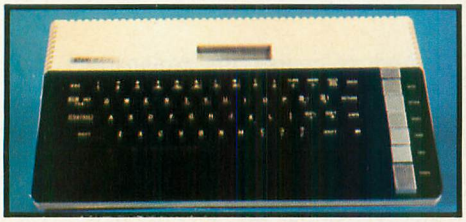
Well, that's about it. (I told you this was easy.) Incidentally, just in case you decide that you prefer to use FILL, here's a brief introduction to using X10 18:

1. PLOT a point at the lower left-hand corner of your object.
2. DRAWTO the upper right-hand corner.
3. DRAWTO the upper left-hand corner.
4. POSITION the cursor at the lower right-hand corner.
5. POKE location 765 with the number of the color register you want to FILL with. (This number is the same as that you would use with the color command.)
6. Now do X10 18,#6,0,0,"S:".

There is one problem with FILL; the area that you're FILLing has to be empty. If there are any points already turned on within the area, FILL won't work correctly.

When you want to talk Atari

XL/XE HARDWARE



CMO SPECIAL
Atari 800XL **$89.99**

65XE 109.00
130XE 149.00

INTERFACES
ICD
P:R Connection 61.99
Printer Connection 41.99
Supra
1150 . . . . . 39.99
1151 (1200 XL) 40.99
Xetec
Graphix Interface 38.99
Atari
850 Interface 109.00

XL/XE ENHANCEMENTS
Axlon 32K Mem. Board (400/800) . 19.99
Atari 80 Column Card 79.99
ICD
BBS Express (ST) 52.99
Sparta DOS Construction Set 28.99
US Doubler/Sparta DOS 47.99
Real Time Clock 48.99
Rambo XL 29.99
US Doubler 28.99

MODEMS
Atari
SX212 300/1200 (ST) 89.99
XMM301 . . . . . 44.99
Anchor
VM520 300/1200 ST Dir. Con 119.00
Avatex
1200 HC 89.99
2400 . . . . . 169.00
Supra
2400 Baud XL/XE or ST 169.00
2400 Baud (no software) 149.00

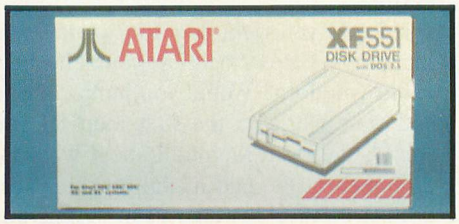
MONITORS
Magnavox
CM 8505 14" Composite/RGB/TTL . . . 199.00

ST HARDWARE

Call For Current Information
On The Entire ST Line!

ATARI SM1224 RGB/Color Monitor $329

520ST FM RGB/Color System . . . 789.00
SM124 Monochrome Monitor 179.00



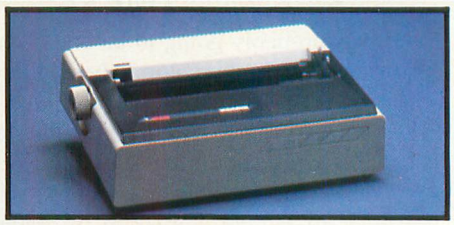
CMO PACKAGE EXCLUSIVE
Atari 800XL & XF551 Drive
w/5 Undocumented ROMS Asteroids, Defender, Missile Command, QIX, Star Raiders **$279**

DRIVES

Atari
ST 314 DS/DD 219.00
XF551 Drive (XL/XE) 179.00
SHD204 20 Meg Hard Drive 599.00
I.B.
5¼" 40 Track (ST) 219.00
5¼" 80 Track (ST) 279.00
I.C.D.
FA•ST 20 Meg 629.00
FA•ST 30 Meg 869.00
FA•ST Dual Hard Drives . . . . . Call
Indus
GTS 100 3½" DS/DD (ST) 199.00
GT 1000 5¼" DS/DD (ST) 209.00
GT Drive (XL/XE) 189.00
Supra
FD-10 10MB Removable Floppy . 889.00
30 Meg Hard Drive (ST) 689.00

CALL FOR DISKETTE SPECIAL

PRINTERS



Atari XDM121 LQ (XL/XE) $189

Atari
1027 LQ XL/XE 129.00
XM-M801 XL/XE Dot Matrix 199.00
XM-M804 ST Dot Matrix 199.00
Brother
M-1109 100 cps Dot Matrix 169.00
M-1509 180 cps Dot Matrix 389.00
HR-20 22 cps Daisywheel 339.00
Citizen
120D 120 cps Dot Matrix 149.00
180D 180 cps Dot Matrix 179.00
Premier-35 35 cps Daisywheel . . 549.00
Epson
LX-800 150 cps, 80 col 189.00
FX-850 264 cps, 80 col. . . . . Call
LQ-500 180 cps, 24-wire . . . . . Call
LQ-850 330 cps, 80 col. . . . . Call
NEC
P2200 pinwriter 24-wire 379.00
Okidata
Okimate 20 color printer 129.00
ML-182 +120 cps, 80 column . . 229.00
ML-390 +270 cps, 24-Wire 539.00
Panasonic
KX-P1080i 144 cps, 80 col 169.00
KX-P1091i 194 cps, 80 col 199.00
Star Micronics
NX-1000 140 cps, 80 column . . . 179.00
Toshiba
P321-SL 216 cps, 24-wire 499.00

ACCESSORIES

Allsop Disk Holders
Disk File 60-5¼" 9.99
Disk File 30-3½" 9.99
Curtis
Emerald 39.99
Safe Strip 19.99
Universal Printer Stand 14.99
Tool Kit 22.99

WE SHIP 90%
OF ALL ORDERS
WITHIN 24 HOURS

SELECT FROM
OVER 3000
PRODUCTS

COMPUTER MAIL ORDER

.....you want to talk to us.

SPECIALS XL/XE

#AAB822P 822 Printer Paper.....$3.49
#AA14746 T.V. Switch Box.....3.49
#AA4010 Tic-Tac-Toe.....5.49
#AA4011 Star Raiders.....5.49
#AA4012 Missile Command.....5.49
#AA4013 Asteroids.....5.49
#AA4022 Pac Man.....5.49
#AA4025 Defender.....5.49
#AA4027 QIX.....5.49
#AA4102 Kingdom (Cass.).....1.99
#AA4112 States & Capitals (Cass.).1.99
#AA4121 Energy Czar.....1.99
#AA4123 Scram (Cass.).....1.99
#AA4126 Speed Reading.....2.99
#AA4129 Juggle's Rainbow.....1.99
#AA415 File Manager.....9.99
#AA4204 1020 Color Pens.....1.99
#AA5047 Timewise (D).....3.99
#AA5049 Visicalc (D).....24.99
#AA5050 Mickey Outdoors.....5.99
#AA5081 Music Painter (D).....9.99
#AA6006 Counseling Procedure....1.99
#AA7102 Arcade Champ (No J. Stk)6.99
#AA8030 E.T. Phone Home.....3.99
#AA8048 Millipede.....4.99

CLOSEOUTS XL/XE

ROM CARTS (XL/XE)
$3.49 ea or 5 for $14.99

Loose/Undocumented
Choose from: Space Invaders, Star Raiders, Missile Command, Asteroids, Pac Man, Galaxian, Defender, QIX, Super Breakout, E.T., Eastern Front, Robotron.

Rocklyn
Gorf.....3.49
Anti-Sub (Disk).....3.49
Journey to Planet.....3.49
Atari Program Exchange
10 Different Cassettes For.....$11.99

SPECIALS XL/XE

Access
Leaderboard Golf.....13.99
Accolade
Hardball.....19.99
Atari
Atariwriter Plus.....35.99
Broderbund
Printshop.....26.99
Datasoft
Alternate Reality (City).....23.99
221 Baker St.....20.99
Electronic Arts
Auto Duel.....23.99
Firebird
Silicon Dreams.....19.99
Jewels of Darkness.....19.99
Microprose
Top Gunner.....16.99
F-15 Eagle Strike.....22.99
Origin Systems
Ultima 4.....36.99
Strategic Simulations
Gemstone Warrior.....11.99
Sublogic
Flight Simulator.....32.99

ST SOFTWARE

Access
Leaderboard Golf.....22.99
Accolade
Pinball Wizard.....21.99
Activision
Hacker II/Music Studio (ea.).....28.99
Antic
CAD 3-D.....29.99
Avant Garde
PC Ditto.....59.99
Batteries Included
Degas Elite.....37.99

Data East
Speed Buggy **$25.99**

ST SOFTWARE

Epyx
Dive Bomber.....29.99
Firebird
Jewels of Darkness.....19.99
The Sentry/Tracker (ea.).....12.99
FTL
Dungeonmaster.....29.99
Metacomco
ISO Pascal.....59.99
Michtron
Leatherneck.....29.99
Microprose
Gunship.....28.99
F-15 Strike/Silent Service (ea.)...24.99
Miles Software
ST Wars.....24.99
Mindscape
Road Runner.....35.99
Mark of the Unicorn
PC Intercom.....79.99
Mark Williams
C.....119.00
Paradox
Wanderer (3D).....24.99

ST SOFTWARE

Progressive Computer
Graphic Artist 1.5.....119.00
Psygnosis
Obliterator.....29.99
Soft Logik Corp.
Publishing Partner.....54.99
Strategic Simulation
Questron II.....35.99
Sublogic
Flight Simulator II.....33.99
Timeworks
Swiftcalc/Wordwriter.....(ea.) 45.99
Desktop Publisher.....79.99

Word Perfect **$179**

In the U.S.A. and in Canada

Call toll-free: 1-800-233-8950

Outside the U.S.A. call 717-327-9575, Fax 717-327-1217
Educational, Governmental and Corporate Organizations call toll-free 1-800-221-4283
CMO, 101 Reighard Ave., Dept. B7, Williamsport, PA 17701

MEMBER MMC MICROCOMPUTER MARKETING COUNCIL of the Direct Marketing Association, Inc.

OVER 350,000 SATISFIED CUSTOMERS • ALL MAJOR CREDIT CARDS ACCEPTED • CREDIT CARDS ARE NOT CHARGED UNTIL WE SHIP

POLICY: Add 3% (minimum $7.00) shipping and handling. Larger shipments may require additional charges. Personal and company checks require 3 weeks to clear. For faster delivery, use your credit card or send cashier's check or bank money order. Credit cards are not charged until we ship. Pennsylvania residents add 6% sales tax. All prices are U.S.A. prices and are subject to change, and all items are subject to availability. Defective software will be replaced with the same item only. Hardware will be replaced or repaired at our discretion within the terms and limits of the manufacturer's warranty. We cannot guarantee compatibility. All sales are final and returned shipments are subject to a restocking fee. We are not responsible for typographic or photographic errors.

B712

CIRCLE #108 ON READER SERVICE CARD.

STAY ON THE CUTTING EDGE!

12 Issues Only $28

Save $14 Off The Cover Price!

12 Issues *With Disk* Only $79

New Lower Price!

SUBSCRIBE TO ST-LOG AND SAVE!

The world of ATARI-ST continues to grow by leaps and bounds and ST-LOG is there every step of the way! SUBSCRIBE and SAVE today!

☐ 12 Issues $28 (MCLYW) ☐ 12 Issues with Disk $79 (DCLYW)

☐ PAYMENT ENCLOSED ☐ BILL ME

CHARGE MY: ☐ VISA ☐ MC #_____

EXPIRATION DATE SIGNATURE

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615.

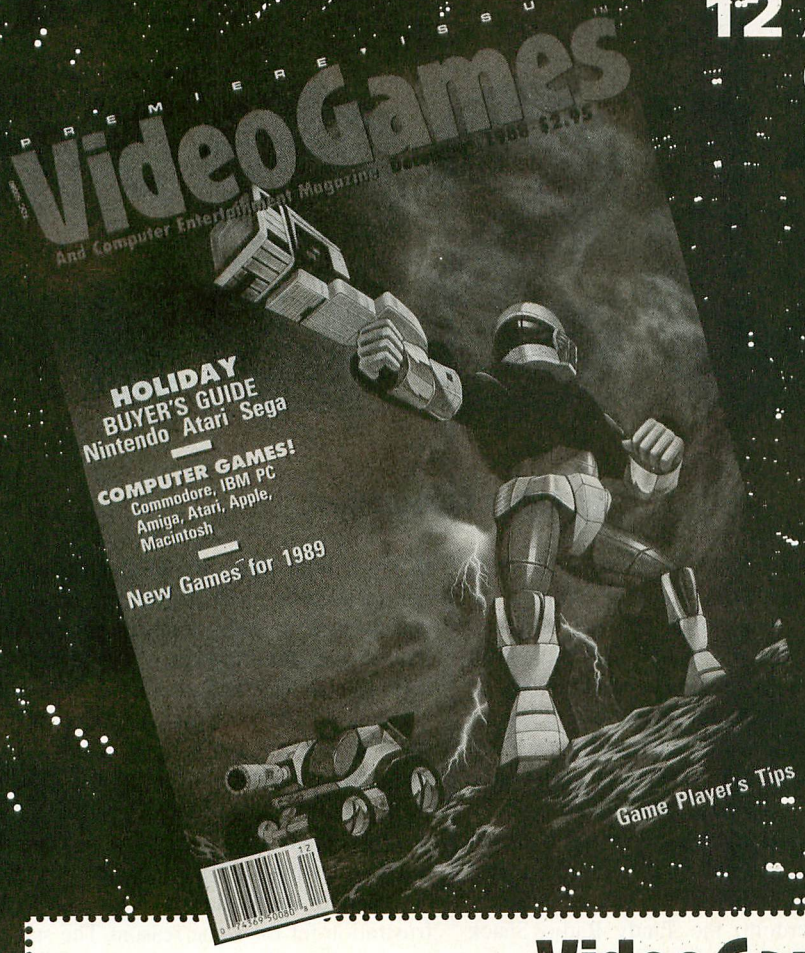
Offer expires March 29, 1989.

ST LOG
THE ATARI ST MONTHLY MAGAZINE

INTRODUCING

VideoGames

And Computer Entertainment Magazine



12 ALL COLOR ISSUES ONLY $19.95

Save over $15 off the cover price!

- GAME REVIEWS
- ARCADE ACTION
- STRATEGY GUIDES
- TECHNICAL REPORTS
- COMPUTER SOFTWARE

VideoGames
And Computer Entertainment Magazine

P.O. BOX 16927, N. HOLLYWOOD, CA 91615

☐ **Yes!** Sign me up for 12 issues for only $19.95—I'll save over $15!

☐ Payment Enclosed — Charge My ☐ VISA ☐ MC

_____ EXP_____

SIGNATURE_____

NAME_____

ADDRESS_____

CITY_____ STATE_____ ZIP_____

Your first issue will arrive in 6 to 8 weeks.
WATCH FOR IT!!
Offer Expires 3/1/89 .

Money back on unused subscriptions if not satisfied!
Foreign—add $7 for postage.

CIYYA

End User

The fall COMDEX in Las Vegas will have just concluded and as of now, big things are promised there. The future of Atari in the U.S. is being formed right now.

by Arthur Leyenberger

In the June 1988 "End User," I discussed my experiences using a Zenith Z-181 laptop computer on the road, and later back home to transmit files to an 8-bit Atari or ST. You may recall that I liked the machine; in fact it was (and still is) one of the best laptop computers currently available. You may also recall that I ultimately found the "11-pound Zenith Z-181" to be too heavy for comfortable computing on the go.

I then proceeded to wax poetic about what I consider to be one of the best kept secrets in computerdom: the Tandy Radio Shack Model 102 laptop computer. If you want to primarily do some writing while on the move, the Radio Shack Model 102 has become the standard to which other laptop computers are compared. It is used by hundreds, maybe thousands, of journalists, students and all types of people for portable applications such as writing and telecommunicating.

Chief among its strengths are its minimal weight (only three pounds), ease of use and moderate cost (about $600 for a maxed-out 32K-byte memory machine and separate portable disk drive). However, despite the relative ease with which the Model 102 allows you to become productive while traveling, you do have to make some sacrifices. The most significant drawback to the Model 102 is its 40-character by eight-line LCD screen.

Doing serious work on a computer with a small screen, often in a dimly lit room, is a real eyestrain. The eight lines of text aren't really enough to see previous material while you are composing and writing new work. Further, using a spreadsheet program designed for the Model 102 is an exercise in frustration for the same reason: The screen is not the standard 80 characters by 20 lines that most computers now use. (Even the 8-bit Atari has had 80-column cards, emulators and software for many years.)

My point in mentioning laptop computing in conjunction with using your ST or 8-bit Atari is to tell you about a new computer that has stolen first place in the laptop sweepstakes, as far as I am concerned. It's the Toshiba T-1000 MS-DOS laptop computer. It has one built-in, 720K, 3½-inch floppy-disk drive, 512K of RAM (Random Access Memory) and is a true PC-compatible com-

puter. Best of all, it weighs in at only 6½ pounds, not much more than my Model 102 and separate disk-drive system.

The T-1000 has MS-DOS in ROM (Read Only Memory), so that MS-DOS does not take up any valuable space in RAM. It has a fold-down, supertwist LCD screen that shows 80 columns by 20 lines. You can also add a 768K memory card that can be configured for expanded RAM, a RAMdisk and a number of other ways.

The thing that makes this computer so exciting is the ease with which you can transfer files to the Atari ST. The 3½-inch disks that the T-1000 uses are identical to those the ST uses. It gets even better, though. As you may know, the ST can read MS-DOS disks directly. For example, many users have an external 5¼-inch disk drive connected to their ST with which they can read MS-DOS files. A number of programs such as *WordPerfect, VIP Professional* (a Lotus 1-2-3 clone) and *Easy Draw* (.GEM files) that run on the ST can use files created by similar MS-DOS programs on a PC.

To transfer files from the Toshiba T-1000 to the ST, all you do is insert the MS-DOS floppy disk in the ST drive. Using the GEM Desktop, you can easily copy any file from it to an ST disk. Just remember, any file can be copied, but MS-DOS programs will not run on the ST (unless you are using an emulator program such as *pc-ditto*, but that's a topic for another column). If you want to transfer files from the Toshiba to an 8-bit Atari, you'll need a terminal program running on both computers and have to use a transfer protocol such as Xmodem. (See the June 1988 "End User" column for detailed instructions for uploading and downloading files to and from Atari and laptop computers.)

The street price of the stock Toshiba T-1000 is about $700. As an MS-DOS computer with one built-in drive, 512K of RAM, five hours of use per battery charge and the MS-DOS operating system contained in ROM, the T-1000 is an excellent machine. In fact, $700 is not much more than the cost of a Radio Shack Model 102 with full memory and a separate disk drive.

I have been using a T-1000 for about a month now and highly recommend it.

Second-class users

One of the continuing issues that Atari users have to wrestle with is the lack of responsiveness and support we get from Atari. I won't complain again about the "game image," but related to that problem is the lack of emphasis that Atari seems to be placing on the United States market. There has not been any major advertising for the STs since 1986. Things have gotten so bad that Atari could probably now advertise a stealth computer (the ST) without worrying about false advertising.

Equally important to the success of the ST as advertising is the availability of new products. Developers see no advertising for the ST so they figure the market is stagnant or nonexistent. Atari attempted to court developers in the early days of the ST by selling them $5,000 ST systems while Apple was *giving* Macs to their developers. Further, the few developers left must continue to deal with the poor support, documentation and programming tools supplied by Atari.

There are plenty of other confusing and inconsistent actions on the part of Atari that continue to irritate existing as well as potential ST owners. I have mentioned before, for example, the lack of upgradability of the ST computer. Despite Atari's claim since the very introduction of the product that they would support the ST and its user, one still has to buy from third-party developers memory upgrades that void your warranty. Atari's alternative for a memory upgrade is to buy a new machine.

Whatever happened to the blitter chip that was supposed to improve the graphics on the ST tenfold? It first was going to be an upgrade to existing 520 and 1040 STs, then it wasn't. Now it appears that the blitter is not anywhere near as good as it was originally touted as and for most people the attitude is "Who cares?"

Problems exist too. It has taken a long time for a new version of TOS in ROM to appear. Atari claims it has been working on it and that we should see the new TOS ROMs this fall. Have you seen them?

A variety of other things continue to plague the Atari market as well. When a new product like the 520STFM with an internal double-sided drive is introduced, some stores get the product before others. In fact, some dealers claim that Atari's own Federated Electronic Stores get the new version of the product first, while the existing supply of the old version gets distributed to everyone else.

Apparently, Canada also gets first chance at some new Atari products. The PCF-554 5¼-inch disk drive originally designed for the PC1 (Atari's PC clone) works with Megas and STs and has been available in Canada since late August. If you live up north, you have also had available a math coprocessor board for the Mega, CD-ROM developer kits and the Atari PC Clone for several months.

I recently asked Neil Harris of Atari Corp. about some of these problems. He agreed that communications was a big issue and has hired someone to produce a series of dealer and developer newsletters. In addition, a user group and a MIDI (Musical Instrument Digital Interface) newsletter are also being published.

Neil also said that Atari has hired a new public relations agency to get the word about Atari ST computers and products to the press. I agree that a PR campaign to industry insiders is equally as important, if not more so, than a consumer advertising campaign. It seems to me that having a consumer ad campaign alone is trying to do the job in a vacuum. People need to see Atari products mentioned in other than Atari-specific magazines.

Editorial coverage can certainly supplement good advertising. Sam Tramiel, president of Atari Corp., has previously promised "a major ad campaign in the fourth quarter of 1988."

Neil also told me that a dealer council has been established, consisting of 14 of the best dealers nationwide. They meet regularly with top Atari officials to air their views and hear about forthcoming products and programs. In addition, Atari is continuing to put special promotions together to help dealers move products.

(Ed: As this issue was going to print, Neil Harris announced that he was leaving Atari after four years with them. He is taking a position in the Marketing Department of the GEnie telecommunications service.)

It seems clear that Atari is attempting to address some of their problems. However, many Atari users have the perception that the top brass of Atari are either unaware of the issues that affect the user or just don't care. It is only by actions that we can judge how Atari Corp. feels.

By the time you read this, we should know the answer. The promised ad campaign should have happened and hopefully been a success. There should be more dealers, not to mention happier dealers thanks to Atari's efforts. The fall COMDEX in Las Vegas will have just concluded and, as of now, big things are promised there. The future of Atari in the U.S. is being formed right now.

Those that would accuse me of "Atari bashing" have not been paying attention. Most ST users would like to hear some good news for a change. It is really up to Atari, and has been since the Tramiels took over. We who are deeply involved with the Atari community all want the same thing: for Atari to be successful. Isn't that obvious?

Déjà vu all over again

David vs. Goliath. You know how the story goes: big multimegabuck company vs. the small start-up company. For the last couple of years, Apple Computer has played the role of Goliath without a corresponding David.

You may recall how Apple threatened a lawsuit against Digital Research Inc., makers of the GEM Desktop and GEM application programs. As a result, DRI was forced to redo the "visual look" of the GEM Desktop for the PC. Consequently there were no more disk or trash-can icons, no more resizable windows (up to four) and no more ease of use.

The "new" GEM Desktop consists of two fixed windows covering the entire screen, with all functions having to be chosen from the drop-down menus. (For you PC GEM users, here's a tip: The GEM Desktop is nothing more than another application program. So, by installing the latest version of GEM 3.0, and then copying certain files from the v1.0 Desktop [DESKHI.ICN, DESKLO.ICN, DESKTOP.APP, DESKTOP.RSC AND DESKTOP.INF], you can have the functionality of the original Desktop with the latest version of GEM.)

Atari has hired a new public relations agency to get the word about Atari ST computers and products to the press.

More recently, Apple filed a major lawsuit against Microsoft, the largest supplier of software for the Macintosh. The suit claims that Microsoft Windows for the PC infringed on Apple's copyrighted Macintosh user interface. There is a subplot going on concerning a 1985 contract between Apple and Microsoft. In this previously confidential agreement, Microsoft acknowledges that the visual displays of Windows 1.0 and five other Macintosh programs are "derivative" of the Macintosh. In return, Apple granted Microsoft ". . . license to use these derivative works in present and future software programs. . . ."

The case between Apple and Microsoft may turn out to be nothing more than a contract case. However, Apple alleges that it is a copyright case. If true, we could finally get to the root of the Macintosh visual-interface controversy in court. That is, that Apple created the Macintosh interface based in some part on the work at the Xerox Palo Alto Research Center (PARC) and from the Stanford Research Institute.

Here's where David comes in. Apple also sued Hewlett-Packard for copyright infringement of the Macintosh interface. They claimed HP's MS-Windows-based New Wave applications environment infringed on Apple's graphical user-interface copyrights (called "audiovisual works" in the suit). I have not seen HP's New Wave but I have heard it outdoes the Macintosh in its power and ease of use.

I never thought I would describe Hewlett-Packard as a David to Apple's Goliath but the title fits since HP is countersuing Apple, alleging unfair business practices and antitrust violations. Specifically, HP charged that "Apple's copyrights are invalid because the concepts that the audiovisual works are based on are derivative of work done by Xerox Corp.; are, if protectable, under the domain of patent law not copyright law; were misrepresented by Apple before the U.S. Copyright Office and the public; and have been used as instruments of monopolization."

Ain't that a mouthful?

Who knows when the litigation will actually appear before the court. But I hope it does come to court so that the issue of graphical user-interfaces can be settled, hopefully once and for all. Neither Apple or any other company should singlehandedly prevent the state of the user-interface art from improving. As a computer user, I have every right to expect that free competition will take place in the market, and that the result will be an improved product.

The GEM Desktop, for instance, is not perfect but it sure is a step in the right direction for an easy-to-use computer interface. When Apple forced DRI to change the Desktop, the result was inferior. That is certainly not progress and represents a sham on Apple's part.

As a holiday wish, let me hope that Apple and Hewlett-Packard have their day in court and that HP wins. And, as a result, companies will again be permitted to create and design the best software they can without fear of reprisal from a company like Apple Computers. I'm raising my glass to you, David, hoping you overpower Goliath for the users.

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

THE A.N.A.L.O.G. #67 DISKETTE CONTAINS 17 MAGAZINE FILES. THEY ARE LISTED BELOW.

SIDE 1:

FILENAME.EXT	LANG.	LOAD	COMMENTS
BAS2BIN .OBJ	ML	(#3)	BASIC TO BINARY
DUNGEON .OBJ	ML	(#3)	DUNGEONLORDS
GRAPHICS.ACT	ACTION!	(#1)	ACTION! GRAPHICS, L1
KOALA .ACT	ACTION!	(#1)	ACTION! GRAPHICS, L2
MAKEDATA.BAS	BASIC	LOAD	ACTION! GRAPHICS, L3
CHECK .ACT	ACTION!	(#1)	D:CHECK IN ACTION!
GDW1 .LST	BASIC	ENTER	GAME DESIGN WRKSHP, P1
GDW2 .LST	BASIC	ENTER	GAME DESIGN WRKSHP, P2
GDW3 .LST	BASIC	ENTER	GAME DESIGN WRKSHP, P3
GDW4 .LST	BASIC	ENTER	GAME DESIGN WRKSHP, P4
GDW5 .LST	BASIC	ENTER	GAME DESIGN WRKSHP, P5
GDW .BAS	BASIC	LOAD	GAME DESIGN, MERGED
MLEDITOR.BAS	BASIC	LOAD	M/L EDITOR
EDITORII.LST	BASIC	ENTER	BASIC EDITOR II

SIDE 2:

FILENAME.EXT	LANG.	LOAD	COMMENTS
BAS2BIN .M65	MAC/65	LOAD	BASIC TO BINARY, SRC
DUNGEN1 .ASM	ASSEM.	ENTER	DUNGEONLORDS, SRC PT1
DUNGEN2 .ASM	ASSEM.	ENTER	DUNGEONLORDS, SRC PT2

TO LOAD YOUR A.N.A.L.O.G. DISK

1) INSERT BASIC CARTRIDGE (NOT REQUIRED FOR XL OR XE 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.

NOTE: ONLY PROGRAMS WITH THE ".BAS" OR ".OBJ" EXTENTION 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 OSS MAC/65 ASSEMBLER
.AMA	REQUIRES THE ATARI MACRO ASSEMBLER
.ASM	REQUIRES THE ATARI ASSEMBLER/EDITOR
.ACT	REQUIRES THE OSS ACTION! CARTRIDGE
.LGO	REQUIRES THE ATARI LOGO CARTRIDGE
.SYN	REQUIRES THE SYNAPSE SYN ASSEMBLER
.STB	REQUIRES ST BASIC

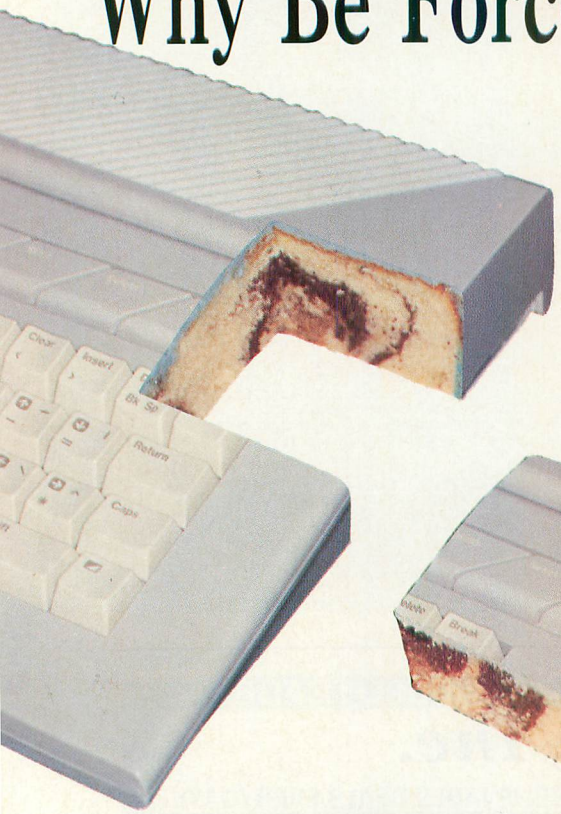
LOADING NOTES

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: SEE ST BASIC MANUAL.

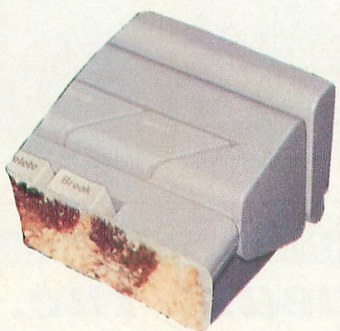
You Own an Atar

Why Be Forced to Read a Magazine that

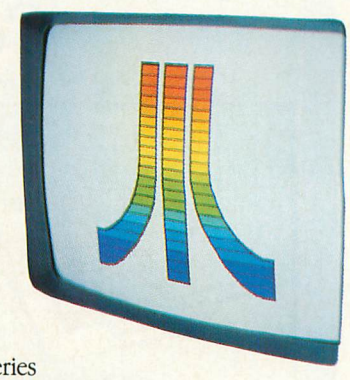


YOUR ATARI RESOURCE CENTER

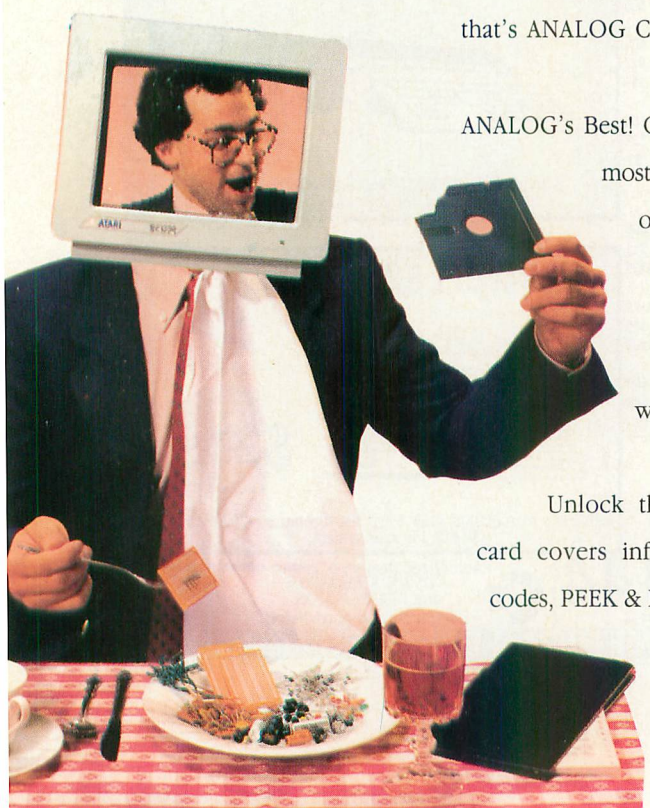
ANALOG Computing continues to offer exciting products for you and your Atari Computer. And we're the only magazine for the Atari 8-bit computer line that hasn't allowed its content to be virtually taken over by coverage of the Atari ST. We include only a minimal amount of ST material so that you can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



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



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

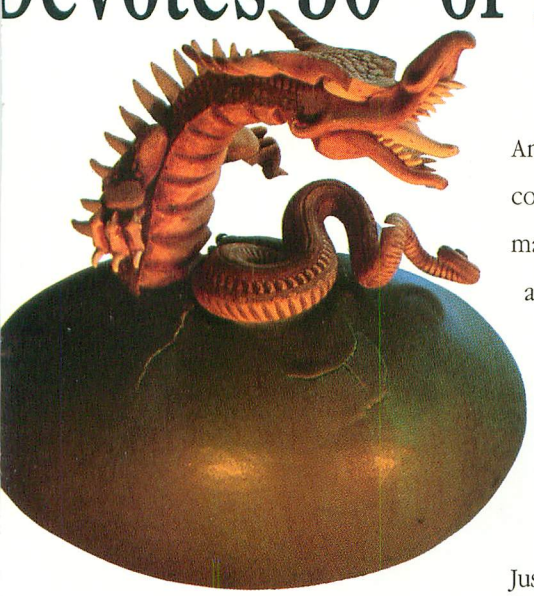


ANALOG's Best! Over 88 of ANALOG Computing's best and most requested programs are now available on this series of ten diskettes. The programs are all ready to run and come with complete documentation on the flip side of each floppy diskette. Select from Graphics, Educational, Utilities 1, Utilities 2,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

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

6502 Computer.

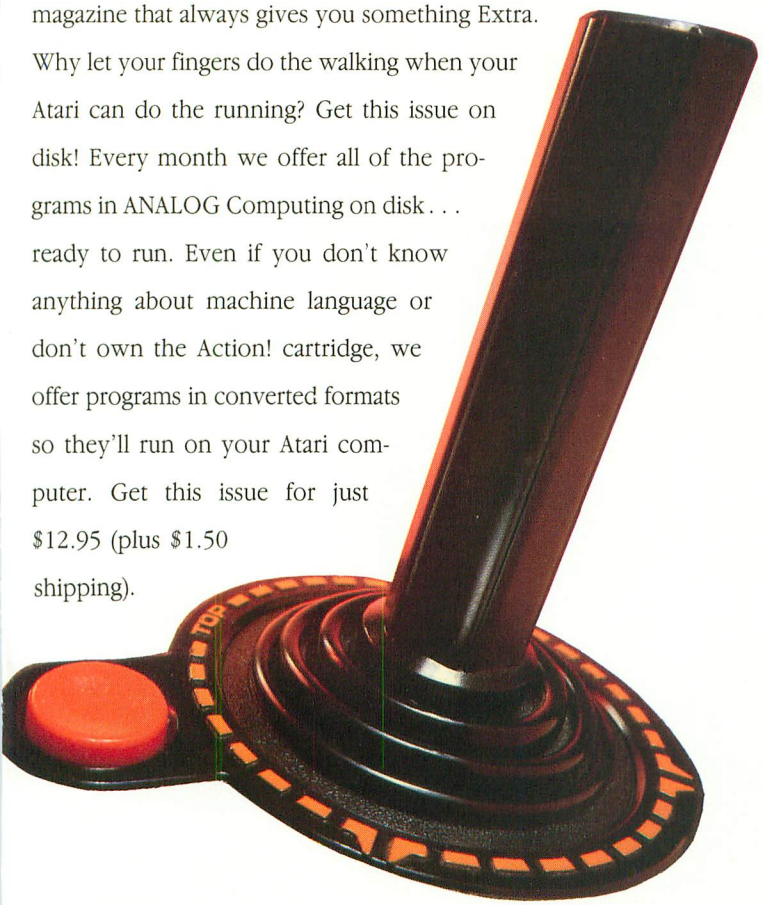
Devotes 50% of Its Pages to the Atari ST?



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

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

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



ANALOG COMPUTING

ANALOG COMPUTING OFFICIAL ORDER FORM

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

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

☐ Payment Enclosed Charge My ☐ VISA ☐ Master Card

Card # _____ Exp. _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

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

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

Seasons Greetings from . . .

INSIDE THIS ISSUE

NEW
MASTER MEMORY MAP

PLUS
PANAK STRIKES
DATABASE DELPHI