THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS

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

T.M.

JANUARY 1989 ISSUE 68 U.S.A. $3.95 CANADA $4.95

Two Arcade Games Inside

Stellar Arena

Inferno

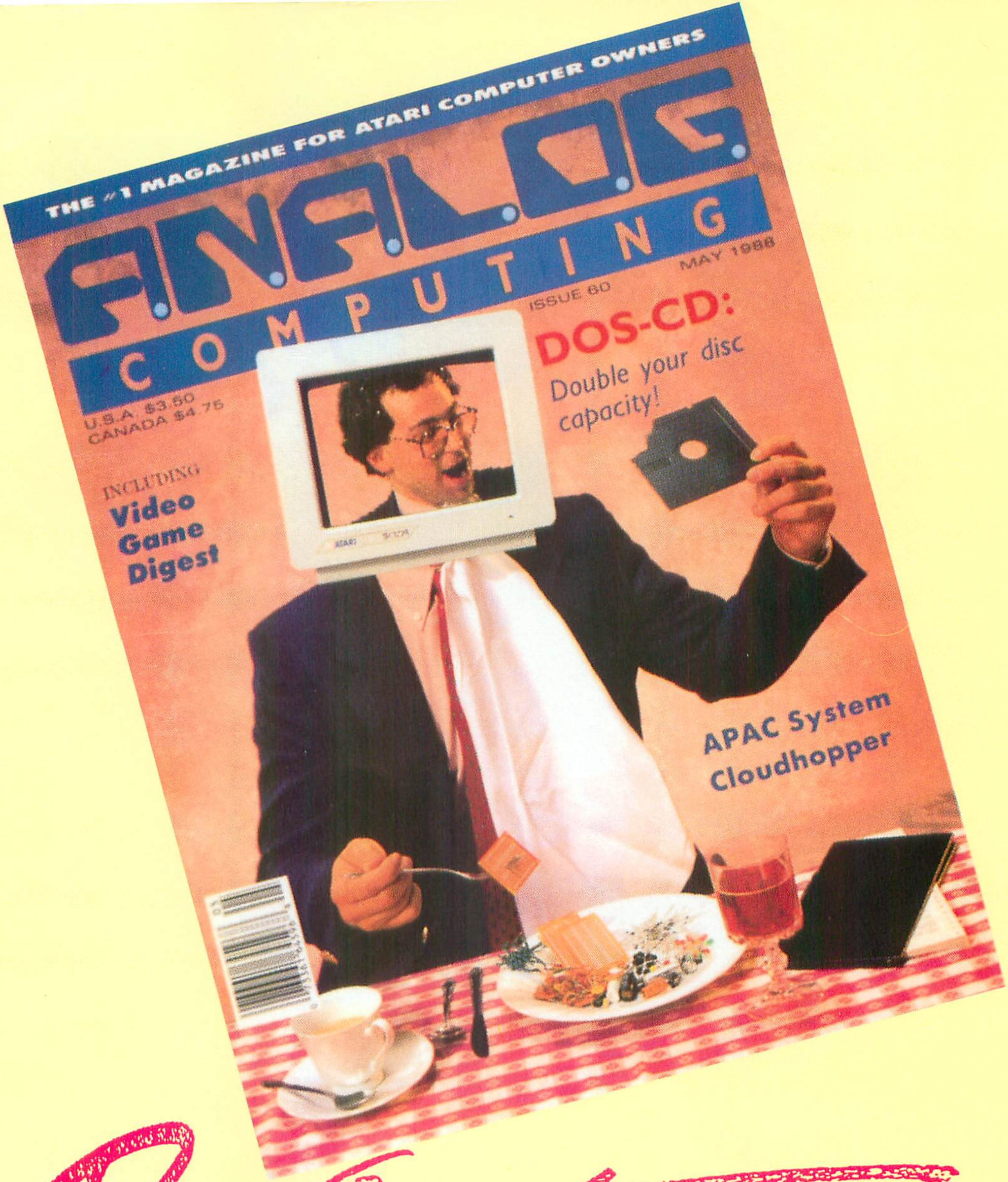
Also:

...er Memory Map

...Magic

...e Design Workshop

...Bomber review



Boot UP BIG SAVINGS!

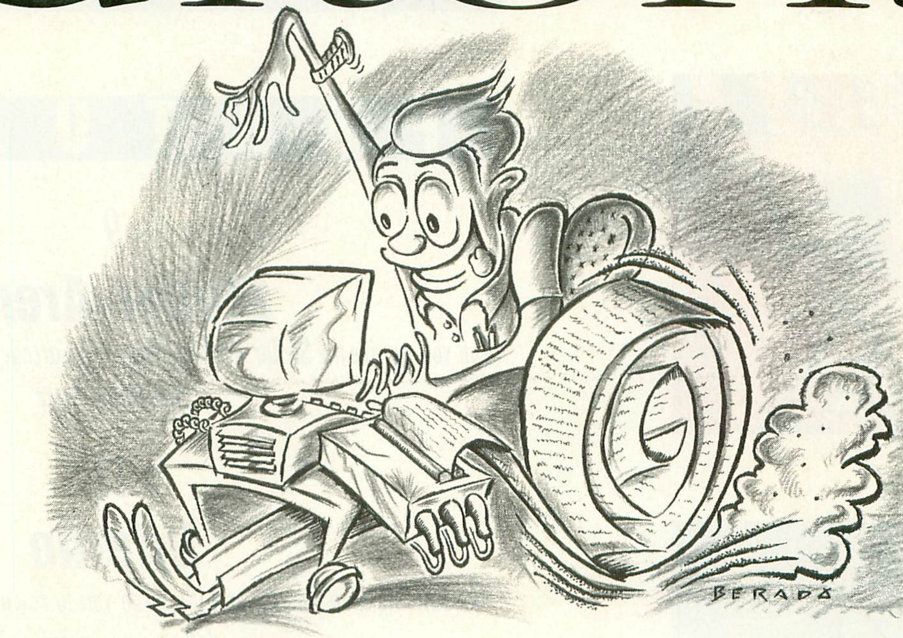
You can save time, and save a lot of money by subscribing to A.N.A.L.O.G. Computing Magazine. Save $19 off the cover price with the convenience of having A.N.A.L.O.G. delivered directly to your door before it even hits the newsstands. To order use the handy postage-paid order card located in the back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

1 YEAR FOR ONLY $28
SAVE **$19** OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

new low price

Editorial



by Andy Eddy
Associate Editor

On the whole, 1988 was not a kind year for Atari. For example, it bought the Federated chain of Western-based, consumer electronics stores, and now the deal is in dispute because of Federated's assets' worth. Atari feels that the Federated assets were overstated by about $43 million, not a drop in the bucket by any means. This situation appears destined for a courtroom decision.

It has also gone through a great many employees. One of the stranger "turnovers" was Chuck Babbitt, who was hired as president of the computer division. Within a couple of months' time, he was out and the position was again open. Atari's explanation was that Babbitt's contract was for short-term consulting—an explanation that very few people took seriously.

Recently (October), the biggest blow to Atari's public image was the loss of Neil Harris. Harris was considered by most to be Atari's mouthpiece, not only speaking at conventions and users' group meetings, but also appearing on the major telecommunications services, answering questions and, though he was restrained near the end by new Atari policies, passing along new product information. Harris will continue in information dissemination: His new career will take him to GEnie as a product marketing specialist.

Perhaps the most disheartening situation was Atari's decision to sell overseas most of the STs it manufactured, where it feels its market is better. It says that the U.S. problem was due to the DRAM shortage. The lack of product prevented it from doing major advertising in the U.S. because the inventory couldn't support the demand.

All of these problems don't help enhance Atari's image. Dave Small, creator of the *Magic Sac* Macintosh emulation cartridge (among other things), uploaded a stinging diatribe (brought on by Harris' resignation) on CompuServe concerning what he considers to be Atari's impending demise. He cited a number of instances that pointed out Atari's poor managerial skills.

Now there are those who will consider me wimpy, but I'm not ready to close the book on Atari. First, a number of sources have stated that Atari may make a stunning showing at the November COMDEX. And, if this indeed takes place (you'll know by the time you read this), it could become a phoenix of sorts, rising out of the flames of a crumbling image.

Unfortunately, none of this helps the 8-bit line of computers. Atari is selling the XE Game System, but it's left to be seen whether buyers will use it for anything other than what its title implies: a game machine. Speaking with a Federated employee the other day, though, I was told that the 8-bit line is in demand, and that Federated is still going through XE *computers* at a brisk pace. Perhaps the death knell has sound prematurely?

If you look back through recent history, you'll see scores of computer companies peaking and valleying, Atari right up there among them. The main hurdle Atari has had to overcome has been its game-machine image. There are those, myself included, who feel that Atari should have taken the same path that Commodore did when it called its computer *Amiga*, a name that stands apart from the company that makes it. Serious computer users, unfairly, find it hard to take *any* computer Atari puts out seriously, given its gaming roots, no matter how powerful the performance statistics.

There's no doubt that Atari has created powerful machines. Trouble arises, in these demanding times, when you don't continually update your hardware, and Atari hasn't done much in that department recently.

Even as I write this, the Summer Olympics are ending. A couple of times in the past, Atari has come close to winning a medal in their computer-industry quest. Some feel that the race is already over and that Atari has pulled up lame; others feel that the competition is coming into the homestretch with Atari far behind, but still with a chance of recovery. The deciding point was COMDEX. As you read this, COMDEX is already in the past, but if Atari made an impressive showing (I've confirmed that it's working on an ST Laptop and a 68030 ST, but Atari hasn't decided whether it'll be showing these products), it could renew itself. We'll pass the information on to you as we find it out.

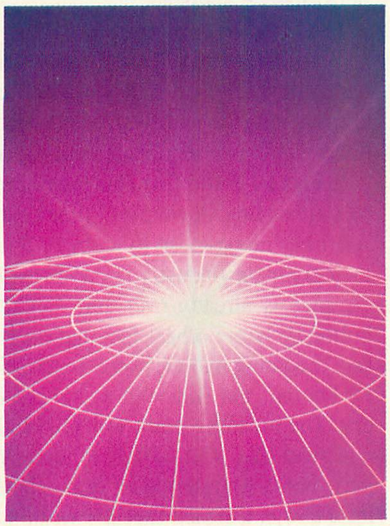
In the meantime, we wish you and yours a Happy New Year, from all of us at ANALOG.

Conte

FEATURES

Page 12

Page 68



Page 9

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1988 L.F.P., Inc. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. All rights reserved on entire contents; nothing may be reproduced in whole or in part without written permission from the publisher. 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.

nts

Statement of Ownership Management and Circulation

Title	ANALOG Computing
Frequency of issue	Monthly
No. of issues annually	12
Annual subscription price	$28.00
Mailing address	LFP Inc. 9171 Wilshire Blvd., Suite 300 Beverly Hills, CA 90210-5515
Headquarters	LFP, Inc. 9171 Wilshire Blvd., Suite 300 Beverly Hills, CA 90210-5515
Publisher	Lee Pappas
Editor	Clayton Walnum
Managing Editor	Dean Brierly
Owner	LFP Inc. Larry Flynt
Total no. copies printed	44,701
Circulation data (average per issue for preceding 12 months)	
Dealers and vendors	13,862
Mail subscription	13,610
Total paid circulation	27,472
Free distribution	990
Total distribution	28,462
Copies not distributed	1,886
Return from news agents	14,353
Total	44,701

ANALOG COMPUTING STAFF

Publisher
LEE H. PAPPAS

Executive Editor
CLAYTON WALNUM

Art Director
KRISTEL PECKHAM

Associate Editor
ANDY EDDY

Managing Editor
DEAN BRIERLY

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
CHARLES F. JOHNSON

Contributing Editors
LEE S. BRILLIANT, M.D.;
MICHAEL BANKS; FRANK COHEN;
MAURICE MOLYNEAUX;
STEVE PANAK; CRAIG PATCHETT;
ROBIN SHERER; KARL E. WIEGERS
MATTHEW J. W. RATCLIFF

Entertainment Editors
DAVID PLOTKIN

Cover Photography
MARK CHEN

Illustrations
JOHN BERADO
ALAN HUNTER

Copy Chief
KATRINA VEIT

Copy Editors
SARAH BELLUM
ANNE DENBOK
PAT ROMERO
KIM TURNER

Chief Typographer
KLARISSA CURTIS

Typographers
DAVID BUCHANAN
JUDY VILLANUEVA

Contributors
BILL BODENSTEIN
JOHN MANOR
FRANK MARTONE
MARK ODENDAHL
JOHN ORTIZ

Vice President, Production
DONNA HAHNER

Advertising Production Director
JANICE ROSENBLUM

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

Corporate Ad Director
PAULA THORNTON

Subscriptions Director
IRENE GRADSTEIN

Vice President, Sales
JAMES GUSTAFSON

JANUARY 1989
ISSUE 68

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

No portion of this magazine may be reproduced in any form without written permission from the publisher. Many programs are copyrighted and not public domain.

Due, however, to many requests from Atari club libraries and bulletin-board systems, our new policy allows club libraries or individually run BBSs to make certain programs from **ANALOG Computing** available during the month printed on that issue's cover. For example, software from the July issue can be made available July 1.

This does not apply to programs which specifically state that they are not public domain and, thus, are not for public distribution.

In addition, any programs used must state that they are taken from **ANALOG Computing** Magazine. For more information, contact **ANALOG Computing** at (213) 858-7100, ext. 163.

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**

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 #103 ON READER SERVICE CARD.

8-Bit NEWS

Merrill Ward vs. Total Control Systems

Previously, ANALOG reported that Merrill Ward Associates (MWA) was negotiating with Atari Corp. to license a new 8-bit desktop operating system for the XE/XL computer. The hopes were that *GOE* would become the new standard operating system for the XE/XL and renew interest in the XE/XL computer. A renaissance in the Commodore 64 has been spurred by the release of the GEOS desktop system last year.

GOE was written by David Sullivan of Total Control Systems (TCS). Sullivan was first contacted by Shelly Merrill last year. Merrill desired to obtain the marketing rights to GOE with the understanding that Merrill had the contacts needed to approach Atari Corp. with a licensing proposal. Sullivan gave a verbal approval to approach Atari Corp., then sent a demonstration version of GOE to be shown at the meeting with Atari Corp.

A proposal was drafted between TCS and MWA, although in the end the two could not come to an agreement. After Atari Corp. declined Merrill's proposal, the situation turned to the worse. MWA announced the GOE at the Consumer Electronics Show last summer, and sold copies of the demonstration version to 8-bit end users. MWA did this without informing Sullivan.

TCS has since completed the GOE system and is now publishing *and* marketing the system to the Atari user community. Atari Corp. has again been approached, but no word has been given on acceptance. In the meantime, TCS is making the GOE system available as a 64K cartridge that includes the basic operating environment and documentation of its usage. Additional cartridges can be plugged into the top *and* bottom of the GOE cartridge, making it compatible with other cartridge-based products.

A painting program on the par with *Neochrome* for the ST and a word processor that supports fonts and graphics is being completed and should soon be released.

A developer's kit is now available. Released on several disks, the developer's kit includes a linker, resource construction set and example programs. TCS expects developers to use the *MAC/65* assembler, as they do not provide their own assembler.

Total Control Systems
4156 Tolowa Street
San Diego, CA 92117
(619) 270-0111

Merrill Ward & Associates
255 North El Cielo Rd., Ste. 222
Palm Springs, CA 92262
(619) 328-8728

CIRCLE #151 ON READER SERVICE CARD.

So Long Sandy

There is a new dance that's very popular in Sunnyvale, California. It's called the Atari Shuffle. The more successful shufflers have been Jerry Brown, Charles Babbitt, Tony Gould, Richard Frick—and most recently Sandy Austin. Up until June, Sandy was Atari's Users' Group coordinator, a role that put her in charge of all the Atarifests.

Austin was hired into Atari in the summer of 1986 and immediately placed in charge of directing communications between Atari and the many dedicated, Atari Users' Groups. She was the primary force behind the Atarifest concept, bringing Atari promotions to individual cities with the help of local user groups. The promotion was a huge success for Atari, with more than 14 Atarifests held in 1987. At a very small cost, Atari brought its name and advertising to home and small business end users on a personal and professional level.

Unfortunately, the rule at Atari seems to be "Do well and begone." So, Sandy Austin is no longer at Atari. Cindy Claveran has been promoted into Austin's old position. Claveran has been with Atari for the past two years as the developer-relations coordinator, a role that has made her the favorite of most 8-bit and ST developers.

Rumor du Jour

ADOS is Atari's announced-but-not-shipped disk-operating system that allows the new Atari XF551 disk drive to work in double-density, double-side format. XF551 users will have access to 360K of storage space on one 5¼-inch floppy disk. Unfortunately, ADOS is not yet available, making SpartaDOS X from ICD the only choice. When the XF551 was ready to ship last summer, ADOS was intended to be shipped with the drive as a public-domain operating system. Unfortunately, ADOS was not ready, so Atari shipped the old DOS 2.5.

It seems that Atari is renaming ADOS to *DOS XE* and will be selling the new operating system as a commercial product. Although no official word on DOS XE release date has been given, Atari XE/XL users should expect the new system within the next few months. By the way, the new DOS requires at least 64K of memory, making it incompatible with the old Atari 800 computer series.

Atari Corp.
1196 Borregas Ave.
Sunnyvale, CA 94086
CIRCLE #154 ON READER SERVICE CARD.

SpartaDOS X Ships!

ICD is now shipping *SpartaDOS X*, a new disk-operating system for your Atari XE/XL computer. SpartaDOS X is a plug-in cartridge with *fast* disk I/O routines. Loading DOS takes half the time it takes to load the current system utilities. Since SpartaDOS X is cartridge based, it offers the most free memory of any available DOS. ANALOG had a chance to test SpartaDOS X at the Glendale Atarifest, and the speed results were very good.

SpartaDOS X is fully compatible with Indus GT drives, Atari XF551 drives, US Doubler-enhanced 1050 drives and Happy 1050 drives. All densities are supported (single, dual and double), so owners of the new Atari XF551 will finally be able to access the extra 180K of storage in the double-side double-density mode.

SpartaDOS X adds hard-disk compatibility using ICD's *Multi I/O* (MIO) board. Directory sizes have been improved to 1423 files (other DOS directories are far more limited). You will also find *Archive* support built into SpartaDOS X. Archive—ARC for short—is a program that compacts programs and data into smaller files for transmission to bulletin-board systems and other online services.

Several other new features have been included: supported are 16 I/O channels, nine drives (with D9: as a RAMdisk), up to one-megabyte RAMdisks and full parameter passing for batch files.

ICD
1220 Rock St.
Rockford, IL 61101-1437
(815) 968-2228 **CIRCLE #155 ON READER SERVICE CARD.**

Been Looking for Old Stuff?

Many times during your XE/XL's lifetime, you might have a need to buy some old equipment or software. For example, do you remember the Adam computer? In 1984, Coleco was on top of the video-games trend with the Colecovision game machine. However, shortly after the 1985 release of the Adam, Coleco was nowhere to be seen in the computer market.

M.W. Ruth Co, Inc. still has programs and cartridges for Colecovision and Adam. They also stock Atari VCS and XE/XL cartridges. They even have Atari 5200 game cartridges. Most of their catalog of products is filled with inexpensive consumer items: joysticks, computer covers, etc.

M.W. Ruth Co., Inc.
3100 W. Chapel Ave.
Cherry Hill, NJ 08002
CIRCLE #153 ON READER SERVICE CARD.

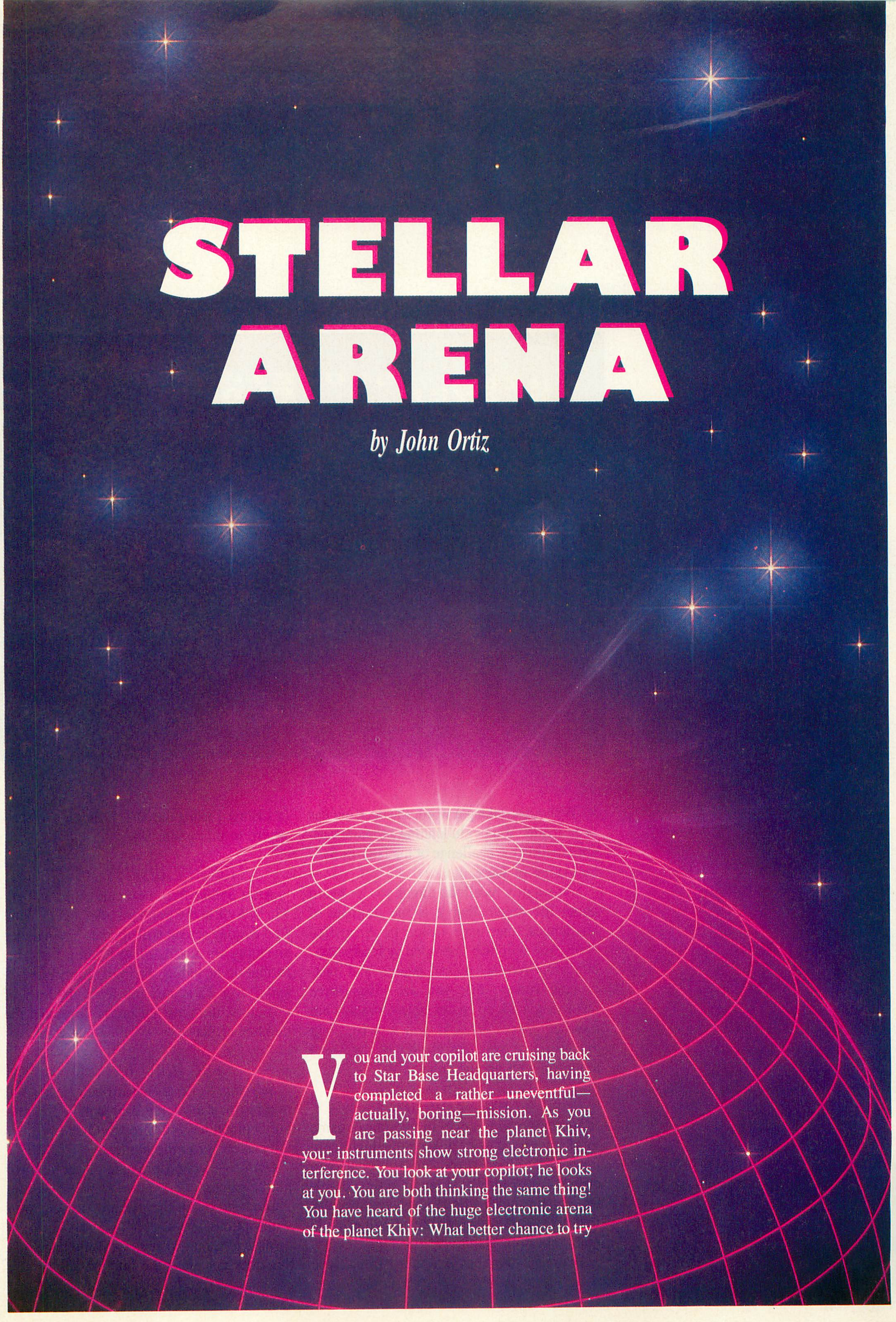
Users' Group Highlight

The Bakersfield Atari Computer Enthusiasts publishes *BACE Line*, a monthly newsletter of information and help for Atari 8-bit and ST users and programmers. Of interest to 8-bit users, the newsletter includes an interesting series of tutorials for programming with Atari BASIC. Typical topics include looping constructs, error reporting and other necessary functions.

BACE Line
228 Plymouth Ave.
Oildale, CA 93308
CIRCLE #156 ON READER SERVICE CARD.

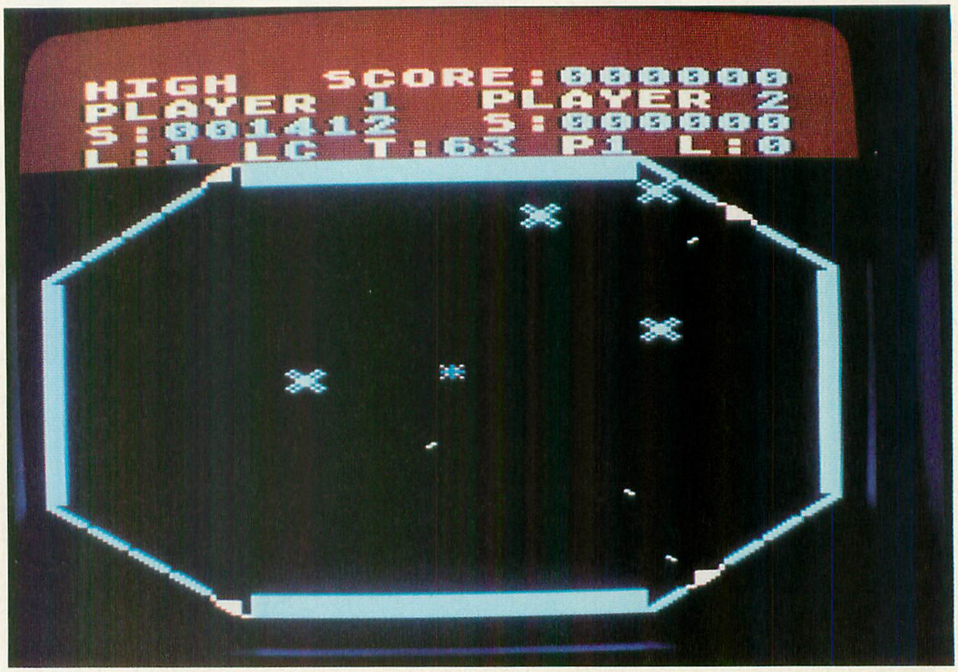
STELLAR ARENA

by John Ortiz



You and your copilot are cruising back to Star Base Headquarters, having completed a rather uneventful—actually, boring—mission. As you are passing near the planet Khiv, your instruments show strong electronic interference. You look at your copilot; he looks at you. You are both thinking the same thing! You have heard of the huge electronic arena of the planet Khiv: What better chance to try

STELLAR



```
HIGH    SCORE:000000
PLAYER 1    PLAYER 2
S:001412    S:000000
L:1 LC T:63 P1 L:0
```

it? Without speaking a word, the decision has been made. You begin landing preparations, anticipating the challenge of the famous Stellar Arena of Khiv.

You silently recall the things you've heard about the arena: that it is the ultimate of challenges; that once inside its electrified walls, you will be bombarded by laser shots, attacking robots, pulsar beams and fast-moving spikes; that you are given three chances to rack up an enviable score and are awarded bonus lives if you survive long enough; that you can challenge the arena alone or with another player.

Getting started

Stellar Arena is a one- or two-player game for preteens and older. It keeps the high score for each session, so you can try to outscore even yourself. Stellar Arena requires excellent joystick agility. Its 15 levels of difficulty will not be mastered easily and will provide many hours of challenge.

To create your copy of Stellar Arena, type in Listing 1 using the M/L Editor found elsewhere in this issue.

The first display screen

After Stellar Arena executes, you will get a brief look at the program title, and then the first display screen will appear. You will see an empty arena and, above it, the scoring information including the high score; Player 1 and Player 2's scores and lives, the level of difficulty, the time left on the level and the number of players.

The function keys

Selection—to choose a difficulty level. You may begin on any level A through I. (You may not begin at levels J through O). Level A is the easiest, O is the hardest.

Option—to choose a one-player or two-player game.

Start—to begin the game or to start over with options intact.

System Reset—to return to title page, but this does not erase the high score.

ARENA

Pause—to pause any time, you press the space bar or any letter key. A joystick movement by either player will bring you out of pause.

Using the joystick

In Stellar Arena you will use the joystick for three reasons: to move your player, fire your weapon and determine the direction of the shot. To move your player, simply move the joystick in any of eight directions. To fire your weapon, press the red fire button and, at the same time, move the joystick in the direction you wish to shoot.

You may shoot as often as you like, but each new shot erases the one before it; so give your shot time to hit its target before you shoot again.

Inside the arena of Khiv

The challenge of Stellar Arena is to score as many points as possible by both dodging and shooting the obstacles inside. You earn points as long as you stay alive, but to get a good score you must knock out as many obstacles as you can.

The arena is divided into five different zones, each with its own color and three levels of difficulty. (Each level is slightly faster than the one before it.)

Zone blue contains four cybernetic robots. The robots always appear at either the top or bottom wall of the arena. They move within the arena in one direction until they are deflected by a wall or another robot, when shot it is automatically replaced; so there are always four robots present. Shooting a robot gets you ten points times the level (Level A=10 pts., Level B=10 pts. x 2; through Level O=10 pts. x 15). Zone blue, like all other zones, contains a laser gun in each corner which moves randomly back and forth across the corner. The laser guns get 50 times the level points, but if it is shot, it is out of commission for the remainder of that level.

Zone green contains five cybernetic robots, four laser guns and one pulsar. The pulsar moves up and down along the left side of the arena. Before releasing its deadly beam horizontally across the entire arena, it will hum for a second or two—your warning to get out of its path. The pulsar is too powerful to be knocked out.

Zone purple contains six robots, four lasers, two pulsars (one at each side that shoot at the same time, but not in the same place) and one spike. Spikes move around randomly, can be shot for 100 times the level points and are out for the rest of the level once shot.

Zone gold has seven robots, four lasers, two pulsars and one *fast* spike. You also move twice as fast in this zone.

Zone red has eight robots, four lasers, two pulsars and two speedy spikes—*good luck!* This zone moves you twice as fast as the previous one.

In all zones, collision with anything, including the electrified walls, results in immediate death. You are given three lives to start and earn an additional one every 10,000 points.

In a two-player game, the players should agree beforehand whether they will fight each other or not. Shooting another player, planned or accidentally, earns 250 times the level points. Collision with the other player (unless he is in suspended animation) results in the death of both players.

Ready to take the challenge?

When you are ready to enter the arena, press Start. You will be positioned in the left center of the arena (in a two-player game, the other player is in the right center of the arena). You will see the robots and the laser guns. You are now in suspended animation and invulnerable to any attack. You will always start this way when getting a new life, but, as soon as you press the fire button, you're under attack and should move. While you are invulnerable your score does not advance, *but* the timer is running; so it is to your advantage to get in as quickly as possible.

Ready? Press Start and the fire button, and good luck in the Stellar Arena of Khiv.

continued on page 36

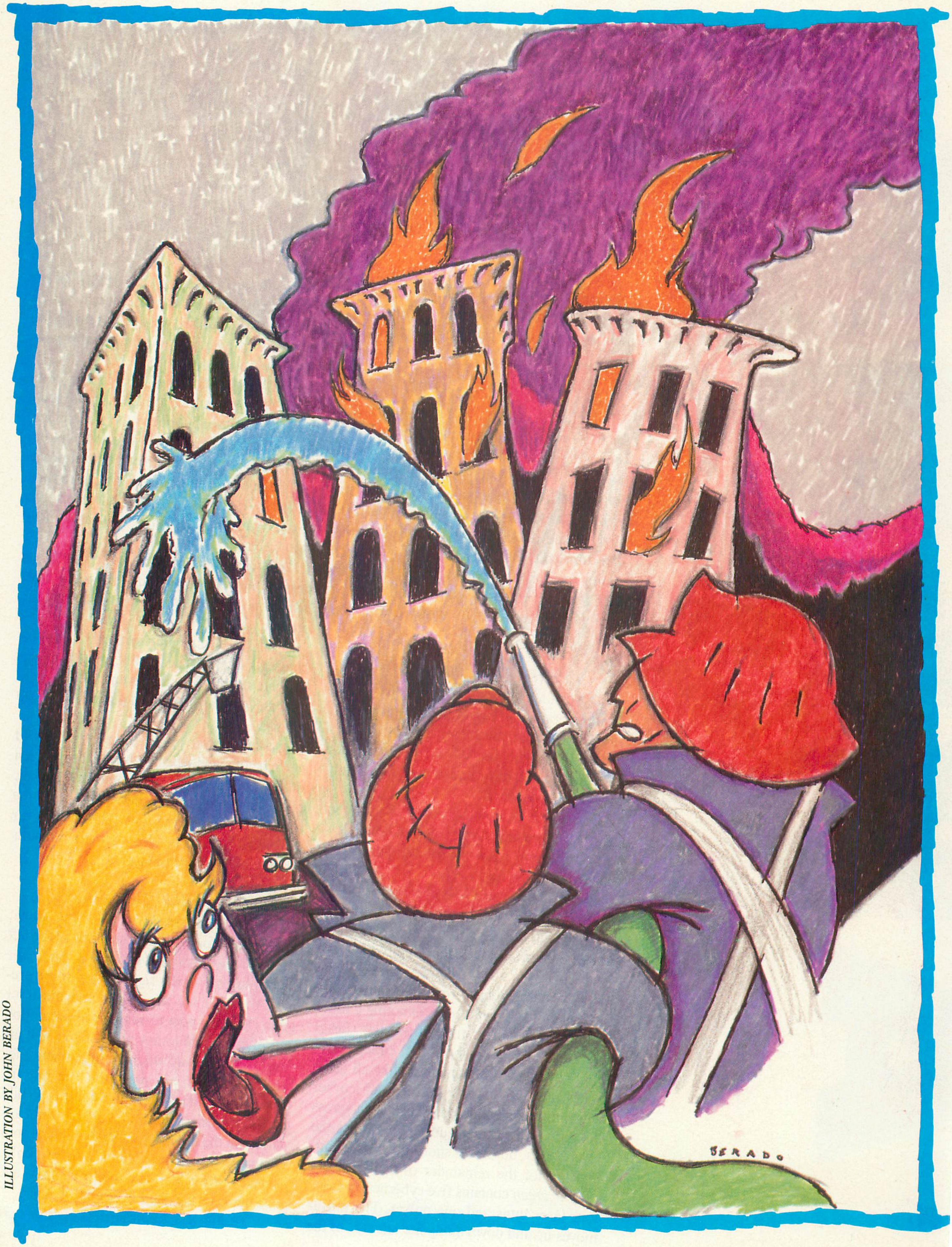


ILLUSTRATION BY JOHN BERADO

INFERNO

by Frank Martone

You are a daring firefighter racing against time to save the victims trapped in a burning building.

In the beginning of the game, you will see a diagram of the building you are in, and a flashing arrow will show you what floor you are on. Each building consists of seven floors; there are five people on each floor. So, there are 35 people in each building. The goal of *Inferno* is to successfully reach the top of the third building.

You will start out with five firefighters, this is displayed at the upper right corner of the playfield. Travel from floor to floor, trying to save as many people as you can. To save a person merely touch them. Beware of the flames and explosions; touching these is naturally fatal.

Travel from floor to floor trying to save as many people as you can, but beware of flames and explosions.

Cheer up, your firefighter is not left totally defenseless. You have a powerful fire extinguisher. Pressing the joystick button will activate it.

It surrounds the firefighter with a stream of water. You can use it to clear away the flames or explosions. Be careful. If you accidentally spray a person with the extinguisher, he will die of suffocation. Your fire extinguisher will not last forever; its water supply must be replenished. Every time you use the extinguisher it will cost you ten units of water. To refill the extinguisher touch a nearby water canister. It will give you 100 units of water. You will start out with 300 units of water. The current water supply is displayed at the top in the middle of the playfield.

You don't receive points immediately when you save a person. Your score for a floor is given when you leave it. You may not exit a floor until either all, or some of the five people on that floor, are either killed or rescued. When you are allowed to leave the floor, your man and the screen walls will turn green. You may then exit through the top of the screen.

After you leave a screen (floor), you will see the words "FLOOR SECURED" on the screen. You will then receive points based on the number of people that were saved on that floor times the point value of the people in that building. In the first building, all people are worth 100 points, in the second building the people are worth 200 points, etc.

When you reach the top of a building, you will receive a free firefighter, and you will also get points based on the number of people saved in the entire building. You will receive 1,000 points for each person saved. So, the perfect score would be 35,000 because there are 35 people in each building. If you save all 35 people, you will receive 50,000 points as a bonus.

Important variables:

SC—Score
PN—Number of people
WP—Water supply
FMEN—Number of firefighters
DL—Difficulty level

continued on page 29

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

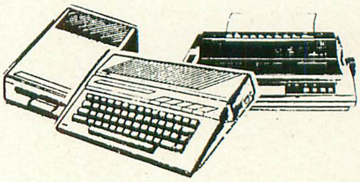
MasterCard VISA

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



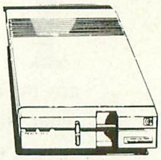
$429.95

Call For More ATARI Hardware Information.

INDUS

GTS-100

- Atari ST Drive
- 3.5" DSDD



$195.95

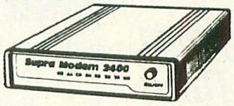
ATARI

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

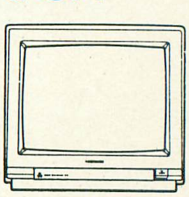
$969.95

• Hi Res color monitor included!

MAGNAVOX

CM-8502

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

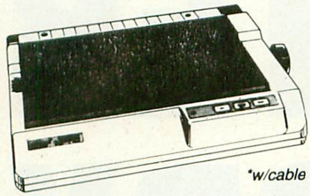


$179.95

1-800-233-8760

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

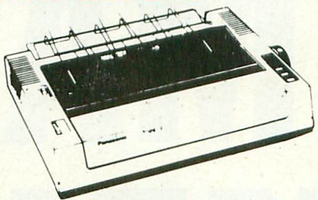
NX-1000	$169.95*
NX-1000C	$169.95
NX-1000 Color	$225.95
NX-1000C Color	$229.95
NX-15	$289.95

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 OA

1080i Model II



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

$149.95*

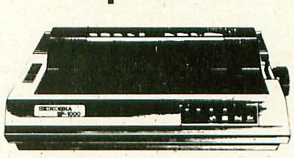
*quantities limited

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

*quantities limited

SEIKOSHA

Sp180Ai



$125.95*

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

*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

CIRCLE #102 ON READER SERVICE CARD.

MASTER MEMORY MAP PART VI

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.

CSTAT
648 0288

CSTAT is the cassette status register.

WMODE
649 0289

This location tells the cassette handler whether the cassette is to be read from (0) or written to (128).

BLIM
650 028A

When the cassette handler reads in a record, the 132 bytes in that record are stored in CASBUF (1021). BLIM tells how many of those bytes are data that we want to give to the user. It is set according to one of the control bytes in the record, and since this probably doesn't make any sense to you, you should go read the description of CASBUF if you want more information.

Noname
651-655 028B-028F

More spare bytes that you shouldn't use because future versions of the OS might use them. Locations 651 and 652 are already used by version "B" as part of the interrupt handler routines.

The display handler uses the next 48 locations. Note that not all locations are used in all graphics modes.

In the case of a graphics mode with a text window, the display handler takes care of the screen, while the screen editor takes care of the text window. Two separate IOCBs are

continued on page 54

Game Design Workshop

Player/Missile Graphics

by Craig Patchett

So far we've seen a whole bunch of different ways to move things around on the screen, but they all had limitations that were difficult to deal with. What we really need is something that will allow us to quickly and easily move an object around independently of whatever else is on the screen. Fortunately for us, the Atari has a special feature called Player/Missile Graphics (PMG) that will allow us to do just that.

Before we take a look at exactly how to use PMG, it may help to understand what exactly it is. To do that, we'll compare PMG to a cartoon animator and his work. In producing a cartoon, the animator has the same kind of problem that we do. She wants to be able to move characters around the screen without having them affect the background scene. How does she do this? She could draw the character on the background, film it, then redraw the background, put the character in a different position and repeat the whole process over and over again. Obviously though, that would take a long time to do. Instead she draws the background once, then traces and paints the character onto a clear plastic sheet and places the sheet over the background painting. Then when she wants to change the character's position, all that needs to be done is to change the position on the acetate sheet.

Pretty ingenious, don't you think? Well, so did the folks at Atari. It gave its computers five of these "sheets," called "players," each eight dots wide and as high as the screen. It also gave one of the players the ability to break up into four separate sheets, called "missiles," each two dots wide and also as high as the screen.

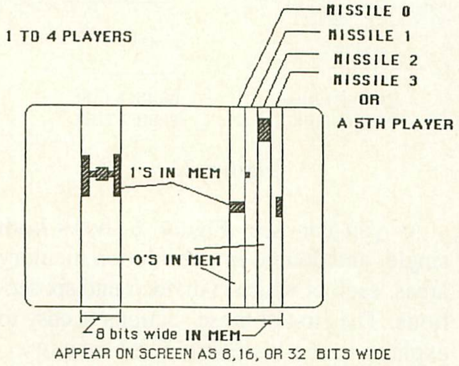


Figure 1.

That gave Atari computers the ability to move objects all over the screen without affecting whatever else was on the screen, an ability that no other home computer had at the time.

How Much Memory Do I Need?

Now that we know what we're going to be doing, let's take a look at how it's done. It will help if you go back and review the section on redefining the character set, since PMG shares a lot of the same techniques. For example, like character sets, PMG needs some space in memory of its own. How much space? Well, that depends. Unfortunately, and I only use the word because this may seem a little complicated at first, PMG has two modes, in the same way there are 12 graphics modes. In the first mode, called double-line resolution, each dot is the same height as the graphics mode 7. In the second mode, called single-line resolution, the dots are half as high or as high as a graphics mode 8 dot. In both modes, the dots are as wide as a graphics mode 7 dot.

How does this affect you? Single-line resolution mode takes up twice as much memory. Double-line resolution takes up 1K of memory, and single-line resolution takes up 2K. Sounds backward, doesn't it?

What mode you will use depends both on how much memory is available and the size of the pixels you want on the screen. If you're short on memory, you will have to use double-line resolution. Both will be demonstrated very soon.

How to Protect It

Now that we know how much memory is required, where do we find it? In the same place we found it for character sets—at the end of memory. First we have to make that memory safe to use, however, and we again do this by changing the value in location 106. The following lines give examples for both single- and double-line resolution:

```
100 PB=PEEK(740)-4:POKE
106,PB-4:GRAPHICS 0:REM
SINGLE-LINE RESOLUTION

100 PB=PEEK(740)-8:POKE
106,PB-4:GRAPHICS 0:REM
DOUBLE-LINE RESOLUTION
```

Don't forget that you need some kind of GRAPHICS command after you change location 106 in order to make sure memory is really protected.

You should also recall that we subtract an extra four from PB when we change location 106 to make absolutely sure our reserved area is safe from intrusion by the statement or a listing that might scroll upward into what the computer thinks is unused memory. Also, if you're using one of graphics modes 7 through 11, you should change location 106 by a multiple of 16. We discussed the reason for this in the section on redefining character sets. As you will recall, these graphics modes need to start on either an even 2K or 4K boundary.

PB, incidentally, stands for Player/missile Base, which is just a fancy expression for the beginning of our PMG memory area. You can call it whatever you like. If you need a safe place for both the character set and PMG, first create a safe area for your character set, move it, then create another safe area for the PMG data, and place the data needed there. So, now that we have our PMG memory safely tucked away, let's take a look at how it's used.

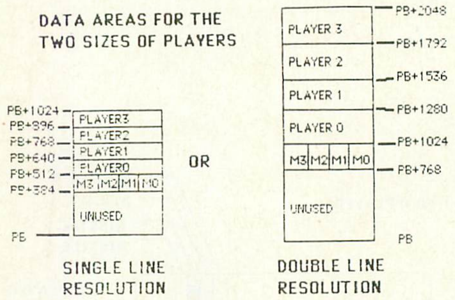


Figure 2.

As you can see, Figure 2 shows both single- and double-line resolution memory areas, each of which is divided into six sections. The first of these sections is easy to explain, since it's just unused memory.

That's right, these bytes are not used at all by PMG, and since they've been reserved, you can use them to hold machine-language routines (as we'll do later in this chapter), data or nothing at all! The sections of memory after the unused area hold the data for the players and missiles, so this is probably a good time to explain how players and missiles are created.

We already know that a player is the equivalent of a sheet or strip of plastic placed on top of the background. This strip is eight dots wide, which should ring a bell for you. Why? The Atari characters are also eight dots wide, and in fact it might be easier for you if you think of a player as a very tall character. Whereas a character is eight bytes high, a player is 128 bytes for single-line resolution, 256 bytes for double-line. So, each dot in a player is represented by a bit in a byte, just like with characters (if you haven't done so already, go back and review the section on character sets). What about the missiles? The missiles are the same height as the players, but they are only two bits wide instead of eight. There are four crammed into the same set of 128 or 256 bytes. Figure 3 gives an example.

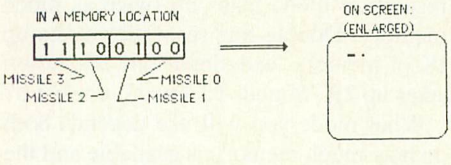


Figure 3.

Now we have all the basics out of the way and are ready to get into the mechanics. Before we do, however, let's take time out for a quick review.

1. Players and missiles move independently of the background.

2. There are four players and four missiles, or you can group all the missiles together for a fifth player.

3. Single-line resolution gives you dots that are half as tall as double-line resolution.

4. You must reserve 2K of memory for single-line resolution, 1K for double-line.

5. In single-line resolution, players and missiles are 256 bytes high. In double-line they are 128 bytes high.

6. Players are eight dots or one byte wide. Missiles are two dots or a quarter of a byte wide.

7. Players are stored in memory in the same way as characters; so are missiles, except that four of them are crammed into the same byte.

With all this in mind, we're now ready to get some real live players up on the screen.

For our first example we'll do something simple—a spaceship. Anyone can build one of these. (Want to go for a ride?) We'll use player 0 and single-line resolution. Afterwards we'll see how it would be done in double-line resolution. Let's get started by drawing our spaceship as shown in Figures 4, 5 and 6.

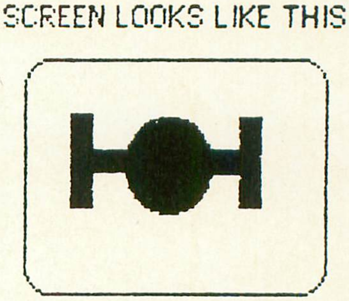
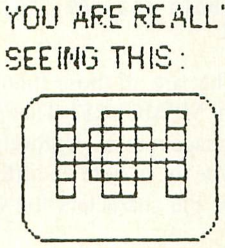
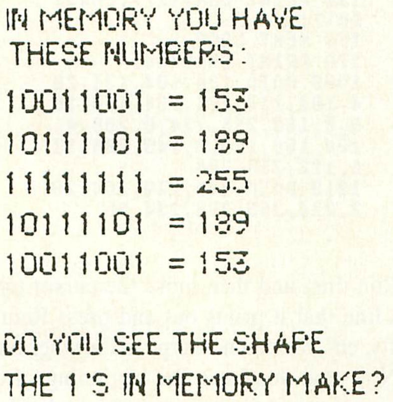


Figure 4.



```
YOU ARE REALLY
SEEING THIS:
```

Figure 5.



```
IN MEMORY YOU HAVE
THESE NUMBERS:
10011001 = 153
10111101 = 189
11111111 = 255
10111101 = 189
10011001 = 153

DO YOU SEE THE SHAPE
THE 1'S IN MEMORY MAKE?
```

Figure 6.

In order to show this (Fig. 4) on the screen, we have to convert it to dots (Fig. 5), and then into the correct number to place in memory to represent those dots (Fig. 6).

Now we're ready to put these values into memory. The best way is to use a character set editor.

Notice that so far we have only used eight of the 256 bytes available to us up and down the screen. You don't have to use all of those 256. Any bytes, or any dots for that matter, that aren't used will be invisible on the screen, just like the portions of the cartoonist's plastic that aren't drawn on. And as you'll discover soon, you want to leave a lot of empty bytes right in the middle of the player, leaving the first and last 124 bytes empty (124=(256−8)/2). Before we do that, however, we have to make sure that our PMG memory area is empty.

When we reserved this memory, there probably was already information stored in it, and we have to get rid of that information before we start putting our own data in there. We could do this with a simple FOR/NEXT loop like this:

```
110 FOR X=0 TO 2047:POKE
PB*256+X,0:NEXT X
```

But as you can see by running this loop, it is very slow. Better to use machine language. *Oh no*, you think, *I didn't realize I was going to have to use machine language!* Relax, all you have to do is use it; we'll do the programming for you. As a matter of fact, all you have to do to make our machine-language routine work is use the following command:

```
X=USR(ADR(MEMCLR$),START
,LENGTH)
```

In this command, MEMCLR$ is the string, which you'll get to in a minute, that holds the machine-language routine. START is the starting address of the memory you want to clear, and LENGTH is the number of bytes you want to clear. So for our PMG example, you'll use:

```
X=USR(ADR(MEMCLR$),PB*25
6,2048)
```

Now that isn't too painful, is it? And you can even use the routine for stuff other than PMG, whenever you have to clear some memory.

Incidentally, we have to multiply PB by 256 because PB is the number of pages, not the actual address.

Let's put together the beginning of our program (finally). We'll reserve some memory, clear it, and then put our spaceship data into the middle of player 0:

```
100 DIM MEMCLR$(36):MEMC
LR$="hh┐Lh┐Kh*)♥ X♥P h(
)♥┌K◢@)PyfLJ◆)P j◆"
110 PB=PEEK(740)-8
120 POKE 106,PB-4
130 GRAPHICS 0
140 PMBASE=PB*256
150 X=USR(ADR(MEMCLR$),P
MBASE,2048)
160 FOR BYTE=126 TO 130
170 READ DAT
180 POKE PMBASE+1024+BYT
```

```
E,DAT
190 NEXT BYTE
1000 DATA 153,189,255,18
9,153
```

Uh, oh, what are all those funny-looking characters in MEMCLR$? That's just the machine-language routine. Unfortunately, it's difficult to type in, so here's a little program that will type the characters for you:

```
100 GRAPHICS 0
110 PRINT "100 DIM MEMCL
R$(36):MEMCLR$=";CHR$(34
);
120 FOR LOOP=1 TO 36
130 READ DAT
140 PRINT CHR$(27);CHR$(
DAT);
150 NEXT LOOP
170 PRINT CHR$(34)
1000 DATA 104,104,133,20
4,104,133,203,104,170,16
9,0,160,255,224,0,208,4,
104,168,169,0,145,203,13
6,192,255,208
1010 DATA 249,230,204,20
2,224,255,208,234,96
```

Run this, and then move the cursor up to the line that it prints out and press Return. Nifty, eh? You can now type in the other lines.

Now that you have the beginning of our program typed in, run it and see what happens. Nothing, right? That's because the Atari doesn't know that you want to use PMG yet. You have to tell it first, and to do that you use locations 559, 53277 and 54279. Wow, how come so many?

53277: Tells the Atari that we want to use PMG.

559: Tells it that we want to store the player/missile data in memory. This location also tells the Atari whether we are using single- or double-line resolution.

54279: Tells it where in memory to look for the player data.

Here are the values we need to POKE into each location:

```
POKE 54279,PB:POKE 559,6
2:POKE 53277,3:REM SINGL
E-LINE RESOLUTION

POKE 54279,PB:POKE 559,4
6:POKE 53277,3:REM DOUBL
E-LINE RESOLUTION
```

There are several possible numbers that can go into these memory locations. Here are two charts showing the values you can POKE into locations 54272 and 53277 and their effects:

DMACTL = 54272 (559): Choose from the following options and add the total to get the value to POKE into 559.

For	*Add*
Wide playfield	3
Standard playfield	2
Narrow playfield	1
No playfield	0
Enable missile DMA	4
Enable player DMA	8
2-Line "thick" players, or	0
1-Line "thick" players	16
Enable instruction fetch DMA	32

Choose only one.

GRACTL = 53277: POKE 53277,n. Choose from the following and add the total to get the values to POKE into 53277.

To turn on missile	1
To turn on players	2
To remember joystick values	4

Now let's add the following line to our program:

```
200 POKE 54279,PB:POKE 5
59,62:POKE 53277,3
```

Now run the whole thing and see if we get our spaceship. No? What's wrong now? We still have to tell the Atari how far across the screen we want the spaceship to appear. In other words, the Atari needs to know it's horizontal position. Right now it thinks that position is zero, which is off the screen on the left-hand side. A player can have a horizontal position between zero and 255, with 255 being off the right-hand side of the screen. Each player has a location that holds its current horizontal position. These locations are:

Player Location	*Missile Locations*
53248 (Player 0)	53252 (Missile 0)
53249 (Player 1)	53253 (Missile 1)
53250 (Player 2)	53254 (Missile 2)
53251 (Player 3)	53255 (Missile 3)

Location 53248 holds the position of player 0, so let's set it to 128 and put our spaceship in the middle of the screen:

```
210 POKE 53248,128
```

Now run everything together and, *voila*, it works! There is our spaceship in the middle of the screen. Aren't computers wonderful? Notice how the spaceship is black. If black is not your favorite color, you can easily change it.

What if we want the spaceship color to be, say, red? Each of the players has its own color register, which means that each player can be a different color. It also means that the players have different colors than the playfield (the background), which in turn means that we can have up to nine colors on the screen at the same time (more if you use the GTIA modes). For now though, let's just worry about changing the color of our spaceship.

The four player colors are stored in locations 704, 705, 706 and 707 (the missiles have the same colors as the players). Let's suppose that we want our spaceship to be a dark red.

Red has a color value of four. We also need a brightness value, so let's pick two. That will give us a dark red. Unfortunately, now we are faced with a problem. We have two values specifying the color and only one location to put them in. How do we get around this? As you may remember from our first discussion on color registers way back, to store a color and brightness value in one memory location we use the formula:

COLOR* 16+BRIGHTNESS

So we multiply 4 by 16 and add 2 to get 66 (4*16+2=64+2=66), which we then POKE into location 704:

```
220 POKE 704,66
```

Now our spaceship is red. By the way, you can play around with the spaceship position and color from the immediate mode. Just type in POKE 53248,n or POKE 704,n (where n is a value between 0 and 255) without a line number, and press Return. Notice how the spaceship isn't affected on your screen when you do this, and the text on the screen scrolls up. As I mentioned earlier, players and missiles are completely independent of the rest of the screen. You have complete control of this independence using something called Priorities which we will get to in a moment.

Before we go on, I should point out one more thing. You've probably noticed by now that when you Break the PMG program, the player remains on the screen. What if we want to get rid of it? Can we just simply turn it off? No. Prove this to yourself by typing in the following with the spaceship on the screen:

```
POKE 53277,0:POKE 559,34
```

Kind of messy, right? I won't get into the reason for this, since it isn't really important. I will, however, give you the solution. Ready? Simply set all the player and missile horizontal positions to 0. This moves everything off the screen. For our spaceship, we would use the following:

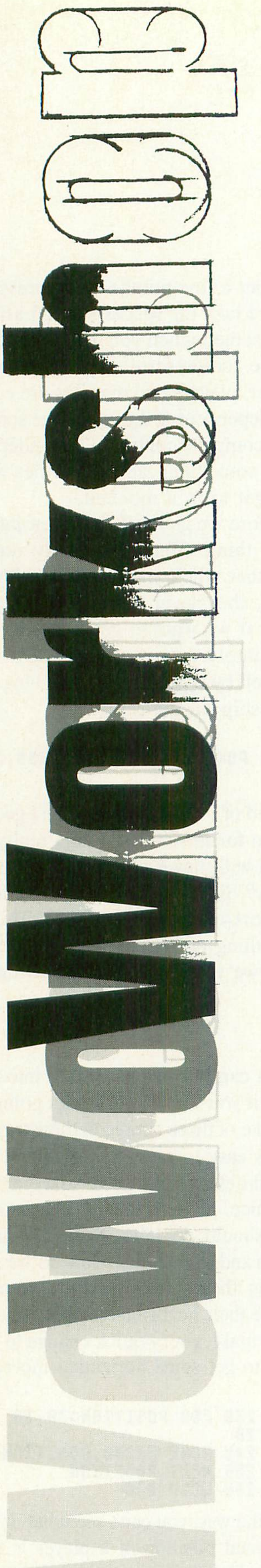
```
POKE 53248,0
```

You can put this statement into the program if you want, but I'm not going to. For the sake of these simple demonstrations, it's just as easy to press System Reset.

So far everything we've talked about is all very nice, but let's face it, players and missiles should do more than just sit on the screen and look pretty. How do we go about moving them? We've already seen how to change their horizontal position and can actually make a very quick change to our program to get some horizontal movement:

```
230 FOR POSITION=30 TO 2
20
240 POKE 53248,POSITION
250 NEXT POSITION
260 GOTO 230
```

By the way, you can't see what the current horizontal position of a player is by using PEEK. This is because the horizontal position locations (53248 through 53255) are in a special part of the computer that you can only POKE into. If you want to be able to tell what the current position is, you have to



keep track of it in a variable.

For example, whenever you change the horizontal position of player 0, you might do something like this:

```
HORZP0=X:POKE 53248,X
```

Then the variable HORZPO will always have the current horizontal position of player 0 as its value.

So much for horizontal movement, now what about vertical? Where are the memory locations for the vertical positions? Unfortunately, there aren't any. Why not? Let's go back a little bit. Do you remember that a single-line resolution player is 256 bytes high, with each of those bytes representing a bunch of dots that are as high as a graphics mode 8 dot? Well, you probably also know by now that in graphics mode 8, the screen is only 192 dots high. That means that 64 (256-192) of the bytes in a player are off the screen. So what? When Atari designed its computers, it figured out since a player was already higher than the screen, there was no point in putting in vertical position registers. After all, whoever was programming could just move the data for the spaceship (or whatever else was to be moved) up and down inside the player. It was as simple as that.

Well, this is great for all of those Atari programmers who use machine language, but for the rest of us using slower languages like BASIC, it doesn't work as fast as we need. Let's take a look at what I mean.

Put the following lines into our program and run it again:

```
230 FOR VPOS=125 TO 14 S
TEP -1
240 FOR BYTE=0 TO 4
250 POKE PMBASE+1024+VPO
S+BYTE,PEEK(PMBASE+1025+
VPOS+BYTE)
260 NEXT BYTE
270 POKE PMBASE+1030+VPO
S,0
280 NEXT VPOS
```

By now, with all this information spinning around in your head, especially all the different locations that have to be remembered, you may be getting a little confused. Let's take a break and summarize all the different locations that are used in PMG:

53277 (GRACTL): To tell the Atari that we want to use PMG.

559 (SDMCTL): To tell the Atari that we want to store PMG data in memory.

54279 (PMBASE): To tell the Atari where we're going to store the PMG data in memory.

53248 (HPOSPO): To move player 0 horizontally (specify the horizontal position).

53249 (HPOSP1): To move player 1 horizontally.

53250 (HPOSP2): To move player 2 horizontally.

53251 (HPOSP3): To move player 3 horizontally.

53252 (HPOSMO): To move missile 0 horizontally.

53253 (HPOSM1): To move missile 1 horizontally.

53254 (HPOSM2): To move missile 2 horizontally.

53255 (HPOSM3): To move missile 3 horizontally.

704 (PCOLRO): To specify the color of player/missile 0.

705 (PCOLR1): To specify the color of player/missile 1.

706 (PCOLR2): To specify the color of player/missile 2.

707 (PCOLR3): To specify the color of player/missile 3.

That's everything we've covered so far, but there's lots more still to go, so be sure you're familiar with this before you continue. This is probably a good time for you to sit down and try programming your own player and/or missile. When you've got everything under control, read on, because the best is yet to come!

Before we go on to look at all the special tricks that are available to you with PMG, let's take a closer look at movement. We've already seen how to move a player or missile both horizontally and vertically.

How about moving it with a joystick though? That shouldn't be too difficult; all we have to do is check to see which direction the joystick is moved in and then move the player or missile in the same direction.

Well, it takes a lot of time to check a joystick and figure out direction, especially if the program has other things to worry about as well.

You end up with a player that moves slowly and jerkily, and that's obviously not something you want. So, once again, it's machine language to the rescue. The next section, PMOVE, will provide us with an excellent one-routine-fits-all tool.

PMOVE

PMOVE is a machine-language routine that will automatically take care of player/missile movement for you. Let's look at exactly what makes it so wonderful. PMOVE looks at the joysticks during VBLANK and then moves the players accordingly. What this means to you is that you never have to worry about moving the players from within your program.

So, for example, if you were using PMOVE in a program and pressed the break key, the program would stop, of course, but you could still move the players around! This ability greatly increases the usefulness of PMG in your BASIC programs, as you can probably imagine. Without any further ado, then, let's see how to use PMOVE.

Basically, there are two steps to getting PMOVE to do what you want. The first one is putting it into memory and getting it ready to go. The second is telling it exactly what you want to do. We may as well do this in order, so let's present the program necessary to get PMOVE in memory.

```
100 GRAPHICS 0:? "Make s
ure you have saved a cop
y of":? "this program be
fore RUNning it":FOR X=1
 TO 1050:NEXT X
110 ? :?
120 DIM LN(2):FOR X=1 TO
 2:READ DAT:LN(X)=DAT:NE
XT X
130 DATA 41,657
140 FOR X=1 TO 2:TOT=0:N
=0:GOSUB 1000
150 FOR N=1 TO LN(X):REA
D DAT:TOT=TOT+DAT
160 IF N/25<>INT(N/25) T
HEN 190
170 T=TOT:TOT=0:READ DAT
:IF DAT<>T THEN ? "...ER
ROR":STOP
180 GOSUB 1000
190 NEXT N:READ DAT:IF D
AT<>TOT THEN ? "...ERROR
":STOP
200 NEXT X
210 RESTORE 20000
220 FOR X=1 TO 2:L=29500
+500*X:GOSUB 1010
230 FOR N=1 TO LN(X):REA
D DAT:? CHR$(27);CHR$(DA
T);
240 IF N/25=INT(N/25) TH
EN READ DAT
250 IF N/90=INT(N/90) TH
EN GOSUB 1020:L=L+10:GOS
UB 1010
260 NEXT N:READ DAT:GOSU
B 1020
270 NEXT X
280 END
1000 ? :? "CHECKING LINE
 ";19000+1000*X+10*INT(N
/25);:RETURN
1010 GRAPHICS 0:POSITION
 2,4:? L;" MLANG$=";CHR$
(34);:RETURN
1020 ? CHR$(34);":RETURN
":? "CONT":POSITION 0,0:
POKE 842,13:STOP
1030 POKE 842,12:RETURN
20000 DATA 104,104,133,2
07,104,133,206,104,133,2
09,104,133,208,104,170,1
60,255,138,208,2,104,168
,177,206,145,3719
20010 DATA 208,136,192,2
55,208,247,230,207,230,2
09,202,224,255,208,233,9
6,3340
21000 DATA 104,104,104,1
41,188,6,104,104,141,228
,6,141,231,6,141,234,6,1
41,237,6,238,237,6,141,2
40,3235
21010 DATA 6,238,240,6,1
69,127,141,199,6,162,9,1
60,4,173,47,2,41,16,240,
9,169,255,141,199,6,2765
21020 DATA 162,19,160,8,
140,200,6,160,9,189,206,
6,153,189,6,202,136,16,2
46,169,7,174,240,6,160,2
969
21030 DATA 108,32,92,228
,96,32,238,6,189,152,6,2
4,109,200,6,168,205,199,
6,144,3,172,199,6,189,28
09
21040 DATA 152,6,56,237,
200,6,141,201,6,136,177,
204,200,145,204,136,240,
5,204,201,6,176,242,169,
0,3450
21050 DATA 145,204,96,32
,238,6,189,152,6,56,237,
200,6,168,176,2,160,0,18
9,152,6,24,109,200,6,275
9
21060 DATA 141,201,6,200
,177,204,136,145,204,200
,204,199,6,240,7,204,201
,6,144,239,240,237,169,0
,145,3855
21070 DATA 204,96,138,72
,162,4,32,238,6,104,170,
189,160,6,56,237,200,6,1
68,176,2,160,0,189,160,2
```

```
935
21080 DATA 6,24,109,200,
6,141,201,6,136,177,204,
61,202,6,145,204,200,200
,189,202,6,73,255,49,204
,3206
21090 DATA 136,136,17,20
4,145,204,200,200,204,19
9,6,176,7,204,201,6,144,
221,240,219,189,202,6,49
,204,3719
21100 DATA 145,204,136,1
89,202,6,49,204,145,204,
96,138,72,162,4,32,238,6
,104,170,189,160,6,24,10
9,2994
21110 DATA 200,6,168,205
,199,6,144,3,172,199,6,1
89,160,6,56,237,200,6,14
1,201,6,200,177,204,61,3
152
21120 DATA 202,6,145,204
,136,136,189,202,6,73,25
5,49,204,200,200,17,204,
145,204,136,136,240,5,20
4,201,3699
21130 DATA 6,176,224,189
,202,6,49,204,145,204,20
0,189,202,6,49,204,145,2
04,96,189,189,6,133,204,
24,3445
21140 DATA 216,173,188,6
,125,194,6,133,205,169,0
,133,77,96,162,0,188,128
,6,48,106,185,120,2,41,2
707
21150 DATA 8,208,23,189,
148,6,221,136,6,240,43,1
69,0,133,77,254,148,6,18
9,148,6,157,0,208,208,29
31
21160 DATA 28,185,120,2,
41,4,208,21,169,0,133,77
,189,148,6,221,132,6,240
,9,222,148,6,189,148,265
2
21170 DATA 6,157,0,208,1
88,128,6,185,120,2,41,2,
208,17,189,152,6,221,144
,6,240,30,254,152,6,2668
21180 DATA 32,229,6,138,
16,21,185,120,2,41,1,208
,14,189,152,6,221,140,6,
240,6,222,152,6,32,2385
21190 DATA 226,6,232,224
,4,208,140,162,0,189,164
,6,240,83,189,168,6,240,
50,16,23,222,156,6,222,3
182
21200 DATA 156,6,189,156
,6,157,4,208,201,47,176,
32,169,0,157,164,6,240,5
3,254,156,6,254,156,6,29
59
21210 DATA 189,156,6,157
,4,208,201,208,144,9,169
,0,157,164,6,240,106,208
,196,189,172,6,240,57,16
,3208
21220 DATA 23,222,160,6,
222,160,6,32,232,6,189,1
60,6,201,16,176,39,169,0
,157,164,6,240,74,254,29
20
21230 DATA 160,6,254,160
,6,32,235,6,189,160,6,24
,216,105,16,205,199,6,17
6,4,41,240,208,7,169,283
0
21240 DATA 0,157,164,6,2
40,42,189,176,6,61,0,208
,240,13,169,255,157,176,
6,157,184,6,169,0,157,29
38
21250 DATA 164,6,189,180
,6,61,8,208,240,13,169,2
55,157,180,6,157,184,6,1
69,0,157,164,6,232,224,3
141
21260 DATA 4,208,145,76,
98,228,0,759
```

This may look a little funny, so let me explain exactly what's going on. PMOVE is, obviously, a machine-language routine, which means that it is made up of a whole bunch of numbers. You can see those numbers in the above DATA statements. Unfortunately, numbers take up a lot of memory. There's a way to store machine language that doesn't take up nearly as much memory, and that's to use ATASCII characters (letters, digits, graphics characters, etc.). Unfortunately (again), it's very difficult for you to type in such characters. What we've done, therefore, is to present a combination of the two. When you run the above program, it will read all the DATA statements, checking to make sure you typed in the numbers correctly, and then it will change the numbers to graphics characters, right before your very eyes! Once it's done, you should LIST the new lines to disk with the following command.

LIST"D:PMOVE",30000,30570

Still with me? Now type in the following, NEW, RETURN, then:

```
10 FOR BYTE=1 TO 40
20 READ DAT
30 POKE 1737+BYTE,DAT
40 NEXT BYTE
50 DATA 252,243,207,63,0
,128,0,128,128,2,2,3,3,1
,0,0,0,0,0,4,5,6,7,3,76,
128,64,76,80,64,76,177,6
4,76,5,65,76
60 DATA 88,65,0
100 DIM MEMCLR$(36):MEMC
LR$="hh┐Lh┐Kh✱)♥ ◆♥P┤h(
)♥┌K╱@)PyfLJ◆)P j◆"
110 PB=PEEK(740)-8
120 POKE 106,PB-4
130 GRAPHICS 0
140 PMBASE=PB*256
```

```
150 X=USR(ADR(MEMCLR$),P
MBASE,2048)
151 DIM MLANG$(90),MOVME
M$(41)
152 GOSUB 30000:MOVMEM$=
MLANG$
153 MEM=PMBASE
154 FOR SEC=0 TO 7
155 GOSUB 30500+10*SEC
156 X=USR(ADR(MOVMEM$),A
DR(MLANG$),MEM,LEN(MLANG
$)-1)
157 MEM=MEM+LEN(MLANG$)
158 NEXT SEC
200 POKE 54279,PB:POKE 5
59,62:POKE 53277,3
205 X=USR(PMBASE,PB,PB)
999 END
```

Remember that Line 100 was created previously. And ENTER PMOVE back into the computer with the following:

ENTER: "D:MOVE"

Well, so much for the hard part. What you now have in your computer is a program that will set up the computer for PMG and also get PMOVE ready to go as well. Here's a breakdown of exactly what it does:

10-60: These lines set up some data for PMOVE in page 6. This is actually the equivalent of initializing some variables.

100-150: We've seen these lines already in our previous programs.

151: MLANG$ is used for temporary storage as you'll soon see. MOVMEM$ will hold a routine to move memory.

152: The GOSUB 30000 puts the move-memory routine into MLANG$ where we transfer it into MOVMEM$.

153: We initialize MEM to point to PMBASE. This is where we will put PMOVE. If you're using double-line resolution, then you'll have to change this line to MEM=PMBASE−512.

154: PMOVE is in a total of eight segments.

155: We GOSUB to transfer each segment into MLANG$.

156: This is the USR instruction that makes MOVMEM$ work. The MOVMEM format is: X=USR(ADR(MOVMEM$), FROM, TO, LENGTH−1) where FROM is the beginning of the memory segment to be moved, TO is the beginning of the memory area to move to, LENGTH is the length of the segment to be moved.

157: Now we update MEM to point to the end of the segment we just moved.

158: That's the end of our loop.

200: You've seen this before, but note that it must come before the following statement (205).

205: This is the statement that gets PMOVE going. Its format is: X=USR(START,PB,ST) where START is the starting address of the PMOVE routine, PB is the high byte of the P/M address, ST is the high byte of START.

Note that PMOVE must being on a page boundary (START must be an even multiple of 256).

By the way, after you run any program with PMOVE in it, you must press System Reset before running it again.

Now (finally) we're ready to actually do something with PMOVE. But first (is there no end?), we have to tell PMOVE what we expect it to do. We do this by POKEing various values into memory. Let's look at these locations and values:

1664-1667: PMOVE lets you decide which players to control with which joysticks. You can control one player with one joystick, four players with four joysticks, four players with one joystick, and so forth. If you want to attach player n (n is 0, 1, 2 or 3) to joystick x (x is 0, 1, 2 or 3), then:

POKE 1664+n,x

So if we wanted joystick 0 to control players 1 and 2, we would:

POKE 1665,0:POKE 1666,0

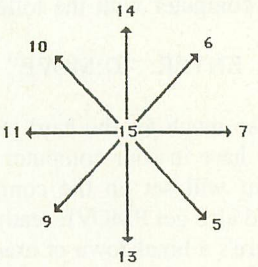
If you don't want player n to be attached to any joystick:

POKE 1664+n,255

Finally, there is also a way to make a player move without having it connected to a joystick. I suggest, however, that you skip over this until you understand everything else. What you do:

POKE 1664+n where n is a joystick number between 43 and 46. Then:

POKE 675+[the joystick value] with the direction you want the player to move. Possible directions are illustrated below:



Joystick values

As you may have noticed, these are the same as the joystick directions. All you are doing with all of this is tricking PMOVE into looking at a joystick that doesn't exist. As we said before, however, for the benefit of those of you who read this despite previous warnings, don't worry too much about it until you've played around with PMOVE a bit.

1668-1671: These locations let you specify a left-hand limit for each player. For example, if you don't want player 3 to be able to go left of horizontal position 48 (the left-hand edge of the screen), you would:

POKE 1668+3,48

1672-1675: These locations let you specify a right-hand limit for each player, and work in the same way as 1668-1671 above.

1676-1679: These locations let you specify a top limit for each player.

1680-1683: These locations let you specify a bottom limit for each player.

1684-1687: These locations specify the horizontal position of each player. Unlike the previous locations, however, PMOVE will update these locations as things move around on the screen. You must set the initial positions. So, if you were starting player 1 at horizontal position 128, you would POKE 1684+1, 128. Then, after you had gotten going, PEEKing location 1685 would tell you the current position of player 1.

1688-1687: These locations specify the current vertical position of each player. They work in the same way as locations 1684-1687 above.

1692-1695: Now we move on to the missiles. These are the locations for the current horizontal position of each missile.

1696-1699: These locations specify the current vertical positions of each missile.

1700-1703: These locations allow you to start and stop each missile. POKE 1700+n, 1 will start missile n moving. POKE 1700+n, 0 will stop it.

1704-1707: These locations allow you to specify the horizontal direction for each missile: 128 = left, 1 = right, 0 = no horizontal movement. I'll explain this a little more with the next set of locations.

1708-1711: These locations allow you to specify the vertical direction for each missile: 128 = up, 1 = down, 0 = no vertical movement.

Let's take a look at how locations 1704-1711 work. Suppose you wanted missile 0 to move diagonally up and to the right:

POKE 1704,1 = the horizontal direction (right)

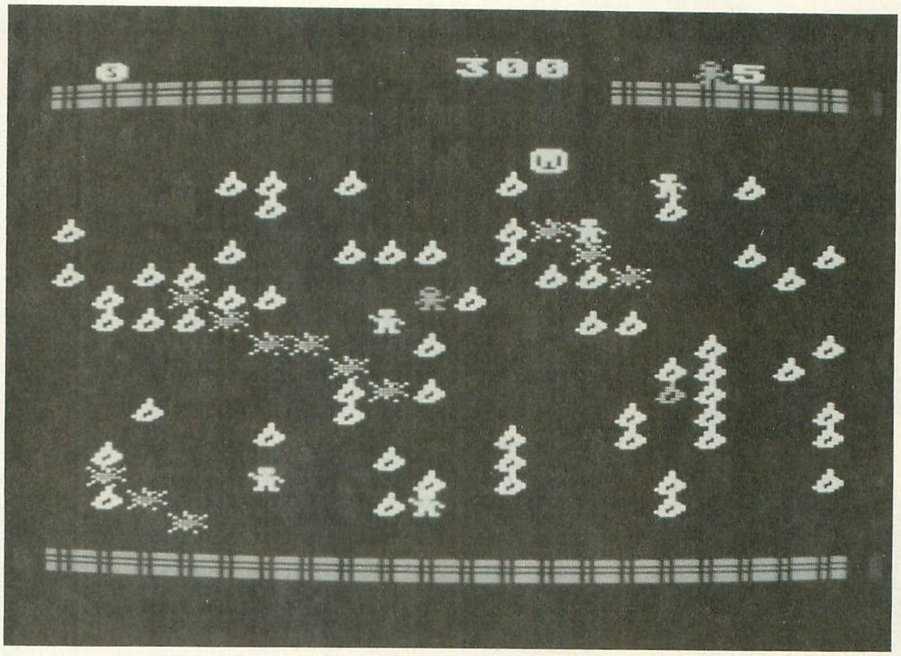
POKE 1708,128 = the vertical direction (up)

You would also have to POKE 1700,1 when you were ready to start the missile moving. If you wanted to move missile 3 straight up, it would be:

POKE 1707,0 and POKE 1711,128.

continued on page 50

INFERNO



continued from page 13

```
VG 10 REM INFERNO BY FRANK MARTONE 1/87
IX 20 REM COPYRIGHT 1988 BY ANALOG COMPUT
   ING
GZ 30 GRAPHICS 17:POSITION 1,10:? #6;"SYS
   TEM INITIALIZING ":POKE 708,14:POKE 71
   2,135
RC 40 GOSUB 1650
TO 50 SCR=PEEK(88)+256*PEEK(89):GOTO 2020
YX 60 GRAPHICS 17:POKE 756,CH/256:SETCOLO
   R 0,8,10:POKE 77,0
CO 70 SC=0:PN=5:SOUND 0,0,0,0:WP=300:FMEN
   =5:DL=20:AP=21:FN=1:BU=1:SA=0:LF=0:SP=
   0:PV=100:GOTO 1130
XY 80 POKE 559,0:P1=INT(RND(0)*14)+4:P2=I
   NT(RND(0)*17)+3:P3=INT(RND(0)*14)+4:P4
   =INT(RND(0)*17)+3:P5=INT(RND(0)*14)+4
ML 90 P6=INT(RND(0)*17)+3:P7=INT(RND(0)*1
   4)+4:P8=INT(RND(0)*17)+3:P9=INT(RND(0)
   *14)+4
IL 100 P10=INT(RND(0)*17)+3:POSITION 0,1:
   ? #6;"▄▄▄▄▄▄▄▄▄▄▄        ▄▄▄▄▄▄▄▄▄▄"
TE 110 POSITION 0,22:? #6;"▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
   ▄▄▄▄▄▄▄▄▄▄":POSITION RND(0)*16+4,RND(0)*17
   +3
IX 120 POSITION P1,P2:GOSUB 700:POSITION
   P3,P4:GOSUB 710:POSITION P5,P6:GOSUB 7
   20:POSITION P7,P8:GOSUB 730
EL 130 POSITION P9,P10:GOSUB 740
KB 140 X=9:Y=10:SETCOLOR 0,8,10:L1=1:L2=1
   :L3=1:L4=1:L5=1:POSITION X,Y:? #6;" "
CQ 150 POKE SCR+X+20*Y,10:SOUND 0,0,0,0:P
   OSITION 16,0:? #6;"♟"
GI 160 POKE 559,34:POSITION RND(0)*15+2,R
   ND(0)*17+3:? #6;"J":POKE 710,117
VI 170 REM MAIN-LOOP
ZK 180 SCR=PEEK(88)+256*PEEK(89)
DJ 190 TP=SCR+X+20*Y:SOUND 1,0,0,0:POKE 7
   7,0
QR 200 ST=STICK(0):TR=STRIG(0):POKE 709,5
   6:SETCOLOR 0,8,10:POKE 711,14
OW 210 IF LF=5 THEN POKE 710,212
RW 220 POSITION 1,0:? #6;SC:POSITION 17,0
   :? #6;FMEN:POSITION 10,0:? #6;WP;"  ";
JN 230 IF L1=1 THEN LOCATE P1,P2,W1:IF W1
   <>59 THEN GOSUB 750
VS 240 IF L2=1 THEN LOCATE P3,P4,W2:IF W2
   <>58 THEN GOSUB 780
EF 250 IF L3=1 THEN LOCATE P5,P6,W1:IF W1
   <>63 THEN GOSUB 810
SQ 260 IF L4=1 THEN LOCATE P7,P8,W1:IF W1
   <>47 THEN GOSUB 840
EQ 270 IF L5=1 THEN LOCATE P9,P10,W1:IF W
   1<>43 THEN GOSUB 870
KY 280 SOUND 1,0,0,0:IF TR=0 THEN GOSUB 2
   730
PK 290 IF ST=14 THEN POKE TP,0:Y=Y-1
LV 300 IF ST=13 THEN POKE TP,0:Y=Y+1
OW 310 IF ST=7 THEN POKE TP,0:X=X+1
LF 320 IF ST=11 THEN POKE TP,0:X=X-1
WW 330 IF TR=0 THEN GOSUB 2730
DI 340 LOCATE X,Y,BC:GOSUB 580
JZ 350 IF BC=250 THEN GOTO 920
YM 360 IF BC=1 THEN GOTO 920
KB 370 IF BC=74 THEN FOR W=55 TO 40 STEP
   -3:SOUND 0,W*8,10,8:NEXT W:WP=WP+100
GT 380 TP=SCR+X+20*Y:POKE 712,0:POKE TP,1
   0+128
HE 390 IF X>17 THEN POKE TP,0:X=X-1
GV 400 IF X<2 THEN POKE TP,0:X=X+1
GC 410 IF Y<4 AND X>6 AND X<14 AND LF=5 T
   HEN GOTO 1240
KF 420 IF Y<4 THEN POKE TP,0:Y=Y+1
KD 430 IF Y>19 THEN POKE TP,0:Y=Y-1
GF 440 POKE SCR+RND(0)*300+100,58+192:SOU
   ND 0,F,8,3:F=F+1:POKE 711,F:IF F=80 TH
   EN F=75
SE 450 IF PEEK(53770)<DL THEN GOSUB 540
NK 460 IF FN>2 THEN DL=40
RO 470 IF FN>4 THEN DL=80
KM 480 IF FN>5 THEN DL=190
AV 490 IF FN>6 AND BU=3 THEN DL=255
OU 500 GOTO 180
VI 510 REM END OF MAIN-LOOP
QU 520 REM
TH 530 REM EXPLOSION
QN 540 E1=RND(0)*15:E2=RND(0)*15+3
QC 550 POSITION E1,E2:? #6;"(":POSITION E
   1+1,E2+1:? #6;"(":POSITION E1+2,E2+2:?
    #6;"(":FOR D=1 TO 5:NEXT D
FT 560 POKE 712,14:COLOR 1:PLOT E1,E2:DRA
   WTO E1+2,E2+2:SOUND 0,100,0,10:POKE 71
   2,0
ZP 570 RETURN
OH 580 IF BC=59 THEN FOR W=15 TO 0 STEP -
   11:FOR U=20 TO 33:SOUND 0,U,10,W:NEXT
   U:NEXT W:SA=SA+1:L1=0:LF=LF+1
OB 590 IF BC=59 THEN SP=SP+1:RETURN
QX 600 IF BC=58 THEN FOR W=15 TO 0 STEP -
   11:FOR U=20 TO 33:SOUND 0,U,10,W:NEXT
   U:NEXT W:SA=SA+1:L2=0:LF=LF+1
NA 610 IF BC=58 THEN SP=SP+1:RETURN
SV 620 IF BC=63 THEN FOR W=15 TO 0 STEP -
   11:FOR U=20 TO 33:SOUND 0,U,10,W:NEXT
   U:NEXT W:SA=SA+1:L3=0:LF=LF+1
LH 630 IF BC=63 THEN SP=SP+1:RETURN
XQ 640 IF BC=47 THEN FOR W=15 TO 0 STEP -
   11:FOR U=20 TO 33:SOUND 0,U,10,W:NEXT
   U:NEXT W:SA=SA+1:L4=0:LF=LF+1
ML 650 IF BC=47 THEN SP=SP+1:RETURN
ZP 660 IF BC=43 THEN FOR W=15 TO 0 STEP -
```

```
   11:FOR U=20 TO 33:SOUND 0,U,10,W:NEXT
   U:NEXT W:SA=SA+1:L5=0:LF=LF+1
KT 670 IF BC=43 THEN SP=SP+1:RETURN
ZS 680 RETURN
DK 690 POSITION 13,6:? #6;"█":GOSUB 1380
BZ 700 ? #6;";":RETURN
BQ 710 ? #6;":":RETURN
DV 720 ? #6;"?":RETURN
XD 730 ? #6;"/":RETURN
VN 740 ? #6;"+":RETURN
FO 750 POKE 712,56:FOR R=1 TO 20:SOUND 0,
   R*5,6,10:NEXT R:SOUND 0,0,0,0:L1=0
HP 760 POSITION P1,P2:? #6;"█":GOSUB 900:
   POSITION P1,P2:? #6;"█":GOSUB 900:POSI
   TION P1,P2:? #6;"█":GOSUB 900
CS 770 POKE 712,0:P1=0:P2=0:LF=LF+1:RETUR
   N
IM 780 POKE 712,56:FOR R=1 TO 20:SOUND 0,
   R*5,6,10:NEXT R:SOUND 0,0,0,0:L2=0
FD 790 POSITION P3,P4:? #6;"█":GOSUB 900:
   POSITION P3,P4:? #6;"█":GOSUB 900:POSI
   TION P3,P4:? #6;"█":GOSUB 900
FF 800 POKE 712,0:P3=0:P4=0:LF=LF+1:RETUR
   N
KR 810 POKE 712,56:FOR R=1 TO 20:SOUND 0,
   R*5,6,10:NEXT R:SOUND 0,0,0,0:L3=0
BY 820 POSITION P5,P6:? #6;"█":GOSUB 900:
   POSITION P5,P6:? #6;"█":GOSUB 900:POSI
   TION P5,P6:? #6;"█":GOSUB 900
IL 830 POKE 712,0:P5=0:P6=0:LF=LF+1:RETUR
   N
LH 840 POKE 712,52:FOR R=1 TO 20:SOUND 0,
   R*5,6,10:NEXT R:SOUND 0,0,0,0:L4=0
ZM 850 POSITION P7,P8:? #6;"█":GOSUB 900:
   POSITION P7,P8:? #6;"█":GOSUB 900:POSI
   TION P7,P8:? #6;"█":GOSUB 900
LR 860 POKE 712,0:P7=0:P8=0:LF=LF+1:RETUR
   N
OF 870 POKE 712,52:FOR R=1 TO 20:SOUND 0,
   R*5,6,10:NEXT R:SOUND 0,0,0,0:L5=0
JT 880 POSITION P9,P10:? #6;"█":GOSUB 900
   :POSITION P9,P10:? #6;"█":GOSUB 900:PO
   SITION P9,P10:? #6;"█":GOSUB 900
PZ 890 POKE 712,0:P9=0:P10=0:LF=LF+1:RETU
   RN
WM 900 FOR D=1 TO 70:NEXT D:RETURN
OX 910 REM FIREMAN KILLED
ST 920 POKE SCR+X+20*Y,10
ZJ 930 FOR R=15 TO 0 STEP -1:FOR S=1 TO 4
   :SOUND 0,5,8,R:POKE 712,R:NEXT S:NEXT
   R
AB 940 SETCOLOR 0,12,14:SOUND 0,0,0,0:SOU
   ND 1,0,0,0
IN 950 FMEN=FMEN-1
SL 960 POSITION 1,0:? #6;SC:POSITION 17,0
   :? #6;FMEN:POSITION 10,0:? #6;WP;"  ";
HT 970 POKE SCR+X+20*Y,5
FH 980 FOR L=0 TO 95 STEP 5.9:SOUND 0,L,0
   ,10:SOUND 1,L,10,5:SOUND 2,L,8,3:POKE
   712,255:POKE 712,0:NEXT L:POKE 712,0
EM 990 POKE SCR+X+20*Y,0:X=10:Y=10:FOR S=
   0 TO 2:SOUND S,0,0,0:NEXT S
RV 1000 IF FMEN=0 THEN GOTO 2330
IO 1010 FOR D=1 TO 15:SOUND 0,D*4,8,2:NEX
   T D
MM 1020 POSITION X,Y:? #6;" "
QT 1030 GOTO 180
KO 1040 POSITION 1,0:? #6;"floor ";FN:POS
   ITION 10,0:? #6;"building ";BU
NE 1050 FOR F=1 TO 3:POSITION 2,AP:? #6;"
   .":FOR R=1 TO 30:NEXT R:SOUND 0,50,10,
   10:POSITION 2,AP:? #6;" "
HK 1060 FOR R=1 TO 30:NEXT R
BT 1070 POKE 709,221
DE 1080 SOUND 0,100,10,10:NEXT F:SOUND 0,
   0,0,0
QJ 1090 POSITION 2,AP:? #6;".":FOR D=1 TO
    350:NEXT D
AA 1100 AP=AP-3:FN=FN+1
BL 1110 GRAPHICS 17:POKE 756,CH/256:GOTO
   80
PJ 1120 GOTO 1050
OQ 1130 GRAPHICS 17:POKE 756,CH/256:J=2:I
   F FN=8 THEN GOTO 1400
FH 1140 POKE 559,0:POKE 712,97:POKE 710,3
   4:LF=0:POKE 708,14
PV 1150 IF BU=2 THEN POKE 710,135:POKE 70
   8,44
DO 1160 IF BU=3 THEN POKE 710,0:POKE 708,
   255
IK 1170 IF BU>3 THEN GOTO 2510
QQ 1180 POSITION 0,J:? #6;"      ████████████████
   ███████":J=J+1
BN 1190 POSITION 0,J:? #6;"      ███e███e███e█
   █e███":J=J+1
FI 1200 POSITION 0,J:? #6;"      ███z███z███z█
   █z███":J=J+1
IB 1210 IF J>21 THEN POKE 559,34:GOTO 104
   0
RK 1220 GOTO 1180
CV 1230 REM BONUS
IS 1240 GRAPHICS 18:POKE 756,CH/256:IF SP
   =0 THEN FN=FN+1:GOTO 1130
SG 1250 G=4:POKE 710,135:POKE 709,44:POKE
    710,14:POKE 708,87:SOUND 1,0,0,0:SOUN
   D 0,0,0,0
UI 1260 D=PEEK(560)+256*PEEK(561)+4:POKE
   D+4,6:POKE 712,3
CL 1270 POSITION 4,2:? #6;"floor secured"
   :POSITION 8,3:? #6;"bonus"
LU 1280 POSITION 0,0:? #6;"_______________
   _______":POSITION 0,10:? #6;"__________
   ____________"
YE 1290 FOR T=1 TO SP:POSITION G,6:? #6;"
   █":G=G+1:FOR I=15 TO 0 STEP -1:SOUND 0
   ,10,10,I:NEXT I
JM 1300 NEXT T
IE 1310 POSITION 9,6:? #6;"█":GOSUB 1380
RI 1320 POSITION 10,6:? #6;PV:GOSUB 1380
NL 1330 POSITION 13,6:? #6;"█":GOSUB 1380
LC 1340 FOR H=1 TO 4:POSITION 15,6:? #6;P
   V*SP:GOSUB 1390
TX 1350 POSITION 14,6:? #6;"        ":FOR
   E=1 TO 50:NEXT E:NEXT H
VL 1360 SC=SC+PV*SP:SP=0
KK 1370 SP=0:GRAPHICS 17:POKE 756,CH/256:
   GOTO 1130
NU 1380 FOR I=15 TO 0 STEP -1:SOUND 0,50,
   10,I:FOR D=1 TO 2:NEXT D:NEXT I:RETURN
    :RETURN
DZ 1390 FOR I=15 TO 0 STEP -1:SOUND 0,20,
   10,I:FOR D=1 TO 2:NEXT D:NEXT I:RETURN
YA 1400 GRAPHICS 18:POKE 756,CH/256
KT 1410 G=1:K=6:POKE 711,115:POKE 709,44:
   POKE 710,14:POKE 708,87
UL 1420 D=PEEK(560)+256*PEEK(561)+4:POKE
   D+5,6
MK 1430 POSITION 1,2:? #6;"building ";BU;
   " secured"
WF 1440 SC=SC+SA*1000:FMEN=FMEN+1:FOR R=1
    TO 3
YU 1450 FOR T=-25 TO 25 STEP 1:SOUND 0,AB
   S(T)+35,10,10:POKE 709,ABS(T):NEXT T:N
   EXT R:SOUND 0,0,0,0
SC 1460 FOR D=1 TO 100:NEXT D:POSITION 0,
   4:? #6;" BONUS 1 FREE MAN *":GOSUB 139
   0
RZ 1470 FOR T=1 TO SA:POSITION G,K:? #6;"
   █":G=G+1:FOR I=15 TO 0 STEP -3:SOUND 0
   ,10,10,I:NEXT I
YW 1480 IF G=10 THEN G=1:K=K+1
KP 1490 NEXT T
HM 1500 POSITION 11,8:? #6;"█ 1000":GOSUB
    1380
TO 1510 FOR E=1 TO 5:POSITION 8,11:? #6;1
   000*SA:GOSUB 1390
PG 1520 POSITION 8,11:? #6;"          ":
   FOR D=1 TO 50:NEXT D:NEXT E
RG 1530 IF SA=35 THEN GOSUB 2650
WP 1540 POSITION 8,11:? #6;1000*SA
EP 1550 IF SA=35 THEN POSITION 8,11:? #6;
   1000*SA+50000
HC 1560 FOR D=15 TO 0 STEP -1:SOUND 0,255
   ,10,D:SOUND 1,254,10,D:FOR E=1 TO 20:N
   EXT E:NEXT D
ZV 1570 FOR D=255 TO 240 STEP -1:POKE 708
   ,D:POKE 709,D:POKE 710,D:POKE 711,D:PO
   KE 712,D:NEXT D
ZP 1580 SA=0:BU=BU+1:FN=1:PV=PV+100:AP=21
   :GOTO 1130
QU 1590 GOTO 1510
NC 1600 RESTORE 1640
WR 1610 READ A,B:SOUND 0,A,10,10:SOUND 1,
   B,10,10:FOR D=1 TO 20:NEXT D
SS 1620 IF A=1 THEN RETURN
QQ 1630 GOTO 1610
NA 1640 DATA 60,144,182,68,0,243,72,182,0
   ,81,193,0,91,217,1,1
GU 1650 REM REDEFINE CHARACTERS
TT 1660 CH=(PEEK(106)-8)*256:FOR I=0 TO 7
   :POKE CH+I,0:NEXT I
CC 1670 FOR I=0 TO 512:POKE CH+I,PEEK(573
   44+I):NEXT I
PF 1680 RESTORE 1750
UB 1690 READ A:IF A<0 THEN RETURN
YV 1700 POKE 708,A
UD 1710 FOR J=0 TO 7:READ B:POKE CH+A*8+J
   ,B:NEXT J
MJ 1720 POKE 711,A+50
US 1730 GOTO 1690
AK 1740 REM CHARACTER DATA
GP 1750 DATA 10,24,36,60,25,62,88,20,54
```

```
WI 1760 DATA 1,16,157,74,60,126,57,80,132
VE 1770 DATA 14,0,4,6,255,6,4,0,0
TG 1780 DATA 4,0,0,4,0,16,8,64,0
KZ 1790 DATA 5,1,66,0,34,8,64,20,128
JZ 1800 DATA 6,0,0,4,80,8,20,0,0
FG 1810 DATA 58,8,8,24,28,54,38,108,56
WL 1820 DATA 2,215,215,0,215,0,215,215,21
   5
YE 1830 DATA 55,56,56,144,124,58,56,40,10
   8
RV 1840 DATA 42,124,254,186,186,170,170,1
   98,124
YB 1850 DATA 27,56,56,144,124,58,56,40,10
   8
HR 1860 DATA 49,16,64,8,1,20,64,2,16
WO 1870 DATA 5,0,146,84,0,198,0,84,146
JD 1880 DATA 32,255,129,129,255,129,129,1
   89,255
BV 1890 DATA 29,0,6,39,23,86,124,18,48
WE 1900 DATA 30,0,0,8,11,159,255,0,0
QL 1910 DATA 12,1,64,8,32,57,60,124,56
UP 1920 DATA 11,56,56,144,124,58,56,40,10
   8
VO 1930 DATA 31,56,56,144,124,58,56,40,10
   8
WR 1940 DATA 15,56,56,144,124,58,56,40,10
   8
XR 1950 DATA 26,56,56,144,124,58,56,40,10
   8
RC 1960 DATA 7,0,0,126,0,0,126,0,0
ZR 1970 DATA 28,0,66,36,24,24,36,66,0
LT 1980 DATA 8,0,2,80,56,24,4,64,0
LY 1990 DATA 9,255,171,255,171,255,171,25
   5,255
EF 2000 DATA -1
BV 2010 REM TITLE PAGE
IN 2020 GRAPHICS 17:POKE 710,0:POKE 756,C
   H/256:POKE 708,145
VV 2030 DL=PEEK(560)+256*PEEK(561)+4:POKE
    DL+7,7:FOR I=1 TO 6:? #6:NEXT I:POKE
   DL+8,3:POKE DL+12,3
KO 2040 ? #6;"           inferno"
XY 2050 ? #6;"                BY FRANK MARTO
   NE"
LP 2060 POSITION 0,5:? #6;"______________
   ______"
QN 2070 POSITION 0,9:? #6;"______________
   ______":POKE 711,89
MS 2080 POKE 708,172:FOR D=15 TO 0 STEP -
   0.2:SOUND 0,255,10,D:SOUND 1,254,10,D:
   FOR E=1 TO 2:NEXT E:POKE 53274,D+30
LE 2090 NEXT D:POSITION 11,12:? #6;"YOU A
   RE THE FIREMAN"
OC 2100 POSITION 3,17:? #6;"* ":FOR D=1 T
   O 230:NEXT D
TF 2110 POSITION 4,12:? #6;"YOU MUST SAVE
    THE TRAPPED PEOPLE"
UM 2120 POSITION 12,16:? #6;"Z Z Z":POSIT
   ION 13,17:? #6;"W W":POSITION 13,18:?
   #6;"Z Z Z":FOR K=1 TO 7
OT 2130 POSITION 9,17:? #6;".":FOR E=1 TO
    40:NEXT E:POSITION 9,17:? #6;" ":FOR
   E=1 TO 40:NEXT E:NEXT K
OZ 2140 POSITION 2,12:? #6;"BEFORE THEY A
   RE KILLED IN THE INFERNO":FOR D=1 TO 1
   00:NEXT D:P1=13:P2=17
RC 2150 POSITION P1,P2:? #6;",":GOSUB 900
   :POSITION P1,P2:? #6;"=":GOSUB 900:POS
   ITION P1,P2:? #6;">":GOSUB 900
GX 2160 P1=P1+2:IF P1=17 THEN GOTO 2180
QW 2170 GOTO 2150
OG 2180 POSITION 0,12:? #6;"        PRESS
   FIRE TO USE EXTINGUISHER        ":X=3:Y=
   17:FOR D=1 TO 140:NEXT D
DP 2190 POSITION X,Y-1:? #6;"Q":POSITION
   X-1,Y:? #6;"Q":POSITION X+1,Y:? #6;"Q"
   :POSITION X,Y+1:? #6;"Q"
ME 2200 FOR E=1 TO 140
VW 2210 NEXT E:POSITION X,Y-1:? #6;" ":PO
   SITION X-1.Y:? #6:" ":POSITION X+1,Y:?
    #6;" ":POSITION X,Y+1:? #6;" "
QL 2220 FOR E=1 TO 200:NEXT E
HP 2230 POSITION 0,12:? #6;"TO REFILL EXT
   INGUISHER PICK UP CANISTER    ":FOR D=
   1 TO 100:NEXT D
IN 2240 POSITION 5,17:? #6;"J":FOR D=1 TO
    200:NEXT D
GX 2250 POSITION 12,16:? #6;"     ":POSIT
   ION 13,17:? #6;"   ":POSITION 13,18:?
   #6;"     "
UU 2260 POSITION 10,16:? #6;") ) ) ":POSI
   TION 10,17:? #6;") ) )      ":POSITION 1
   0,18:? #6;") ) )      "
ML 2270 POSITION 0,12:? #6;"          COMPL
   ETE ALL 3 BUILDINGS         ":FOR D=1 TO
    450:NEXT D
```

```
TS 2280 POSITION 0,12:? #6;"
     PRESS START                  ":FOR D
   =1 TO 20:NEXT D
BH 2290 IF PEEK(53279)=6 THEN GOTO 2490
SC 2300 IF STRIG(0)=0 THEN GOTO 2490
DY 2310 POSITION 0,12:? #6;"
                                 ":FOR D=
   1 TO 20:NEXT D
SK 2320 GOTO 2280
XK 2330 FOR R=15 TO 0 STEP -1:FOR S=1 TO
   4:SOUND 0,5,10,R:POKE 712,R:NEXT S:NEX
   T R
NF 2340 FOR I=5 TO 15:POSITION 0,I:? #6;"
                         ":NEXT I
BD 2350 D=PEEK(560)+256*PEEK(561)+4:POKE
   D+8,7
AQ 2360 POSITION 0,5:? #6:? #6:? #6;"
     game over ":FOR I=255 TO 0 STEP -1:P
   OKE 53274,I:SOUND 0,I,4,10
DJ 2370 SOUND 1,I,8,6:NEXT I:POKE 712,0:P
   OKE 708,255:POKE 711,89
WD 2380 J=0
VB 2390 SOUND 0,0,0,0:SOUND 1,0,0,0
CR 2400 FOR D=1 TO 150:NEXT D:Q=0
UB 2410 ? #6:? #6;" press start or fire"
DN 2420 FOR D=1 TO 25:NEXT D:POKE 712,0:P
   OKE 711,RND(0)*5+60:Q=Q+1:IF Q=24 THEN
    GOTO 2460
AT 2430 IF PEEK(53279)=6 THEN GOTO 2490
SQ 2440 IF STRIG(0)=0 THEN GOTO 2490
QT 2450 GOTO 2420
AV 2460 FOR T=22 TO 0 STEP -1:POSITION 0,
   T:? #6;"████████████████████":FOR D=1
   TO 4:NEXT D:NEXT T
SY 2470 FOR T=22 TO 0 STEP -1:POSITION 0,
   T:? #6;"                    ":FOR D=1
   TO 4:NEXT D:NEXT T
PG 2480 GOTO 2020
RZ 2490 GRAPHICS 18:SOUND 0,228,10,10:SOU
   ND 1,230,10,10
FV 2500 POSITION 5,5:? #6;"fire alert":FO
   R R=1 TO 215:POKE 53274,R:NEXT R:SOUND
    1,0,0,0:GOTO 70
NH 2510 GRAPHICS 17:POKE 756,CH/256:POKE
   712,115:POKE 559,0
ZN 2520 POKE 708,6:POKE 710,0
WG 2530 FOR I=15 TO 18
EK 2540 POSITION 5,I:? #6;")) ) ))) ))))
   )) ))":NEXT I
WS 2550 POSITION 8,14:? #6;") )))"
UE 2560 FOR I=19 TO 21
RZ 2570 POSITION 5,I:? #6;")) ) ))) ))))
   )) ))":NEXT I
CD 2580 POSITION 0,22:? #6;"______________
   ________"
BU 2590 D=PEEK(560)+256*PEEK(561):POKE D+
   6,7:POSITION 3,1:? #6;"congratulations
   "
AE 2600 ? #6:? #6;" you have saved the
         city":POKE 559,34:GOSUB 1380
HA 2610 ? #6:? #6;"FINAL SCORE ";SC:POSIT
   ION 10,14:? #6;"*":GOSUB 1390
DN 2620 FOR D=1 TO 220:POKE 53274,PEEK(53
   770):SOUND 0,RND(0)*10+200,8,9:POSITIO
   N RND(0)*19,RND(0)*4+8:? #6;"!"
WR 2630 POSITION RND(0)*19,RND(0)*4+8:? #
   6;" ":POKE 711,PEEK(53770):NEXT D:SOUN
   D 0,0,0,0
YY 2640 FOR D=255 TO 240 STEP -1:POKE 708
   ,D:POKE 709,D:POKE 710,D:POKE 711,D:PO
   KE 712,D:NEXT D:GOTO 50
KI 2650 GRAPHICS 18:POKE 756,CH/256:POKE
   711,115:SC=SC+50000
GA 2660 POKE 708,14:FOR F=1 TO 5:POSITION
    4,1:? #6;"PERFECT SCORE":GOSUB 1380
ZL 2670 POSITION 4,1:? #6;"perfect score"
   :GOSUB 1390:NEXT F
GN 2680 SOUND 0,0,0,0:POSITION 2,4:? #6;"
   EXTRA BONUS 50000"
SJ 269A GOSUB 1600:POSITION 7,6:? #6;"HUR
   RAY"
XR 2700 POSITION 10,7:? #6;"*":POSITION 5
   ,9:? #6;" WWWWWWWWK"
XN 2710 FOR D=1 TO 275:NEXT D:RETURN
LH 2720 REM FIRE EXTINGUISHER
EG 2730 IF WP<=0 THEN SOUND 0,255,10,10:R
   ETURN
DM 2740 POSITION X,Y-1:? #6;"Q":POSITION
   X-1,Y:? #6;"Q":POSITION X+1,Y:? #6;"Q"
   :POSITION X,Y+1:? #6;"Q"
WR 2750 POSITION X,Y-1:? #6;" ":POSITION
   X-1,Y:? #6;" ":POSITION X+1,Y:? #6;" "
   :POSITION X,Y+1:? #6;" "
VO 2760 WP=WP-5:SOUND 0,3,8,10:SOUND 1,7,
   8,10:RETURN
```

End User

by Arthur Leyenberger

Welcome. Once again we find ourselves together for a look at what's happening in and around the world of Atari computers. Before I get down to the details, I want to share a few thoughts about technology with you. This is an especially good time to do it since a new decade is just around the corner.

Technology is constantly improving. Now an ST costs less money than the early 8-bits and has 20 times the features.

Way back in the mid-1970s, I bought a calculator. Not just any old calculator, but a marvelous new computing device called the Hewlett-Packard HP-65. What made it so exciting was that it was the first "programmable pocket" calculator ever made.

The machine had 100 steps of program memory and was programmed by "recording" the keystrokes pressed from the keyboard. It was fairly straightforward to do. You simply put the calculator in "write" mode and pressed the keystrokes as you would if you were solving the problem directly with the calculator functions. You could program a request for input from the keyboard which would cause the machine to pause when the program was run. Then you would enter a number and resume the program, causing it to include that number in the calculations.

The HP-65 was powerful. It had conditional testing which meant that you could compare two numbers against each other and have the program branch depending on whether they were equal or unequal. It also had a flag that could be set and tested for additional program branching. Since the HP-65 was a scientific calculator, it had a slew of built-in math, trig and other scientific functions.

Once a program was written, you could save it on a small strip of magnetic tape. This was useful since once you turned off the machine, the memory contents were lost. To re-enter the program, the magnetic card was inserted back into the miniature card reader inside the machine. All of the program steps were displayed as "keycodes" or row/column locations of the specific key on the keyboard.

The HP-65 was not really a consumer product, at least not for the average consumer. It cost $800 (in 1975) which was expensive. But for me it was worth it. I learned the fundamentals of programming on that gadget and have been thankful ever since. At the time, my family and friends thought I was a fool to spend that kind of money for something like that, but I can't imagine what my future would have been like without it.

You see, once I learned to program that calculator, I started to learn to program in BASIC using a computer time-sharing service that was used by the company I worked for. I became very proficient at programming in BASIC and some of the proprietary languages

The new ROMs from Atari make GEM slightly faster by fixing some long-standing bugs and providing a different file-selector box.

used by the time-sharing company. Eventually I took a job with the outfit which lasted for almost four years. That led to another job and where I am today.

Recently, I bought another HP calculator. Oh, there were a few other calculators in between, but the new machine is at least ten times more powerful than the HP-65. It costs $100 which is almost $\frac{1}{10}$ the price. If you figure inflation over eight years into the difference, you might have a price difference of 50 times.

My point? Technology is constantly improving. Those of us who bought an Atari 800 in the early 1980s were amazed with 48K of memory and the things the computer could do. Now, an ST costs less money than the early 8-bit Atari computers and has 20 times the features and performance. Technology is relative. It must boggle the mind of my parents who grew up with horses, no mass communications, unsanitary conditions (by today's standards), etc., to live in the world of today.

The consumer electronics of the 1980s were a tidal wave of technology. The new entertainment, educational and commercial products have already had profound impact on our lifestyles. The industry does over $30 billion worth of business annually and we are part of it. Ten or 15 years ago the notion of a "computer store" was completely foreign to most people, even those who should have known. Now, it seems there is a computer store on every block. Unfortunately, not many of them sell Atari computers.

What does all of this have to do with Atari computer users? Everything! Atari is in the business of selling electronics products. Whether it is video games or computers, if there is money to be made, Jack Tramiel and company will be in that business. This is perfectly reasonable. We've seen Atari continually introduce new and more powerful computers over the last several years, and there is no reason why it won't continue.

The good news

Unfortunately, we have not seen Atari do a good job of marketing the ST computers, i.e., by keeping dealers happy with a good supply of products in the U.S. and by communicating with the dealers and the users. But the good news is Atari seems to be changing. At the least, Atari is finally beginning to recognize some of these problems and take steps to rectify them.

Here are some of the things Atari is doing right. They have cut off mail-order dealers, who have provided little or no support and often undercut the retailers who *do* provide service for what they sell. There has been a lot of controversy about this, but the user will benefit in the long run by having more retail dealers willing to sell Atari products and therefore better support and good prices driven by competition.

Atari has set up a dealer council made up of several large dealers around the country. Their mission is to meet on a regular basis and discuss the retail business problems with Atari top brass. Atari has also started publishing a dealer newsletter to foster better communications. In fact, this dealer newsletter will also be sent to the SYSOPS of CompuServe, GEnie and DELPHI in order to improve communications on these information services. Atari has made an effort to be more visible on these services so that individual users can ask questions and get direct replies.

Atari now has a policy of not pre-announcing new products. One of the biggest complaints in the past was that Atari would announce a bunch of new products that would take forever to become available, if at all. With the lack of information or misinformation, rumors are created which causes further problems.

By the time you read this, Atari will have introduced the new ROMs for the ST. These new ROMs make GEM slightly faster by fixing some long-standing bugs, and providing a different file-selector box. Also, the Blitter chip is finally making it into production.

Advertising has been one of the missing links in the last couple of years. I understand Atari's point of view: It couldn't advertise in the U.S. because it didn't have machines to sell. If they did advertise, they would have unhappy consumers who couldn't get products at their local retailer. This is apparently about to change. Atari has opened a new plant or two in the Far East and also is about to open a new plant in Texas. With increased production Atari should be able to supply the new demand.

Lastly, Atari has big plans for the recently acquired Federated Electronics stores. One of Atari's major goals for these stores is for them to have an excellent service facility. As

Mavis Beacon Teaches Typing ***is a program that uses artificial-intelligence techniques.***

most of us know, there are very few places one can go to get an Atari computer fixed. The goal is to start with service on a district basis, covering up to eight stores. The service centers would pick up and deliver machines to Federated Stores within its district for quicker repair turnaround.

In addition, Atari wants these stores to be full-service centers that will carry the entire Atari computer line as well as other products. Further, these stores would become a base for an outside sales force dedicated to large customers and commercial applications. Of course, some of the other improvements mentioned above such as advertising and negating the Atari "game image" have to happen too in order for the outside sales force to have someone to sell to.

More Federated stuff

The Atari-owned Federated Electronics stores have been losing money in the last couple of quarters of 1988. Atari itself has said that it will take another quarter or two before the stores become profitable, but they can see the light at the end of the tunnel. Now it seems that part of the problem with the stores may have been out of Atari's control from the start.

Atari recently sued the former owners of The Federated Group (and their financial advisors) for inflating the value of the company's assets and thereby increasing the price of the deal. Atari purchased 67 stores in August 1987 for about $67 million. At the time, some analysts criticized the deal and suggested the price was too high, but Jack Tramiel stood behind his decision to purchase the stores. Now, Atari is claiming that they paid $43 million more for the stores than they should have.

Other companies have tried operating their own stores to gain a greater market share. However, such companies as IBM, Xerox and Digital Equipment have not been unsuccessful with this technique and have, in some cases, sold their storefronts. On the other hand, Tandy has used this method successfully to get products on shelves.

Learning can be fun

One of the neat things about writing this column every month is that I get to pick out my own subheadings. Maybe not a major perk but still exciting. The above subhead may sound goofy but it is true. I won't bore you with the details of experimental studies, but it has been proven that learning takes place more rapidly and with better retention if the process is fun. Plus, if something is fun, you are more apt to do it.

This is true with one category of software that has been around since the early days of home computers: typing tutors. Ever since my first Atari 800, I have seen and used typing programs that have been said to increase your typing speed while using the power of the computer to aid the process. This has generally been true. Interestingly, as computers have become more powerful, so have the typing tutorial programs.

A new program in this class has just been released for the ST. It is called (gulp) *Mavis Beacon Teaches Typing* (distributed by Electronic Arts for Software Toolworks, One Toolworks Plaza, 13557 Ventura Blvd., Sherman Oaks, CA 91423; 818-907-6789).

I'll call it Mavis for short hereafter. Mavis is a highly advanced program that claims to use artificial-intelligence (I hate that phrase) techniques to customize individualized lessons. I don't know if it does or not but this is a *super* program, the best typing tutor I've seen on any computer.

There are a number of things that make this an excellent program. One is that mindless letter combinations and word repetitions are not used for the typing drill. Instead, meaningful sentences taken from *Guinness Book of World Records*, famous quotations, jokes, great writings, riddles and other sources are used for the training. This makes for interesting lessons that hold your attention while you practice your typing.

Mavis consists of five screen displays, the first of which is the chalkboard. Here the new or returning student is greeted, areas requiring more work are noted and exercises are recommended. Next stop is the classroom where you actually practice your typing. The screen consists of a miniature monitor on the top and a ST keyboard on the bottom. A pair of ghostly transparent hands are properly positioned on the keyboard at all times for guidance.

As you type each letter the hands show the correct finger placement, but if you find them distracting, they can be turned off. When the hands are turned off, the keys on the keyboard light up when they are touched. The monitor shows both the lesson text as well as your typing, and the whole process is intuitive.

The workshop screen is used to practice particular skills. It is here where you display what you learned from the classroom. This screen includes a clock, metronome and accuracy gauge. The metronome is used to keep a steady beat while typing. Apparently studies have shown that some people when learning to type hesitate slightly before trying a difficult finger position. This slows down the typing rate, and the metronome can help overcome the problem.

Most typing programs come with a game, and Mavis is no different. But the road-race game is better than shooting down aliens and probably more effective since large amounts of text must be entered. The screen shows the dashboard of the car you are in and another car on the right. The faster you type the faster your car goes, pulling ahead from the other one. The speedometer shows words-per-minute typed, and another meter shows accuracy.

Throughout the program, many graphs are used to show your typing speed in a variety of ways. These bar graphs show raw speed with or without error penalties (one keystroke is subtracted for each mistake), progress by letter, percent error by key, etc. In addition, your progress can be displayed either cumulatively or by lesson.

Mavis Beacon Teaches Typing uses the ST's capabilities quite well. The graphics are superb, sound is used effectively and mouse control is good. There are also plenty of help screens available. Further, clear objectives are always given before every lesson.

About the artificial-intelligence deal, the program does use these types of techniques to tailor the lessons to the user's progress and problems. Software Toolworks is the company that did *Chessmaster 2000*, and I have no doubt that the Mavis program is equally as sophisticated. Mavis is the program you should get to learn or to improve your touch-typing skills. Best of all, Ms. Beacon won't rap your knuckles when you make a mistake. ■

CIRCLE #104 ON READER SERVICE CARD.

continued from page 11

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,32,65,27,66,0,9,31,7
,44,7,169,60,141,2,8825
1010 DATA 211,24,96,169,179,133,10,169
,70,133,11,96,0,0,0,179,1445
1020 DATA 180,165,172,172,161,178,0,0,
161,178,165,174,161,0,0,0,3832
1030 DATA 0,0,0,35,47,48,57,50,41,39,4
0,52,0,209,217,216,3952
1040 DATA 211,0,0,0,0,0,0,0,106,111,
104,110,0,111,114,9699
1050 DATA 116,105,122,0,0,0,0,0,0,104,
105,103,104,0,0,115,8365
1060 DATA 99,111,114,101,90,16,16,16,1
6,16,16,0,0,240,236,225,3893
1070 DATA 249,229,242,0,17,0,0,240,236
,225,249,229,242,0,18,0,7785
1080 DATA 0,115,90,16,16,16,16,16,16,0
,0,115,90,16,16,16,5474
1090 DATA 16,16,16,0,0,236,218,16,0,23
6,33,0,244,218,25,25,3978
1100 DATA 0,240,18,0,236,218,16,0,112,
112,112,112,112,112,71,54,3891
1110 DATA 65,112,112,112,112,70,74,65,
112,112,70,94,65,65,194,65,3932
1120 DATA 112,112,112,70,200,57,6,70,2
00,58,134,70,0,57,6,6,9694
1130 DATA 6,6,6,6,6,6,6,70,0,58,6,6,6,
6,6,6,2924
1140 DATA 6,6,6,65,218,65,0,0,0,0,0,0,
0,0,0,0,2916
1150 DATA 0,0,0,0,0,0,255,255,255,255,
255,255,255,255,240,240,10
1160 DATA 240,240,28,66,23,67,240,240,
240,240,15,15,15,15,15,15,2120
1170 DATA 15,15,3,7,14,28,56,112,224,1
92,192,224,112,56,28,14,4398
1180 DATA 7,3,3,12,28,60,124,124,128,1
28,192,48,56,60,62,62,2220
1190 DATA 1,1,1,1,62,62,60,56,48,192,1
28,128,124,124,60,28,2742
1200 DATA 12,3,128,64,0,0,0,0,0,0,0,0,
32,16,0,0,2498
1210 DATA 0,0,0,0,0,0,8,4,0,0,0,0,0,0,
0,0,1298
1220 DATA 2,1,1,2,0,0,0,0,0,0,0,0,4,8,
0,0,1399
1230 DATA 0,0,0,0,0,0,16,32,0,0,0,0,0,
0,0,0,1598
1240 DATA 64,128,0,85,170,0,85,170,0,0
,66,165,90,60,60,90,1761
1250 DATA 165,66,0,0,24,60,60,24,0,0,0
,0,34,20,0,20,3681
1260 DATA 34,0,0,0,8,8,62,8,8,0,0,16,3
6,26,88,36,4872
1270 DATA 8,0,240,96,126,240,240,126,9
6,240,15,6,126,15,15,126,4730
1280 DATA 6,15,0,0,0,255,1,255,1,0,1,1
,0,1,255,1,8780
1290 DATA 255,0,255,255,0,235,236,237,
1,21,20,19,255,120,72,138,7238
1300 DATA 72,169,66,162,0,69,79,37,78,
141,10,212,141,9,212,142,5996
1310 DATA 26,208,104,170,104,88,64,230
,178,230,229,230,206,230,207,230,7944
1320 DATA 212,230,24,67,19,68,219,230,
224,162,7,189,193,2,157,19,7385
1330 DATA 208,202,208,247,173,244,2,14
1,9,212,169,192,141,14,212,169,1302
1340 DATA 0,162,4,213,214,240,2,214,21
4,202,208,247,138,172,2,6,290
1350 DATA 136,153,0,60,172,3,6,136,153
,0,61,174,8,6,189,0,1199
1360 DATA 59,41,252,157,0,59,174,9,6,1
89,0,59,41,243,157,0,3471
1370 DATA 59,169,0,133,205,174,4,6,173
,0,6,24,125,216,66,141,4250
1380 DATA 0,6,141,0,208,232,173,2,6,24
,125,216,66,141,2,6,2693
1390 DATA 165,176,201,10,144,8,230,205
,165,205,201,1,240,215,174,10,1226
1400 DATA 6,173,6,6,24,125,216,66,24,1
25,216,66,141,6,6,141,3601
1410 DATA 4,208,232,173,8,6,24,125,216
,66,24,125,216,66,141,8,4805
1420 DATA 6,170,189,0,59,9,1,157,0,59,
165,205,201,1,240,10,5197
1430 DATA 165,177,201,1,240,89,169,0,1
33,205,174,5,6,173,1,6,3305
1440 DATA 24,125,216,66,141,1,6,141,1,
208,232,173,3,6,24,125,3707
1450 DATA 216,66,141,3,6,165,176,201,1
0,144,8,230,205,165,205,201,1737
1460 DATA 1,240,215,174,11,6,173,7,6,2
4,125,216,66,24,125,216,5426
1470 DATA 66,141,7,6,141,5,208,232,173
,9,6,24,125,216,66,24,3934
1480 DATA 125,216,20,68,15,69,66,141,9
,6,170,189,0,59,9,4,9752
1490 DATA 157,0,59,165,176,201,7,176,3
,76,154,68,169,0,133,201,6732
1500 DATA 174,16,6,173,12,6,24,125,216
,66,141,12,6,141,2,208,3389
1510 DATA 232,173,14,6,24,125,216,66,1
41,14,6,168,162,8,189,191,6664
1520 DATA 66,153,0,62,136,202,208,246,
230,201,165,176,201,10,144,60,1336
1530 DATA 165,201,201,1,240,202,165,17
6,201,13,144,48,174,17,6,173,7136
1540 DATA 13,6,24,125,216,66,141,13,6,
141,3,208,232,173,15,6,4456
1550 DATA 24,125,216,66,141,15,6,168,1
62,8,189,191,66,153,0,63,4834
1560 DATA 136,202,208,246,230,201,165,
201,201,3,240,208,165,212,201,17,4202
1570 DATA 208,26,169,0,133,212,160,8,1
62,8,165,178,201,2,240,2,7220
1580 DATA 162,16,189,159,66,153,7,66,2
02,136,208,246,169,0,133,205,692
1590 DATA 162,2,189,1,6,149,202,202,20
8,248,165,178,201,2,240,47,1577
1600 DATA 169,90,141,18,208,141,192,2,
169,134,141,19,208,141,193,2,7935
1610 DATA 162,8,189,175,66,164,203,153
,0,60,198,203,164,204,153,0,9511
1620 DATA 61,198,204,202,230,205,165,2
05,201,9,144,230,76,95,228,169,3357
1630 DATA 134,141,18,208,141,192,2,169
,90,141,19,208,141,193,2,169,8349
1640 DATA 0,133,16,69,11,70,178,162,16
,76,224,68,138,72,166,230,8403
1650 DATA 224,3,240,31,166,227,224,0,2
40,16,202,202,202,134,227,142,3629
1660 DATA 2,210,169,172,141,3,210,76,8
7,69,230,230,169,252,133,227,4193
1670 DATA 76,87,69,166,189,224,200,176
,15,232,232,134,189,142,2,210,2338
1680 DATA 169,166,141,3,210,76,87,69,1
69,0,141,3,210,173,230,58,7921
1690 DATA 201,16,208,60,162,51,134,191
,166,225,224,1,240,25,166,228,2197
1700 DATA 224,70,176,19,232,232,134,22
8,142,6,210,169,164,141,7,210,1229
1710 DATA 169,0,133,225,76,159,69,169,
1,133,225,166,228,224,10,144,707
1720 DATA 221,202,202,134,228,142,6,21
0,169,164,141,7,210,76,159,69,9280
1730 DATA 169,0,141,7,210,166,190,224,
252,176,23,224,251,240,14,232,5032
1740 DATA 134,190,142,4,210,169,168,14
1,5,210,76,188,69,169,0,141,7820
1750 DATA 5,210,166,191,224,51,240,21,
176,14,232,134,191,142,6,210,516
1760 DATA 169,138,141,7,210,76,215,69,
169,0,141,7,210,104,170,104,7775
1770 DATA 64,162,0,160,9,217,133,70,24
0,5,232,232,136,208,246,96,3088
1780 DATA 173,10,210,41,6,201,0,208,2,
9,2,96,166,176,202,189,7625
1790 DATA 51,70,141,198,2,189,66,70,13
3,183,189,82,70,133,182,189,9978
1800 DATA 98,70,12,70,7,71,133,180,189
,114,70,133,181,138,24,105,6718
1810 DATA 33,141,226,58,32,11,81,96,22
4,2,240,6,162,2,56,233,6046
1820 DATA 200,96,162,0,24,105,200,96,1
40,92,136,14,48,136,138,140,6298
1830 DATA 216,218,220,104,106,108,248,
250,252,72,74,76,4,4,4,5,3434
1840 DATA 5,5,6,6,6,7,7,7,8,8,8,0,10,1
2,14,16,3078
1850 DATA 18,20,22,24,26,28,30,32,34,3
6,38,0,38,36,34,32,5938
1860 DATA 30,28,26,24,22,20,18,16,14,1
2,10,0,42,40,38,36,5212
1870 DATA 34,32,30,28,26,24,22,20,18,1
6,14,0,42,57,157,142,9205
1880 DATA 10,11,9,13,5,7,6,14,15,19,23
7,21,235,71,72,73,1693
1890 DATA 74,133,134,133,134,21,19,235
,237,203,207,206,210,207,211,207,6990
1900 DATA 211,203,207,203,207,206,210,
203,207,60,201,252,51,169,194,141,5208
1910 DATA 48,2,169,65,141,49,2,169,0,1
70,157,0,6,202,208,250,8547
```

```
1920 DATA 133,20,133,19,169,1,133,176,
133,177,133,226,169,17,141,111,9226
1930 DATA 2,169,56,141,7,212,169,3,141
,29,208,169,62,141,47,2,4908
1940 DATA 169,243,141,0,2,169,66,141,1
,2,169,192,141,14,212,169,7737
1950 DATA 6,162,67,160,13,32,92,228,32
,101,228,169,38,133,180,169,9440
1960 DATA 0,141,8,71,3,72,8,210,141,1,
210,169,200,141,0,210,8284
1970 DATA 169,22,141,16,2,169,69,141,1
7,2,165,16,9,1,133,16,9867
1980 DATA 141,14,210,169,2,133,220,162
,4,189,174,70,149,187,202,208,2692
1990 DATA 248,165,19,201,1,144,250,169
,218,141,48,2,169,65,141,49,7330
2000 DATA 2,169,0,170,157,0,6,157,0,59
,157,0,60,157,0,61,1374
2010 DATA 157,0,62,157,0,63,202,208,23
5,172,11,212,192,112,144,249,3145
2020 DATA 157,0,57,157,0,58,202,208,24
7,169,0,141,5,210,133,186,9983
2030 DATA 133,187,169,3,133,185,162,40
,189,113,65,157,199,57,189,153,383
2040 DATA 65,157,199,58,202,208,241,16
9,16,141,223,58,141,238,58,165,1923
2050 DATA 177,24,105,16,141,234,58,162
,10,169,130,157,4,57,157,184,7708
2060 DATA 58,202,208,247,162,119,160,5
,169,132,157,0,57,157,156,57,7577
2070 DATA 138,56,233,19,170,169,131,15
7,0,57,157,156,57,138,24,105,6014
2080 DATA 39,170,136,208,227,160,5,162
,4,169,133,157,0,57,157,115,7191
2090 DATA 58,169,64,232,157,115,58,202
,202,157,0,57,232,138,24,105,8163
2100 DATA 19,170,136,208,228,160,5,162
,15,169,134,157,0,57,157,85,6826
2110 DATA 58,169,64,232,157,0,57,202,2
02,157,85,58,232,138,24,105,8433
2120 DATA 21,170,4,72,255,72,136,208,2
28,162,131,142,100,57,232,142,1771
2130 DATA 99,58,169,0,162,16,157,255,6
5,202,208,250,162,5,189,45,521
2140 DATA 70,157,195,2,202,208,247,169
,30,141,194,2,141,195,2,165,9527
2150 DATA 226,133,176,32,246,69,162,57
,134,198,232,134,200,165,179,201,4689
2160 DATA 1,240,101,173,31,208,141,2,2
08,141,3,208,201,6,240,77,9382
2170 DATA 201,3,240,28,201,5,240,3,76,
65,72,166,176,232,224,10,9122
2180 DATA 144,2,162,1,134,176,134,226,
134,77,32,246,69,76,135,72,7708
2190 DATA 166,177,134,77,224,1,240,13,
202,134,177,138,24,105,16,141,7369
2200 DATA 234,58,76,135,72,232,76,119,
72,165,20,24,105,9,162,0,3281
2210 DATA 142,2,208,142,3,208,197,20,2
40,169,76,142,72,165,226,133,1504
2220 DATA 176,169,1,133,179,76,67,71,1
69,0,133,179,133,184,169,19,7933
2230 DATA 141,223,58,166,177,224,1,240
,3,141,238,58,169,100,141,0,8274
2240 DATA 6,169,130,141,2,6,224,1,240,
6,141,3,6,169,156,141,6007
2250 DATA 1,6,162,10,169,0,149,211,202
,208,249,173,10,210,41,15,9003
2260 DATA 24,105,3,133,213,165,176,201
,7,144,20,162,1,134,186,160,8836
2270 DATA 10,141,16,6,201,10,144,7,134
,187,160,12,141,17,6,164,4528
2280 DATA 183,32,0,73,251,73,91,76,136
,208,250,160,6,169,16,153,8863
2290 DATA 202,58,153,212,58,153,156,65
,153,166,65,136,208,241,165,176,3488
2300 DATA 133,226,173,31,208,201,6,208
,3,76,155,72,201,7,176,3,6235
2310 DATA 76,67,71,169,0,133,77,169,13
0,141,186,58,165,224,201,45,436
2320 DATA 144,3,32,49,80,32,212,80,173
,252,2,201,255,240,3,76,9925
2330 DATA 124,81,165,185,74,176,65,173
,120,2,32,219,69,224,0,240,9069
2340 DATA 52,173,132,2,201,1,240,45,16
9,0,141,10,6,141,4,6,1593
2350 DATA 172,8,6,185,0,59,41,252,153,
0,59,173,0,6,105,3,1762
2360 DATA 141,6,6,173,2,6,233,3,141,8,
6,142,10,6,169,0,792
2370 DATA 133,189,76,146,73,142,4,6,16
5,177,201,2,208,71,165,185,9609
2380 DATA 74,74,176,65,173,121,2,32,21
9,69,224,0,240,52,173,133,8947
2390 DATA 2,201,1,240,45,169,0,141,11,
6,141,5,6,172,9,6,611
2400 DATA 185,0,59,41,243,153,0,59,173
,1,6,105,3,141,7,6,638
2410 DATA 173,3,6,233,3,141,9,6,142,11
,6,169,0,133,189,76,3906
2420 DATA 223,73,142,5,6,165,207,197,1
80,176,3,76,87,74,169,0,6307
2430 DATA 133,207,160,4,185,17,6,201,0
,208,6,136,208,246,76,87,8608
2440 DATA 74,162,252,73,247,74,2,192,3
,176,2,162,0,149,197,185,8869
2450 DATA 150,70,129,197,134,195,173,1
0,210,41,1,192,3,240,7,192,8237
2460 DATA 1,240,3,24,105,2,24,105,1,13
3,194,170,185,17,6,24,3221
2470 DATA 125,142,70,166,195,149,197,1
61,197,217,150,70,240,19,166,194,3702
2480 DATA 224,1,240,9,224,3,240,5,202,
138,76,31,74,232,76,61,7052
2490 DATA 74,181,197,153,17,6,185,146,
70,129,197,76,245,73,76,231,755
2500 DATA 74,165,214,201,1,176,247,165
,176,201,4,144,241,165,206,197,5511
2510 DATA 180,144,235,169,0,133,206,16
5,213,201,0,240,2,198,213,172,3471
2520 DATA 22,6,174,24,6,148,197,169,13
1,129,197,152,201,180,208,13,1742
2530 DATA 169,236,141,22,6,162,2,142,2
4,6,76,156,74,201,80,208,7122
2540 DATA 5,162,0,142,24,6,173,22,6,24
,105,20,141,22,6,149,1284
2550 DATA 197,169,89,129,197,165,176,2
01,7,144,163,172,23,6,174,25,7436
2560 DATA 6,148,197,169,132,129,197,15
2,201,19,208,13,162,0,142,25,7237
2570 DATA 6,169,219,141,23,6,76,216,74
,201,119,208,5,162,2,142,7662
2580 DATA 25,6,173,23,6,56,233,20,141,
23,6,149,197,169,90,129,7079
2590 DATA 197,165,176,201,4,176,3,76,1
25,75,165,213,201,0,240,3,8661
2600 DATA 76,125,248,74,243,75,75,165,
214,201,1,240,27,201,2,240,1338
2610 DATA 99,230,214,169,50,133,215,16
9,200,141,5,210,169,163,141,4,835
2620 DATA 210,169,252,133,190,76,125,7
5,165,215,201,0,240,3,76,125,9485
2630 DATA 75,169,30,133,215,230,214,16
9,211,133,196,174,24,6,172,22,9771
2640 DATA 6,200,148,197,152,160,18,165
,196,129,197,246,197,136,208,249,7186
2650 DATA 169,0,133,190,165,176,201,7,
176,3,76,125,75,174,25,6,5154
2660 DATA 172,23,6,136,148,197,160,18,
165,196,129,197,214,197,136,208,4762
2670 DATA 249,76,125,75,165,215,201,0,
208,17,133,196,133,214,173,10,605
2680 DATA 210,41,15,24,105,3,133,213,7
6,45,75,165,176,201,7,176,8253
2690 DATA 3,76,218,75,160,0,185,12,6,2
01,51,144,31,201,196,176,9316
2700 DATA 36,185,14,6,201,217,176,7,20
1,79,144,34,76,207,75,32,6881
2710 DATA 234,69,201,6,208,30,24,105,1
0,76,198,75,32,234,69,24,4976
2720 DATA 105,2,76,198,75,32,234,69,24
,105,10,76,198,75,32,234,6742
2730 DATA 69,24,105,6,182,186,224,0,24
0,3,153,16,6,165,176,201,9089
2740 DATA 13,144,5,200,192,1,240,174,1
65,219,197,181,176,3,76,183,2306
2750 DATA 76,169,0,133,219,164,183,185
,25,6,201,0,240,21,190,43,7984
2760 DATA 6,149,244,75,239,76,197,161,
197,201,1,240,16,169,0,153,110
2770 DATA 25,6,153,43,6,198,184,136,20
8,225,76,183,76,190,69,6,8965
2780 DATA 185,25,6,24,125,234,66,190,4
3,6,201,200,144,3,32,30,5072
2790 DATA 70,149,197,161,197,201,0,240
,14,173,10,210,41,7,24,105,5661
2800 DATA 1,153,69,6,76,5,76,181,197,1
33,195,134,194,185,25,6,8167
2810 DATA 190,43,6,149,197,169,0,129,1
97,165,195,166,194,149,197,153,4302
2820 DATA 25,6,138,153,43,6,169,1,129,
197,76,5,76,165,184,197,8562
2830 DATA 183,144,1,96,169,5,133,195,1
85,25,6,201,0,240,1,96,6358
```

continued on page 52

EDIT MAGIC

by Bill Bodenstein

Of all the differences between assembly language and higher-level languages, the one that has given me the biggest headaches is the considerable amount of instructions in an average MAC/65 program listing. Even a simple assembly-language program can often exceed several hundred lines in length. And when editing, it can be a tedious activity to LIST through this code over and over again correcting mistakes.

So to make editing chores somewhat easier, I decided to improve the screen-editor routines in the operating system, interfacing the code with many additional features. The result, *Edit Magic*, should hopefully prove to be a boon to all BASIC and MAC/65 programmers.

Edit Magic, because it copies the OS from ROM to RAM and replaces part of the code, will only work on XL and XE Atari systems.

Getting started

I'm afraid Edit Magic, because it copies the operating system (OS) from ROM to RAM and replaces part of the code, will only work on XL and XE Atari systems. (If you have a 400/800 or 1200 model, now's a good time to purchase a newer Atari computer!) But if you have an XL/XE, type in the data statements from Listing 1 using M/L Editor. Refer to the instructions to M/L Editor, and create a binary file called EDTMAGIC.OBJ. Rename as AUTORUN.SYS if you'd like the program loaded automatically at power-up.

Also, take a look at Listing 2, the MAC/65 source code. If you have the assembler, you may want to type in Listing 2 instead. I tried to make the program fairly modifiable so that you can later mold the utilities to fit your programming needs.

Edit Magic information line

Okay, you've loaded Edit Magic into memory (refer to your DOS manual for instructions on loading a binary file). Go immediately to BASIC (or to MAC/65—the instructions to follow are appropriate for the assembler as well). Notice the 25th line added to the top of your screen. This line is divided, from left to right, into four parts: the List Box, Message Box, File Box and Save Counter. You should also see a "V" prompt in the List Box, informing you that Edit Magic is now examining your keyboard input.

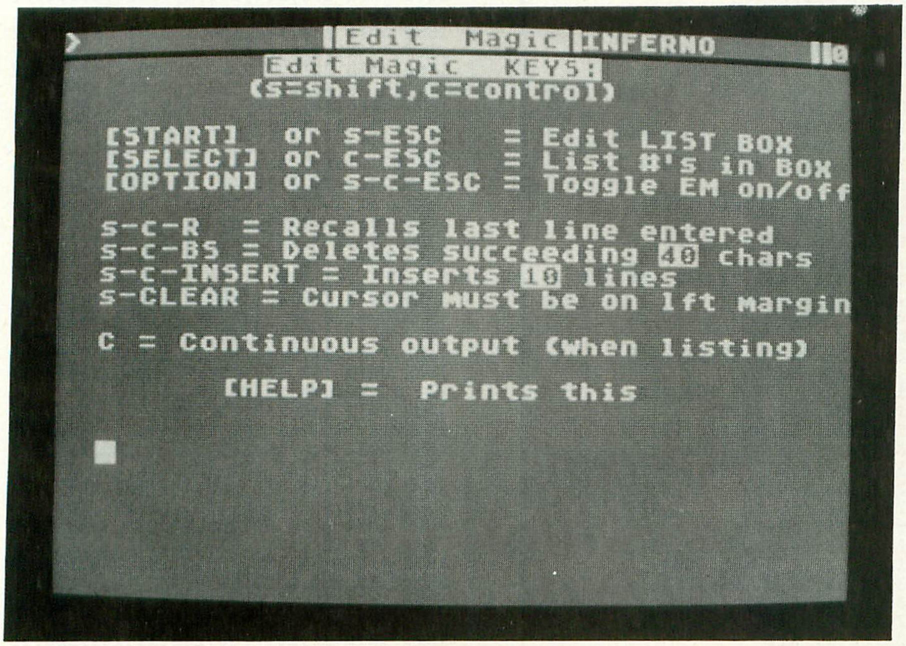
Loading and saving

Let's begin by loading a BASIC program from disk. Whenever Edit Magic sees you've entered a LOAD command, it will search the syntax for the filename and place it in the File Box. Now you won't forget the name of the resident BASIC (or MAC/65) program.

Try saving your program. Edit Magic, upon discovering an inputted SAVE, increments the Save Counter and puts that number (0-9) in inverse. After saving, try changing any line in your program. The number in the Save Counter box becomes non-inverse again whenever a line is modified. So before you ever NEW memory or abort to DOS, check the Save Counter to make sure your program has been recently saved to disk or cassette.

Listing features

With your BASIC program in memory, type LIST and hit Return. I added my *Fast Print* routine (see Issue 61) to speed up screen output and to add a screen pause feature: Every 21 lines of output the speaker clicks, a "press a key" message is put in the Message Box, and Edit Magic waits for a keypress or Break. To prevent a pause, press *C* for con-



```
        Edit  Magic INFERNO                 0
           Edit Magic  KEYS:
          (s=shift,c=control)

[START]  or s-ESC   = Edit LIST box
[SELECT] or c-ESC   = List #'s in Box
[OPTION] or s-c-ESC = Toggle EM on/off

s-c-R  = Recalls last line entered
s-c-BS = Deletes succeeding 40 chars
s-c-INSERT = Inserts 10 lines
s-CLEAR = Cursor must be on lft margin

C = Continuous output (when listing)

      [HELP] = Prints this
```

tinuous output anytime after typing LIST. Or a POKE 847,0 will remove it for good (for good until a value of one or greater is stored in this location to re-enable the automatic pause).

Faster listings are nice, but must we still type LIST (plus a line number range) every time? Not with Edit Magic! Press either Start or, if you prefer, Shift-Escape (press Escape while holding down the Shift key). You should find the cursor now located in the List Box next to the "V" prompt. This is where you enter a range of line numbers, using the standard LIST syntax: an optional first line, comma, then last line. Except, if you do not specify a last line and end with a comma, your program will list up to Line 32767. So to list from Line 100 to the end of your program, just enter "100," and then hit Return. If you make a mistake, press the Back Space key to erase a character, or hit Escape or Break to exit the box. No other keys are allowed while editing the List Box.

For now, just press Return from the List Box to list all lines. The screen will clear and display the lines. When the screen pauses, hit Break and move the cursor up to any line and change it. Now press Select or Control-Escape, and the same set of lines defined in the List Box is printed. But also notice the line you changed is marked with an inverse right arrow on the left margin. This is to remind you which lines have been edited since last listing from the List Box. Up to 24 line numbers will be remembered for marking.

Macros

There are currently four macros defined within Edit Magic (though with just a little work, you can add your own):

Help: Shows you all keys Edit Magic uses.

Shift-Control-R: Recalls last line you've entered.

Shift-Control-Backspace: Deletes 40 characters on the same logical line after the cursor.

Shift-Control-Insert: Opens space on the screen by inserting ten blank lines.

A table of macro entries begins at location $C600 (50688 decimal). Each entry consists of three bytes: the internal keypress value and the two-byte address of the string to be printed whenever this key is pressed while Edit Magic is active. The last entry must have a keypress value of 255. Character strings must end with a zero value (not printed). Feel free to create your own macros using your friendly POKE statement and binary, saving your results (see your DOS manual again for help). Assembly-language programmers will find it much easier to make additions directly to the MAC/65 source code.

Turning off Edit Magic

Option or Shift-Control-Escape (all three keys pressed simultaneously) will toggle all the Edit Magic features on and off. Since Edit Magic is buried in the operating system, not only does it not take up any of your usable RAM, but it also shouldn't interfere with any program if kept off. If you do have problems, you can always re-enable the OS ROM by simply pressing Reset. And if you'd like to enable Edit Magic again, from BASIC just POKE 54017,252. It's that easy.

Notes and stuff

Yes, there's still more features to note!

Pressing Control-1 does nothing while you're entering a line of text. (Why would you want to pause during screen input?)

Pressing Shift-V (the clear key), except when the cursor is at the beginning of a line, does nothing. This is to prevent your screen from clearing accidentally when you meant to type ")" or "V." (It happens to me a lot!)

Faster key response, brighter characters and a lighter border color are a few minor changes—you've probably noticed them by now.

I've only mentioned the MAC/65 Assembler in this article because that's the one I use and prefer (I have the disk version, incidently). Will Edit Magic work with other Atari assemblers? Probably. In fact, Edit Magic should have compatibility—though possibly limited—with any language or program that accepts a LIST, LOAD and SAVE command— and *does not* ever use the added RAM from $C000 to $FFFF. Because of the latter restriction, BASIC XE and XL versions of Sparta DOS will not run properly with Edit Magic loaded.

That's all the stuff you need to worry about. Take plenty of time getting used to Edit Magic. I'm sure you'll soon agree with me that it makes editing a program an almost bearable task!

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,5,167,5,173,247,25
5,201,2,240,3,76,141,5,6342
1010 DATA 173,1,211,170,41,254,205,1,2
11,240,242,133,212,169,0,133,1477
1020 DATA 214,169,192,133,215,173,47,2
,72,169,0,141,14,212,141,47,5185
1030 DATA 2,141,0,212,120,160,0,142,1,
211,177,214,206,1,211,145,9669
1040 DATA 214,200,208,243,230,215,240,
13,165,215,201,208,208,233,169,216,777
3
1050 DATA 133,215,76,49,5,88,169,64,14
1,14,212,104,141,47,2,165,4435
1060 DATA 212,141,1,211,162,0,169,142,
157,68,3,169,5,157,69,3,3030
1070 DATA 157,73,3,169,9,157,66,3,32,8
6,228,173,4,228,141,147,6974
1080 DATA 194,173,5,228,141,148,194,17
3,6,228,141,149,194,173,7,228,1252
1090 DATA 141,150,194,96,73,110,115,11
6,97,108,108,105,110,103,32,197,6160
1100 DATA 228,233,244,160,205,225,231,
233,227,32,46,46,46,155,226,2,8633
1110 DATA 227,2,0,5,224,2,225,2,0,5,0,
5,37,5,173,68,8428
1120 DATA 2,208,1,96,162,255,154,169,1
,133,9,32,57,231,169,0,5027
1130 DATA 141,68,2,165,6,240,10,173,25
3,191,41,4,240,3,108,250,8465
1140 DATA 191,108,10,0,144,194,144,194
,1,151,194,183,194,10,0,0,4532
1150 DATA 0,0,0,0,0,0,0,76,79,65,68,32
,83,65,86,69,8634
1160 DATA 32,76,73,83,84,32,51,50,55,5
4,55,253,242,0,175,195,6831
1170 DATA 114,197,6,115,204,5,250,204,
3,93,205,7,92,115,204,156,8246
1180 DATA 250,204,220,93,205,118,132,2
05,255,125,32,32,32,32,32,32,3304
1190 DATA 32,32,197,228,233,244,160,20
5,225,231,233,227,160,160,203,197,8317
1200 DATA 217,211,186,155,32,32,32,32,
32,32,32,40,115,61,115,104,1027
1210 DATA 105,102,116,44,99,61,99,111,
110,116,114,111,108,41,155,155,6004
1220 DATA 91,83,84,65,82,84,93,32,32,1
11,114,32,115,45,69,83,1334
1230 DATA 67,32,32,32,61,32,69,100,105
,116,32,76,73,83,84,32,617
1240 DATA 66,79,88,155,91,83,69,76,69,
67,84,93,32,111,114,32,1915
1250 DATA 99,45,69,83,67,32,32,32,61,3
2,76,105,115,116,32,35,109
1260 DATA 39,115,32,105,110,32,66,79,8
8,155,91,79,80,84,73,79,2747
1270 DATA 78,93,32,111,114,32,115,45,9
9,45,69,83,67,32,61,32,9843
1280 DATA 84,111,103,103,108,101,32,69
,77,32,111,110,47,111,102,102,3110
1290 DATA 155,115,45,99,45,82,32,32,61
,32,82,101,99,97,108,108,2379
1300 DATA 115,32,108,97,115,116,32,108
,105,110,101,32,101,110,116,101,4299
1310 DATA 114,101,100,155,115,45,99,45
,66,83,32,61,32,68,101,108,1563
1320 DATA 101,116,101,115,32,115,117,9
9,99,101,101,100,105,110,103,32,4051
1330 DATA 180,176,32,99,104,97,114,115
,155,115,45,99,45,73,78,83,3507
1340 DATA 69,82,84,32,61,32,73,110,115
,101,114,116,115,32,177,176,5946
1350 DATA 32,108,105,110,101,115,155,1
15,45,67,76,69,65,82,32,61,1741
1360 DATA 32,67,117,114,115,111,114,32
,109,117,115,116,32,98,101,32,3251
1370 DATA 111,110,32,108,102,116,32,10
9,97,114,103,105,110,155,67,32,4054
1380 DATA 61,32,67,111,110,116,105,110
,117,111,117,115,32,111,117,116,5422
1390 DATA 112,117,116,32,40,119,104,10
1,110,32,108,105,115,116,105,110,4874
1400 DATA 103,41,155,155,32,32,32,32,3
2,32,91,72,69,76,80,93,624
1410 DATA 32,61,32,32,80,114,105,110,1
16,115,32,116,104,105,115,155,5452
1420 DATA 155,0,254,254,254,254,254,25
4,254,254,254,254,254,254,254,254,5357
1430 DATA 254,254,254,254,254,254,254,
254,254,254,254,254,254,254,254,254,59
74
1440 DATA 254,254,254,254,254,254,254,
254,254,254,0,157,157,157,157,157,6400
1450 DATA 157,157,157,157,157,0,144,19
7,150,197,112,80,66,179,197,16,8476
1460 DATA 66,153,197,240,197,2,2,2,2,2
,2,2,2,2,2,2,4610
1470 DATA 2,2,2,2,2,2,2,2,2,2,2,2,65,1
44,197,0,7442
1480 DATA 0,0,0,0,0,0,0,0,0,0,0,252,
165,228,233,4214
1490 DATA 244,128,128,173,225,231,233,
227,252,0,0,0,0,0,0,0,1292
1500 DATA 0,0,0,0,0,252,16,165,228,233
,244,128,128,173,225,231,4203
1510 DATA 233,227,240,242,229,243,243,
128,225,128,235,229,249,0,198,9,4202
1520 DATA 198,244,63,197,247,104,197,2
32,183,194,255,96,202,190,207,169,6916
1530 DATA 30,141,179,197,162,0,32,167,
207,173,152,194,24,105,16,44,5856
1540 DATA 153,194,240,2,9,128,141,218,
197,173,144,194,240,47,169,0,81
1550 DATA 141,154,194,169,32,141,217,2
,169,4,141,218,2,173,138,194,9148
1560 DATA 201,76,240,25,169,0,141,14,2
12,169,76,141,138,194,169,213,1256
1570 DATA 141,139,194,169,206,141,140,
194,169,64,141,14,212,166,46,32,7817
1580 DATA 44,207,132,35,192,128,176,4,
201,155,240,1,96,162,0,129,7157
1590 DATA 36,173,144,194,240,121,173,1
28,5,201,155,240,114,162,255,232,5268
1600 DATA 48,109,189,128,5,157,183,194
,201,155,208,243,169,0,157,183,2788
1610 DATA 194,162,255,232,48,89,189,12
8,5,201,155,240,82,201,32,240,1782
1620 DATA 242,201,48,240,238,134,242,1
89,128,5,201,49,144,68,201,58,9336
1630 DATA 176,64,169,0,141,153,194,173
,155,194,133,224,201,24,176,47,633
1640 DATA 169,55,162,195,198,224,48,13
,32,53,207,48,34,24,105,5,2063
1650 DATA 144,242,232,208,239,133,212,
134,213,166,242,160,255,200,192,5,5589
1660 DATA 240,10,189,128,5,145,212,232
,201,32,208,241,238,155,194,76,3933
1670 DATA 165,203,169,161,162,194,32,5
3,207,192,4,144,56,232,189,128,428
1680 DATA 5,48,81,201,58,208,246,169,0
,141,152,194,141,153,194,141,1991
1690 DATA 155,194,160,0,153,205,197,20
0,192,12,208,248,160,0,232,189,3383
1700 DATA 128,5,48,48,201,34,240,44,32
,97,207,153,205,197,200,192,2281
1710 DATA 12,208,235,240,31,169,166,16
2,194,32,53,207,192,4,144,20,7595
1720 DATA 169,255,141,153,194,238,152,
194,173,152,194,201,10,144,5,169,996
1730 DATA 0,141,152,194,169,0,141,179,
197,162,0,134,242,169,155,164,1970
1740 DATA 35,96,169,0,141,156,194,141,
220,2,169,0,141,255,2,173,8661
1750 DATA 252,2,201,255,208,96,166,17,
240,24,173,220,2,201,17,240,421
1760 DATA 70,173,31,208,41,7,201,7,208
,23,169,0,141,160,194,76,6971
1770 DATA 187,203,108,181,194,104,104,
169,155,32,156,207,160,128,132,17,9124
1780 DATA 96,160,253,200,200,200,190,1
75,195,224,7,240,189,217,175,195,6877
1790 DATA 208,241,173,160,194,208,179,
238,160,194,185,176,195,133,212,185,70
78
1800 DATA 177,195,133,213,108,212,0,17
3,144,194,240,150,169,198,162,195,5009
1810 DATA 32,125,207,76,230,203,172,14
4,194,240,36,160,253,200,200,200,6492
1820 DATA 48,29,190,0,198,224,255,240,
22,217,0,198,208,239,162,255,5839
1830 DATA 142,252,2,185,1,198,190,2,19
8,32,125,207,76,187,203,160,933
1840 DATA 253,200,200,200,190,185,195,
224,255,240,135,217,185,195,208,241,5
1850 DATA 162,255,142,252,2,185,186,19
5,133,212,185,187,195,133,213,108,4854
1860 DATA 212,0,173,144,194,208,3,76,1
87,203,162,11,169,0,157,180,9073
1870 DATA 197,202,16,250,162,0,165,93,
129,94,169,128,141,180,197,169,1736
1880 DATA 0,141,159,194,32,182,207,192
,0,48,83,141,157,194,201,44,9213
1890 DATA 240,8,201,48,144,25,201,58,1
76,21,174,159,194,224,11,144,9425
1900 DATA 3,76,149,204,32,97,207,157,1
80,197,238,159,194,208,17,201,3786
1910 DATA 126,208,13,174,159,194,240,8
,169,0,157,180,197,206,159,194,3232
1920 DATA 174,159,194,169,128,157,180,
197,173,157,194,201,155,208,8,169,3512
1930 DATA 0,157,180,197,76,250,204,201
```

```
,27,240,3,76,149,204,174,159,2023
1940 DATA 194,169,0,157,180,197,76,230
,203,173,144,194,208,3,76,187,1901
1950 DATA 203,162,5,189,170,194,157,12
7,5,202,208,247,236,159,194,240,6738
1960 DATA 12,189,180,197,24,105,32,157
,133,5,232,208,239,201,44,208,2112
1970 DATA 14,160,0,185,176,194,157,133
,5,232,200,192,5,208,244,138,2965
1980 DATA 24,105,133,133,36,169,5,133,
37,169,0,141,162,2,141,255,7482
1990 DATA 2,169,125,32,156,207,173,155
,194,141,154,194,169,0,141,155,1276
2000 DATA 194,104,104,104,104,160,1,13
2,35,76,165,203,169,1,77,144,6669
2010 DATA 194,141,144,194,240,3,76,187
,203,173,145,194,141,48,2,173,9723
2020 DATA 146,194,141,49,2,173,151,194
,141,197,2,169,0,141,200,2,7125
2030 DATA 76,187,203,173,144,194,240,6
,165,99,197,82,208,8,169,118,258
2040 DATA 141,252,2,76,227,203,76,187,
203,141,157,194,172,144,194,240,5670
2050 DATA 4,164,87,240,3,76,156,207,16
4,17,208,3,76,142,206,172,9610
2060 DATA 255,2,208,244,166,82,240,113
,228,99,144,109,174,154,194,240,4927
2070 DATA 104,201,48,144,100,201,58,17
6,96,160,255,200,177,243,201,48,3971
2080 DATA 240,249,169,55,133,224,169,1
95,133,225,152,24,101,243,133,226,4030
2090 DATA 165,244,105,0,133,227,173,15
4,194,133,228,160,0,198,228,48,1992
2100 DATA 56,177,226,170,41,127,209,22
4,208,31,201,32,240,14,200,192,2255
2110 DATA 5,240,9,138,16,235,177,224,2
01,32,208,13,198,95,160,255,2652
2120 DATA 169,223,145,94,230,95,76,42,
206,24,169,5,101,224,133,224,175
2130 DATA 165,225,105,0,133,225,76,236
,205,173,157,194,172,79,3,240,2352
2140 DATA 111,172,9,210,192,18,208,30,
160,255,140,252,2,76,161,206,1581
2150 DATA 201,125,208,18,160,0,204,254
,2,208,11,204,162,2,208,6,7574
2160 DATA 140,156,194,76,166,205,201,1
55,240,6,166,85,228,83,144,64,581
2170 DATA 238,156,194,32,156,207,174,1
56,194,224,21,144,50,160,0,140,8993
2180 DATA 156,194,140,31,208,162,11,32
,167,207,169,255,141,252,2,205,2776
2190 DATA 252,2,208,14,166,17,208,247,
162,0,32,167,207,160,128,132,267
2200 DATA 17,96,141,252,2,160,0,140,25
5,2,162,0,32,167,207,96,7422
2210 DATA 166,85,228,83,176,16,170,41,
127,201,125,176,8,201,32,176,8910
2220 DATA 8,201,27,144,4,138,76,156,20
7,138,32,97,207,160,0,145,7925
2230 DATA 94,230,85,230,99,230,94,208,
2,230,95,177,94,133,93,73,9290
2240 DATA 128,145,94,96,173,144,194,24
0,76,173,49,2,201,197,240,44,983
2250 DATA 141,146,194,173,48,2,141,145
,194,165,88,141,151,197,165,89,1032
2260 DATA 141,152,197,169,144,141,48,2
,169,197,141,49,2,169,1,141,6183
2270 DATA 79,3,173,197,2,201,12,240,3,
141,151,194,169,12,141,197,9940
2280 DATA 2,141,23,208,169,2,141,200,2
,173,6,228,141,70,3,173,7085
2290 DATA 7,228,141,71,3,104,168,104,1
70,104,64,173,148,194,72,173,9945
2300 DATA 147,194,72,96,133,212,134,21
3,166,242,160,255,200,192,5,240,5951
2310 DATA 21,189,128,5,41,127,201,96,1
44,3,56,233,32,209,212,208,843
2320 DATA 8,232,201,32,208,230,160,128
,96,165,212,166,213,192,0,96,1918
2330 DATA 142,158,194,72,41,127,170,10
4,224,96,176,12,224,32,176,5,7783
2340 DATA 24,105,64,144,3,56,233,32,17
4,158,194,96,133,212,134,213,2127
2350 DATA 160,0,140,158,194,177,212,24
0,17,32,156,207,192,0,48,10,7047
2360 DATA 172,158,194,200,208,236,230,
213,208,232,96,168,173,150,194,72,5675
2370 DATA 173,149,194,72,152,96,160,0,
189,219,197,153,193,197,232,200,6008
2380 DATA 192,11,208,244,96,173,37,228
,72,173,36,228,72,96,4,228,9293
2390 DATA 7,228,95,202,153,205,93,242,
94,242,179,203,0,0,0,0,6199
```

LISTING 2: ASSEMBLY

```
0100        .OPT NO LIST
0110 ********************************
0120 *     E d i t    M a g i c     *
0130 *         Version 2.0          *
0140 *     By Bill Bodenstein       *
0150 *    For XL/XE systems only    *
0160 *          1/23/87             *
0170 ********************************
0180 ;
0190 ;This program will provide
0200 ;many simple utilities to
0210 ;make editing BASIC and MAC/65
0220 ;listings easier and faster.
0230 ;
0240 ;
0250 *** EQUATES ***
0260 ;
0270 INITCODE = $0500
0280 MAINCODE = $CA60
0290 DISPLIST = $C590
0300 MACLIST = $C600
0310 MISC =  $C290
0320 XITVBCODE = $C28A
0330 ;
0340 INITADR = $02E2
0350 RUNADR = $02E0
0360 ;
0370 LMARGIN = $52
0380 RMARGIN = $53
0390 COLCRS = $55
0400 LOGCOL = $63
0410 OLDADR = $5E
0420 OLDCHR = $5D
0430 SAVMSC = $58
0440 DINDEX = $57
0450 ;
0460 BRKKEY = $11
0470 CH  =   $02FC
0480 KBCODE = $D209
0490 HELPFG = $02DC
0500 KRPDEL = $02D9
0510 KEYREP = $02DA
0520 CONSOL = $D01F
0530 ESCFLG = $02A2
0540 DSPFLG = $02FE
0550 SSFLAG = $02FF
0560 ;
0570 CIX =   $F2
0580 INBUFF = $F3
0590 TEXTBUFF = $0580
0600 ICSTAZ = $23
0610 ICBALZ = $24
0620 ICIDNO = $2E
0630 ICPTL = $0346
0640 ICCOM = $0342
0650 ICBAL = $0344
0660 ICBLL = $0348
0670 PAUSEFLG = $034F
0680 CIO =   $E456
0690 ;
0700 SDMCTL = $022F
0710 DMACTL = $D400
0720 SDLSTL = $0230
0730 NMIEN = $D40E
0740 PORTB = $D301
0750 COLOR1 = $02C5
0760 COLOR4 = $02C8
0770 COLPF1 = $D017
0780 ;
0790 TRAMSZ = $06
0800 DOSVEC = $0A
0810 COLDST = $0244
0820 ;
0830 EHANDTAB = $E400
0840 KHANDTAB = $E420
0850 ;
0860 PUTREC = 9
0870 SPACE = 32
0880 ESC =   27
0890 CLEAR = 125
0900 DELETE = 126
0910 RETURN = 155
0920 BREAK = 128
0930 NOKEY = 255
0940 ;
0950 ;
0960     *=  INITCODE
0970 ;
0980 *** INITIALIZATION ***
0990 ;    ---------------
1000 ;  Copy o.s. from ROM to RAM and
1010 ;disable ROM to allow rest of
1020 ;Edit Magic code to load there.
1030 ;
```

```
1040 INIT
1050 ;
1060      LDA $FFF7   ;Is o.s. from
1070      CMP #2      ;XL/XE machine?
1080      BEQ DISABLED? ;Br if yep
1090 GOEXIT
1100      JMP EXIT    ;Else do nothing
1110 ;
1120 DISABLED?
1130      LDA PORTB   ;Is os ROM/RAM?
1140      TAX         ;Save ROM status
1150      AND #255-1  ;Make disabled
1160      CMP PORTB   ;Is it already?
1170      BEQ GOEXIT  ;Br if yup
1180      STA $D4     ;Save new status
1190 ;
1200      LDA # <$C000 ;Start copying
1210      STA $D6     ;o.s. from
1220      LDA # >$C000 ;$c000 to $ffff
1230      STA $D7
1240 ;
1250      LDA SDMCTL  ;Save DMA status
1260      PHA
1270      LDA #0      ;Disable NMI
1280      STA NMIEN   ;so no VBI while
1290      STA SDMCTL  ;moving o.s.
1300      STA DMACTL  ;Turn screen off
1310      SEI         ;No IRQ
1320 ;
1330      LDY #0
1340 COPYO.S.
1350      STX PORTB   ;Enable ROM
1360      LDA ($D6),Y ;Get a byte
1370      DEC PORTB   ;Enable RAM
1380      STA ($D6),Y ;Put o.s. there
1390      INY
1400      BNE COPYO.S. ;And move more
1410 ;
1420      INC $D7     ;Next page
1430      BEQ NMION   ;Done if $0000
1440      LDA $D7     ;Skip over
1450      CMP #$D0    ;hardware chips
1460      BNE COPYO.S.
1470      LDA #$D8    ;Set next page
1480      STA $D7     ;to $D800
1490      JMP COPYO.S. ;and loop back
1500 ;
1510 NMION
1520      CLI         ;Re-enable IRQ
1530      LDA #64     ;and non-mskable
1540      STA NMIEN   ;interrupts
1550      PLA         ;And turn scr
1560      STA SDMCTL  ;back on
1570 ;
1580      LDA $D4     ;Make ROM
1590      STA PORTB   ;disabled
1600 ;
1610 ;Let user know what's going on
1620 ;with a short message.
1630 ;
1640      LDX #$00    ;Use scr editor
1650      LDA # <INITMSG
1660      STA ICBAL,X ;Point to our
1670      LDA # >INITMSG ;message
1680      STA ICBAL+1,X
1690      STA ICBLL+1,X ;Whatever len
1700      LDA #PUTREC ;Print record
1710      STA ICCOM,X
1720      JSR CIO     ;CIO prints it
1730 ;
1740 ;Save editor handler's vectors
1750 ;for GET and PUT operations so
1760 ;Edit Magic can call them later.
1770 ;
1780      LDA EGET    ;Get vector
1790      STA SAVEGETV
1800      LDA EGET+1
1810      STA SAVEGETV+1
1820      LDA EPUT    ;Put vector
1830      STA SAVEPUTV
1840      LDA EPUT+1
1850      STA SAVEPUTV+1
1860 ;
1870 EXIT RTS         ;Let DOS load
1880 ;
1890 INITMSG .BYTE "Installing Edit Ma
gic ...",155
1900 ;
1910      *=  INITADR
1920      .WORD INIT
1930 ;
1940 ;
1950      *=  MAINCODE
1960 ;
1970 *** MAIN CODE ***
1980 ;    ---------
```

```
1990 ;  Edit Magic will be stored
2000 ;over unused o.s. code.
2010 ;(Intrn'l charset,warmstart,
2020 ; and self-test screen data.)
2030 ;There are four entry points:
2040 ;1) From GET operations
2050 ;2) Prior to every keypress
2060 ;3) From PUT operations
2070 ;4) At the end of every VBI
2080 ;
2090 ;GET editor entry:
2100 ;-----------------
2110 ;  Handle features before and
2120 ;after getting a line of text
2130 ;from the screen editor.
2140 ;
2150 GETENTRY
2160 ;
2170      LDA #'>-32  ;">" in listbox
2180      STA LISTBOX ;says active
2190 ;
2200      LDX # <[MSG1-MSGS]
2210      JSR PUTMSG  ;Display title
2220 ;
2230      LDA SAVECNT ;Get # times
2240      CLC         ;file SAVEd
2250      ADC #16     ;Conv to intrnal
2260      BIT SAVEDFLG ;If recently
2270      BEQ PUTCNT  ; saved,
2280      ORA #128    ; inverse #
2290 PUTCNT
2300      STA BAKUPCNT ;Put # on scr
2310 ;
2320      LDA EMFLG   ;Is Edit Magic
2330      BEQ GOGET   ;on?
2340 ;
2350 ;
2360      LDA #0      ;Clear lnno cntr
2370      STA STCKCNT ;for later use
2380 ;
2390      LDA #32     ;Shorter key
2400      STA KRPDEL  ;repeat delay
2410      LDA #4      ;and delay
2420      STA KEYREP  ;between repeats
2430 ;
2440      LDA XITVBCODE ;Has our jmp
2450      CMP #76     ;been set yet?
2460      BEQ GOGET   ;Br if yes
2470      LDA #0      ;Prevent VBI
2480      STA NMIEN
2490      LDA #76     ;JMP op-code
2500      STA XITVBCODE ;Put jmp to
2510      LDA # <XITVBI ; our vbi exit
2520      STA XITVBCODE+1 ;routine
2530      LDA # >XITVBI ; in place of
2540      STA XITVBCODE+2 ;system's
2550      LDA #64     ;Allow VBI again
2560      STA NMIEN
2570 ;
2580 ;
2590 GOGET
2600      LDX ICIDNO  ;Get iocb index
2610      JSR EGETCHAR ;Let os do work
2620      STY ICSTAZ  ;Save status
2630      CPY #BREAK  ;Exit if <brk>
2640      BCS GOCIO   ;pressed
2650      CMP #RETURN ;Or not <ret>
2660      BEQ PUTEOL
2670 GOCIO RTS        ;Let CIO finish
2680 ;
2690 PUTEOL
2700      LDX #0      ;Put EOL at
2710      STA (ICBALZ,X) ;end of line
2720 ;
2730      LDA EMFLG   ;Skip features
2740      BEQ RETCIO2 ;if E.M. off
2750 ;
2760 ;
2770 ;A line of text is in the text
2780 ;buffer. Save line and see if
2790 ;it has a LOAD or SAVE command,
2800 ;or a line number.
2810 ;
2820      LDA TEXTBUFF ;If blank line
2830      CMP #RETURN
2840      BEQ RETCIO2 ; do nothing
2850 ;
2860 ;
2870      LDX #255    ;Save inputted
2880 SAVCHR ;         line of text
2890      INX
2900      BMI RETCIO2 ;Emergency exit
2910      LDA TEXTBUFF,X ;Move line
2920      STA SAVTBUFF,X ;to storage
2930      CMP #RETURN ;   until eol
2940      BNE SAVCHR
```

```
2950 ;
2960      LDA #0       ;For macro: a
2970      STA SAVTBUFF,X ;zero ends
2980 ;
2990      LDX #255
3000 SKIPLEADING
3010      INX
3020      BMI RETCIO2
3030      LDA TEXTBUFF,X ;Skip over
3040      CMP #RETURN ;until eol
3050      BEQ RETCIO2
3060      CMP #SPACE  ;leading blanks
3070      BEQ SKIPLEADING
3080      CMP #'0     ;and leading 0's
3090      BEQ SKIPLEADING
3100 ;
3110      STX CIX     ;Save index
3120 ;
3130 LNNO?
3140      LDA TEXTBUFF,X
3150      CMP #'1     ;Is line begun
3160      BCC LOAD?   ;by linenum?
3170      CMP #'9+1
3180      BCS LOAD?
3190 ;
3200      LDA #0      ;Remind user
3210      STA SAVEDFLG ;program has
3220 ;                been modified
3230      LDA LNNOCNT ;Save linenum in
3240      STA $E0     ;stack if room
3250      CMP #24     ;Only sav 24 #'s
3260      BCS RETCIO2
3270 ;
3280      LDA # <LNNOSTACK ;Is num in
3290      LDX # >LNNOSTACK ;stack?
3300 LOOKSTACK
3310      DEC $E0     ;Look thru stack
3320      BMI PUTINSTACK ;Br when done
3330      JSR CMPTEXT ;Compare nums
3340      BMI RETCIO2 ;Exit if # there
3350 ;
3360      CLC         ;Each lnno in
3370      ADC #5      ;stack is 5 chrs
3380      BCC LOOKSTACK ;Loop back
3390      INX
3400      BNE LOOKSTACK
3410 ;
3420 PUTINSTACK
3430      STA $D4     ;Save addr of
3440      STX $D5     ;open spot
3450      LDX CIX     ;Index 1st char
3460      LDY #255
3470 MOVCHR1
3480      INY
3490      CPY #5      ;Limit of 5 chrs
3500      BEQ INCSTCK ;Br when done
3510      LDA TEXTBUFF,X
3520      STA ($D4),Y
3530      INX
3540      CMP #SPACE  ;Move 'til blank
3550      BNE MOVCHR1
3560 INCSTCK
3570      INC LNNOCNT ;Inc stack size
3580 ;
3590 RETCIO2
3600      JMP RETCIO  ;Exit to CIO
3610 ;
3620 ;
3630 LOAD?
3640      LDA # <LOAD ;Compare first 5
3650      LDX # >LOAD ;chrs to "LOAD "
3660      JSR CMPTEXT
3670      CPY #4      ;Match if "LOAD"
3680      BCC SAVE?   ;Br if not match
3690 ;
3700 FINDCOLON
3710      INX         ;Find filename
3720      LDA TEXTBUFF,X
3730      BMI RETCIO  ;Exit if <ret>
3740      CMP #':     ;Colon precedes
3750      BNE FINDCOLON
3760 ;
3770      LDA #0      ;Reset save cntr
3780      STA SAVECNT
3790      STA SAVEDFLG ;File not saved
3800      STA LNNOCNT ;and stack cntr
3810      LDY #0
3820 CLRFIRST
3830      STA FILEBOX,Y ;Clear filebox
3840      INY
3850      CPY #12     ;12 chars in box
3860      BNE CLRFIRST
3870 ;
3880 MOVFN
3890      LDY #0      ;Transfer fname
3900 MOVCHR2 ;        to filebox
```

```
3910      INX
3920      LDA TEXTBUFF,X
3930      BMI RETCIO  ;Exit if <ret>
3940      CMP #'"     ;or quotes
3950      BEQ RETCIO
3960      JSR CONVCHAR ;Make internal
3970      STA FILEBOX,Y ;And poke it
3980      INY
3990      CPY #12     ;Max 12 chars
4000      BNE MOVCHR2
4010      BEQ RETCIO
4020 ;
4030 ;
4040 SAVE?
4050      LDA # <SAVE ;See if "SAVE"
4060      LDX # >SAVE ;in text line
4070      JSR CMPTEXT
4080      CPY #4      ;Blnk not needed
4090      BCC RETCIO  ;Br if nope
4100 ;
4110      LDA #255    ;Program saved
4120      STA SAVEDFLG
4130 ;
4140      INC SAVECNT ;Inc # of saves
4150      LDA SAVECNT ;Make sure<9
4160      CMP #10
4170      BCC RETCIO
4180      LDA #0      ;Else reset to 0
4190      STA SAVECNT
4200 ;
4210 ;
4220 RETCIO
4230      LDA #SPACE-32 ;Remove ">"
4240      STA LISTBOX ;  prompt
4250      LDX #0
4260      STX CIX     ;Reset char indx
4270      LDA #RETURN ;Last char=<ret>
4280      LDY ICSTAZ  ;Set status
4290      RTS         ;Return to CIO
4300 ;
4310 ;
4320 ;
4330 ;Get key from editor:
4340 ;--------------------
4350 ;  Before editor handles
4360 ;keypress or <break>, check for
4370 ;Edit Magic keys, macro keys and
4380 ;console buttons.
4390 ;
4400 EGETKEY
4410      LDA #0      ;Clear pauser
4420      STA ROWCNTR ;for scr output
4430      STA HELPFG  ;& <help> press
4440 ;
4450 WAITFORKEY
4460      LDA #0      ;So ctrl-1 does
4470      STA SSFLAG  ;nothing
4480 ;
4490      LDA CH      ;Key pressed?
4500      CMP #NOKEY
4510      BNE CMPKEY1 ;Br if yep
4520 ;
4530      LDX BRKKEY  ;<BRK> pressed?
4540      BEQ RETEGET ;Exit if yep
4550 ;
4560      LDA HELPFG  ;[HELP] pressed?
4570      CMP #17
4580      BEQ HELPUSER ;Br if yep
4590 ;
4600      LDA CONSOL  ;Button pressed?
4610      AND #7
4620      CMP #7
4630      BNE CMPBUTTN ;Br if yep
4640      LDA #0      ;Need to know if
4650      STA CONSOLREL ;console has
4660 ;                     been released
4670      JMP WAITFORKEY ;Loop if nope
4680 ;
4690 ;
4700 RETEGET
4710      JMP (EGETKV) ;Return to o.s.
4720 ;
4730 EXITEGET
4740      PLA         ;Abort editor
4750      PLA         ;routine in o.s.
4760      LDA #RETURN ;Print <ret> to
4770      JSR EPUTCHAR ;move cursor
4780      LDY #BREAK  ;And tell CIO
4790      STY BRKKEY  ;<brk> pressed
4800      RTS
4810 ;
4820 ;
4830 CMPBUTTN
4840      LDY #0-3    ;Look thru E.M.
4850 CMPCONSOL ;      consol buttons
4860      INY         ;3-byte entries
```

```
4870      INY
4880      INY
4890      LDX EMCONSOL,Y ;Exit if end
4900      CPX #7         ;   of entries
4910      BEQ WAITFORKEY
4920      CMP EMCONSOL,Y ;Match?
4930      BNE CMPCONSOL ;Loop if nope
4940 ;
4950      LDA CONSOLREL ;Console been
4960      BNE WAITFORKEY ;released?
4970      INC CONSOLREL ;Set flag
4980 ;
4990 GOCONSOL
5000      LDA EMCONSOL+1,Y
5010      STA $D4        ;If found,
5020      LDA EMCONSOL+2,Y
5030      STA $D5        ;jump to addr
5040      JMP ($D4)
5050 ;
5060 ;
5070 HELPUSER
5080      LDA EMFLG      ;Ignore <help>
5090      BEQ EGETKEY ;if E.M. off
5100      LDA # <HELPMSG ;Print help
5110      LDX # >HELPMSG ;info
5120      JSR PRINTMSG
5130      JMP EXITEGET ;Abort input
5140 ;
5150 ;
5160 CMPKEY1
5170      LDY EMFLG      ;If E.M. off
5180      BEQ CMPKEY2 ;br over macros
5190      LDY #0-3       ;Look for match
5200 CMPMAC ;            in macro keys
5210      INY            ;3-byte entries
5220      INY
5230      INY
5240      BMI CMPKEY2 ;Emergency exit
5250      LDX MACKEYS,Y
5260      CPX #255       ;End of keys?
5270      BEQ CMPKEY2 ;Br if yup
5280      CMP MACKEYS,Y ;Match?
5290      BNE CMPMAC     ;Loop if nope
5300 ;
5310      LDX #NOKEY     ;Clear keypress
5320      STX CH
5330      LDA MACKEYS+1,Y ;Get addr of
5340      LDX MACKEYS+2,Y ;macro msg
5350      JSR PRINTMSG ;Print it
5360      JMP WAITFORKEY ;And go back
5370 ;
5380 CMPKEY2
5390      LDY #0-3       ;Look thru E.M.
5400 CMPEM ;             special keys
5410      INY            ;3-byte entries
5420      INY
5430      INY
5440      LDX EMKEYS,Y ;Last entry?
5450      CPX #255
5460      BEQ RETEGET ;Exit if yep
5470      CMP EMKEYS,Y ;Match?
5480      BNE CMPEM      ;Loop if nope
5490 ;
5500      LDX #NOKEY     ;Clear keypress
5510      STX CH
5520      LDA EMKEYS+1,Y ;Jump to addr
5530      STA $D4        ;   in entry
5540      LDA EMKEYS+2,Y
5550      STA $D5
5560      JMP ($D4)
5570 ;
5580 ;
5590 ;If [START] or SHIFT-ESC
5600 ;pressed, enter LIST BOX.
5610 ;
5620 *** ENTER LIST BOX ***
5630 ENTERLBOX
5640      LDA EMFLG      ;Exit if Edit
5650      BNE CLRLBOX ;Magic off
5660 JBACK JMP WAITFORKEY ;Loop back
5670 ;
5680 CLRLBOX
5690      LDX #11        ;Blank out box
5700      LDA #0         ;ie. clr scr mem
5710 CLRBYTE
5720      STA LISTBOX+1,X
5730      DEX            ;Box = 12 chars
5740      BPL CLRBYTE ;plus ">" prompt
5750 ;
5760 MAKECURSOR
5770      LDX #0         ;Turn curs off
5780      LDA OLDCHR     ;by restoring
5790      STA (OLDADR,X) ;non-inv char
5800      LDA #128       ;And put in box
5810      STA LISTBOX+1 ;pseudo-cursor
5820 ;
5830 STARTINPUT
5840      LDA #0         ;Initialize
5850      STA CHARINDX ;index of char
5860 ;
5870 ;Receive input from keyboard
5880 ;and display in LIST BOX.
5890 ;
5900 GETINPUT
5910      JSR KGETCHAR ;Get a keypress
5920      CPY #0         ;Check for error
5930      BMI EXITLBOX ;Exit if <brk>
5940      STA SAVCHAR ;Save keypress
5950 ;
5960 NUMBER?
5970      CMP #',        ;Allow comma
5980      BEQ CHCKINDX
5990      CMP #'0        ;and 0-9 only
6000      BCC DELETE?
6010      CMP #'9+1      ;in listbox
6020      BCS DELETE?
6030 ;
6040 CHCKINDX
6050      LDX CHARINDX ;Ignore key if
6060      CPX #11        ; box is full
6070      BCC PUTINBOX ;Br if not full
6080      JMP GETINPUT ;Loop if it is
6090 ;
6100 PUTINBOX
6110      JSR CONVCHAR ;Put num/comma
6120      STA LISTBOX+1,X ;in box
6130      INC CHARINDX ;Point nxt char
6140      BNE PUTCURSOR ;and branch
6150 ;
6160 DELETE?
6170      CMP #DELETE ;Allow <delete>
6180      BNE PUTCURSOR ;Br if it isnt
6190 ;
6200 CLRCURSOR
6210      LDX CHARINDX ;Backspace if
6220      BEQ PUTCURSOR ;chars in box
6230      LDA #0         ;Erase last char
6240      STA LISTBOX+1,X
6250      DEC CHARINDX ;One less char
6260 ;
6270 PUTCURSOR
6280      LDX CHARINDX ;Put curs block
6290      LDA #128       ;in listbox
6300      STA LISTBOX+1,X
6310 ;
6320 RETURN?
6330      LDA SAVCHAR ;Done if <ret>
6340      CMP #RETURN
6350      BNE ESC?       ;Br if not
6360      LDA #0         ;Clr p-cursor
6370      STA LISTBOX+1,X
6380      JMP LISTLINES ;List contents
6390 ;
6400 ESC?
6410      CMP #ESC       ;Abort if <esc>
6420      BEQ EXITLBOX ;Br if yep
6430      JMP GETINPUT ;Else loop back
6440 ;
6450 ;Leave input from LIST BOX and
6460 ;move back to normal screen.
6470 ;
6480 EXITLBOX
6490      LDX CHARINDX ;Remove
6500      LDA #0         ;pseudo-cursor
6510      STA LISTBOX+1,X
6520      JMP EXITEGET ;Abort input
6530 ;
6540 ;
6550 ;If [SELECT] or CTRL-ESC
6560 ;pressed, list contents of LIST
6570 ;BOX by returning LIST+lnnos
6580 ;to CIO as inputted line.
6590 ;
6600 *** LIST LINES ***
6610 ;
6620 LISTLINES
6630      LDA EMFLG      ;Exit if EM off
6640      BNE MOVLIST
6650      JMP WAITFORKEY ;Loop back
6660 ;
6670 MOVLIST
6680      LDX #5         ;Move "LIST "
6690 MOVCHR3 ;           into text buffer
6700      LDA LIST-1,X
6710      STA TEXTBUFF-1,X
6720      DEX
6730      BNE MOVCHR3
6740 ;
6750 MOVCHR4
6760      CPX CHARINDX ;Put chars in
6770      BEQ MOVHILNNO? ; listbox
6780      LDA LISTBOX+1,X ;after
```

```
6790      CLC        ;    "LIST "
6800      ADC #32    ;Conv to ascii
6810      STA TEXTBUFF+5,X
6820      INX
6830      BNE MOVCHR4
6840 ;
6850 MOVHILNNO?
6860      CMP #',    ;If last chr=","
6870      BNE SETICB ;put "32767"
6880 ;             (sorry MAC/65 users)
6890      LDY #0
6900 MOV32767
6910      LDA HILNNO,Y ;"32767" in
6920      STA TEXTBUFF+5,X ;ascii code
6930      INX
6940      INY
6950      CPY #5
6960      BNE MOV32767
6970 ;
6980 SETICB
6990      TXA        ;Tell editor
7000      CLC        ;where input is
7010      ADC #128+5
7020      STA ICBALZ ;Zero-page IOCB
7030      LDA # >TEXTBUFF
7040      STA ICBALZ+1
7050 ;
7060 SETUPSCR
7070      LDA #0     ;Clear esc flag
7080      STA ESCFLG ;in case pressed
7090      STA SSFLAG ;and ctrl-1 flg
7100      LDA #CLEAR ;so screen will
7110      JSR EPUTCHAR ;clear and list
7120 ;
7130      LDA LNNOCNT ;Save size of
7140      STA STCKCNT ;stack and
7150 ;
7160      LDA #0     ;Reset stack of
7170      STA LNNOCNT ;lnnos edited
7180      PLA        ;Throw out
7190      PLA        ;return addrs
7200      PLA
7210      PLA
7220      LDY #1     ;Set status to
7230      STY ICSTAZ ;no error
7240      JMP RETCIO ;Go right to CIO
7250 ;
7260 ;
7270 ;If [OPTION] or SHIFT-CTRL-ESC
7280 ;pressed, toggle Edit Magic
7290 ;on/off. When turned on, VBI
7300 ;routine will re-display top
7310 ;status line.
7320 ;
7330 *** TOGGLE EDIT MAGIC ***
7340 TOGEDTMAG
7350      LDA #1     ;Toggle Edit
7360      EOR EMFLG  ;Magic flag
7370      STA EMFLG  ;on/off
7380      BEQ EMOFF  ;Br if now off
7390 ;
7400 EMON
7410      JMP WAITFORKEY ;Get s'more
7420 ;
7430 EMOFF
7440      LDA SAVSDLSTL ;Restore old
7450      STA SDLSTL  ;  display list
7460      LDA SAVSDLSTL+1
7470      STA SDLSTL+1
7480 ;
7490      LDA SAVCOLR1 ;Restore char
7500      STA COLOR1  ;brightness
7510      LDA #0      ;and black
7520      STA COLOR4  ;border color
7530 ;
7540      JMP WAITFORKEY ;Go back
7550 ;
7560 ;
7570 ;If <CLEAR> key (SHIFT-<)
7580 ;pressed and cursor not on left
7590 ;margin, ignore it.
7600 ;
7610 *** HANDLE SHIFT-CLEAR ***
7620 SHIFTCLR
7630      LDA EMFLG   ;Exit if EM off
7640      BEQ CLRIT
7650 ;
7660      LDA LOGCOL  ;Start of
7670      CMP LMARGIN ;logical line?
7680      BNE IGNOREIT ;Br if nope
7690 CLRIT
7700      LDA #54+64  ;Replace
7710      STA CH      ;keypress
7720      JMP RETEGET ;and exit
7730 IGNOREIT
7740      JMP WAITFORKEY ;Ignore <clr>
```

```
7750 ;
7760 ;
7770 ;
7780 ;Put character to editor:
7790 ;-----------------------
7800 ;  Use faster print routine to
7810 ;speed up screen output. Also
7820 ;allow pause, and check for
7830 ;line number preceding text.
7840 ;If lnno in stack, mark it.
7850 ;
7860 PUTENTRY
7870      STA SAVCHAR ;Save char
7880 ;
7890      LDY EMFLG   ;Exit if Edit
7900      BEQ GO.OS   ;Magic off
7910 ;
7920      LDY DINDEX  ;If not txt mode
7930      BEQ BREAK?
7940 GO.OS
7950      JMP EPUTCHAR ;let o.s. print
7960 ;
7970 BREAK?
7980      LDY BRKKEY  ;Abort if <brk>
7990      BNE CTRL1?
8000      JMP ABORTOUTPUT
8010 ;
8020 CTRL1?
8030      LDY SSFLAG  ;Loop if CTRL-1
8040      BNE BREAK?  ;pressed
8050 ;
8060 ;If start of text line, check
8070 ;for linenumber. If line has one
8080 ;and matches one in stack, mark
8090 ;the line.
8100 ;
8110      LDX LMARGIN ;Room to mark?
8120      BEQ PAUSEON? ;Br if nope
8130      CPX LOGCOL  ;Start of line?
8140      BCC PAUSEON? ;Br if nope
8150 ;
8160      LDX STCKCNT ;Have any lnnos
8170      BEQ PAUSEON? ;been edited?
8180 ;
8190      CMP #'0     ;Is char part
8200      BCC PAUSEON? ;of a lnno?
8210      CMP #'9+1
8220      BCS PAUSEON? ;Br if nope
8230 ;
8240      LDY #255    ;Skip over
8250 SKIP0 ;           leading zeroes
8260      INY
8270      LDA (INBUFF),Y ;Points to #
8280      CMP #'0     ;Zero char?
8290      BEQ SKIP0   ;Loop if yup
8300 ;
8310      LDA # <LNNOSTACK ;We'll look
8320      STA $E0     ;    thru stack
8330      LDA # >LNNOSTACK ;to see if
8340      STA $E1     ;    same lnno
8350 ;                has been edited
8360      TYA         ;Indxes 1st char
8370      CLC
8380      ADC INBUFF
8390      STA $E2     ;Point to lnno
8400      LDA INBUFF+1
8410      ADC #0
8420      STA $E3
8430 ;
8440      LDA STCKCNT ;Save counter
8450      STA $E4
8460 ;
8470 LOOKLNNO
8480      LDY #0
8490      DEC $E4     ;Done when all
8500      BMI PAUSEON? ;lnnos checked
8510 LOOKCHR
8520      LDA ($E2),Y ;Char from line
8530      TAX         ;Save it
8540      AND #127    ;In case inverse
8550      CMP ($E0),Y ;Char in stack
8560      BNE NEXTLNNO
8570      CMP #SPACE  ;Done if blank
8580      BEQ MATCHLNNO
8590      INY         ;Next char
8600      CPY #5      ;Max 5 chars
8610      BEQ MATCHLNNO ;Assume match
8620      TXA         ;Last # char?
8630      BPL LOOKCHR ;Loop if not yet
8640 ;
8650      LDA ($E0),Y ;Is next char
8660      CMP #SPACE  ;a blank?
8670      BNE NEXTLNNO ;Br if no
8680 ;
8690 MATCHLNNO ;     Match found!
8700      DEC OLDADR+1 ;Mark line by
```

```
8710      LDY #255      ;putting arrow
8720      LDA #'→+64    ;left of cursor
8730      STA (OLDADR),Y
8740      INC OLDADR+1
8750      JMP PAUSEON?
8760 ;
8770 NEXTLNNO
8780      CLC
8790      LDA #5        ;Look at next
8800      ADC $E0       ;lnno in stack
8810      STA $E0
8820      LDA $E1
8830      ADC #0
8840      STA $E1
8850      JMP LOOKLNNO
8860 ;
8870 ;
8880 PAUSEON?
8890      LDA SAVCHAR
8900      LDY PAUSEFLG
8910      BEQ FASTPRNT
8920 ;
8930      LDY KBCODE  ;If last key='C'
8940      CPY #18
8950      BNE COUNTLN
8960      LDY #NOKEY    ;clear keypress
8970      STY CH
8980      JMP FASTPRNT ;Don't pause
8990 ;
9000 CLEAR?
9010      CMP #CLEAR    ;If <clr> char
9020      BNE COUNTLN  ;reset row cntr
9030      LDY #0        ;only if screen
9040      CPY DSPFLG    ;will clear
9050      BNE COUNTLN  ;Br if not
9060      CPY ESCFLG
9070      BNE COUNTLN  ;Br if not again
9080      STY ROWCNTR ;Clear # lines
9090      JMP GO.OS     ;and print <clr>
9100 ;
9110 COUNTLN
9120      CMP #RETURN ;Keep count of
9130      BEQ INCLN     ;lines printed
9140      LDX COLCRS
9150      CPX RMARGIN
9160      BCC FASTPRNT
9170 INCLN INC ROWCNTR
9180 ;
9190 ;Every 22nd line printed, pause
9200 ;output and wait for keypress.
9210 ;
9220 PAUSESCR?
9230      JSR EPUTCHAR ;Print char
9240      LDX ROWCNTR ;Time to pause?
9250      CPX #21
9260      BCC LEAVE     ;Br if no
9270 ;
9280      LDY #0        ;Reset # rows
9290      STY ROWCNTR ;printed
9300      STY CONSOL    ;Click speaker
9310 ;
9320 PUTPROMPT
9330      LDX # <[MSG2-MSGS]
9340      JSR PUTMSG    ;Put "press key"
9350 ;                     msg on top line
9360      LDA #NOKEY    ;Clear keypress
9370      STA CH
9380 WAITFORKEY2
9390      CMP CH        ;Wait for user
9400      BNE CONTOUTPUT ;to hit a key
9410      LDX BRKKEY    ;   or <brk>
9420      BNE WAITFORKEY2
9430 ;
9440      LDX # <[MSG1-MSGS]
9450      JSR PUTMSG    ;Clr prompt
9460 ;
9470 ABORTOUTPUT
9480      LDY #BREAK    ;<BREAK> pressed
9490      STY BRKKEY
9500      RTS
9510 ;
9520 CONTOUTPUT
9530      STA CH        ;Clear keypress
9540      LDY #0        ;Clear ctrl-1
9550      STY SSFLAG    ;pause
9560 CLRPROMPT
9570      LDX # <[MSG1-MSGS]
9580      JSR PUTMSG    ;Restore title
9590 ;
9600 LEAVE RTS      ;Go back to CIO
9610 ;
9620 ;Put character directly in
9630 ;screen memory unless scrolling
9640 ;or control character.
9650 ;
9660 FASTPRNT
```

```
9670      LDX COLCRS  ;Will it scroll?
9680      CPX RMARGIN
9690      BCS PRINTCHR ;Br if maybe
9700      TAX           ;Save char
9710      AND #127
9720      CMP #125      ;Is char a
9730      BCS PRINTIT ;non-control
9740      CMP #32       ;char?
9750      BCS PUTIT
9760      CMP #27
9770      BCC PUTIT
9780 ;
9790 PRINTIT
9800      TXA           ;Regain char
9810 PRINTCHR
9820      JMP EPUTCHAR ;Let o.s. print
9830 ;
9840 PUTIT
9850      TXA           ;Regain char
9860      JSR CONVCHAR ;Make internal
9870      LDY #0        ;Put char in
9880      STA (OLDADR),Y ;curs pos in
9890 ;                     screen mem
9900 MOVCURS
9910      INC COLCRS    ;Move cursor
9920      INC LOGCOL    ;pointers to
9930      INC OLDADR    ;next column
9940      BNE PUTCURS
9950      INC OLDADR+1
9960 ;
9970 PUTCURS
9980      LDA (OLDADR),Y ;Save char
9990      STA OLDCHR    ;under curs and
010000    EOR #128      ;inverse it
010010    STA (OLDADR),Y
010020 ;
010030 EXITPUTCHR RTS   ;Return to CIO
010040 ;
010050 ;
010060 ;
010070 ;Handle VBI:
010080 ;-----------
010090 ;Before exiting vert. blank
010100 ;int, make sure Edit Magic's
010110 ;settings are still intact.
010120 ;
010130 XITVBI
010140    LDA EMFLG     :Edit Magic on?
010150    BEQ EXITVBI ;Br if nope
010160 ;
010170    LDA SDLSTL+1 ;If not using
010180    CMP # >DLIST ;our disp list,
010190    BEQ SETCOLR
010200    STA SAVSDLSTL+1 ;save addr
010210    LDA SDLSTL   ;   of theirs
010220    STA SAVSDLSTL
010230    LDA SAVMSC    ;Let dl know
010240    STA SCRMEM    ;where scr is
010250    LDA SAVMSC+1
010260    STA SCRMEM+1
010270    LDA # <DLIST ;and use ours
010280    STA SDLSTL
010290    LDA # >DLIST
010300    STA SDLSTL+1
010310 ;
010320    LDA #1        ;Reset pause for
010330    STA PAUSEFLG ;listing
010340 ;
010350    LDA COLOR1    ;If new char
010360    CMP #12       ;brightness
010370    BEQ SETCOLR
010380    STA SAVCOLR1 ;save it
010390 ;
010400 SETCOLR
010410    LDA #12       ;Brighten chars
010420    STA COLOR1
010430    STA COLPF1
010440    LDA #2        ;Lighten border
010450    STA COLOR4
010460 ;
010470    LDA EPUT      ;Make sure our
010480    STA ICPTL     ;put char vect
010490    LDA EPUT+1    ;is used
010500    STA ICPTL+1
010510 ;
010520 EXITVBI ;       Restore regs
010530    PLA           ;(Tracy DuMont
010540    TAY           ;said she'd pay
010550    PLA           ;me a dollar if
010560    TAX           ;I put her name
010570    PLA           ;here.)
010580    RTI           ;Leave vbi
010590 ;
010600 ;
010610 ;
010620 *** SUBROUTINES ***
```

```
010630 ;  -----------
010640 ;
010650 ;Jump to o.s. GET from editor
010660 ;
010670 EGETCHAR
010680   LDA SAVEGETV+1
010690   PHA
010700   LDA SAVEGETV
010710   PHA
010720   RTS
010730 ;
010740 ;
010750 ;Compare start of entered text
010760 ;with up to 5 characters.
010770 ;
010780 CMPTEXT
010790   STA $D4       ;Passed addr of
010800   STX $D5       ;text to compare
010810   LDX CIX       ;Indx of 1st chr
010820   LDY #255
010830 CMPCHR
010840   INY           ;Check up to 5
010850   CPY #5        ;characters
010860   BEQ MATCH     ;Br if done
010870   LDA TEXTBUFF,X ;Char from ln
010880   AND #127      ;In case inverse
010890   CMP #96       ;Lowercase?
010900   BCC SAMECHR? ;Br if no
010910   SEC           ;Make uppercase
010920   SBC #32
010930 SAMECHR?
010940   CMP ($D4),Y ;Char from E.M.
010950   BNE NOMATCH
010960   INX
010970   CMP #SPACE  ;Blank ends
010980   BNE CMPCHR
010990 ;
011000 MATCH
011010   LDY #128      ;Set N-flg
011020   RTS
011030 ;
011040 NOMATCH
011050   LDA $D4       ;Restore arg adr
011060   LDX $D5
011070   CPY #0        ;Clear N-flg
011080   RTS
011090 ;
011100 ;
011110 ;Convert char from ASCII to
011120 ;internal (screen).
011130 ;
011140 CONVCHAR
011150   STX SAVREG  ;Save xreg
011160   PHA         ;Acc=ascii char
011170   AND #127    ;Make non-inv
011180   TAX         ;Save in xreg
011190   PLA         ;Restore char
011200   CPX #96     ;Lowercase?
011210   BCS NOTRNS  ;Br if yep
011220   CPX #32     ;Graphics char?
011230   BCS TRNS2   ;Br if nope
011240 TRNS1
011250   CLC         ;Conv grphic chr
011260   ADC #64
011270   BCC NOTRNS
011280 TRNS2
011290   SEC         ;Conv uppercs
011300   SBC #32
011310 NOTRNS
011320   LDX SAVREG  ;Restore xreg
011330   RTS
011340 ;
011350 ;
011360 ;Print a message.
011370 ;Passed: Acc&xreg contain addr
011380 ;of message. Print char by
011390 ;char until zero value found.
011400 ;
011410 PRINTMSG
011420   STA $D4       ;Save lo addr
011430   STX $D5       ;Save hi addr
011440   LDY #0
011450 PRINTCHAR
011460   STY SAVREG  ;Save index
011470   LDA ($D4),Y ;Get msg char
011480   BEQ EXITPRNT ;Exit if 0
011490   JSR EPUTCHAR
011500   CPY #0        ;<brk> pressed?
011510   BMI EXITPRNT ;Exit if <brk>
011520   LDY SAVREG  ;Restore index
011530   INY
011540   BNE PRINTCHAR ;Loop back
011550   INC $D5
011560   BNE PRINTCHAR
011570 ;
011580 EXITPRNT
011590   RTS
011600 ;
011610 ;
011620 ;Print a single character.
011630 ;
011640 EPUTCHAR
011650   TAY           ;Acc=char
011660   LDA SAVEPUTV+1 ;Push vector
011670   PHA           ;  to stack
011680   LDA SAVEPUTV
011690   PHA
011700   TYA           ;Restore char
011710   RTS           ;Jump to o.s.
011720 ;
011730 ;
011740 ;Put a message in MSG BOX in
011750 ;Edit Magic screen line.
011760 ;
011770 PUTMSG
011780   LDY #0
011790 PUTBYTE
011800   LDA MSGS,X  ;Xreg indxs msg
011810   STA MSGBOX,Y
011820   INX
011830   INY
011840   CPY #11       ;Msgbox=11 chars
011850   BNE PUTBYTE
011860   RTS
011870 ;
011880 ;
011890 ;Get a keypress.
011900 ;
011910 KGETCHAR
011920   LDA KHANDTAB+5 ;Jump to o.s.
011930   PHA
011940   LDA KHANDTAB+4
011950   PHA
011960   RTS
011970 ;
011980 ;
011990     .IF *>$CFFF
012000     .ERROR "EDIT MAGIC CODE TOO
 LARGE!"
012010     .ENDIF
012020 ;
012030 ;
012040 *** SCREEN DATA  ***
012050 ;
012060   *=  DISPLIST
012070 ;
012080 DLIST
012090   .BYTE $70,$50,66
012100   .WORD EMLINE
012110   .BYTE $10,66
012120 SCRMEM *= *+2
012130   .BYTE 2,2,2,2,2,2,2
012140   .BYTE 2,2,2,2,2,2,2,2,2,2,2
012150   .BYTE 2,2,2,2,2,65
012160   .WORD DLIST
012170 ;
012180 EMLINE
012190 LISTBOX .SBYTE "            ||"
012200 MSGBOX .SBYTE "Edit  Magic |"
012210 FILEBOX .SBYTE "           ||"
012220 BAKUPCNT .SBYTE "0"
012230 ;
012240 ;
012250 MSGS
012260 MSG1 .SBYTE "Edit  Magic"
012270 MSG2 .SBYTE "press a key"
012280 ;
012290 ;
012300 *** MACROS ***
012310 ;
012320   *=  MACLIST
012330 ;
012340 ;Table for macro keys.
012350 ;Entries consist of 3 bytes:
012360 ;1st is internal value of
012370 ;keypress, 2nd & 3rd are
012380 ;address of text to print.
012390 ;Text is printed until zero
012400 ;found.
012410 ;
012420 MACKEYS
012430   .BYTE 52+64+128 ;s-c-BS
012440   .WORD SHFTCTRLDEL
012450   .BYTE 55+64+128 ;s-c-INSERT
012460   .WORD SHFTCTRLINS
012470   .BYTE 40+64+128 ;s-c-R
012480   .WORD SAVTBUFF
012490   .BYTE 255
012500 ;
012510 ;
012520 *** MISC STORAGE ***
012530 ;
```

```
012540  *= MISC
012550 ;
012560 EMFLG .BYTE 1
012570 SAVSDLSTL *= *+2
012580 SAVEGETV *= *+2
012590 SAVEPUTV *= *+2
012600 SAVCOLR1 .BYTE 10
012610 SAVECNT .BYTE 0
012620 SAVEDFLG .BYTE 0
012630 STCKCNT .BYTE 0
012640 LNNOCNT .BYTE 0
012650 ROWCNTR .BYTE 0
012660 SAVCHAR .BYTE 0
012670 SAVREG .BYTE 0
012680 CHARINDX .BYTE 0
012690 CONSOLREL .BYTE 0
012700 ;
012710 LOAD .BYTE "LOAD "
012720 SAVE .BYTE "SAVE "
012730 LIST .BYTE "LIST "
012740 HILNNO .BYTE "32767"
012750 ;
012760 EGETKV .WORD $F2FD
012770 ;
012780 SAVTBUFF .BYTE 0
012790  *= *+127
012800 ;
012810 LNNOSTACK
012820  *= *+[24*5]
012830 ;
012840 ;
012850 ;Table for function keys.
012860 ;EMCONSOL/EMKEYS has 3-byte
012870 ;entries: console button/key
012880 ;plus address to jump to if
012890 ;pressed.
012900 ;
012910 EMCONSOL
012920  .BYTE 6      ;    [start]
012930  .WORD ENTERLBOX
012940  .BYTE 5      ;    [select]
012950  .WORD LISTLINES
012960  .BYTE 3      ;    [option]
012970  .WORD TOGEDTMAG
012980  .BYTE 7
012990 ;
013000 ;
013010 EMKEYS
013020  .BYTE 28+64 ;    shift-ESC
```

```
013030   .WORD ENTERLBOX
013040   .BYTE 28+128 ;   ctrl-ESC
013050   .WORD LISTLINES
013060   .BYTE 28+64+128 ;s-c-ESC
013070   .WORD TOGEDTMAG
013080   .BYTE 54+64 ;    shift-<
013090   .WORD SHIFTCLR
013100   .BYTE 255
013110 ;
013120 ;
013130 HELPMSG
013140   .BYTE "K          Edit Magic  K
EYS:",155
013150   .BYTE "        (s=shift,c=cont
rol)",155,155
013160   .BYTE "[START]  or s-ESC    =
Edit LIST BOX",155
013170   .BYTE "[SELECT] or c-ESC    =
List #'s in BOX",155
013180   .BYTE "[OPTION]` or s-c-ESC =
Toggle EM on/off",155
013190   .BYTE "s-c-R  = Recalls last
line entered",155
013200   .BYTE "s-c-BS = Deletes succe
eding 40 chars",155
013210   .BYTE "s-c-INSERT = Inserts 1
0 lines",155
013220   .BYTE "s-CLEAR = Cursor must
be on lft margin",155
013230   .BYTE "C = Continuous output
(when listing)",155,155
013240   .BYTE "        [HELP] =  Prints
 this",155,155
013250   .BYTE 0
013260 ;
013270 ;
013280 SHFTCTRLDEL .BYTE "◀◀◀◀◀◀◀◀◀◀◀◀
◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀◀",0
013290 ;
013300 SHFTCTRLINS .BYTE "▶▶▶▶▶▶▶▶▶▶",
0
013310 ;
013320 ;
013330      .IF *>DISPLIST
013340      .ERROR "MISC. STORAGE AREA
TOO LARGE!"
013350      .ENDIF
013360 ;
013370   *=  $F25D
```

```
013380   .WORD EGETKEY
013390 ;
013400   *=  EHANDTAB
013410 ;
013420 EOPEN *= *+2
013430 ECLOSE *= *+2
013440 EGET .WORD GETENTRY-1
013450 EPUT .WORD PUTENTRY-1
013460 ESTAT *= *+2
013470 ESPEC *= *+2
013480 EINIT *= *+3
013490 ;
013500 ;
013510 ;The code has now been stored
013520 ;in o.s. RAM. If loaded as an
013530 ;AUTORUN.SYS file,control will
013540 ;eventually revert back to the
013550 ;coldstart routine--which has
013560 ;been overwritten! So, check
013570 ;for this situation, and do
013580 ;coldstart work so we needn't
013590 ;return there.
013600 ;
013610   *=  INITCODE ;Here again
013620 ;
013630 RUN
013640   LDA COLDST   ;Powerup?
013650   BNE FINISH.COLD ;Br if yes
013660   RTS          ;Return to DOS
013670 ;
013680 FINISH.COLD
013690   LDX #255     ;Reset stack
013700   TXS          ;pointer
013710   LDA #1       ;Successful
013720   STA $09      ;disk boot
013730   JSR $E739
013740   LDA #0       ;So no reboot
013750   STA COLDST
013760   LDA TRAMSZ   ;Lft cart?
013770   BEQ GO.DOS   ;Br if no
013780   LDA $BFFD    ;Go to cart?
013790   AND #$04
013800   BEQ GO.DOS   ;Br if no
013810   JMP ($BFFA)  ;Jump to cart
013820 GO.DOS
013830   JMP (DOSVEC) ;Jump to DOS
013840 ;
013850 ;
013860   *=  RUNADR
013870   .WORD RUN
```

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

```
THE A.N.A.L.O.G. #68 DISKETTE CONTAINS 18
MAGAZINE FILES.  THEY ARE LISTED BELOW.

SIDE 1:

FILENAME.EXT  LANG.   LOAD  COMMENTS
------------  -----   ----  --------

NUMED    .SRC  BASIC   ENTER NUMBER EDITOR
NUMED2   .BAS  BASIC   LOAD  NUMBER EDITOR, L2
STELLAR  .OBJ  ML      (#3)  STELLAR ARENA
EDTMAGIC.OBJ   ML      (#3)  EDIT MAGIC
INFERNO  .BAS  BASIC   LOAD  INFERNO
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
GDW6     .LST  BASIC   ENTER GAME DESIGN WRKSHP, P6
GDW7     .LST  BASIC   ENTER GAME DESIGN WRKSHP, P7
GDW8     .LST  BASIC   ENTER GAME DESIGN WRKSHP, P8
MLEDITOR.BAS   BASIC   LOAD  M/L EDITOR
EDITORII.LST   BASIC   ENTER BASIC EDITOR II

SIDE 2:

FILENAME.EXT  LANG.   LOAD  COMMENTS
------------  -----   ----  --------

NUMED    .M65  MAC/65  LOAD  NUMBER EDITOR SOURCE
STELLAR  .M65  MAC/65  LOAD  STELLAR ARENA SOURCE
EDTMAGIC.M65   MAC/65  LOAD  NUMBER EDITOR SOURCE


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.
```

continued from page 28

Collision Checking

So far we've seen all of PMOVE's capabilities as far as moving things around is concerned. But PMOVE also has the ability to check for specific types of collisions between the missiles and the rest of the screen. This means that you can have PMOVE check for a collision between missile 1 and player 2, for example, and when one occurs it will set a flag and stop the missile. If missile 1 collides with player 3, however, nothing will happen. The same idea holds for missile to playfield collisions.

1712-1715: Are used to specify the missile to playfield collisions that you want PMOVE to watch out for.

Now the question is, how? A while back, before any of this PMOVE stuff, we talked about the collision registers, and how four different collisions could be stored in one location. Remember? If not, go back and review for a second. Locations 1712-1715 are set up in exactly the same manner as the collision registers. So, for example, if you wanted PMOVE to check for collisions between missile 1 and playfield 2 and 3, you would POKE 1713,6. As I said, go back and reread the section on the collision registers if you don't understand why.

Incidentally, when PMOVE encounters one of the collisions you told it to look for, it will turn off the missile and set the corresponding location here to 255 so you know what kind of collision occurred. If you were looking for more than one collision, you can then check the collision register itself to find out the exact culprit.

1716-1719: These are the locations for missile to player collisions and act in the same way as 1712-1715 above.

1720-1724: Finally, the last of the locations! Actually, these four don't really do that much. When missile n has a collision that you told PMOVE to watch out for, location 1720+n gets set to 255 (you should set it to something else when you first turn on the missile). Why even bother with this when the corresponding collision location gets set to 255 as well (see locations 1712-1719 above)? Well, it just makes it a little easier (and faster) to check for a collision, since you only have to check one place instead of two.

Whew! Now you can see what I meant when I said that there was a little work involved in getting PMOVE going. Don't give up hope though, it isn't really as difficult as it looks. To show you what I mean, let's start by putting a player on the screen and attaching it to joystick 0:

```
160 FOR BYTE=126 TO 130
170 READ DAT
180 POKE PMBASE+1024+BYT
E,DAT
190 NEXT BYTE
200 POKE 54279,PB:POKE 5
59,62:POKE 53277,3
201 POKE 1664,0:POKE 166
5,255:POKE 1666,255:POKE
 1667,255
202 POKE 1668,49:POKE 16
72,200:POKE 1676,34:POKE
 1680,221
203 POKE 1684,128:POKE 1
688,128
205 X=USR(PMBASE,PB,PB)
210 POKE 53248,128
220 POKE 704,72
1000 DATA 153,189,255,18
9,153
```

These lines should be added to the combination of the previous two examples. Here's an explanation of what these lines are doing:

160-190: Set up the player.

200: Turn on PMG.

201: Connect player 0 to joystick 0 and turn off the other players.

202: Set up the left, right, top and bottom boundaries, respectively.

203: Tell PMOVE the initial horizontal and vertical positions.

210: Position the player horizontally.

220: Color the player.

That's fairly simple, isn't it? Now let's give the program the ability to fire a missile. We'll complicate things a little by firing the missile in the direction the joystick is moved when the fire button is pressed. This will slow things down a little, but it is a handy thing to know how to do.

```
230 IF PEEK(1700)=1 OR S
TRIG(0)=1 THEN 230
235 DIR=STICK(0):IF DIR=
15 THEN 230
240 X=USR(ADR(MISCLR$),P
MBASE+768,255,252):PPOS=
PEEK(1688)
250 POKE PMBASE+768+PPOS
-1,3:POKE PMBASE+768+PPO
S,3:POKE PMBASE+768+PPOS
+1,3
260 POKE 1696,PPOS:POKE
1692,PEEK(1684)+2:POKE 5
3252,PEEK(1692)
270 POKE 1704,0
280 IF DIR<12 THEN POKE
1704,128
290 IF DIR<8 THEN POKE 1
704,1
300 POKE 1708,0
310 IF DIR/2=INT(DIR/2)
THEN POKE 1708,128:GOTO
330
320 IF DIR<>11 AND DIR<>
7 THEN POKE 1708,1
330 POKE 1700,1
340 GOTO 230
```

Add these lines to the program above. They won't work by themselves. List the resulting program to disk.

We need a special routine to clear a missile, since we only want to clear two bits in each byte. Here's the program to generate Line 105 for you to do this. Type New, then run this, type New again, move cursor to Line 105 and press Return ENTER above saved program.

```
100 GRAPHICS 0
110 PRINT "105 DIM MISCL
R$(26):MISCLR$=";CHR$(34
);
120 FOR LOOP=1 TO 26
130 READ DAT
140 PRINT CHR$(27);CHR$(
DAT);
150 NEXT LOOP
170 PRINT CHR$(34)
1000 DATA 104,104,133,20
7,104,133,206,104,104,16
8,104,104,133,208,177,20
6,37,208,145,206,136,192
,255,208,245,96
```

Now here's how MISCLR$ is used:

```
X=USR(ADR(MISCLR$),MISBA
SE,RES,MASK)
```

MISBASE is the address of the beginning of the missile area. It's equal to PMBASE+768 for single-line resolution, and PMBASE+384 for double-line resolution. RES tells what kind of resolution is being used and is equal to 255 for single-line and 127 for double. Finally, MASK is used to specify which missile you want to clear and has one of the following values:

MISSILE 0: 252
MISSILE 1: 243
MISSILE 2: 207
MISSILE 3: 63

As a final example, we'll create a program that will move player 0 around automatically, without having it attached to a joystick. Just add these lines to our spaceship program:

```
201 POKE 1664,43:POKE 16
65,255:POKE 1666,255:POK
E 1667,255:POKE 675,10
221 IF PEEK(1688)=PEEK(1
680) OR PEEK(1688)=PEEK(
1676) THEN POKE 675,5+5*
(PEEK(675)=5)
230 IF PEEK(1700)=1 OR S
TRIG(0)=1 THEN 221
235 DIR=STICK(0):IF DIR=
15 THEN 221
340 GOTO 221
```

Once again, the explanation:

201: By POKEing 1664 with a 43, we connect player 0 to a nonexistent joystick that we can give our own direction value to. See the previous explanation to locations 1664-1667 for more details. We then set location 675, our nonexistent joystick, to 10, which corresponds to left and up.

221: We now check to see if the player has reached either its upper or lower limit and, if it has, change its direction.

230: The only change here is the line number after the THEN statement.

235: Again, only the line number after the THEN statement has been changed.

340: One more line-number change.

Congratulations! You now have more control over PMG than most other BASIC programmers. Hopefully PMOVE will help you create the programs you've always wanted to create but haven't been able to because BASIC has been too slow for PMG.

STAR ARENA

continued from page 37

```
2840 DATA 173,10,210,41,2,170,153,43,6
,173,10,210,41,7,224,2,4709
2850 DATA 240,9,24,105,26,153,25,6,76,
143,76,24,105,166,153,25,4493
2860 DATA 6,149,197,161,197,201,0,240,
25,169,0,153,25,6,198,195,8760
2870 DATA 165,195,201,0,208,202,173,10
,210,41,7,24,105,1,153,69,5014
2880 DATA 6,96,230,184,169,1,129,197,9
6,164,183,32,91,76,136,208,350
2890 DATA 250,160,4,185,61,6,201,0,240
,6,136,208,246,76,250,76,1400
2900 DATA 185,17,6,201,0,240,243,24,12
1,154,70,162,0,192,3,144,7654
2910 DATA 2,162,2,149,197,161,197,201,
0,208,223,181,197,153,61,6,1195
2920 DATA 138,153,240,76,235,77,65,6,1
85,158,70,129,197,76,200,76,9932
2930 DATA 165,216,201,0,240,3,76,143,7
7,162,5,160,3,196,176,176,9551
2940 DATA 7,200,200,200,202,76,7,77,13
4,216,160,4,190,65,6,185,8482
2950 DATA 61,6,149,197,161,197,217,162
,70,144,14,169,0,153,61,6,6465
2960 DATA 153,65,6,136,208,230,76,143,
77,217,166,70,176,3,76,37,7492
2970 DATA 77,217,170,70,240,20,192,3,1
76,8,24,105,1,129,197,76,6137
2980 DATA 45,77,56,233,1,129,197,76,45
,77,169,0,129,197,185,61,8265
2990 DATA 6,24,121,154,70,190,65,6,201
,200,144,3,32,30,70,149,5715
3000 DATA 197,161,197,201,0,240,3,76,3
7,77,138,153,65,6,181,197,8236
3010 DATA 153,61,6,185,166,70,192,3,17
6,5,129,197,76,45,77,24,5235
3020 DATA 105,3,76,132,77,169,0,133,19
2,173,4,208,201,0,240,6,8657
3030 DATA 141,0,6,32,162,79,173,12,208
,201,0,240,6,141,0,6,4818
3040 DATA 32,162,79,173,5,208,201,0,24
0,6,141,1,6,32,233,79,6073
3050 DATA 173,13,208,201,0,240,6,141,1
,6,32,233,79,160,0,185,6731
3060 DATA 0,208,201,0,240,74,74,72,144
,18,32,76,79,169,0,129,5014
3070 DATA 197,162,10,134,222,32,107,80
,169,25,133,191,104,74,144,38,7530
3080 DATA 32,76,236,77,231,78,79,161,1
97,201,75,176,29,56,233,71,256
3090 DATA 170,189,151,70,72,169,0,157,
18,6,104,166,193,129,197,162,221
3100 DATA 50,134,222,32,107,80,169,25,
133,191,230,192,169,0,153,10,9203
3110 DATA 6,153,6,6,200,192,1,240,170,
160,0,185,8,208,74,72,8171
3120 DATA 144,28,192,1,208,24,165,185,
74,176,19,140,0,6,32,162,5190
3130 DATA 79,160,1,162,250,134,222,32,
107,80,169,25,133,191,104,74,9113
3140 DATA 72,144,27,192,0,208,23,165,1
85,74,74,176,17,140,1,6,4701
3150 DATA 32,233,79,162,250,134,222,32
,107,80,169,25,133,191,104,74,9466
3160 DATA 72,144,41,169,0,141,16,6,141
,12,6,141,14,6,133,186,3709
3170 DATA 153,6,6,153,10,6,162,100,134
,222,32,107,80,169,25,133,6956
3180 DATA 191,162,0,138,157,0,62,202,2
08,250,230,192,104,74,144,41,1492
3190 DATA 169,0,141,17,6,141,13,6,141,
15,6,133,187,141,7,6,2552
3200 DATA 141,11,6,162,100,134,222,32,
107,80,169,25,133,191,162,0,7898
3210 DATA 138,157,0,63,202,208,250,230
,192,200,192,1,208,3,76,33,28
3220 DATA 78,165,192,201,0,240,0,141,3
0,208,165,20,201,16,176,3,7506
3230 DATA 76,15,79,169,0,133,20,173,23
1,58,201,16,240,6,206,231,1623
3240 DATA 58,76,232,78,227,79,15,79,17
3,230,58,201,16,240,11,206,740
3250 DATA 230,58,169,25,141,231,58,76,
15,79,165,176,201,15,176,2,7655
3260 DATA 230,176,32,246,69,169,25,141
,230,58,141,231,58,173,238,58,2231
3270 DATA 201,16,240,3,76,35,79,173,22
3,58,201,16,208,3,76,214,9062
3280 DATA 81,162,2,189,5,6,201,47,144,
21,201,204,176,17,189,7,7929
3290 DATA 6,201,67,144,10,201,223,176,
6,202,208,231,76,28,73,169,1013
3300 DATA 0,157,9,6,157,5,6,76,59,79,1
69,0,153,10,6,230,4209
```

```
3310 DATA 192,185,6,6,56,233,47,162,0,
201,8,144,7,56,233,8,5541
3320 DATA 232,76,91,79,134,194,185,8,6
,56,233,64,162,0,201,8,6680
3330 DATA 144,7,56,233,8,232,76,112,79
,134,195,224,10,176,21,169,9945
3340 DATA 0,133,193,165,194,224,0,240,
6,24,105,20,202,208,250,166,2712
3350 DATA 193,149,197,96,169,2,133,193
,138,56,233,10,170,76,133,79,9166
3360 DATA 230,192,165,185,74,176,58,16
9,0,133,224,133,191,174,223,58,2975
3370 DATA 224,16,240,44,202,142,223,58
,165,185,9,1,133,185,172,2,8786
3380 DATA 6,162,8,169,0,153,0,60,136,2
02,208,249,141,4,6,174,9091
3390 DATA 223,58,224,16,240,10,169,100
,141,0,6,169,130,141,2,6,4861
3400 DATA 96,169,228,79,223,80,100,141
,0,6,96,230,192,165,185,74,898
3410 DATA 74,176,58,169,0,133,224,133,
191,174,238,58,224,16,240,44,2329
3420 DATA 202,142,238,58,165,185,9,2,1
33,185,172,3,6,162,8,169,7011
3430 DATA 0,153,0,61,136,202,208,249,1
41,5,6,174,238,58,224,16,315
3440 DATA 240,10,169,156,141,1,6,169,1
30,141,3,6,96,169,156,141,7831
3450 DATA 1,6,96,165,185,74,144,20,173
,223,58,201,17,144,13,173,8985
3460 DATA 132,2,201,0,208,6,165,185,41
,2,133,185,165,177,201,2,9652
3470 DATA 208,26,165,185,74,74,144,20,
173,238,58,201,17,144,13,173,9134
3480 DATA 133,2,201,0,208,6,165,185,41
,1,133,185,96,72,152,72,7681
3490 DATA 201,1,240,45,160,6,132,221,1
65,176,72,185,202,58,201,25,1231
3500 DATA 176,36,105,1,153,202,58,192,
12,240,70,192,2,240,40,164,178
3510 DATA 221,202,208,231,166,222,198,
176,208,225,104,133,176,104,168,104,54
29
3520 DATA 96,160,16,76,116,80,169,16,1
53,202,58,192,11,240,3,136,8722
3530 DATA 208,201,164,221,76,121,80,16
0,0,185,223,58,201,25,240,14,248
3540 DATA 24,105,1,153,223,58,169,252,
133,227,169,0,133,230,76,141,2722
3550 DATA 80,160,15,76,183,80,165,218,
201,0,240,1,96,165,20,201,128
3560 DATA 4,240,224,80,219,81,9,201,8,
240,5,201,12,240,1,96,8294
3570 DATA 169,1,133,218,165,185,74,72,
176,9,162,1,160,0,134,222,9151
3580 DATA 32,107,80,104,74,176,9,162,1
,160,1,134,222,32,107,80,6714
3590 DATA 96,162,4,189,17,6,201,0,240,
6,202,208,246,76,35,81,9327
3600 DATA 189,129,70,157,17,6,76,20,81
,165,176,201,4,144,82,173,8491
3610 DATA 22,6,201,0,208,12,173,24,6,2
01,2,240,5,169,100,141,7915
3620 DATA 22,6,165,176,201,7,144,57,17
3,23,6,201,0,208,10,169,7395
3630 DATA 99,141,23,6,169,2,141,25,6,1
62,0,189,12,6,201,0,3345
3640 DATA 208,20,169,72,157,14,6,169,1
28,157,12,6,169,1,149,186,7294
3650 DATA 141,2,208,141,3,208,232,224,
2,176,6,165,176,201,10,176,1554
3660 DATA 218,96,169,255,141,252,2,165
,190,72,169,251,133,190,165,176,6129
3670 DATA 72,169,1,133,176,169,0,174,1
0,6,172,11,6,141,10,6,2373
3680 DATA 141,11,6,173,31,208,201,6,20
8,7,104,133,176,104,76,155,9457
3690 DATA 72,201,7,176,7,104,133,176,1
04,76,67,71,173,121,2,201,8361
3700 DATA 15,240,3,76,199,81,173,120,2
,201,15,240,214,142,10,6,8249
3710 DATA 140,11,6,104,133,176,104,133
,190,76,76,73,160,0,185,203,104
3720 DATA 58,217,220,81,45,82,213,57,1
44,7,208,25,200,192,6,144,9496
3730 DATA 241,160,0,185,213,58,217,213
,57,144,41,208,25,200,192,6,668
3740 DATA 144,241,76,26,82,160,6,185,2
02,58,153,212,57,153,126,65,28
3750 DATA 136,208,244,76,231,81,160,6,
185,212,58,153,212,57,153,126,2271
3760 DATA 65,136,208,244,160,6,185,202
,58,153,156,65,185,212,58,153,2683
3770 DATA 166,65,136,208,241,76,67,71,
226,2,227,2,179,70,0,0,5886
```

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

MASTER MEMORY MAP PART VI

continued from page 16

used for this purpose (see locations 832 to 959), along with two separate cursors.

You should look at SWPFLG (123) for additional information about locations 656 to 667.

TXTROW
656 0290

The row that the text-window cursor is currently in. Because there are only four rows in the text window, TXTROW ranges from zero to three.

TXTROW is the text window equivalent of ROWCRS at location 84.

TXTCOL
657,658 0291,0292

The column that the text-window cursor is currently in. There are 40 columns in the text window, so TXTCOL can range from 0 to 39. "Ah, ah," you say. That means location 658 never gets used (since it's only needed when the column number is greater than 255). This is true under normal circumstances, but if you change the text window to be something other than GRAPHICS 0, you may need it.

TINDEX
659 0293

While we're on the subject of changing the text-window graphics mode, TINDEX tells the OS what graphics mode the text window is (also see DINDEX at location 87). If you decide you'd like a different text window, you'll have to change the display list as TINDEX. Use the program for location SDLSTL (560,561) to look at the display list and see where the text window is. I won't go over it here because for most uses, you'll probably just want to mix graphics modes. If that s not the case, however, it's easy to figure out how to make the necessary changes. Just look for the CHR 2 commands at the end of the display list. See location 559 for more information.

TXTMSC
660,661 0294,0295

The address of the upper left-hand corner of the text-window screen memory. See SAVMSC (88,89) for the address of regular screen memory.

TXTOLD
662-667 0296-029B

Check out locations 90 through 95, okay? These six locations are the text window equivalent, so I won't bother explaining them again.

TMPX1
668 029C

This, along with the next three locations, is used for temporary storage. They are used in one or more of the computer's routines as a place to store information during the routine. Once the routine is over, the values in them are no longer meaningful.

TMPX1, in case it wasn't clear, is a temporary location.

HOLD3
669 029D

A temporary location (the location isn't temporary, its usage is).

SUBTMP
670 029E

Another temporary location.

HOLD2
671 029F

And yet another temporary location.

DMASK
672 02A0

Way, way back at location SHFAMT (111), we had a little discussion about masking and making changes to individual pixels in the graphics modes. Remember? Well, go back and refresh your memory anyway.

DMASK holds the mask for the pixel that we want to make changes to. Somewhere way up near the end of the OS ROM, there is a list of all the possible masks. The display handler decides which one is needed and loads it into DMASK. Figure 1 has the different values DMASK can have, as well as the graphics modes they are used with.

By way of explanation, the "1s" are used to look at individual bits and the "0s" to ignore them.

Now why, you may ask, do we need more than one mask for most graphics modes? Graphics modes need anywhere from one to eight bits to represent a character or a pixel. Suppose a particular mode, such as mode 9, needs four bits per pixel. That means that each byte holds two different pixels, right (since a byte is eight bits)? So we need two masks to be able to mask out either pixel. This may be a little confusing to you, but don't worry. Unless you're programming in machine language, it's something that is nice to know, but that you'll never need.

11111111	for modes zero, one, and two.
11110000	for modes nine, ten, and eleven.
00001111	
11000000	
00110000	for modes three, five, and seven.
00001100	
00000011	
10000000	
01000000	
00100000	
00010000	for modes four, six, and eight.
00001000	
00000100	
00000010	
00000001	

Figure 1: DMASK bit chart

TMPLBT
673 02A1

More temporary storage space.

ESCFLG
674 02A2

When the Esc key is pressed, ESCFLG is set to 128, and the next key pressed gets an Esc flag attached to it (for example, pressing Esc twice would cause the second Esc to print a special character on the screen). After the next key has been pressed ESCFLG is reset to zero.

ESCFLG is initialized to zero.

TABMAP
675-689 02A3-02B1

TABMAP tells the OS what columns to move the cursor to when the Tab key is pressed.

When the Tab key is pressed, the cursor is moved to the next column, after the one the cursor is on, that has a tabstop. What's a tabstop? It's nothing more than a flag saying, "Hey, Tab, stop here, okay?" TABMAP is where these tabstops or flags are kept. Since you can set a tabstop on any one of the 120 columns in a logical line, TABMAP is 15 bytes long. What? How do you get 120 from 15? Well, since the tabstop for each column is either turned on or turned off, we only need one bit for each column. Fifteen bytes times eight bits per byte equals 120. Oh!

How do you set the tabstops? From BASIC, all you have to do is use the TAB-SET (Shift-Tab) and TAB-CLR (Control-Tab) keys. From within a program, however, you must change TABMAP yourself. To do this, start with 120 zeroes on a piece of paper. These represent the 120 columns, numbered from 0 on the far left to 119 on the far right. Now change the zeroes to ones in the columns where you want your tabstops. So far so good. The next step is to break the 120 digits into groups of eight and convert them to decimal. See the section on bits and bytes for help in doing this. The last step, now that you have 15 decimal numbers, is to POKE these numbers into TABMAP. You now have your own customized Tab settings.

What restores TABMAP to its original values? Pressing System Reset or using a GRAPHICS command (OPENing S: or E: as well). What are its original values? A value of one in every byte. That translates to tabstops at 7, 15, 23, 31, . . . 119.

A few final words. TABMAP works in graphics mode 0 and in text windows only. Also, the left edge of the screen will always to be a tabstop, whether you set it to be or not.

LOGMAP
690-693 2B2-2B5

When you're writing or editing your BASIC program, the screen editor needs to know where each logical name begins. Why? Just so that it can make sense out of what's on the screen. Remember, a program listing on the screen may make sense to you, but to the computer it's just a bunch of bytes in memory. With the help of LOGMAP, at least it knows on what row of the screen a program line begins.

LOGMAP works in much the same way as the preceding TABMAP. There are 24 rows in a GRAPHICS 0 screen, so there are 24 bits in LOGMAP. Actually, there are 32 bits (four bytes), just the last byte doesn't get used. The first byte handles rows 0 through 7, the second handles 8 through 15, and the third handles 16 through 23. If a logic line begins on a certain row, then the corresponding bit is set to one. If the row is part of a previous logical line, then the bit is set to zero.

All the bits in LOGMAP are set to one when the computer is first turned on, System Reset is pressed, a GRAPHICS command is used, the text screen is OPENed, or the text screen is cleared. This is because all the lines are blank, and therefore considered to be the start of a logical line.

TABMAP is updated when you first enter a logical line (with the Return key), edit a line, delete a line or insert a line. Under all these circumstances, the position of the logical lines on the screen will be altered, thus the need for updating.

INVFLG
694 02B6

INVFLG works similarly to ESCFLG

MASTER MEMORY MAP PART VI

except it keeps track of the inverse video key (the Atari logo key) instead of the Esc key. It is initialized to zero, which means that all the characters you type in will be normal. But when you press the inverse video key, INVFLG gets set to 128, and characters that you type now will appear in inverse video (black on white instead of white on black). Pressing the inverse video key again will restore INVFLG to zero and get things back to normal.

You should be aware that changing INVFLG will only affect characters that are typed in after you change it. That means that you can't use it like this:

```
POKE 694,128:PRINT "INVE
RSE?"
```

Machine-language programmers might be interested to know that INVFLG is always XORed with the character value, regardless of INVFLG's value (this should tell you that the value for an inverse video character is just that for a regular character with bit seven set, i.e., the regular value plus 128). This means that you can have fun with the keyboard by POKEing INVFLG with something other than zero or 128. Try it, it's fun!

FILFLG
695 02B7

If FILFLG is not equal to zero, then we're in the middle of a FILL.

TMPROW
696 02B8

A temporary storage place for the value in ROWCRS (84).

TMPCOL
697,698 02B9, 02BA

More temporary storage, this one for the value in COLCRS (85,86)

SCRFLG
699 02BB

This one is somewhat complicated. First of all, it keeps track of how many physical lines (as compared to logical lines) have been scrolled off the top of the screen. If you keep pressing Return, it will eventually count up to 255 and then wrap back around to zero. No problem so far. According to the OS listing, however, it is also used during the character insertion process (when you press Shift-Insert). Apparently, if you insert a character, SCRFLG gets set to zero. If the insertion caused the screen to scroll up, then the number of lines it scrolled (which depends on how long the logical line at the top of the screen was, so it could be from one to three) is stored in SCRFLG. The value in SCRFLG is then used to reposition the cursor, leaving SCRFLG with a final value of 255.

HOLD4
700 02BC

HOLD4 is used to temporarily hold the value of ATACHR (763) during the FILL routine.

HOLD5
701 02BD

An unidentified storage location.

SHFLOK
702 02BE

When SHFLOK is set to 0, all text typed in will be in lowercase. Set it to 64, and all text will be in uppercase. Finally, 128 will give you all control and graphics characters.

The following key combinations affect SHFLOK:

Caps/Lowr sets it to zero.
Shift-Caps/Lowr sets it to 64.
Control-Caps/Lowr sets it to 128.

In addition, POKE in 192 and only numbers and punctuation will be recognized if pressed. Finally 255 in this location will not allow any letters at all to be recognized.

Remember that these two POKEs work on input from the keyboard only. You can still write letters to the screen or printer. This means their practical use is to prevent inputs you don't want.

Note that SHFLOK does not indicate whether or not the Shift or Control keys are pressed.

SHFLOK is initialized to 64.

BOTSCR
730 02BF

BOTSCR tells how many lines of text are available for use by the screen editor. What does this mean? Well, it can use all 24 lines in graphics mode 0, so BOTSCR would have a value of 24. In a mode with a text window, there are four lines in the text window that it can use, so BOTSCR would have a value of four. In all other modes there are none, so BOTSCR has a value of zero. "What about graphics modes 1 and 2?" you say. In these modes the screen editor takes care of the text window, while the display handler takes care of the rest (at least in terms of PRINTing text, which is what we're really talking about here).

Try the following program:

```
100 GRAPHICS 0
110 POKE 703,4
120 FOR ROW=0 TO 19
130 POSITION 2,ROW
140 PRINT #6;"We have to
 print #6 up here"
150 NEXT ROW
160 PRINT "But now we ha
ve a text window here!"
170 PRINT
180 GOTO 160
```

Color

The next nine locations, 704 through 712, are called "color registers." This is just a fancy way of saying that they tell ANTIC what colors to put on the screen. How do you convert a color into a number that you can store here? The Atari has a total of 16 colors that you can choose from, and each is assigned a number. The exact colors vary slightly from television set to television set, so it's difficult to describe them exactly. Bear that in mind when you consult the chart in Figure 2.

COLOR	VALUE	COLOR	VALUE
Black	0	Blue	8
Rust	1	Deep blue	9
Reddish orange	2	Dull blue	10
Dark orange	3	Olive green	11
Red	4	Green	12
Purplish blue	5	Dark green	13
Cobalt blue	6	Orangey green	14
Ultramarine	7	Orange	15

Figure 2. Color value chart

Okay, now somewhere in the back of your mind you're probably thinking, *Wait a minute, aren't there supposed to be 256 possible colors on the Atari?* Yes and no. There are only 16 colors, but there are also 16 shades of each color, resulting in a total of 256 (26 times 16) possible "colors." That's not even true either. Even though you can specify a brightness value from 0 to 15, 0 and 1 will be the same brightness, as will 2 and 3 and so forth. That gives us a true total of 128: 128 combinations of color and brightness, however, will be more than you need, or can use at one time.

So now we have a color value and a brightness value. Since each color register is only one byte, we're obviously going to have to somehow combine these two values together. If you're familiar with hexadecimal, you will probably know how already. Recall that in hexadecimal, each byte has two digits, each of which can have a value from 0 to F (15). All we do to combine our color and brightness is have the first digit be the color, and the second the brightness. If you're using decimal, you want to multiply the color value by 16 and add the brightness. That's how you figure out the value to POKE into the appropriate color register (you can also use the SETCOLOR command for the playfield registers).

PCOLR0
704 02C0

This is the color register for player 0 and missile 0. It is also used to hold the background color in GTIA mode 10.

The SETCOLOR command will not work on this or any of the next three locations.

PCOLR0 is a shadow register for COLPM0 at location 53266.

PCOLR1
705 02C1

The color register for player 1 and missile 1. It is a shadow register for COLPM1 at location 53267.

PCOLR2
706 02C2

The color register for player 2 and missile 2. It is a shadow register for COLPM2 at location 53268.

PCOLR3
707 02C3

The last player/missile color register, this time for player 3 and missile 3. It is a shadow register for COLPM3 at location 53269.

COLOR0
708 02C4

Lots of information for this guy. This is the color of playfield 0. It is also called color register 0, is a shadow register for COLPF0 at location 53270, specifies the color of *uppercase* letters in graphics modes 1 and 2, and can be set by the BASIC SETCOLOR command (as can the next four locations). Whew!

COLOR1
709 02C5

This holds the color value for playfield 1, is called color register 1, is a shadow register for COLPF1 at location 53271, and specifies the color of *lowercase* letters in graphics modes 1 and 2.

COLOR1 is also used to specify the brightness of the characters in graphics mode 0, and of the pixels in the graphics mode 8. As you know, you can only draw with one color in graphics mode 8, right? Well, not quite. Through a process called "artifacting," you can get up to four.

Briefly, because the pixels are so small in graphics mode 8, a pixel in an odd-numbered column will have a different color than one in an even-numbered column. Don't ask why, just try the following program:

```
100 GRAPHICS 8
110 COLOR 1
```

MASTER MEMORY MAP PART VI

```
120 FOR COL=10 TO 20 STE
P 2
130 PLOT COL,10
140 DRAWTO COL,20
150 NEXT COL
160 FOR COL=31 TO 41 STE
P 2
170 PLOT COL,10
180 DRAWTO COL,20
190 NEXT COL
```

Voila! Two new colors. But how do we get the regular white, and where does the fourth color come from? You know, you ask a lot of questions. If we plot an even-numbered column and then the following odd-numbered column, we get white. If, on the other hand we plot an odd-numbered column and then the following even-numbered one, we get the fourth color. Make sure you understand the difference between the two. Add the following lines to the preceding program:

```
200 FOR COL=50 TO 60 STE
P 4
210 PLOT COL,10:DRAWTO C
OL,20
220 PLOT COL+1,10:DRAWTO
 COL+1,20
230 NEXT COL
240 FOR COL=71 TO 81 STE
P 4
250 PLOT COL,10:DRAWTO C
OL,20
260 PLOT COL+1,10:DRAWTO
 COL+1,20
270 NEXT COL
```

Doing things this way kind of restricts you in the way you plot and draw, but it *does* give you more colors. You should also note that the CTIA and GTIA chips switch the odd and even colors on the screen. This usually makes red on one computer look like green on another. Also, the colors you *do* get will depend on the values in COLOR1 and COLOR2 (following).

COLOR2
710 02C6

This holds the color value for playfield 2, is called color register 2, is a shadow register for COLPF2 at location 53272, and specifies the color of *inverse uppercase* letters in graphics modes 1 and 2.

In graphics modes 0 and 8, COLOR2 specifies the background color.

COLOR3
711 02C7

Okay, you should be getting the hang of this by now. This is the same as COLOR2, but with threes instead of twos. It's also the color of *inverse lowercase* letters in graphics modes 1 and 2.

COLOR4
712 02C8

Same as the preceding but for the background color. It is a shadow register for COLBK at location 53274. Don't forget that in GTIA modes, PCOLR0 (704) is the background color, while COLOR4 is just a regular color register.

Noname
713-735 02C9-02DF

These 23 are currently unused.

The following four bytes, from 736 to 739, are used by DOS. That means that they are unused if you are not using DOS.

RUNAD
736,737 02E0,02E1

When you load a binary load file from DOS, sometimes it will run automatically and sometimes it won't. What makes the difference? If the binary load file stores an address in RUNAD, then DOS will JSR to that address after the file has been loaded. Otherwise, the DOS menu will stay on the screen. See your DOS manual for more information under the sections on binary loading and saving.

INITAD
738,739 02E2,02E3

Whoops! I lied slightly in the last location. If a binary load file alters INITAD, then DOS immediately JRSs again to the address in INITAD before continuing to load the file. You can use this to do stuff like put a message or picture on the screen

while the rest of the file is loading. Make sure that the routine whose address you're putting in INITAD ends with an RTS. Also, if you want DOS to return to the menu after executing the RUNAD routine, make sure it ends with an RTS instruction.

So where was the lie? If you don't end the INITAD routine with an RTS instruction, the RUNAD routine will never be executed (and you may run into problems with future disk I/O).

RAMSIZ
740 02E4

RAMSIZ has a similar function to RAMTOP (106), so go back and read up on RAMTOP. The main difference is that RAMSIZ doesn't cause the screen memory to move when you change it and do a graphics call. Experiment to see how it works.

MEMTOP
741,742 02E5,02E6

MEMTOP holds the address of the last free memory location. This does not mean the top of RAM. Why not? You're forgetting that the screen memory and display list are put at the end of RAM. MEMTOP is the last location that is unused, and is therefore the location right below the display list. Originally, however, before any graphics mode is set up, it does hold the same address as RAMSIZ.

Anything that results in the display handler changing the screen memory and display list also results in MEMTOP getting changed. That means System Reset, the GRAPHICS command and OPENing the screen.

For more information on MEMTOP's use (yes, I'm going to send you somewhere else again), see APPMHI at locations 14 and 15.

MEMTOP is called HIMEM by BASIC, since BASIC has its own MEMTOP at locations 144 and 145.

MEMLO
743,744 02E7,02E8

Since we have a pointer to the top of free memory, it only makes sense to have one to the bottom of free memory. MEMLO holds the address of the first byte in RAM that is available for your use. Notice that BASIC uses a different pointer for the first free byte, called LOMEM (128,129). Although some sources imply otherwise, MEMLO and LOMEM seem to always contain the same value, which is not touched by the OS after the power-up routine is done.

The first free location in memory is usually at 1792. If you're using DOS, however, DOS needs some extra space for something called the "FMS buffers" (see SABYTE [1801] and DRVBYT [1802]). This means that MEMLO will be greater when DOS is present by 128 for each buffer.

Let's talk about reserving memory for your own private use. We last discussed this at RAMTOP (106), where we saw how to reserve memory above screen memory. But, alas, this technique wasted up to 800 bytes because of a problem with scrolling the screen. So now we come to the alternative of reserving memory at the other end of RAM, below everything else. How do we do it? There are two possibilities. First of all, you could write an AUTORUN.SYS file that loads MEMLO with the values you want. *De Re Atari* has an excellent example of how to do this, but it's obviously a technique that requires a knowledge of machine language. What if you're working in BASIC? Well, there's a problem. Remember that BASIC also keeps a pointer to the bottom of free memory. It's called LOMEM, and I have mentioned it. If we change MEMLO, we also have to change LOMEM. We can do this by POKEing both MEMLO and LOMEM, but that confuses BASIC because it loses some important information that it had already stored in the memory area you just told it not to look at. That's a problem. What happens if you POKE MEMLO and then type NEW (NEW transfers the value of MEMLO into LOMEM and resets all the program pointers)? Nothing bad; in fact it does exactly what we wanted. But we still have a problem: This method only works when you make the changes yourself; it won't work from inside a program. As it turns out, and it makes sense if you think about it, there is nothing you can do from within a BASIC program to reserve memory using MEMLO (without destroying the program). This means that the MEMLO method of reserving memory is only useful if you're programming in machine language (or if you first boot up an AUTORUN.SYS file as described). Sorry folks.

System Reset will restore MEMLO to its original value. The program in *De Re Atari*, as mentioned, uses the System Reset vector to make sure that MEMLO does not get reset.

Only NEW (or turning off the computer) will restore LOMEM.

Noname
745 02E9

Currently unused. This is, however, subject to change in future versions of the OS.

DVSTAT
746-749 02EA-02ED

This one is for experts only, so don't expect it to sound pretty. When you send a GET STATUS command (83) to a device, these bytes are set according to the type of device and its status. They seem to be set only by the printer handler (*not* the disk file manager) and the RS232 handler.

Location 746 gives the command status. Because it is interpreted differently for each device, you should consult either the OS manual or the 850 Interface manual for details (this isn't a cop-out on my part; the information in this byte is useful only to extremely competent machine-language programmers).

MASTER MEMORY MAP PART VI

If the GET STATUS is to a printer, location 747 contains the AUX2 byte of the previous operation. If it is to a disk drive, location 747 holds the value of the status byte of the disk controller chip (if you really need to know more details, find some documentation on the INS1771-1 Floppy Disk Controller chip). Finally, if it is to the 850 Interface, location 747 could indicate one of two things. If concurrent mode I/O is not active, then it will hold information regarding the monitored readiness lines (DSR, CTS and CRX) and the data receive line (RCV) of the specified port. *Please* see your 850 Interface manual for more details.

If concurrent mode I/O were active, location 747, in conjunction with location 748, will hold the number of characters currently in the input buffer.

For the printer and disk drive, a GET STATUS command will return the maximum time-out value for the device. This value is provided by the device controller and is initialized to 31. A value of 64 here represents one second.

Location 749 is only used for the 850 Interface and only if concurrent mode I/O is active at the time of the GET STATUS. In that case, it holds the number of characters currently in the output buffer.

If you got this far and you're confused, don't worry. By the time you have a need to use DVSTAT, it should be easier to understand. I've been programming the Atari for five years and have only recently found a need for it.

CBAUDL, CBAUDH
750,751 02EE,02EF

The speed at which programs load in from cassette is called the "baud rate," and this is what is stored in CBAUDL/H. It's initialized to 1484 by the OS, which represents 600 baud. Unfortunately, sometimes the data on the cassette tape is stored a little slower or faster than 600 baud. This may be due to the speed of the cassette motor, the tape being stretched slightly, or other such minor details. In any case, at the beginning of each cassette record (remember that a record is just a bunch of bytes) are two bytes that have alternating zeroes and ones (01010101; 85). These bytes are used to set the baud rate exactly, so speed variations can be compensated for.

AUDF3 (53764) and AUDF4 (53766) are used to store the baud rate and do the actual timing.

CRSINH
752 02F0

This one should come as a reward to you for trudging through the sludge of the last few locations. CRSINH is used to make the cursor invisible (and visible again). This comes in handy when you've got a message or something on the screen, and you don't want whoever's reading it to see the cursor. All you have to do is POKE CRSINH with something other than a zero. To make the cursor visible again, just POKE it with a zero. That's (almost) all there is to it.

Hold it, what was the "almost" that was trying to hide in the parentheses back there? Well, there is one tiny thing I forgot to mention. The cursor won't disappear (or reappear) until you move it for the first time after you change CRSINH. All that means is you have to have a PRINT of some kind after the POKE. The easiest way around this is just to POKE CRSINH before you print anything on the screen. For example:

```
100 POKE 752,1
110 PRINT
```

CRSINH is set to zero when you turn on the computer, and also when you press Break, press System Reset, use a GRAPHICS command or OPEN either "S:" or "E:".

Also see CHACT at location 755 for another way to tell the cursor to get lost.

Here is a way to place dots all over your screen so that you can check the convergence of the TV or monitor:

```
10 POKE 710,0:POKE 752,1
:POKE 82,0:FOR I=1 TO 95
9:? ".";:NEXT I
20 GOTO 20
```

KEYDEL
753 02F1

A lot of you have probably heard the term "debounce" (no, it's not from a commercial for French shampoo). Some of you probably don't have the slightest idea what it means, so let's talk debounce for a bit.

When you press a key, you're actually bringing two little bits of metal together. When the two touch, electricity flows through them and tells the computer that the key is pressed. Sometimes, when the two first hit each other, they bounce a little. This has the effect that they are touching, then they're not touching, and then they're touching again, which the computer would normally interpret as meaning the key was pressed twice. You only pressed it once, however, so somehow the computer has to be smart enough to realize this. The process it uses is called "debouncing," and it's fairly simple. If a bounce occurs, it happens real fast; too fast for you to have been able to hit the key twice. So, the OS waits a little while after you first press the key before looking to see if you pressed it again. That way, it doesn't see the bounce. KEYDEL tells it how long to wait.

KEYDEL is set to three whenever a key is pressed and then every stage 2 VBLANK it's decremented by one. Until it reaches zero, the OS will not let the same key be pressed again. Unless you can press a key faster than 20 times a second, this won't be a problem for you.

CH1
754 02F2

CH1 is the value of the last key pressed (not the current one). When you press a key, the OS checks its value (stored in CH [764]) against CH1. If they're the same, then KEYDEL is checked to make sure that the key has been debounced. If KEYDEL is equal to zero, or if the two values aren't the same, then the current key code is stored in CH1 and the OS goes on to process that key.

CHACT
755 02F3

CHACT does some neat things to the characters on the screen. The bits are used as summarized in Figure 3.

-------0	inverse character letters are visible
-------1	inverse character letters are invisible
------0-	inverse character backgrounds are visible
------1-	inverse character backgrounds are invisible
-----0--	all characters are right-side up
-----1--	all characters are upside down

Figure 3. CHACT bit chart

What does this mean? Try typing some inverse characters on the screen (use the Atari logo key). Now POKE 755,1. What happened? That's right, the letters disappear. Try POKE 755,2. This makes the background (the solid white part) disappear. Finally, try POKE 755,3 to make everything disappear (everything in the inverse characters). That should give you a good idea of what the first two bits can do. By the way, since the cursor is essentially an inverse character, it will disappear as well when you make the inverse character background disappear.

The last bit is pretty self-explanatory. Just try POKE 755,4 and see what happens.

What can you use CHACT for? Reverse characters add emphasis to text, CHACT lets you add even more emphasis by making inverse characters blink. Try the following:

```
100 GRAPHICS 0
105 POKE 752,1
110 PRINT :PRINT "Add EM
PHASIS to your programs"
120 FOR BLINK=1 TO 10
130 POKE 755,0
140 FOR DELAY=1 TO 50
150 NEXT DELAY
160 POKE 755,2
170 FOR DELAY=1 TO 50
180 NEXT DELAY
190 NEXT BLINK
195 POKE 752,0
```

Try substituting other values for the zero in line 130. Also see location VVBLKD at 548 and 549 for a machine-language routine that uses CHACT to make inverse text blink while you're typing it.

In case you hadn't figured it out already, CHACT is initialized to two.

CHBAS
756 02F4

This is a biggie. (Have I ever lied?) CHBAS holds the address in pages (so you multiply the number here by 256 to get the actual address) of the character set. What is a character set? A character set is a whole bunch of numbers that tell the computer how to draw the various characters on the screen. In other words, it tells the computer what the characters look like. How can numbers describe what a character looks like? First of all, you should go read the section from a couple of months ago on bits and bytes. Then come back here.

Back already? Okay, what do bits and bytes have to do with character descriptions (why am I asking so many questions)? Well, a byte can be thought of as part of a picture. You know, with the bits being dots in the picture. You turn a bit on, and the corresponding dot in the picture gets turned on. You've already seen how this is used in the graphics modes. Well, the text modes also need to turn dots on and off, but they need to change a whole bunch at once for each character. So what the Atari does is store eight bytes for each character in this special thing called the character set. Each of these descriptions is given a number, and to set the right dots for a particular character, the computer just has to say, "Hey, get me the description for character number (whatever) and put it on the screen," and the character will magically appear on the screen.

Let's take a look at how those eight bytes make up a character in Figure 4.

# IN MEMORY	BIT PATTERN
0	00000000
12	00001100
28	00011100
60	00111100
108	01101100
126	01111110
12	00001100
0	00000000

Figure 4. CHBAS bit chart

MASTER MEMORY MAP PART VI

Now you can see how simple creating characters is. First draw the shape of an 8 x 8 pattern of 0s and 1s. Next add up the value of the ON, or 1, bits. Then POKE these numbers in the proper order into memory. Let's go over the details.

Look at those bits again in terms of dots, with the 0s meaning no dot, and the 1s meaning dot;

```
        ####
      ######
    ########
  ####  ####
  ############
        ####
```

Ah, ah! The description we used was for the "4" character. You should now be able to see how the descriptions work.

How are the descriptions ordered within the character set? It's not the same order as ATASCII (the order that CHR$ and ASC use). To convert from ATASCII values, which you can find in your BASIC manual, to the character set order, use Figure 5.

TYPE OF CHARACTER	ATASCII NO.	CHAR. SET NO.
uppercase, numbers, punctuation	32-95	0-63
graphics, characters	0-31	64-95
lowercase	96-127	96-127

Figure 5.

Now, to find the character description of a particular character, find the ATASCII value (either with ASC or by looking it up in the chart in the BASIC manual), use the preceding chart to convert it to the character set value (more commonly called the "internal" value), multiply that by eight (because there are eight bytes for each character) and add it to PEEK(CHBAS)*256. The result is the address of the first byte of the character description you want.

The character set that comes with the Atari is stored starting at location 57344. You can double-check this by PEEKing CHBAS and seeing that it has a value of 224. There are a total of 128 possible characters (not counting inverse ones), so the character set takes up 128*8 equals 1024 bytes.

In graphics modes 1 and 2, you probably know that you can't have upper- and lowercase letters on the screen at the same time. Why not? In these modes the characters can be one of four possible colors. In order to be able to pull this off, two of the bits in the character number have to be used to specify the color. This means that only six bits are left to specify the character. Six bits are enough to give you the numbers 0 through 63. Zero through 63, if you consult the preceding character order chart, are the uppercase characters, numbers and punctuation. So what if you want lowercase? The BASIC manual tells you to POKE 755 (CHBAS) with 226 instead of 224. What does this do? It moves the start of the character set forward by 512 (2*256) bytes. Now I know that right now you're thinking to yourself, *Gee, 0 through 63 is a total of 64 characters, and eight bytes for each character gives me, let me see, uh, 512 bytes!* Hey, you're terrific! What you just caught on to is that changing CHBAS like that simply lets you skip over to the lowercase and graphics characters, the other half of the character set.

Unfortunately, this means that the heart character gets used as a space, so your screen is filled with hearts—romantic, but not what you want. You can get rid of them with SETCOLOR 0,0,0 or by redefining the heart character to a space.

In graphics mode 0, there's not much more to tell. If an inverse video character is requested (see INVFLG [694]), then the eight bytes for that character are reversed (1s changed to 0s and vice versa) before they are put on the screen.

CHBAS is a shadow register for CHBASE at location 54281. For some reason you cannot set CHBAS to an odd number, or garbage will fill the screen. Finally, CHBAS can be set to point to your own character set.

"Hold on there, just a second, wait a minute, time out, take five, whoa! You mean I can design my own character set? And you took all this time before you told me, and now you're going to move on without telling me how to do it? What kind of author are you?"

We'll cover character sets in a future installment of "Master Memory Map."

Noname
757-761 02F5-02F9

More spare bytes. You know, I have to assume that you're going to come to these locations and forget all about that warning I gave you way back when. You remember, "Don't use spare bytes, they may be used in future versions of the OS." But if you did remember and are getting sick of me telling you every time we come across some spare bytes, then what can I say? It's a rough world out there.

CHAR
762 02FA

This is the internal number (value) of the character that was read or written last by the display handler. A lot of the time the handler will move the cursor as the last step of an operation, so PEEKing here will often return a value of 128 or 0 (for a visible or invisible cursor respectively).

ATACHR gives the ATASCII value corresponding to the internal value in CHAR.

ATACHR
763 02FB

ATACHR is used by the display handler, the screen editor and the keyboard handler to hold the ATASCII value of the character last read or written. If we're using a graphics mode rather than a text mode, then it's the value of the graphics byte rather than that of the character (for the display handler only). It's also used in converting ATASCII to internal and vice versa, and FILL uses it to hold the color of the area being filled (in which case it gets its value from FILDAT [765]).

CH
764 02FC

CH is the middle guy between the keyboard and the keyboard handler. When a key is pressed, a keyboard value (yes, yet another kind of character value) gets put into CH. The keyboard handler then picks it up, puts it into CH1 (754), and puts a 255 into CH to indicate that it got the value okay. There are a few exceptions to this. First of all, if we're in the middle of debouncing (see KEYDEL [753], the key is ignored completely; it doesn't even make it to CH. If Control-1 is pressed, then SSFLAG (767) is updated, but CH is not affected. Finally, CH also gets changed by the key repeat process mentioned under SRTIMR at location 555. To repeat a key, the OS takes the value in KBCODE (53769) and stores it in CH.

If you are GETting information from the keyboard, make sure you set CH to 255 before you do your GET. This will make sure that any previous key presses are ignored. For example,

```
100 OPEN #1,4,0,"K:"
110 POKE 764,255
120 GET #1,A
130 PRINT "You pressed k
ey number ";PEEK(754)
140 GOTO 110
```

You can use this program to find out the values for the various keys, or you can look at the chart on page 50 of the OS manual. In either case, you should notice that the Control key adds 128 to a key value, and the Shift key adds 64.

Here is my favorite trick for this location. Say you want your program to load in a tape program and then run it. It would seem that there is no way to do that because someone has to press the Return key after the program loads and you type RUN. Not true. Use location 764 to hold the Return key like this:

```
10 REM YOUR PROGRAM HERE
.
.
2000 POKE 764,12:CLOAD:R
UN
```

FILDAT
765 02FD

Simply put, FILDAT is the data of FILL within the XIO 18 command.

DSPFLG
766 02FE

When DSPFLG is set to a nonzero value, then Control characters like Control-Clear, Control-Delete, Control-arrow and so forth will appear as a character on the screen rather than having some kind of effect on the screen (such as clearing it or moving the cursor). If it's equal to zero, then they have their normal effect.

Note that to type a Control character so that it appears on the screen, you press Esc before you type that character. ESCFLG (764) is ORAed with DSPFLG before the character is processed. That means that the Esc key is not the only way to get Control characters to appear. That's good. Suppose, for example, that you want to print the arrow characters on the screen from BASIC. You can use the Esc key to type them into a string, but when you try to print that string to the screen, BASIC will move the cursor rather than print the arrows. What you have to do is POKE 766,1 before you try and print the string. Be sure to change it back afterward.

SSFLAG
767 02FF

SSFLAG is used to pause a program or a LISTing. When it's set to 0, everything works as usual. When set to 255, however, the pause is in effect and will stay that way until it's set back to 0 again. If the basic idea of this sounds like something you've run across before, that's because it is. The Control-1 key, which you have probably used to pause your LISTings, changes SSFLAG. You can also change SSFLAG yourself, but if you do it from within a BASIC program, keep in mind that the program is paused, so you won't be able to change it back unless somebody presses Control-1! Try this:

```
100 POKE 202,1
110 PRINT "Now try LISTi
ng this program"
```

SSFLAG has no effect on machine-language routines, which is why you can't use Control-1 to pause some programs.

M/L EDITOR

For use in machine-language entry.

by Clayton Walnum

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.).

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.

LISTING 1: BASIC LISTING

```
AZ 10 DIM BF(16),N$(4),A$(1),B$(1),F$(15)
   ,F1$(15)
LF 11 DIM MOD$(4)
BN 20 LINE=1000:RETRN=155:BACKSP=126:CHKS
   UM=0:EDIT=0
GO 30 GOSUB 450:POSITION 10,6:? "Start or
    Continue? ";:GOSUB 500:? CHR$(A)
ZG 40 POSITION 10,8:? "FILENAME";:INPUT F
   $:POKE 752,1:? " "
FE 50 IF LEN(F$)<3 THEN POSITION 20,10:?
   "  ":GOTO 40
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:CHKSUM=0
GH 180 L1=3:FOR X=1 TO 16:POSITION 13*(X<
   10)+12*(X>9),X+2:POKE 752,0:? "BYTE #"
   ;X;": ";:GOSUB 310
KH 190 IF EDIT AND L=0 THEN BYTE=BF(X):GO
   TO 210
FY 200 BYTE=VAL(N$)
OZ 201 MOD$=N$
BU 210 POSITION 22,X+2:? BYTE;"  "
YZ 220 BF(X)=BYTE:CHKSUM=CHKSUM+BYTE*X:IF
    CHKSUM>9999 THEN CHKSUM=CHKSUM-10000
MS 230 NEXT X:CHKSUM=CHKSUM+LINE:IF CHKSU
   M>9999 THEN CHKSUM=CHKSUM-10000
IG 240 POSITION 12,X+2:POKE 752,0:? "CHEC
   KSUM: ";:L1=4:GOSUB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
SY 270 POSITION 22,X+2:? C;"   "
IL 280 IF C=CHKSUM THEN 300
DI 290 GOSUB 440:EDIT=1:CHKSUM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GOSUB 500:IF (A=ASC("Q") OR A=ASC(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACKSP AND (A<4
   8 OR A>57) THEN 320
DX 331 IF A=RETRN AND N$="" 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
   ACKSP) AND L=0 THEN 320
DW 350 IF A=RETRN THEN POKE 752,1:? " ":R
   ETURN
GG 360 IF A<>BACKSP THEN 400
SA 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
AS 380 N$=""
RE 390 ? CHR$(BACKSP);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WX 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHICS 0:END
YT 430 GOSUB 440:POSITION 10,10:? "NO SUC
   H FILE!":FOR X=1 TO 1000:NEXT X:CLOSE
   #2:GOTO 30
FD 440 POKE 710,48:SOUND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:SOUND 0,0,0,0:RETURN
MY 450 GRAPHICS 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 STEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 STEP 2:POKE DL+X,0
   :NEXT X
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
```

PANAK STRIKES

by Steve Panak

Well, it finally happened: I'm out of games. It's an event that has been looming on the horizon for months now, a sad fact that I've lamented endlessly in these pages. It's even become something I've learned to live with. And while I am in the process of searching out the few titles I haven't looked at yet, I decided this month to look at what it takes to make a given game great.

The question that immediately comes to mind is that if I know so much about these games, why don't I just make them? To answer this, some might paraphrase Woody Allen, saying that those who can't do, teach, while those that can't teach become critics. The truth is much sadder. Alas, I feel deep sorrow when I must confess that I am at the point in my life when there's barely time enough to keep up with what I've got going, much less anytime to learn any new skills. I must instead attempt to master those I already have. And programming is not one of them. A firm grasp of BASIC is not enough to create award-winning games. For this reason, I choose to play games, which in turn qualifies me to enumerate and comment on what I perceive to be the required components of each game genre, the qualities a program must have to make the cut. So you developers out there, read carefully, and hopeful programmers lend an ear.

The first genre I'll look at is the war simulation, probably the simplest game to pull off successfully. The hallmark of a great simulation is an uneasy balance between a wide range of command choices and a simple control interface. Give me enough control to make it real, but make it simple enough that I don't need a doctorate to play. For this reason, the joystick is preferred, optimally combined with a menu across the bottom of the screen. This not only allows the player to relax during the lengthy sessions which are required to complete a simulation, but also speeds learning, by offering a continuous reminder of the currently available options.

A simulation should also be a strong opponent. If, once you learn to play the game, you find the computer is a pushover, you'll quickly shelve the game. Graphics, while important, generally take a backseat in a simulation. But they should still be carefully designed so that it is easy to differentiate the various unit icons, as well as the differing battlefield terrains. Finally, the scenario must be engaging. This is the toughest aspect, in my opinion. Too many games center themselves around this or that or the other civil or world war battle. The result is a market cluttered with look alikes. Too few simulations allow space battles or colonization of distant planets, or sword and sorcery for that matter.

Speaking of sword and sorcery, the fantasy game is a specialized simulation which just happens to be my favorite game type. Give me a well-designed fantasy, and I'm apt to deprive myself of sleep until it's finished. The rules for simulations apply here, although the graphics are a little more important. After all,

PANAK STRIKES

I know a good game when I see one: It must be tough, but fair in its never-ending onslaught of dangers and rewards.

they've got to create and enhance a whole new world, rather than simply build upon familiar and easily visualized surroundings. The old saying that a picture is worth a thousand words applies here. Don't depend on the documentation to paint the entire picture.

Operationally speaking, the simulation has the worst track record as far as performance goes. Most games are too large to exist completely in RAM, hence seemingly endless disk accesses and swaps are often necessary to play the game. When this is necessary, the pauses should be placed at natural break points if at all possible—say between phases, or, preferably, between players. And password-based copy-protection schemes should request a response only once during any play session.

Another type of simulation is even simpler to establish the criteria for success, but much harder to implement. I call these the thinking games. Programs which beat man at his own games; for example, chess. Golf games also fall into this grouping. These programs must, of course, know how to play the game. It would be most annoying to run into a chess game which failed to recognize an allowable castle, or a golf opponent who refused to take penalty strokes when lifting out of a water hazard.

In addition to knowing the rules, these games must be good opponents as well. As in the simulation, I didn't pay the price of admission just to win every time to play against a moron. I want to be challenged. And I want to jump right in. For this reason, the control interface is again important. With these games I prefer a scheme which is intuitive, so that you can step right into the game. A good example of this is Electronic Arts' *Chessmaster 2000*, in which you manipulate an animated hand over the pieces, and then pick them up. Menus are again helpful. The point is that you shouldn't have to read a lengthy manual just to partake of a game you've played all your life.

Arcade games are the hardest to peg, which is why I've saved them for last. Who would have thought that a little smiley face that gobbled dots could also manage to gobble up so many quarters. It goes without saying that the images must be colorful and distinct, the action fast and challenging. The skeleton of this genre is as set as any other. The guts of the good game are what is so mysterious. The only thing I can say is that I know a good one when I see one. *Pac-Man*, *Asteroids*, *Donkey Kong*, *Xevious*. The game must be tough, but fair in its never-ending onslaught of always changing dangers and rewards. Extra men should be able to be rewarded indefinitely, a la *Missile Command*. In a great arcade game, the sound, the graphics and the action are almost hypnotic, forcing you to zombie-like play just one more time.

Now a little about documentation. Make it fit the game. An arcade game doesn't need an elaborate story line. You need to know how to load the thing, and from then on the screen should be the source of your information. A well-designed demo or teaser screen will draw the uninitiated right into the game. As the game becomes more complex, so must the manual. Indexes and reference cards are essential. Often the reference card is all I use to learn to play a game.

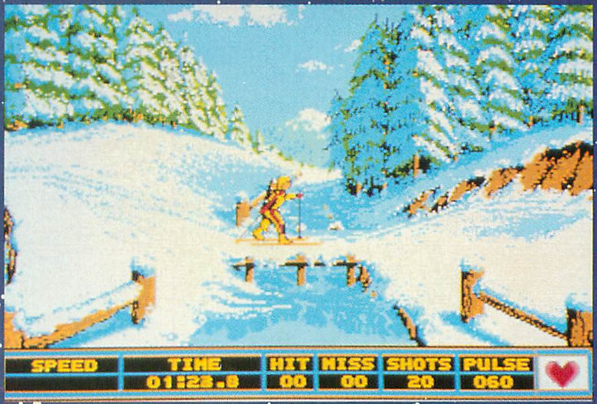
Well, that's it for this month. I hope I've given a couple of designers a few ideas, as well as enlightened some end users. When next we meet, I'll apply these principles to some games I've just received: a couple of arcade games and a new simulation.

INTRODUCING

VideoGames & Computer Entertainment™

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
& Computer Entertainment™
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____________________

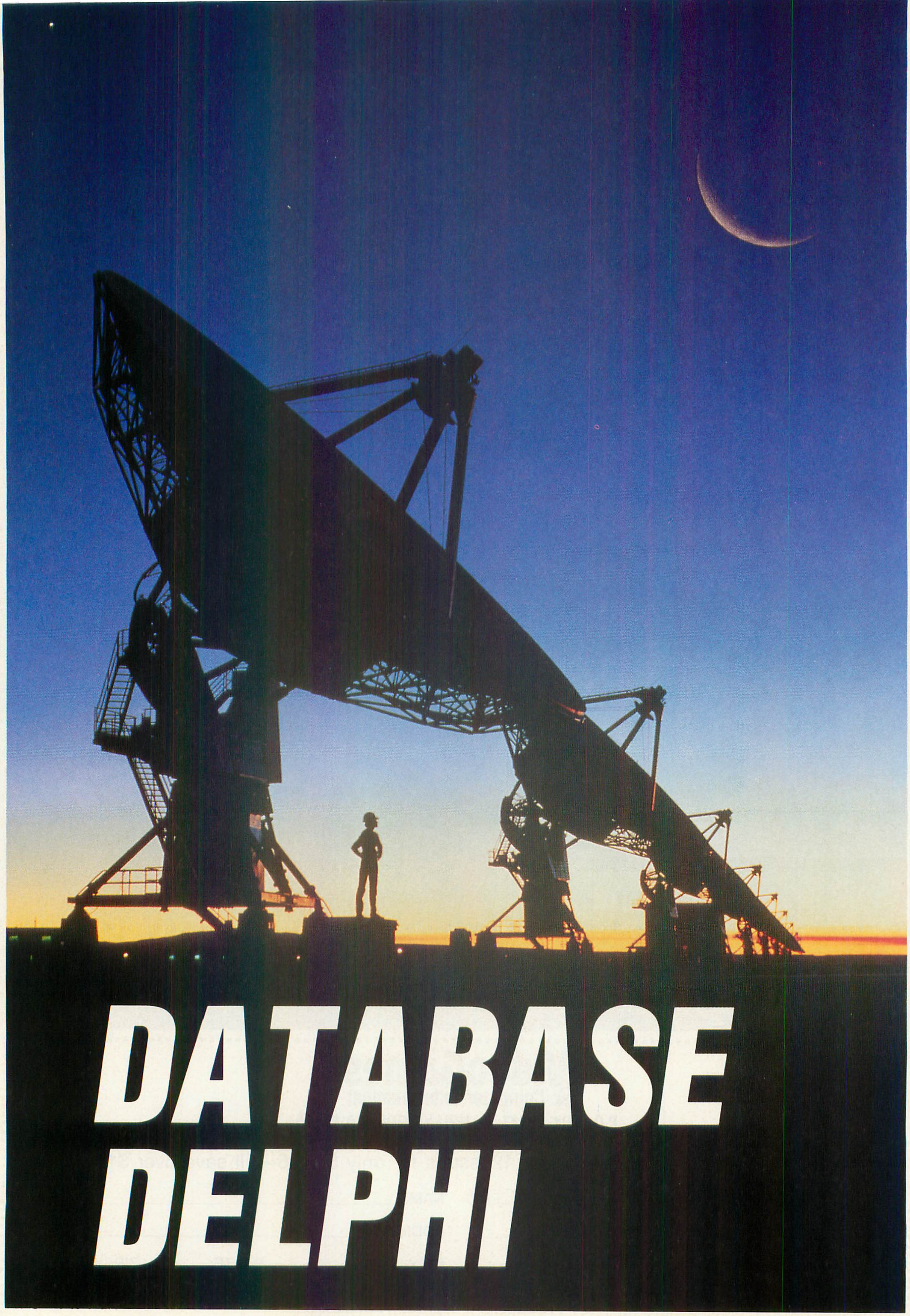
ADDRESS____________________

CITY__________ STATE______ ZIP______

Your first issue will arrive in 6 to 8 weeks.
WATCH FOR IT!!
Offer Expires 3/27/89.

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

MCMYA



DATABASE DELPHI

by Michael A. Banks

HAPPY NEW YEAR! BUENO ANO NUEVO! AKEMASHITE OMEDETO GOZAI MASU!

Well, here we are—at the beginning of the end of what singer Jimmy Buffett calls the "Fabulous Eighties!" I can't say that this decade has been as tumultuous and fascinating as, say, the '60s, but it has been interesting. As we enter into the final year of the decade, allow me to extend my best wishes to all of you for a prosperous and even more interesting new year.

Hm. . . I wonder what the '90s will bring? Certainly more computers, and more things to do with them. But, while you're waiting for the turn of the decade, there's plenty to keep you busy in ANALOG's Atari SIG—like getting to know your fellow SIG members better! Let's take a look. . . .

Finding Out Who's Who Online

In recent columns, I've shown you how to find out who's online, and how to find information such as a member's real name. This time, we'll delve even deeper into finding out about your fellow SIG members.

Short of asking someone in E-mail, Forum or real-time conference, the best way to find out who someone is, and about their likes, dislikes, location and interests, is to read a user's online profile. A profile is information a member enters about him/herself in one of several public areas on DELPHI. The information is entered under keywords and is searchable. There are two areas where users can enter profiles. One is PEOPLE, (a selection of the DELPHI Main Menu). The other is in SIG Member Directories. The two are separate databases. A member's PEOPLE profile usually contains general information. A Member Directory profile contains more specialized information, pertaining to the SIG's special areas interests.

PEOPLE

To see a profile as entered in PEOPLE, type /WHOIS followed by a membername at most any SIG prompt. You'll see what the member in question has entered in his or her profile:

```
ANALOG>What do you want to do? /WHOIS BACHAND

NAME       : Charles Bachand
CAR        : I'm currently driving a 1985 Nissan 300ZX 2+2.  It was the only car
   I could fit into comfortably. <grin>
COMPUTERS : Atari ST
DELPHI     : I'm currently helping out in several of the on-line SIGs here,
   trying to get into CSIX (Computer Shopper), Atari 8-bit (ANALOG), Atari ST
   (ST-Log), and of course, the Hobby SIG, each day.
FADS       : Favorite word this week: "CURRENTLY".
HOBBIES    : Currently I'm into R/C Cars (1/10-scale electrics) with my Kyosho
   Rocky 4WD. I'm also an electronics experimenter and the first one holding a
   screwdriver whenever a new electronic toy comes my way.
LOCATION   : Worcester, MA
OCCUPATION: Software and text author, having worked for ANALOG and ST-Log
   magazines for a number of years.  I'm currently developing Atari ST specific
   software, as well as implement a full blown hobby store on-line in the Hobby
   SIG's Shopping Area.
TERM-SOFT : Flash!
```

Entering information in PEOPLE or SIG databases is not compulsory, however, and if a member hasn't entered any information about him/herself in PEOPLE, DELPHI will inform you, like this:

```
There is no information on file for BINGO
```

Profiles vary in length and content, depending on the individual's preferences. Members may enter whatever they wish about themselves, so if something you read in a profile seems a little fantastic, you might take it with a grain of salt.

To get into the PEOPLE area—either to perform a search (as described later) or to enter your information—type GO PEOPLE at most any Atari SIG prompt, or exit to the DELPHI Main Menu and type PEOPLE. The PEOPLE area operates exactly like the Member Directory.

The commands used in PEOPLE and the Member Directory are the same, so keep in mind that what follows in the description of the Member Directory applies to PEOPLE, as well. The only difference is in where members can access the information entered in the databases.

Member Directory

To access the information entered in the ANALOG Atari SIG's Member Directory, type Member, which is a selection on the SIG menu. You'll see this menu:

```
ANALOG>What do you want to do? MEMBER

MEMBER Directory Menu:

I-Am
Who-Is
List-Keywords
Browse
Search
Help
Exit
```

Who's That? (WHO-IS)

The simplest operation in the Member Directory is viewing the profile of a specific DELPHI member. Type WHOIS (or

WHO-IS or /WHOIS) and a membername to see that member's Member Directory profile. The format is the same as that displayed by PEOPLE, but the content will probably be quite different because, again, PEOPLE and the Member Directory are two different databases.

Browsing the Member Directory (BROWSE)

You don't have to enter a specific member name to access information about other Atari SIG members. You can browse through Member Directory entries in sequential order. Type BROWSE, and you'll be prompted for a letter or letters; this specifies the point at which the Directory should begin displaying profiles. (You can enter numbers, too.) For instance, type *A*, and you'll see information on the first member name beginning with "A." Type *KZ*, and you'll see my Member Directory entry. Type *0*, and you'll see the very first entry in the Directory. Press Return to see ensuing entries in alphabetical order; enter Control-Z to return to the Member Directory menu.

BROWSE is also useful if you want to find information about members whose names you may recall only partially. For example, a user name such as GEODEAMON may be rather difficult to recall, but if you remember at least "GEO," you can enter those three letters and find the full member name and profile (assuming the member in question has entered his/her profile in the Member Directory). When used this way, BROWSE is a useful search tool, especially since it will turn up all user names that begin with the specified string.

Searching Out the Facts (SEARCH)

The Member Directory offers a sophisticated search tool that is based on the keywords under which members enter information about themselves. But its structure differs substantially from that of the SIG software databases, so I need to take a few lines to explain the structure of the database before I can tell you how to use SEARCH.

Rather than having keywords attached to entries in the database, the information in each entry is organized under various keywords. This makes for an important distinction when you enter or search information; rather than searching for keywords, you search the information that is stored under a keyword. As an example, in ANALOG's profile, the searchable information under the keyword NAME is "Clayton Walnum." In the actual search operation, you would specify NAME as the keyword to be searched, and "Clayton" or "Walnum" (or both—or any fragment of either name) as the information to be found, which is also known as the "search string."

Here's a more detailed example: Let's say you want to find people who use the HOMETERM terminal program. In that case, you would type SEARCH, then specify TERM-SOFT (one of the standard keywords used by the database) as the keyword, and HOMETERM as the search string. The search would go like this:

```
DIRECTORY>(I-Am,Who-Is,List,Browse,Search): SEARCH

What KEYWORD:TERM-SOFT

Search for:HOMETERM

Matches found for the following membernames:

113MAMA       AARONFINCH    ADOLPH        AJK           ATARUS
BARRYGRAY     BATTERIES     BELLABS       BERNIEB       BETHEL
BOND007       BRYON         BUDDY         BVECTOR       CANNONBALL
CASHFLOW      CHARLEKOONTZ  CKSCULLION    CLOCKMAN      CONTROLATARI
CSONG         CYMOLER       DALLASATARI   DAVIDMARTIN   DBLAKESLEE
DDMASTER      DDW           DICKB         DICKDERY      DML
DOCTORDAVE    DWAINE        DWGLOVER      EEF           ELECTRODAVE
ESTUS         EXPRESSWAY    FASTMAN       FLIPPER       FLYINGTURTLE
FRANKN        FRETS         GENEDOUGLASS  GEOFFREY      GFBROWN
GLOBE         GRANDPAW      GRAYWOLF      HACKER        HALUEKING
HAP           INET65        JABER         JACE          JAWS
JAYMER        JEFE          JEFFC         JEPU          JERRYCHAM
JJJACKSON     JJJF          JL626         JOEST         JOEWYKS
JOHNNY1       JOHNNYMAC     JOHNO         JOJO          JRQ
KARNOW        KATOOM        KDG           KENMATHESON   KJOHNSON
LEHI          LWB           MAGNATE       MEMBERSHIP    MERLYN
More?N
Found 80.
Would you like the complete profiles printed for the selected
membernames?N
```

Had I typed *Y* at the "More?" prompt, DELPHI would have displayed another screenful of member names. And if I had typed *Y* at the final prompt, DELPHI would continue displaying profiles for all of the above-named members, until I entered Control-Z or Control-C. (Obviously, HOMETERM is a very popular program.)

Note that a keyword search will find any string under a keyword. Thus, if you specified OCCUPATION during a search and entered "write" as the search string, ANALOG4's entry would come up because "writer" is under the specified keyword.

Make a List (LIST-KEYWORDS)

Speaking of keywords, you can see a list of the most commonly used keywords in the Member Directory, or a list of *all* current keywords in use by typing LIST-KEYWORDS at the DIRECTORY prompt. When you do a search, it's handy to know

One of the best ways to learn about the Member Directory database structure is to enter your own profile.

just what's there, so you might want to view and capture the keyword list(s) before you try a search.

Type LIST and you'll be asked whether you wish to see a list of all keywords, or just the more frequently-used keywords (recommended). Seeing the keywords will also help you better understand the database structure.

"Getting to Know You . . . (I-AM)"

Actually, one of the best ways to learn about the Member Directory database structure is to enter your own profile, which you will probably want to do sooner or later, anyway. Going through the process of entering your profile is very instructive.

DELPHI will prompt you and provide instructions every step of the way, and you'll find that you understand the keyword system completely when you're done.

DELPHI will initially ask you to enter information about yourself under these keywords: NAME, COMPUTERS, LOCATION and TERM-SOFT. After that, you can add information under other keywords by typing ADD. (If you've left the I-AM menu, you'll have to type I-AM first.) Almost anything can be used as a keyword, but keep in mind the fact that people will be looking for you, and use logical keywords. Browse through some of the existing profiles to get ideas for keywords.

You can also edit or delete keyword entries later on.

And remember: information you enter in the Member Database is accessible only at the Member Database menu; it cannot be accessed via the /WHOIS command at prompts outside the Member Directory.

Spotlight: ANALOG's Atari SIG Databases

Database offerings continue to grow, and you'll find highlights from the most recent issues of ANALOG in the Current Issue database. And don't forget to check the Recent Arrivals database for new uploads not yet moved to their specific database categories.

Speaking of databases, here's a reminder to ANALOG readers who aren't yet DELPHI members: among the many benefits of DELPHI membership is being able to download program listings that appear in ANALOG. That's a major convenience, when you consider all the time you spend keyboarding programs—and correcting errors. And there are thousands of other programs uploaded by ANALOG Atari SIG members, as well as the latest Atari news and reviews and much, much more. Interested? You can sign up right now: See the accompanying sidebar for online signup information.

Conference Reminder

On Tuesday evenings at 10 p.m., EST, ANALOG's Atari SIG is the place to be! The Atari Users' Group holds a real-time conference that is open to everyone. Come on by and meet other Atari users.

To join in the conference, 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 beneath the group name. 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.

That's it for now. Next month: customizing DELPHI. 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.*

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:EDITORL1.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 "EDITORL1.BAS"*.

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

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

Cassette version:

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

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

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

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

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

(6) Rewind the cassette.

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

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

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

Using the Editor

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

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

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

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

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

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

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

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

Leaving the Editor

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

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

Large listings

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

The post-processor

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

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

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

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

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

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

Murphy's Law

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

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

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

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

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

Listing 1.
BASIC listing.

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

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

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

Listing 2.
BASIC listing.

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

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

ANALOG COMPUTING

SAVE TIME AND MONEY SUBSCRIBE TO ANALOG

SAVE $19 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ANALOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS. GET THE MOST OUT OF YOUR COMPUTER

SUBSCRIBE TO ANALOG TODAY

MCMYY

□ 1 YEAR @ $28 — SAVE $19
FOREIGN — ADD $7 PER YEAR
□ 1 YEAR WITH DISK @ $105
FOREIGN — ADD $15 PER YEAR
□ PAYMENT ENCLOSED □ BILL ME
CHARGE MY: □ VISA □ MC #__________

EXPIRATION DATE ______ SIGNATURE ______

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME ______

ADDRESS ______

CITY ______ STATE ______ ZIP ______

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16927, N. Hollywood, CA 91615.
Your first issue will arrive in 6 TO 8 weeks.

Offer expires 3/27/89. **WATCH FOR IT!**

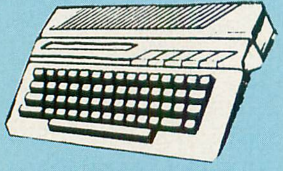
BUY•SELL•TRADE BUY•SELL•TRADE BUY•SELL•TRADE

Computer Repeats, Inc.

TRADE FOR ANYTHING WE SELL!

ALL ORDERS PROCESSED IN 24 HRS!

Atari 130XE



$45 NEW

with trade-in of 800XL, 810 Drive, 1030 Mdm.
$CALL for your system

Used

520ST FM Computer	$399	1040ST Computer	$585
Atari SF314 Drive	$175	130XE Computer	$119
400-16K Computer	$29	800XL Computer	$69
Modems from	$19	Atari 1030 Modem	$35
1050 Drive	$139	Atari 1027 LQ Prn.	$79
IBM Proprinter-II	$189	Okimate 10 w/PIP	$99
Indus GT AT	$139	Slave Drives from	$35
Atari 850 Interface	$69	Modems from	$19
Digitizer/Synths from	$20	Software/Books from	$1

New

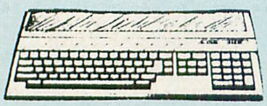
1040ST Plus+ CPU	$719	520ST FM mono sys	$659
Atari SF314 Drive	$209	256K DRAM	$11
Atari 65XE Computer	$109	SC1224 color Mon.	$325
US Doubler w/DOS	$39	XF-551 Drive	$179
1802C Color Monitor	$189	Computereyes Dgtzr	$119
Atari SX-212 Modem	$89	Avatex 1200HC	$99
Avatex 2400HC	$199	Happy Rev. 7.1	$99
256K upgrades 0k from	$35	R-Time 8 Cartridge	$49
Star NX-1000 144cps	$189	ICD R-Time 8 Cart.	¢49

Atari XE Game System
with $75 Free Sftwr and light gun

$86
$50 Atari Rebate
$36 NEW

with trade-in of 130XE OR add $29 with 800XL
$CALL for your system

Atari 520ST FM
with Double Sided Drive



$329 NEW

with trade-in of 800XL, 1050, Color Monitor
$CALL for your system

$Cash for your equipment
Thousands of software & book titles
Plus, MUCH, MUCH MORE!

5721 Arapahoe Suite 1B
Boulder, CO 80303

Atari XF-551

$119 NEW

with trade-in of 1050 Drive OR add $20 with 810 Drive
$CALL for your system

All references to trade-ins assume equipment to be in good working condition. Shipping/handling will be added to all prices. No additional charge for credit cards or COD. Mail order prices shown.
WE CHECK FOR CREDIT CARD THEFT!

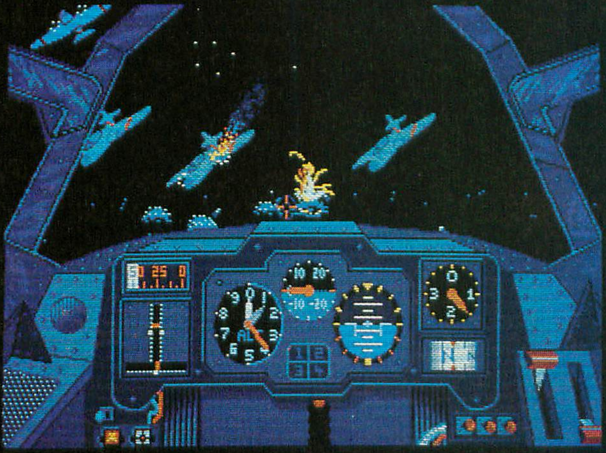
1-800-347-3457
24HR Mdm Sftwr Quotes: 1-303-939-8174 Questns: 1-303-939-8144

Authorized Sales & Service for
COMMODORE/AMIGA
and ATARI ST/XL/XE
Computers and Accessories.

VISA MASTERCARD DISCOVER AMERICAN EXPRESS COD

CIRCLE #109 ON READER SERVICE CARD.

REVIEW



Dive Bomber

Epyx
600 Galveston Drive
Redwood City, CA 94063
$49.95, Color Only

Reviewed by John S. Manor

I flew high above the churning waves. A ship came into view. I pushed the stick forward, putting my Blackhawk dive bomber into a steep dive. At just the right distance from the ship, I released my single torpedo. I watched as it sped through the waves to strike the enemy ship in its center, exploding and sending that ship to the bottom of the ocean.

Dive Bomber, from Epyx, is a slick simulation of airborne naval warfare during World War II. The premise is that you and your plane, the first of its kind, are on loan to the British. The *Bismarck* has just sunk the *HMS Hood*. Your main purpose is to sink the *Bismarck*—to get revenge. Dive Bomber takes you through all the steps in flying the Blackhawk off the aircraft carrier *Ark Royal*. You learn the procedures for takeoff, flight and landing. Then you can try shooting down enemy planes and sinking enemy E-boats or U-boats.

At the start of the game you choose to practice or fly one of five real missions. In practice you can try flying, landing, taking off or attacking the *Bismarck*. You draw straws to choose a real mission. During your mission you must return to the *Ark Royal* to refuel, rearm and to make repairs. While you are in the air, enemy forces will try to sink your carrier. So you have a double duty: Carry out your mission while protecting the *Ark Royal*; otherwise, you may run out of fuel and ammo, and be forced to ditch your plane at sea.

There are four different screen locations in your Blackhawk dive bomber. These are the Pilot's, Engineer's, Navigator's and Tail Gunner's screens. The Pilot's screen is your main flying screen. You see the ocean (sort of like the stars in *Star Raiders*), your control stick and gauges from the window. There's also a lever for dropping your torpedo. The Engineer's screen is used for preparing for takeoff, idling the engine, switching fuel tanks, landing and more. The Navigator's screen shows a map of the area. Geographical locations like the "Isle de Gilligan" are marked on the map. Enemy ships, planes and mines are clearly displayed. U-boats don't show up until they are surfaced. The Tail Gunner's screen allows you to shoot at enemy planes that creep up on you from behind. There is a screen selection box that will light up a number for a screen that needs your attention.

Dive Bomber is fairly easy to learn to play. You can use a joystick and keys or a mouse and keys to control the Blackhawk. The mouse gives you a better feel for the plane. Using it made me feel like I was gliding through the air. When I went into a dive, I felt my stomach lurch.

I found it difficult to locate and destroy enemy ships. After several days of practice I became better at this, though I still usually lost the *Ark Royal* to enemy U-boats. Sinking the *Bismarck* is also a difficult goal to obtain. Even if you get close enough to drop your torpedo, you will probably miss every time at first. I have found that waiting until you're really close and then putting the Blackhawk into a dive helps your torpedo's accuracy.

After you drop a torpedo, the screen changes to a sideways view of the *Bismarck* with your torpedo bearing down on it. You will either see it explode and sink, or get a message that you missed. Either way, your mission is over.

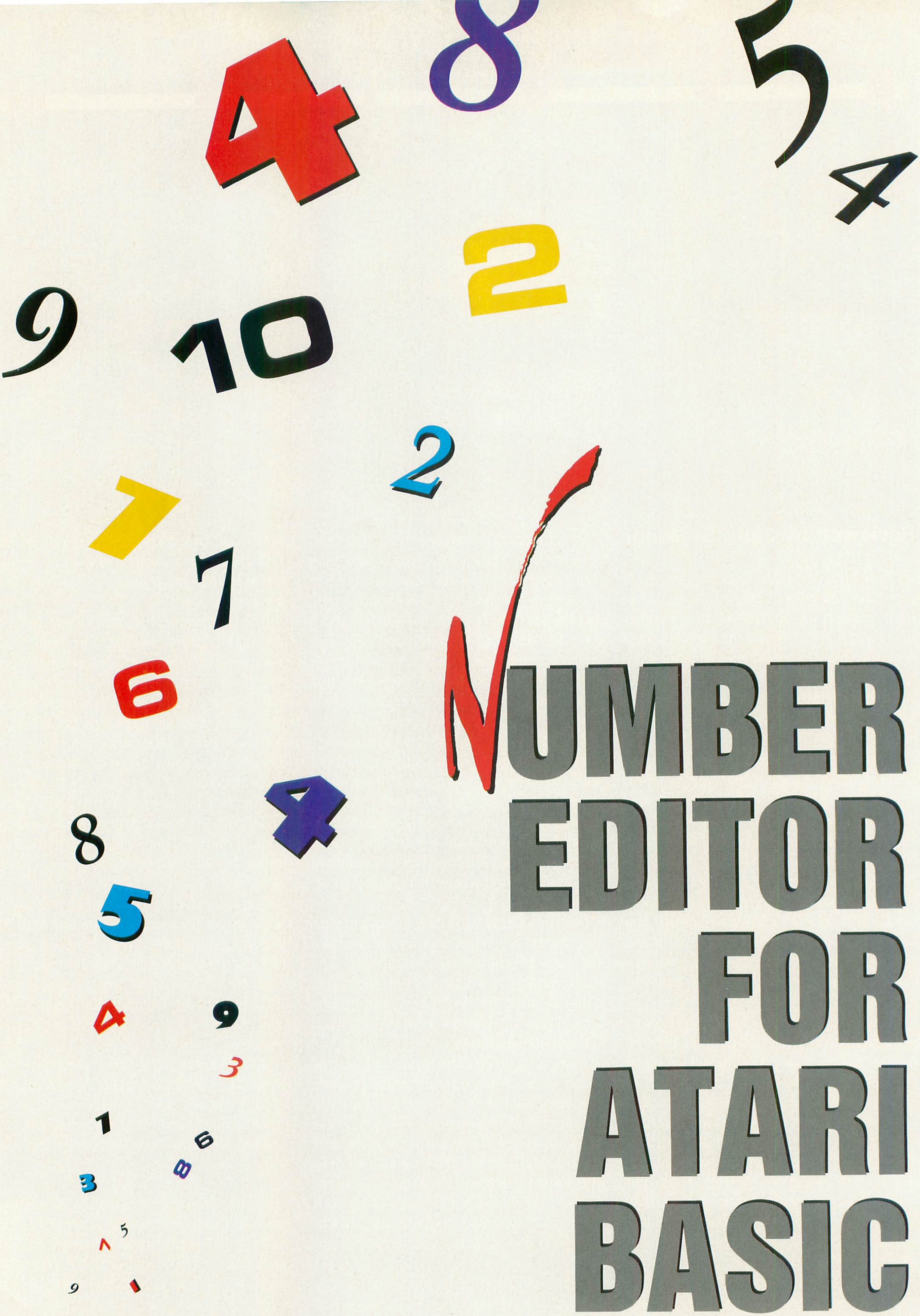
The graphics and sound in Dive Bomber are slightly above average. The game loads with digitized pictures and voices from World War II. When you land on the *Ark Royal*, you roll past the control tower. Floating mines explode in a glorious shower of fire when hit by your guns. Other than the *Bismarck*, though, the enemy ships could have been more detailed.

Dive Bomber comes on two single-sided disks with a clear and detailed instruction booklet. Survival hints and some information on the Blackhawk and *Bismarck* are included in the booklet.

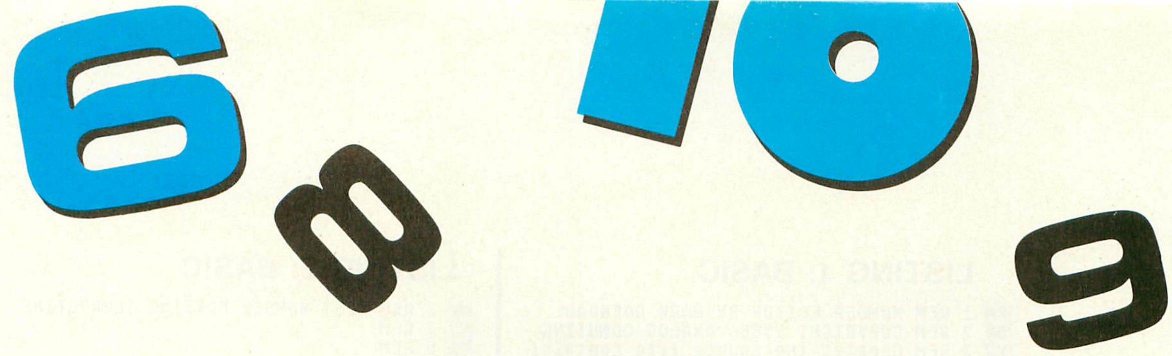
One complaint I have about Dive Bomber is the use of the escape key on the Atari ST to quit a game and return to the main menu. I hit Escape many times when I wanted to hit *1* for the Pilot's screen. My game was ended! Arrgghh! My solution has been to use only the ST's keypad for entering screen numbers.

Dive Bomber is a good game overall. However, it is not a graphic spectacular. It is just a good, solid, interesting naval-air combat simulation. If that's what you are looking for, then Dive Bomber will please you.

John S. Manor is a freelance writer who has had an Atari computer since 1981. His collection of computers now includes an 800, an 800XL, a 130XE and a 520ST. His other interests include astronomy and reading science fiction.



NUMBER EDITOR FOR ATARI BASIC



by **Mark Odendahl**

One of the features missing from Atari BASIC is a PRINT USING statement. In other BASICs, this command formats numeric output so printouts and screen displays are more readable, and is especially useful when printing data in columnar format. The machine-language subroutine described in this article simulates the PRINT USING function.

Listing 1 contains a BASIC program that will create the file NUMED.SRC, which is the BASIC subroutine that places the number editor M/L subroutine into a string variable. Type in the program (using BASIC Editor II to check your work), save a copy to disk and then run it. After a short delay while it reads the data into a string, the program will write the file NUMED.SRC on the disk in Drive 1. NUMED.SRC is then ready to be ENTERed into your own programs.

Listing 2 contains a sample program that demonstrates how to use the M/L subroutine. To see how the subroutine can improve the "look" of numerical output, type in the program, merge in the NUMED.SRC file and then run the resultant program.

Listing 3 is the assembly-language source code. You don't need to type it in. It is included only for those people who are interested in assembly-language programming.

To use the subroutine in your own programs, you must set up a string variable containing an edit mask. An edit mask is simply a set of characters that tells the M/L subroutine how to format the numbers. The edit mask in the sample program is EM$.

The valid symbols for the edit mask are:

Space—Inserts a space into the output. Most effective when used to insert spaces between multiple fields being formatted by the same print statement.

Percent sign (%)—Inserts a percent sign into the output.

Slash (/)—Inserts a slash into the output (useful for dates).

Period (.)—Inserts a decimal point into the output, even if none exists in the number being formatted.

Letter zee (Z)—Replaces leading zeros with blanks, otherwise inserts a digit from the number into output. Not valid to the right of the period in the edit mask.

*Dollar sign ($)*—Replaces leading zeros with blanks, except the leftmost leading zero which is replaced with "$". Not valid to the right of the period in the edit mask.

Number sign (#)—Forces leading or trailing zeros to print.

Comma (,)—Inserts a comma in the output, only if a digit will be printed to the left of this position.

Minus sign (-)—If the number is negative, a minus sign will be inserted at this position, else a space will be inserted.

Guidelines for using the Number Editor

A. The subroutine call (X=USR) and the PRINT statement containing the edit mask must be on the same line, with the PRINT statement immediately following the USR statement.

B. The edit mask must be specified before any of the variables to be printed.

C. Any number of variables can be formatted in the same statement, but there is a limit of 128 characters of output. The number of output characters is computed by multiplying the length of the edit mask by the number of variables and constants being formatted.

D. Only numeric variables and numeric constants can be edited. Strings, arrays and expressions are not allowed. Exception: A minus sign may be specified before a variable or constant.

E. The number of an open IOCB can be specified in the usual manner; that is: 100 X=USR(ADR(SBR$)):?#2,EM$,A,B. The IOCB number may be contained in a numeric variable. Use a comma, not a semicolon, after the IOCB specification.

F. If a number is negative, a minus sign will not print unless a minus sign is explicitly stated in the edit mask. I prefer to have the minus sign print after the number.

G. If the edit mask contains fewer digits (total of $, # and Z) before the decimal point than the number does, the output will be filled with * for the length of the edit mask. Digits after the decimal point are dropped if the edit mask does not allow for them.

H. No rounding is done by the subroutine.

I. If your printer is not set to auto-line feed, remove Line 27080 from the source subroutine (NUMED.SRC). This will cause the M/L subroutine to send line-feed commands to the printer.

While this subroutine is limited in its scope, it will produce much neater printouts than a regular BASIC PRINT. Screen displays can be made more orderly with this subroutine, and printouts with long columns of data will be much more readable. Some experimenting will enable you to learn what combinations of characters in the edit mask will meet your needs.

Mark Odendahl is the supervisor of the technical support group for a major airline in Minneapolis. He has been programming mainframes for 15 years and micros for five years. He has a strong interest in the development of productivity tools for programmers.

NUMBER EDITOR FOR ATARI BASIC

LISTING 1: BASIC

```
FM 1 REM NUMBER EDITOR BY MARK ODENDAHL
DR 2 REM COPYRIGHT 1988  ANALOG COMUTING
YU 3 REM Creates the source file containi
   ng the number editor subroutine string
OM 10 DIM A$(999),Q(50),E(50):FOR I=0 TO
   50:Q(I)=0:E(I)=0:NEXT I
RM 45 Q=0:E=0
AG 50 FOR I=1 TO 467:READ A:IF A=34 THEN
   A=32:Q=Q+1:Q(Q)=I
DG 55 IF A=155 THEN A=32:E=E+1:E(E)=I
PQ 60 A$(I,I)=CHR$(A):NEXT I
IP 75 POKE 766,1
BW 200 OPEN #2,8,0,"D:NUMED.SRC":LN=27000
SI 205 ? #2;LN;" DIM SBR$(467)":LN=LN+10
YC 210 FOR I=1 TO 6:B=I*90-89:E=I*90:IF I
   =6 THEN E=467
HH 220 ? #2;LN;" SBR$(";B;",";E;")=";CHR$
   (34);A$(B,E);CHR$(34)
CO 230 LN=LN+10:NEXT I
GV 250 FOR I=1 TO 50:IF Q(I)>0 THEN C=Q(I
   ):? #2;LN;" SBR$(";C;",";C;")=CHR$(34)
   ":LN=LN+10
OR 255 IF E(I)>0 THEN C=E(I):? #2;LN;" SB
   R$(";C;",";C;")=CHR$(155)":LN=LN+10
GE 260 NEXT I
YL 270 ? #2;LN;" REM REMOVE NEXT LINE IF
   YOUR PRINTER DOES NOT HAVE AUTO LINE F
   EED":LN=LN+10
YT 280 ? #2;LN;" SBR$(433,433)=CHR$(234)"
   :LN=LN+10
KS 295 ? #2;LN;" RETURN"
OB 300 CLOSE #2:POKE 766,0:END
LL 1000 DATA 169,253,133,207,169,3,133,20
   8,162,0,134,215,232,134,218,164,167,17
   7,138,133,167,200,132,212,208
EE 1010 DATA 126,198,218,200,152,113,138,
   133,212,200,152,101,138,133,205,165,13
   9,105,0,133,206,136,177,138,16
FF 1020 DATA 77,240,229,41,127,162,3,10,3
   8,204,202,208,250,24,105,2,144,2,230,2
   04,24,117,134,133,203
FT 1030 DATA 165,204,117,135,133,204,160,
   0,198,218,240,24,48,104,200,177,203,72
   ,16,57,152,105,6,133,212
TC 1040 DATA 200,152,162,4,208,220,104,23
   0,218,208,41,24,177,203,101,140,133,20
   5,200,177,203,101,141,133,206
UN 1050 DATA 200,177,203,168,136,132,213,
   162,0,177,205,201,46,240,10,201,35,208
   ,1,232,136,16,242,162,0
WJ 1060 DATA 134,215,160,0,132,204,132,21
   4,132,220,164,212,200,132,212,177,138,
   48,140,201,18,240,243,201,15
VF 1070 DATA 240,130,144,172,201,28,240,1
   79,144,100,230,220,208,227,144,217,208
   ,215,177,203,133,219,240,17,166
KN 1080 DATA 220,240,4,73,128,133,219,41,
   127,56,233,63,10,24,101,215,133,217,20
   0,177,203,41,240,133,221
UP 1090 DATA 165,213,133,216,164,216,177,
   205,201,32,240,12,201,37,240,8,201,46,
   240,4,201,47,208,38,164
BX 1100 DATA 216,145,207,198,216,16,228,1
   64,217,48,11,240,9,169,42,164,213,145,
   207,136,16,251,56,165,207
AF 1110 DATA 101,213,133,207,144,164,230,
   208,208,160,144,102,201,44,208,25,166,
   217,48,2,208,208,136,177,205
PW 1120 DATA 201,35,208,4,169,44,208,197,
   201,90,208,69,169,32,208,189,201,45,20
   8,6,166,219,48,181,16
KU 1130 DATA 242,164,217,48,43,240,41,198
   ,217,192,11,176,31,192,2,208,6,165,221
   ,208,2,198,217,136,152
JR 1140 DATA 74,168,200,177,203,144,4,41,
   15,176,4,74,74,74,74,9,48,208,136,169,
   48,208,132,201,35
BI 1150 DATA 240,248,201,90,240,187,164,2
   14,208,183,230,214,208,234,168,104,10,
   10,10,10,170,169,11,157,66
ST 1160 DATA 3,152,160,0,201,21,240,30,18
   9,64,3,201,9,240,4,201,6,208,9,169,9,1
   57,66,3,169
OI 1170 DATA 155,208,7,169,10,145,207,200
   ,169,13,145,207,200,152,24,101,207,56,
   233,253,157,72,3,169,0
IA 1180 DATA 157,73,3,169,253,157,68,3,16
   9,3,157,69,3,32,86,228,96
```

LISTING 2: BASIC

```
MK 3 REM TEST NUMBER EDITING SUBROUTINE
NJ 4 REM
NK 5 REM
CM 10 DIM EM$(25)
PJ 20 GOSUB 27000
BU 97 REM
BW 98 REM
BY 99 REM
EO 100 EM$="ZZZZZ,ZZZ.##-  ":GOSUB 500
IC 110 EM$="$$$$$,$$$.##-  ":GOSUB 500
QC 120 EM$="#####,###.##  ":GOSUB 500
UV 130 EM$="ZZZZZZZZZ.####-  ":GOSUB 500
FA 150 ? "DATE EXAMPLE ";:X=USR(ADR(SBR$)
   ):? "##/##/##  ",92686,122586
ZO 160 ? "OVERFLOW EXAMPLE ";:X=USR(ADR(S
   BR$)):? "###",1000
NR 200 END
SC 497 REM
SF 498 REM
SI 499 REM
CE 500 RESTORE
IM 510 FOR I=1 TO 10:READ A,B
SQ 520 X=USR(ADR(SBR$)):? EM$,A,B
GB 530 NEXT I
JG 540 ? :? "PRESS ANY KEY TO CONTINUE":O
   PEN #3,4,0,"K:":GET #3,K:CLOSE #3
ZL 550 RETURN
BC 1000 DATA 100,.509
DE 1010 DATA 4567.34,1234567
AK 1020 DATA 23,.56789
YN 1030 DATA 23.456,98.6
UV 1040 DATA 3.17,.0006
PB 1050 DATA 24,190
LU 1060 DATA 17.5,0
GM 1070 DATA -1,25
QG 1080 DATA 3090,3081
AD 1090 DATA 800,1040
```

LISTING 3: ASSEMBLY

```
10          .SET 1,10
20          .TAB 8,12,30
30          .TITLE "NUMBER EDITOR"
40 *
50 *  File name is NUMED.M65
60 *
70 * This is a subroutine called from
80 * BASIC. Its purpose is to
90 * simulate the BASIC PRINT USING
0100 * statement.
0110 * SAMPLE CALL:
0120 *  X=USR(ADR(SBR$)):PRINT E$,A,B
0130 *
0140 * The routine in SBR$ looks at
0150 * E$ to determine how to format
0160 * A and B.
0170 *
0180 * E$ contains the edit mask.
0190 * A & B in this example are the
0200 * variables to be printed. There
0210 * may be one or more variables
0220 * and/or numeric constants.
0230 * Strings other than the edit
0240 * mask are not allowed.
0250 *
0260 * Expressions are not allowed,
0270 * except that a variable or
0280 * numeric constant may be
0290 * preceded by a minus sign.
0300 *
0310 * The edit mask may be preceded
0320 * by an IOCB specification,
0330 * using either a variable or
0340 * constant.
0350 *
0360 * All parameters must be
0370 * separated by commas; the line
0380 * may end with a semi-colon to
0390 * suppress the line feed.
0400 *
0410 * See the BASIC article and
0420 * documentation for details on
0430 * the uses of this subroutine.
0440 * Zero page usage
0450 * $CB and $CC are used as an
0460 * un-named pointer.
```

```
0470 *
0480     .ORG $CD
0490 EDIT.MASK .DS 1
0500 EDIT.MASK.LO .= EDIT.MASK
0510 EDIT.MASK.HI .DS 1
0520 OUTPUT .DS 1
0530 OUTPUT.LO .= OUTPUT
0540 OUTPUT.HI .DS 1
0550     .ORG $D4
0560 SAVEY .DS 1
0570 EDIT.LENGTH .DS 1
0580 FLOAT .DS 1
0590 EDIT.DECIMALS .DS 1
0600 EDIT.PTR .DS 1
0610 NUMBER.PTR .DS 1
0620 STATE .DS 1
0630 SIGN .DS 1
0640 MINUS.FOUND .DS 1
0650 FIRST.DIGIT.VALUE .DS 1
0660 *
0670 * Printer related equates.
0680 *
0690 CASSETTE.BUFFER .= $03FD
0700 PUT.CHARACTERS .= 11
0710 PUT.RECORD .= 9
0720 CIO.ROUTINE .= $E456
0730 LINE.FEED .= 10
0740 CARRIAGE.RETURN .= 13
0750 EDITOR.DEVICE .= 6
0760 SCREEN.DEVICE .= 9
0770 *
0780 * ATASCII characters
0790 *
0800 SPACE .= 32
0810 NUMBER.SIGN .= '#
0820 DOLLAR.SIGN .= '$
0830 PERCENT .= '%
0840 ASTERISK .= '*
0850 COMMA .= ',
0860 MINUS.SIGN .= '-
0870 DECIMAL.POINT .= '.
0880 SLASH .= '/
0890 ZERO .= '0
0900 Z   .=  'Z
0910 RETURN .= 155
0920 *
0930 * BASIC token values
0940 *
0950 STR.CONSTANT .= 15
0960 SEPARATOR .= 18
0970 SEMI.COLON .= 21
0980 IOCB.IND .= 28
0990 MINUS .= 54
1000     .ORG $8000
1010 * Normally, subroutines pull the
1020 * number of arguments off the
1030 * stack; however, we are going
1040 * to use the current stack
1050 * element as the IOCB number
1060 * for printing.  Since this
1070 * subroutine has no parameters
1080 * passed to it, we are
1090 * defaulting to IOCB #0, the
1100 * Screen Editor.
1110 *
1120 * No error checking is done to
1130 * verify that the number of
1140 * arguments is zero.
1150 *
1160 * Use the Cassette Buffer as the
1170 * Print Buffer.
1180 *
1190     LDA # <CASSETTE.BUFFER
1200     STA OUTPUT.LO
1210     LDA # >CASSETTE.BUFFER
1220     STA OUTPUT.HI
1230 *
1240 * Initialize work areas.
1250 *
1260     LDX #0
1270     STX EDIT.DECIMALS
1280     INX
1290     STX STATE
1300 *
1310 * $8A, $8B (STMCUR) points to
1320 * current BASIC line.
1330 * $A7 contains the displacement
1340 * of the PRINT statement.
1350 *
1360 * To use Y as an offset, load
1370 * $A7.  Get the displacement of
1380 * the next BASIC instruction,
1390 * and store this value back
1400 * into $A7, causing BASIC to
1410 * think he has executed the
1420 * PRINT statement.
1430 *
1440     LDY $A7
1450     LDA ($8A),Y
1460     STA $A7
1470 *
1480 * Increment Y to point to the
1490 * PRINT statement, save this
1500 * value, and let's go do the
1510 * printing ourselves.
1520 *
1530     INY
1540     STY SAVEY
1550     BNE POINT.TO.NEXT
1560 *
1570 STRING.CONSTANT .= *
1580 *
1590 * Assume this is edit mask,
1600 * since this is the only string
1610 * constant allowed.
1620 *
1630 * Set state to 0.
1640 * Increment Y to point to the
1650 * string length, then move this
1660 * pointer to A. Add in the
1670 * string length, and A now
1680 * points to the last byte of
1690 * string.  Save this value.
1700 *
1710     DEC STATE
1720     INY
1730     TYA
1740     ADC ($8A),Y
1750     STA SAVEY
1760 *
1770 * Point to first byte of string.
1780 * Move to A, add in the base,
1790 * and we have the base pointer
1800 * to the edit mask.
1810 *
1820     INY
1830     TYA
1840     ADC $8A
1850     STA EDIT.MASK.LO
1860     LDA $8B
1870     ADC #0
1880     STA EDIT.MASK.HI
1890 *
1900 * Now point Y back to length,
1910 * retrieve the length, and go
1920 * examine the edit mask.
1930 *
1940     DEY
1950     LDA ($8A),Y
1960     BPL GET.MASK.LENGTH
1970 *
1980 * Continue long branch
1990 *
2000 STRING.CONSTANT.A .= *
2010     BEQ STRING.CONSTANT
2020 *
2030 *
2040 VARIABLE.TOKEN .= *
2050 * Turn off hi byte of token #,
2060 * then shift 3 times to
2070 * multiply by 8 and get offset
2080 * in A and $CC.
2090 *
2100     AND #$7F
2110     LDX #3
2120 VAR.LOOP .= *
2130     ASL A
2140     ROL $CC
2150     DEX
2160     BNE VAR.LOOP
2170 *
2180 * Now add 2 to point to 3rd byte
2190 * of variable entry in table.
2200 * This will be the offset for
2210 * the edit mask or the exponent
2220 * byte for a variable.
2230 *
2240     CLC
2250     ADC #2
2260     BCC POINT.TO.VALUE
2270     INC $CC
2280 *
2290 *
2300 POINT.TO.VALUE .= *
2310 *
2320 * Offset is in A (lo) and $CC
2330 * (hi). Now add in base. If we
2340 * are processing a variable
2350 * token, the above code left
2360 * X=0, so the base is VVTP
2370 * ($86).
2380 * If the token is a numeric
2390 * constant, then the base is
2400 * STMCUR ($8A), and that
2410 * routine set X=4.
2420 *
2430     CLC
2440     ADC $86,X
2450     STA $CB
2460     LDA $CC
2470     ADC $87,X
2480     STA $CC
2490     LDY #0
2500 *
2510 * At this point, Y=0 to be used
2520 * as an offset.
2530 * $CB,$CC is pointing to either
2540 * the displacement of the
2550 * string value for the edit
2560 * mask, or to the exponent byte
2570 * of the numeric variable or
2580 * numeric constant.
2590 *
2600 * Now check the STATE to see
2610 * whether this should be an
2620 * IOCB number, the edit mask,
2630 * or a number to format.
2640 *
2650     DEC STATE
2660     BEQ FIND.EDIT.MASK
2670     BMI FORMAT.NUMBER
2680 *
2690 * The number is the IOCB.
2700 * Make Y point to byte after
2710 * exponent, since a valid IOCB
2720 * should not be large.
2730 * Get that byte and put on the
2740 * stack for later retrieval.
2750 * Then go process next token.
2760 *
2770 SAVE.IOCB .= *
2780     INY
2790     LDA ($CB),Y
2800     PHA
2810     BPL POINT.TO.NEXT
2820 *
2830 NUMERIC.CONSTANT.TOKEN .= *
2840 *
2850 * Note that CARRY is clear on
2860 * entry to this routine.
2870 *
2880 * A numeric constant was found.
2890 * Transfer the offset value
2900 * into A and add 6 to point to
2910 * the last byte of the
2920 * constant. Then save this
2930 * pointer.
2940 *
2950     TYA
2960     ADC #6
2970     STA SAVEY
2980 *
2990 * Increment Y to point to
3000 * exponent, transfer to A, then
3010 * set X to point to base.
3020 * Go add in the base address.
3030 *
3040     INY
3050     TYA
3060     LDX #4
3070     BNE POINT.TO.VALUE
3080 *
3090 IOCB.TOKEN .= *
3100 *
3110 * The # token was found,
3120 * indicating the presence of an
3130 * IOCB specification.
```

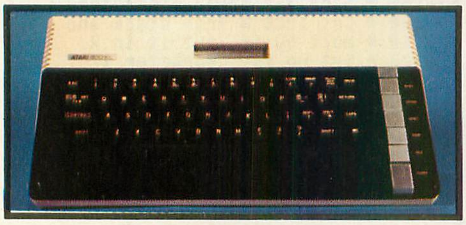
NUMBER EDITOR FOR ATARI BASIC

```
3140 *
3150 * Discard the default from the
3160 * stack (the new value will be
3170 * added when we look at the
3180 * next token), set the STATE to
3190 * to indicate next token is
3200 * an IOcB spec, and then go
3210 * check out the next token.
3220 *
3230      PLA
3240      INC STATE
3250      BNE POINT.TO.NEXT
3260 FIND.EDIT.MASK .= *
3270 *
3280 * A variable was found and state
3290 * was decremented to zero, so
3300 * we assume this is the edit
3310 * mask.  $CB, $CC is pointing
3320 * to offset of value, so add in
3330 * base at STARP ($8C, $8B).
3340 * Then count the number of digit
3350 * positions (#) after the
3360 * decimal point in the edit
3370 * mask.
3380 *
3390      CLC
3400      LDA ($CB),Y
3410      ADC $8C
3420      STA EDIT.MASK.LO
3430      INY
3440      LDA ($CB),Y
3450      ADC $8D
3460      STA EDIT.MASK.HI
3470      INY
3480 *
3490 * Get the string length and
3500 * subtract 1 to convert to the
3510 * max offset value.
3520 *
3530      LDA ($CB),Y
3540 GET.MASK.LENGTH .= *
3550      TAY
3560      DEY
3570      STY EDIT.LENGTH
3580 *
3590 * Count number of decimal places
3600 * in the edit mask.
3610 *
3620      LDX #0
3630 FIND.DECIMALS .= *
3640      LDA (EDIT.MASK),Y
3650      CMP #DECIMAL.POINT
3660      BEQ FOUND.DP
3670      CMP #NUMBER.SIGN
3680      BNE NOT.DIGIT
3690 *
3700 * Increment X only when # found.
3710 *
3720      INX
3730 NOT.DIGIT .= *
3740      DEY
3750      BPL FIND.DECIMALS
3760 *
3770 * If we fall through here, then
3780 * no decimal point was found,
3790 * so there are no decimal
3800 * places.  Set X to indicate
3810 * this.
3820 *
3830      LDX #0
3840 FOUND.DP .= *
3850      STX EDIT.DECIMALS
3860 POINT.TO.NEXT .= *
3870 *
3880 * Examine the token to see what
3890 *
3900 * First, init some variables.
3910 * Clear the high byte of a work
3920 * pointer, indicate a floating
3930 * $, and reset the minus found
3940 * indicator.
3950 *
3960 *
3970      LDY #0
3980      STY $CC
3990      STY FLOAT
4000      STY MINUS.FOUND
4010 CHECK.TOKEN .= *
4020 *
4030 * This is the loop that examines
4040 * each token. First, get the
4050 * last pointer to the BASIC
4060 * line, increment it to point to
4070 * a new token, save this value,
4080 * and get the token.
4090 *
4100      LDY SAVEY
4110      INY
4120      STY SAVEY
4130      LDA ($8A),Y
4140      BMI VARIABLE.TOKEN
4150 *
4160 * If this is a comma, we can
4170 * ignore.
4180 *
4190      CMP #SEPARATOR
4200      BEQ CHECK.TOKEN
4210      CMP #STR.CONSTANT
4220      BEQ STRING.CONSTANT.A
4230 *
4240 * If carry clear, then value
4250 * must be 14.
4260 *
4270      BCC NUMERIC.CONSTANT.TOKEN
4280      CMP #IOCB.IND
4290      BEQ IOCB.TOKEN
4300 *
4310 * If carry clear here, then
4320 * token is EOS, EOL, or ;
4330 *
4340      BCC DONE.EDITING.A
4350 MINUS.TOKEN .= *
4360      INC MINUS.FOUND
4370      BNE CHECK.TOKEN
4380 *
4390 * Continue a long branch
4400 *
4410 POINT.TO.NEXT.A .= *
4420      BCC POINT.TO.NEXT
4430      BNE POINT.TO.NEXT
4440 FORMAT.NUMBER .= *
4450 *
4460 * On entry Y=0, $CB,$CC points
4470 *
4480      LDA ($CB),Y
4490      STA SIGN
4500      BEQ A5
4510 *
4520 * If minus sign was found,
4530 * reverse the sign and save
4540 * this result.
4550 *
4560      LDX MINUS.FOUND
4570      BEQ A2
4580      EOR #$80
4590      STA SIGN
4600 A2   .= *
4610 *
4620 * The number will be formatted
4630 * from right to left.  To align
4640 * the decimal points, subtract
4650 * $3F out of the exponent byte,
4660 * mult by 2, and add in the
4670 * number of decimal places in
4680 * the edit mask.  The result is
4690 * the pointer to the first byte
4700 * of the number that should
4710 * print in the right-most
4720 * position.
4730 *
4740 * If the result is < 0 or > 10,
4750 * then zeros will be forced to
4760 * print, depending on the edit
4770 * mask.
4780 *
4790      AND #$7F
4800      SEC
4810      SBC #$3F
4820      ASL A
4830      CLC
4840      ADC EDIT.DECIMALS
4850 A5   .= *
4860      STA NUMBER.PTR
4870 *
4880 * Get the first digit and save
4890 * it.  This will be used later
4900 * to determine if the first
4910 * digit is an insignificant
4920 * zero.
4930 *
4940      INY
4950      LDA ($CB),Y
4960      AND #$F0
4970      STA FIRST.DIGIT.VALUE
4980      LDA EDIT.LENGTH
4990      STA EDIT.PTR
5000 DIGIT.LOOP .= *
5010 *
5020 * Process the number for the
5030 * length of the edit mask.
5040 *
5050      LDY EDIT.PTR
```

```
5060      LDA (EDIT.MASK),Y
5070 *
5080 * Space, percent, decimal point
5090 * and slash print as-is.
5100 *
5110      CMP #SPACE
5120 *
5130      BEQ D10
5140      CMP #PERCENT
5150      BEQ D10
5160      CMP #DECIMAL.POINT
5170      BEQ D10
5180      CMP #SLASH
5190      BNE D20
5200 D10 .= *
5210      LDY EDIT.PTR
5220      STA (OUTPUT),Y
5230 *
5240 * Decrement pointer to the edit
5250 * mask.  If non-negative, keep
5260 * looping.
5270 *
5280      DEC EDIT.PTR
5290      BPL DIGIT.LOOP
5300 *
5310 * If number ptr > 0, then an
5320 * overflow occurred.  Place all
5330 * asterisks in output area.
5340 *
5350      LDY NUMBER.PTR
5360      BMI INCREMENT.OUTPUT.BASE
5370      BEQ INCREMENT.OUTPUT.BASE
5380      LDA #ASTERISK
5390      LDY EDIT.LENGTH
5400 OVERFLOW.LOOP .= *
5410      STA (OUTPUT),Y
5420      DEY
5430      BPL OVERFLOW.LOOP
5440 *
5450 * We are done with this number,
5460 * so we will reset the output
5470 * base pointer and go get the
5480 * next token.
5490 *
5500 * We have to add the length of
5510 * the edit mask to the current
5520 * output base to get the new
5530 * value.  Since the edit mask
5540 * length was converted to an
5550 * offset by subtracting one, we
5560 * will set the carry first to
5570 * add in 1 extra.
5580 *
5590 INCREMENT.OUTPUT.BASE .= *
5600      SEC
5610      LDA OUTPUT.LO
5620      ADC EDIT.LENGTH
5630      STA OUTPUT.LO
5640      BCC POINT.TO.NEXT.A
5650      INC OUTPUT.HI
5660 *
5670 * This will always branch.
5680 *
5690      BNE POINT.TO.NEXT.A
5700 *
5710 * Continue a long branch.
5720 *
5730 DONE.EDITING.A .= *
5740      BCC DONE.EDITING
5750 D20 .= *
5760      CMP #COMMA
5770      BNE D30
5780 *
5790 * If comma in the edit mask,
5800 * and there are more digits to
5810 * print, then insert the comma.
5820 *
5830 * If there are no digits to
5840 * print then:
5850 * Print a comma if the next
5860 * position will force a zero to
5870 * print (edit mask = #).
5880 * Print a space if the next
5890 * position will zero suppress
5900 * (edit mask = Z).
5910 * If neither of the two above,
5920 * assume $ and print it, then
5930 * turn off the float indicator.
5940 *
5950      LDX NUMBER.PTR
5960      BMI D20A
5970      BNE D10
5980 D20A .= *
5990      DEY
6000      LDA (EDIT.MASK),Y
6010      CMP #NUMBER.SIGN
6020      BNE D21
6030      LDA #COMMA
6040      BNE D10
6050 D21 .= *
6060      CMP #Z
6070      BNE D350
6080 D22 .= *
6090      LDA #SPACE
6100      BNE D10
6110 D30 .= *
6120      CMP #MINUS.SIGN
6130      BNE D35
6140 *
6150 * If number is minus, print the
6160 * minus sign, else print space.
6170 *
6180      LDX SIGN
6190      BMI D10
6200      BPL D22
6210 D35 .= *
6220 *
6230 * Check the ptr to the number.
6240 * If it is negative, force zero.
6250 * If it is zero, check edit mask
6260 * for #,$, or Z.
6270 *
6280      LDY NUMBER.PTR
6290      BMI D300
6300      BEQ D300
6310      DEC NUMBER.PTR
6320 *
6330 * If pointer out of range, then
6340 * we have more digits in edit
6350 * mask then in the number.
6360 * Force to extras to zeros.
6370 *
6380      CPY #11
6390      BCS D200
6400 *
6410 * If processing second digit,
6420 * check if first digit is an
6430 * insignificant zero.
6440 * If it is, decrement the
6450 * counter to bypass.
6460 *
6470      CPY #2
6480      BNE D40
6490      LDA FIRST.DIGIT.VALUE
6500      BNE D40
6510      DEC NUMBER.PTR
6520 D40 .= *
6530 *
6540 * Convert number pointer to an
6550 * offset into value. Then
6560 * increment by 1 to adjust for
6570 * the exponent byte.
6580 *
6590      DEY
6600      TYA
6610      LSR A
6620      TAY
6630      INY
6640      LDA ($CB),Y
6650 *
6660 * Check shift result.
6670 *
6680      BCC LEFT.DIGIT
6690 RIGHT.DIGIT .= *
6700      AND #$0F
6710      BCS D45
6720 LEFT.DIGIT .= *
6730 *
6740 * Move hi nibble to lo nibble
6750 * for processing.
6760 *
6770      LSR A
6780      LSR A
6790      LSR A
6800      LSR A
6810 D45 .= *
6820      ORA #$30
6830 D46 .= *
6840      BNE D10
6850 *
6860 *
6870 D200 .= *
6880      LDA #ZERO
6890      BNE D10
6900 *
6910 *
6920 D300 .= *
6930      CMP #NUMBER.SIGN
6940      BEQ D200
6950      CMP #Z
6960      BEQ D22
6970 *
6980 * Process dollar sign.
6990 *
7000 D350 .= *
7010      LDY FLOAT
7020      BNE D22
7030      INC FLOAT
7040      BNE D46
7050 *
7060 *
7070 DONE.EDITING .= *
7080 *
7090 * Save the token for later.
7100 * Get IOCB # from stack and
7110 * mult by 16, put it in X.
7120 * Set up for character I/O.
7130 *
7140      TAY
7150      PLA
7160      ASL A
7170      ASL A
7180      ASL A
7190      ASL A
7200      TAX
7210      LDA #PUT.CHARACTERS
7220      STA $0342,X
7230 *
7240 * Get token, set Y as output
7250 * offset, then test token to
7260 * determine if the line feed is
7270 * suppressed (token = ;).
7280 *
7290      TYA
7300      LDY #0
7310      CMP #SEMI.COLON
7320      BEQ E10
7330 *
7340 * Now check for Screen or Editor
7350 * device open for the requested
7360 * IOCB. If so, use record I/O
7370 * and forget about line feeds
7380 * and carriage returns (but add
7390 * the ATASCII return to output)
7400 *
7410      LDA $0340,X
7420      CMP #SCREEN.DEVICE
7430      BEQ E1
7440      CMP #EDITOR.DEVICE
7450      BNE E3
7460 E1  .= *
7470      LDA #PUT.RECORD
7480      STA $0342,X
7490      LDA #RETURN
7500      BNE E5
7510 E3  .= *
7520 *
7530 * Move printer line feed.
7540 *
7550      LDA #LINE.FEED
7560      STA (OUTPUT),Y
7570 *
7580 * NOP next instruction if your
7590 * printer is set to auto line
7600 * feed when carriage return is
7610 * received.
7620 * This will cause the carriage
7630 * return to overlay the line
7640 * feed character.
7650 *
7660      INY
7670      LDA #CARRIAGE.RETURN
7680 E5  .= *
7690      STA (OUTPUT),Y
7700      INY
7710 E10 .= *
7720 *
7730 * Adjust output buffer pointer
7740 * to account for line feed and
7750 * carriage return characters.
7760 * Then subtract out beginning
7770 * buffer address to get buffer
7780 * length.
7790 *
7800      TYA
7810      CLC
7820      ADC OUTPUT.LO
7830      SEC
7840      SBC # <CASSETTE.BUFFER
7850      STA $0348,X
7860      LDA #0
7870      STA $0349,X
7880 *
7890 * Set up buffer address.
7900 *
7910      LDA # <CASSETTE.BUFFER
7920      STA $0344,X
7930      LDA # >CASSETTE.BUFFER
7940      STA $0345,X
7950      JSR CIO.ROUTINE
7960      RTS
7970      .END
```

When you want to talk Atari

XL/XE HARDWARE



CMO SPECIAL

Atari 800XL	$69⁹⁹
65XE	109.00
130XE	149.00

INTERFACES

ICD	
P:R Connection	59.99
Printer Connection	34.99
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	74.99
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

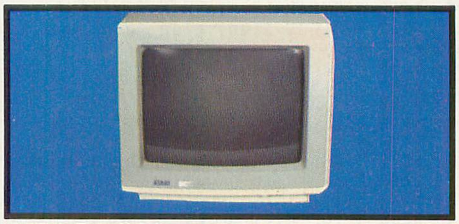
Atari	
SX212 300/1200 (ST)	79.99
XMM301	44.99
Anchor	
VM520 300/1200 ST Dir. Con.	119.00
Avatex	
1200 HC	89.99
2400	169.00
Supra	
2400 Baud XL/XE or ST	169.00
2400 Baud (no software)	139.00

MONITORS

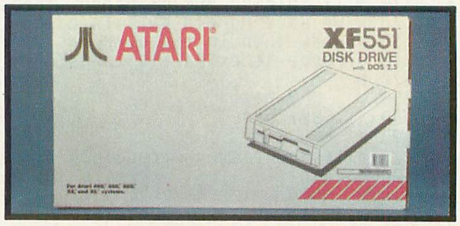
Magnavox	
CM 8505 14" Composite/RGB/TTL	199.00

ST HARDWARE

Call For Current Information On The Entire ST Line!



ATARI SM1224 RGB/Color Monitor	$329
520ST FM RGB/Color System	829.00
SM124 Monochrome Monitor	179.00



CMO PACKAGE EXCLUSIVE

Atari 800XL & XF551 Drive

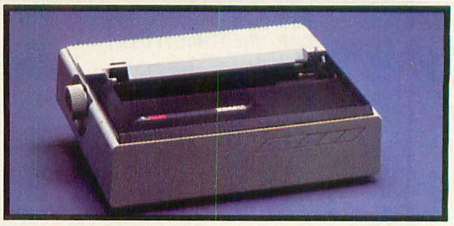
w/5 Undocumented ROMS Asteroids, Defender, Missile Command, QIX, Star Raiders **$259**

DRIVES

Atari	
ST 314 DS/DD	219.00
XF551 Drive (XL/XE)	179.00
SHD204 20 Meg Hard Drive	Call
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	849.00
FA•ST Dual Hard Drives	Call
Indus	
GTS 100 3½" DS/DD (ST)	189.00
GT 1000 5¼" DS/DD (ST)	209.00
GT Drive (XL/XE)	189.00
Supra	
FD-10 10MB Removable Floppy	869.00
30 Meg Hard Drive (ST)	649.00

CALL FOR DISKETTE SPECIAL

PRINTERS



Atari XDM121 LQ (XL/XE)	$189
Atari	
1020 40 col. Color	24.99
1027 Letter Quality XL/XE	89.99
XM-M801 XL/XE Dot Matrix	189.00
Brother	
M-1109 100 cps Dot Matrix	169.00
M-1509 180 cps Dot Matrix	369.00
HR-20 22 cps Daisywheel	349.00
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

SELECT FROM OVER 3000 PRODUCTS

WE SHIP 90% OF ALL ORDERS WITHIN 24 HOURS

CMO

COMPUTER MAIL ORDER

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

SPECIALS XL/XE

#AAB822P 822 Printer Paper	$2.99
#AA14746 T.V. Switch Box	2.99
#AA4010 Tic-Tac-Toe	4.99
#AA4011 Star Raiders	4.99
#AA4012 Missile Command	4.99
#AA4013 Asteroids	4.99
#AA4022 Pac Man	4.99
#AA4025 Defender	4.99
#AA4027 QIX	4.99
#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)
$3⁴⁹ ea or 5 for $14⁹⁹

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	2.99
Anti-Sub (Disk)	2.99
Journey to Planet	2.99
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	34.99

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

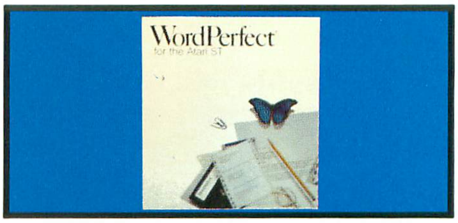
Data East
Speed Buggy **$25⁹⁹**

ST SOFTWARE

Epyx	
Championship Wrestling	19.99
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 U.S.A.

Call: 1-800-233-8950

In Canada call: 1-800-233-8949 All Other Areas 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 and Maryland residents add appropriate 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.

CIRCLE #110 ON READER SERVICE CARD.

B701

Some Call It A Refreshing Change

We Call It The FA▪ST Hard Drive

We named our drive after its swift and aggressive behavior. But it's really not fair to limit this incredible peripheral to just one name.

Call it cool. Cool, calm and collected with its whisper-quiet fan to prevent heated situations.

Call it high-class. With refined style, its sleek design complements your Atari computer system. Quite simply, functional elegance under your monitor that's designed to adjust to your system and lift your sights for easy viewing.

Call it friendly. Our FA▪ST Hard Drive welcomes a host of features like dual DMA ports which invite new devices. Our SCSI expansion is ready when you are. And inside, our drive can handle a partner like no others can. Have the time? The FA▪ST drive does . . . the right time, everytime.

Call it durable. Unwavering dependability from a winning design. Only the best components are found inside our FA▪ST Hard Drive. A full one year warranty and ICD's uncompromised reputation for quality should say it all.

Now, don't let the abundance of features scare you . . . FA▪ST Hard Drives are available in all sizes *and* at prices you can afford. So, to be quite honest, we really don't care what you call our hard drive – as long as you call for it today. And get ready for the best thing that ever happened to your Atari ST.

Call or write for our free catalog today.
1220 Rock Street ▪ Rockford, Illinois 61101 ▪ (815) 968-2228 ▪ MODEM: (815) 968-2229 ▪ FAX: (815) 968-6888

ICD

CIRCLE #101 ON READER SERVICE CARD.

FA▪ST is a trademark of ICD Corporation. *Atari ST is a trademark of Atari Corporation.*