BUILD AN UNINTERRUPTABLE POWER SUPPLY FOR YOUR ATARI COMPUTER!

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

T.M.

FEBRUARY 1989 ISSUE 69

U.S.A. $3.95
CANADA $4.95

TYPE-IN SOFTWARE:
TRIAL BY FIRE
STAR RIDER
VNT EDITOR

EWS:
NT POWER
551 DISK DRIVE

E:
STER MEMORY MAP
ME DESIGN WORKSHOP

FDC 50075

02

0 74369 50075 4

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!

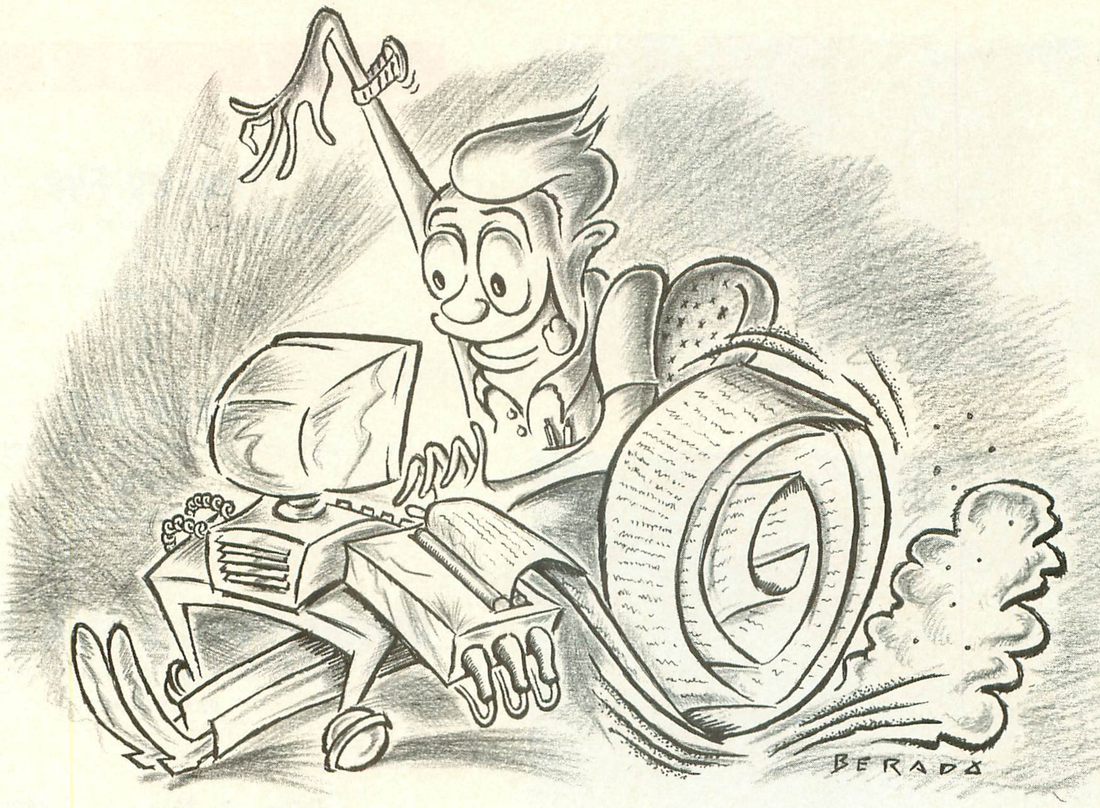
1 YEAR FOR ONLY $28

SAVE **$19** OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

new low price

Editorial



by Clayton Walnum

It's now been over seven years since the first copy of ANALOG Computing hit the stands. Over that period of time, Atari has had its ups and downs, and through it all, ANALOG has been there to provide its readers with the information they needed to get the most from their computers. But after seven years of publishing, it gets very tough to come up with new topics for discussion. Even though the Atari 8-bit computers have had a couple of face-lifts over the years, they are still essentially the same machines they were when they were first released. There is simply nothing new that can be said about them. (We would be delighted if someone out there could send us an article to prove us wrong. That's a challenge!)

It occurred to me that there are a great number of new ANALOG Computing readers—readers that are experiencing Atari computers for the first time. And these new readers have a need for much of the information we Atari veterans consider "old hat." After all, it has been *years* since topics such as player/missile graphics, display lists, page flipping and modified character sets have been covered at a level that the novice programmer could understand. Should we, perhaps, cover these topics again? Write to us and let us know!

I've always envisioned ANALOG's growth as a long line that projected from the past into the future. But with recent developments (or should I say *lack* of developments) it appears to me that maybe ANALOG should start coming full circle, should start providing some of the information that new owners have missed.

One thing that people are constantly asking for is reprints of the early *Boot Camp* columns. This assembly-language tutorial started way back in Issue 13, and is without a doubt the finest series of tutorials of its type ever written. (I can say that without boasting because back then, I was an ANALOG Computing reader just like you, not involved editorially at all. *Boot Camp* was one of ANALOG's features that I most looked forward to each month.) With this issue, Karl Wiegers, our current *Boot Camp* author, is retiring his position. And since most every assembly-language topic has been covered anyway, we have to make a decision. Do we drop *Boot Camp* from our table of contents? Or do we try to replace it?

My suggestion to the publisher was that we dig all the way back to Issue 13 and start reprinting the column. First of all, most of those early issues are out of print, so the initial *Boot Camp* columns are completely unavailable to our readers. Secondly, there is no one I know who can cover the topic better than Tom Hudson, so I don't see any point in having someone rewrite what has already been covered splendidly.

The bottom line is that it's up to you, the readers. Drop us a line and let us know if you'd like to see the early *Boot Camp* articles reprinted. Drop us a line if you *don't* want to see them reprinted. We'll tally the votes and come to a decision based on what you want.

And speaking of making decisions, please take a look at the short survey we've included in this issue. We hope that most of you will take the time to check off your responses and mail it in. If ANALOG is to continue to grow, we have to know what you want. Thanks.

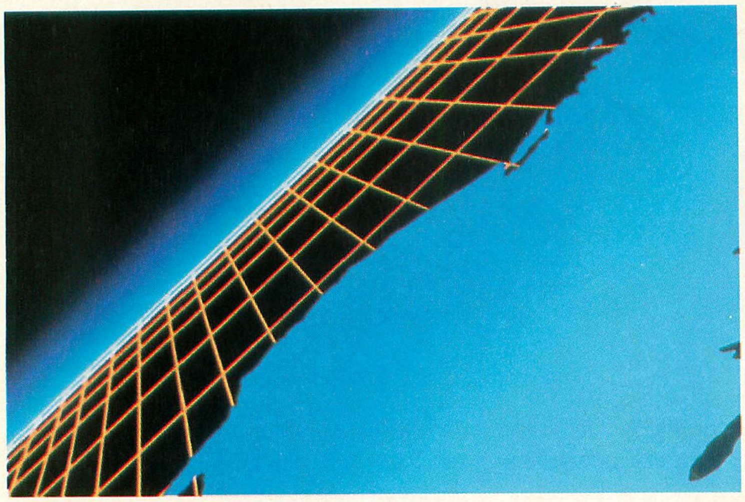
Responses to this editorial should be mailed to:

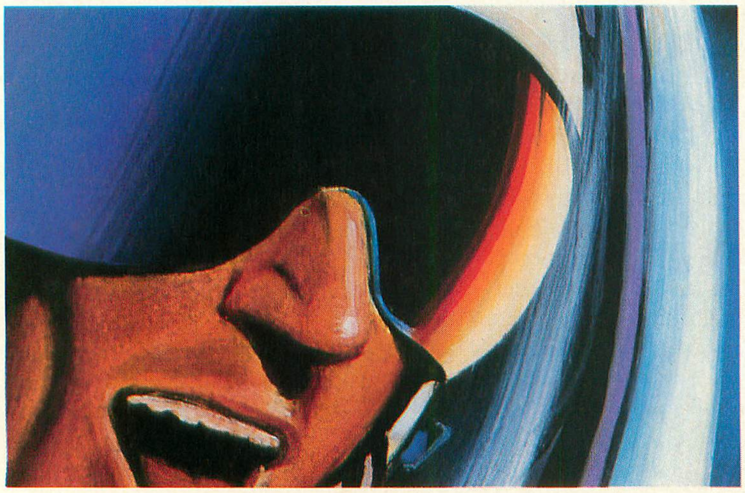
ANALOG Computing
Reader Comment
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

Conte

FEATURES

on page 10


on page 16


on page 74

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1989 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

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; MATTHEW J. W. RATCLIFF; ROBIN SHERER; KARL E. WIEGERS

Entertainment Editor
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

Editorial Assistant
NORMA EDWARDS

Chief Typographer
KLARISSA CURTIS

Typographers
DAVID BUCHANAN
JUDY VILLANUEVA
CLIFFORD LAWSON

Contributors
JACKSON BEEBE
EARL DAVIDSON
GREG KNAUSS
PAUL LAY

Vice President, Production
DONNA HAHNER

Advertising Production Director
JANICE ROSENBLUM

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

Corporate Ad Director
PAULA THORNTON

Subscriptions Director
IRENE GRADSTEIN

Vice President, Sales
JAMES GUSTAFSON

FEBRUARY 1989
ISSUE 69

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**

READER COMMENT

From the Starting Line

After looking through a recent issue of ANALOG Computing, I decided to buy it. When I got it home and looked through it some more, I ordered a subscription.

I like Robin Sherer's *Master Memory Map*. I also like Craig Patchett's column, *Game Design Workshop*. I don't think, however, that Craig should assume that I "already know the basic techniques involved in player/missile graphics, redefining the character set, display lists, and so forth." I don't.

In fact, that was one of the reasons for my not buying your magazine sooner. The articles in past ANALOG magazines assumed that there weren't any beginners out here, only advanced computer wizards. I purchased this magazine and ordered a subscription because it now looks like some help for beginners is going to be offered in your magazine in the coming months.

Until I purchase a MAC/65 assembler or an Action! cartridge, I hope to see more BASIC programs in your magazine. As the above products are expensive, it may be some time before I can get them all.

I hope you will continue to support beginners, as well as computer wizards. I can always cancel my subscription if you don't.

—Michael Pascarelli
Port Richey, FL

Thanks for your subscription, Michael, and we hope that you will find the information you need in future issues of ANALOG. *Yes,* Game Design Workshop *is slanted toward people with some programming experience, although we believe that the information Craig has been presenting each month is complete enough to teach you the basics involved in writing a game, even if you haven't encountered some of the techniques before. However,* GDW *is not a tutorial in BASIC programming. You must be familiar with BASIC to understand the concepts presented.*

One of the stumbling blocks of putting out a magazine like ANALOG *for so many years is that you run out of topics. I doubt there is a single feature of the Atari 8-bit computers that hasn't been covered in depth sometime in the pages of* ANALOG. *Such basic programming techniques as redefining character sets, modifying display lists, performing page flipping and accessing player/missile graphics have all been covered in the past. However, we realize that there are a lot of new Atari owners out there who haven't yet explored the potentials of their machines. In recognition of this fact, we are perfectly willing—if there are enough people interested—to cover these topics again.*

So we must ask a favor of all our readers. Please fill out the short questionnaire found in this issue so that we can accurately assess the direction that we should take ANALOG *in the future. This is your chance to tell us how you feel about the current* ANALOG *and what, if anything, we should do to change it. Please take a couple of moments to respond. Thanks.*

New Address for Abacus

Thank you very much for mentioning Abacus in your August '88 issue of ANALOG Computing in the "ST Notes" column. However, Abacus, a leading publisher of ST products, has a new address. It is 5370 52nd Street SE, Grand Rapids, MI 49508. Phone: (616) 698-0330.

—Jim Oldfield, Jr.
Director of Marketing, Abacus

Another New Address

Omega Soft bought Clearstar Softechnologies in January of last year, and the new address for Clearstar is P.O. Box 140, Harrells, NC 28444. Current products are *Lightspeed C* ($39.95), *The Elite Personal Accountant* ($39.95), *Classy Chassy* ($9.95) and *Time Bomb* ($9.95). Shipping and handling not included.

—John G. Mott
President, Omega Soft

Math's Finer Points

Is there something about the Mandelbrot set that turns the mind to mush?

Actually, James Greco's article (September '88) is in many ways an admirable effort. He handles real and imaginary numbers and squaring complex numbers very well. His explanation of where the different colors come from in plotting the environs of the Mandelbrot set is really fine.

The problem arises from an incorrect formula for the function that generates the Mandelbrot set. The formula, given twice on page 20, should be Z< –Z^2+U. (The square was on the wrong Z in the article.) This notation is analogous to the BASIC line LET Z=Z^2+U. It means "replace Z by its current value squared plus U."

The only trouble is that Z and U are supposed to be complex numbers and BASIC doesn't do complex-number arithmetic on its own. However, if you follow Greco's explanation (page 19), you will see how to square a complex number using high-school algebra. Here is the general formula: (a+bi)^2=a^2–b^2+2abi, where *i* is the famous square root of -1. In the program, the squaring and adding operations are carried out in Line 20. And in spite of the error in the article, the program does the calculation exactly right.

But here is the key point that is missed because of the notation mistake: What makes the Mandelbrot set so complex is that it is not a pure squaring operation. If you square a number, take the result and square it again, and so on, and plot the results the way it is done in plotting the Mandelbrot set, you get a far simpler looking graph. (I think it would be a series of concentric colored rings enclosing a black disk, themselves surrounded by a monochrome background.) The immense complexity comes from combining squaring and adding.

—Jerry Bridgman
Madison, WI

Joytype Correction

Line 1070 of "Joytype" (September '88) should have a semicolon between X$ and the full colon. The omission will not cause an error, but the extra semicolon allows the text files to be read by spelling checkers and other word processors.

—John Pilge
Soquel, CA

8-Bit NEWS

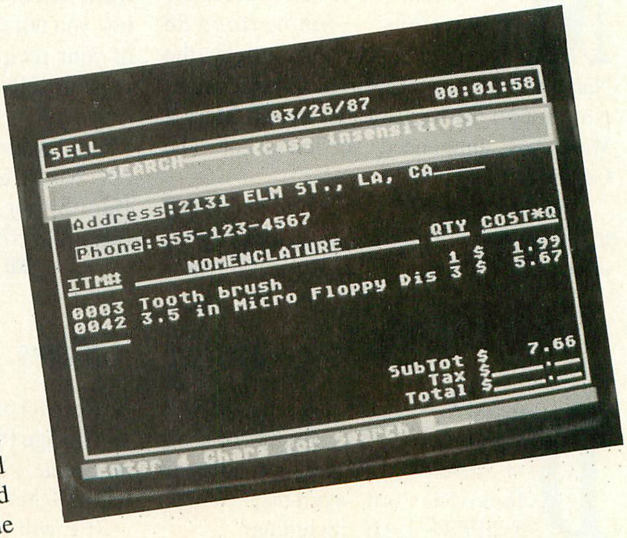
Point-of-Sale System

Xenia Research has released a complete "point-of-sale" (POS) system for use on the Atari XL/XE computers. The Xenia POS will support up to eight computer terminals and three printers, allowing a small business to track sales, inventory, item and employee records, sales tax and other important business data. The software boasts menu-driven operation throughout, as well as the ability to easily generate reports and invoices.

The Xenia POS comes in several versions, depending on your needs. The packages range from $69 for just the POS system itself to $189 for a complete system that'll even read and print bar codes. Each system comes packaged with a free Atari CX85 keypad and 100 two-part invoices, while the higher-priced packages also include a digital bar-code wand.

Xenia Research
P.O. Box 4675
Federal Way, WA 98003
(206) 927-7018

CIRCLE #157 ON READER SERVICE CARD.



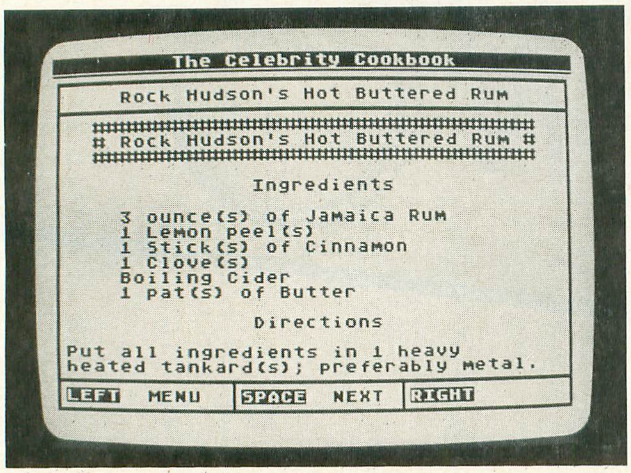
```
                03/26/87        00:01:58
SELL        (case insensitive)
  SEARCH
Address:2131 ELM ST., LA, CA
Phone:555-123-4567            QTY  COST*Q
ITM#     NOMENCLATURE          1 $   1.99
0003 Tooth brush               3 $   5.67
0042 3.5 in Micro Floppy Dis

                        SubTot $     7.66
                           Tax $ ___.__
                         Total $ ___.__
Enter 4 Chars for Search
```

8-Bit NEWS

Virus Perpetrator Convicted

Donald Burleson, an ex-employee of USPA and IRA Co., an insurance and brokerage firm in Fort Worth, Texas, recently discovered that the courts of our land will no longer tolerate the spread of computer viruses. Burleson was convicted of computer sabotage after he planted a computer virus that wiped out 168,000 payroll records on USPA's computers two days after he was fired from his job. David McCown, the prosecutor, believes he is the first lawyer to get a conviction in a computer virus case. Burleson faces up to ten years in prison and up to $5,000 in fines.



```
              The Celebrity Cookbook
        Rock Hudson's Hot Buttered Rum

   ################################
   # Rock Hudson's Hot Buttered Rum #
   ################################

                 Ingredients

      3 ounce(s) of Jamaica Rum
      1 Lemon peel(s)
      1 Stick(s) of Cinnamon
      1 Clove(s)
      Boiling Cider
      1 pat(s) of Butter

                 Directions

 Put all ingredients in 1 heavy
 heated tankard(s); preferably metal.

 LEFT  MENU   SPACE  NEXT  RIGHT
```

Celebrity Cookbook is a Winner

Merrill Ward and Associates' home-catering program, *The Celebrity Cookbook, Vol. 1*, was awarded the prestigious *Innovations.88* computer-software award by the National Consumer Electronics Association. Now hot on the heels of that award, Merrill Ward has announced version 2 of The Celebrity Cookbook.

Version 2 adds a lot more color and speed to the original program. Because the program is written in the innovative GOE (Graphics Operating Environment), it can take advantage of an ST mouse, as well as a joystick, track ball or the keyboard. Version 2 will allow you not only to calculate the serving size of your recipes, but also the cost. The new version sells for $29.95.

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

CIRCLE #159 ON READER SERVICE CARD.

SynFile+ Utilities

Donald R. Seay has just released *SFP*, a group of utility programs for users of *SynFile+*. SFP adds a number of functions not available within SynFile+ itself, including:

★ Print, write to a file, or display on the screen the structure of a SynFile+ file.
★ Alter the values associated with record number and counter fields.
★ Add, change or delete values in look-up tables.
★ Change true/false texts for conditional items.
★ Recover deleted files.
★ Change the justification of any data item.
★ Create, save and produce reports in either list or label format.

SFP will run on a 48K Atari with BASIC and is priced at a reasonable $19.95 (plus $2 shipping).

SFP
4 Forest Drive
Palmyra, VA 22963-2118

CIRCLE #160 ON READER SERVICE CARD.

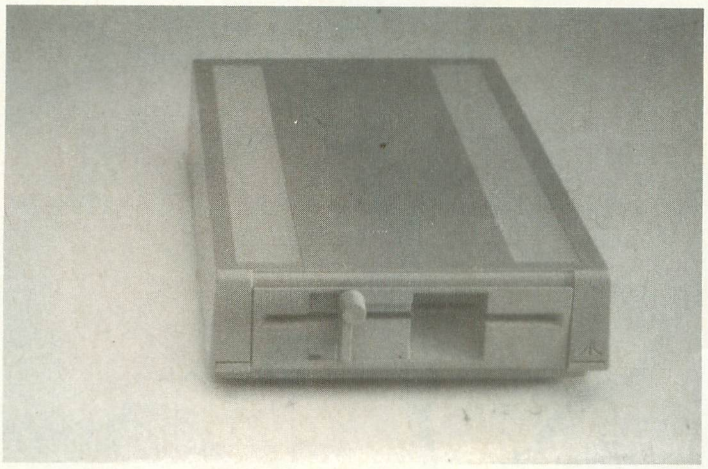
Omega Soft Buys Lightspeed C

Lightspeed C, the popular C program-development language for the 8-bit computers is no longer being sold by Clearstar Softechnologies. Omega Soft, a company that distributes several products for the 8-bit Atari computer, has purchased the rights to sell Lightspeed C exclusively. C is quickly becoming the language of choice of many programmers, and Lightspeed C has been hailed as an exceptional alternative to assembly language on the 8-bit Atari. Its price is $39.95.

Other products available through Omega Soft include the *AL/65 Development System* (a linker-based assembler) and *DISKIO* (a complete disk utility package).

Omega Soft
P.O. Box 139
Harrells, NC 28444
(919) 532-2359

CIRCLE #161 ON READER SERVICE CARD.



Atari XF551 Disk Drive

Atari Corp.
1196 Borregas
Sunnyvale, CA 94806
(408) 745-2000
$299

Reviewed by Matthew J.W. Ratcliff

The Atari XF551 disk-drive is one of the most innovative and confusing products released for the 8-bit computer line in quite some time. It was produced to take the place of the discontinued Atari 1050 disk-drive, and since its release, a controversy has raged, with some people condemning the XF551 while others praise it.

The first things you will notice about the XF551 is that it is shorter and wider than the 1050, incredibly quiet (and with no power light, you can't tell if it's on or off without attempting to access it), and comes with the same old Atari DOS 2.5. This last point has generated mass confusion and still does, many months after its release.

The Atari XF551 is a double-sided, double-density disk-drive, making it capable of storing a full 360K per disk, smashing the 130K barrier imposed on the 1050's double density mechanism with "enhanced" or "dual" density. The confusion stems from the fact that Atari DOS 2.5 can only handle single-sided, single- (88K) or enhanced-density (128K) disks. Since it comes only with DOS 2.5, people assume it is merely a clone of the 1050 drive in a new case. The XF551 is in fact capable of handling 360K disks with the right DOS, namely DOS-XE (formerly A-DOS), from Atari. This product was developed by Optimized Systems Software (now owned by ICD) and completed ages ago, but it still has not been released because of "delays in printing the manual," according to Atari sources.

The story behind "the new drive with an ancient DOS" is that Atari was sued by Nintendo over its XEGS. Nintendo disputed Atari's claim in the XEGS advertisements that you can attach a disk-drive and use it as a real computer. This is true, but Atari had no disk-drives for six months or more. Apparently, when the suit was filed, Atari decided to ship the XF551 with the old DOS 2.5, since DOS-XE (or the manual for it) was not complete. The suit was dropped, and we finally got new disk-drives. The problem is that you will waste over 200K of disk space on each and every floppy if you are still using the old Atari DOS!

"Wait a minute. I can just flip the disks and format the bottom side." Is that what you were thinking? It won't work. The XF551 uses a timing light at *format time* only. When you flip your floppy, the timing hole is on the wrong side. The XF551 doesn't see the timing pulses from the timing hole and refuses to format the flip side of the disk. This may seem rather bizarre when you realize that the disk-drive does not use the timing hole at any other time. When the disk is formatted, timing is written on the disk. The timing hole is used for nothing but the prevention of formatting the flip side of a disk.

This limitation is quite logical when you think about it, however. If you have been using an 810 or 1050 disk-drive, you have probably been formatting the flip side of your floppies for years. Suddenly you start using double-sided, double-density, 360K disks. (Atari assumed you would have a DOS that could do that, of course.) Wouldn't it spoil your day if you inadvertently formatted the flip side of a full 360K diskette? This is where the timing light prevents you from making a serious mistake.

To get the most out of the XF551, while waiting for DOS-XE, you can employ MYDOS 4.0 or SpartaDOS (or the hot new SDX cartridge). These DOSes fully support the XF551. SDX, SpartaDOS in a cartridge, even supports a high-speed disk mode of the XF551, similar to their US Doubler modification for the 1050 disk-drive. (The high-speed mode is a standard, built-in feature of

continued on page 40

Starling

TRIAL BY FIRE

by Greg Knauss

The life of a fireman is tough. Tough like leather, tough like steel, tough like an elementary school teacher, tough like reading all these cliched similes. *I* am tough. And I am a fireman.

I'd tried my hand at various professions other than fire fighting, sure, but none of them seemed to work out. Creative accounting, cat burglary, advising the President: I got caught-in at all of them. Eventually, I ended up being a fireman. I never intended to be a hero.

Oh, I suppose it comes with the territory though. Firemen go through life and death situations every day, and most people never hear of it. Usually, if someone is pulled from a burning building, they'll never find out who did it—just a faceless, nameless person in a funny-looking red hat and a tank of air. Not that it's fame that firemen are after—we fight fires to save lives and help people, not to get famous—but sometimes a fire has a little bizarre twist that gets it in the newspapers. And sometimes, you're involved.

There was a fire like that a little while back in a nursery. Dozens, maybe hundreds, of babies trapped at the top story of a burning building. The stairs were gone, the flames were too big to get a ladder past, and the roof was way too small to land a helicopter on. We had one alternative left: the net. The net is what we use to catch people who have to jump from burning buildings. Unfortunately, some idiot had strung it way too tight. It was more a trampoline than a net. This was going to be a circus—in more ways than one. None of us wanted to use it, but we had nothing else.

Me and a young kid—Tom somebody—grabbed the net and ran to the foot of the building. "Goo goo da-do!" he yelled. Later he insisted that that was baby-talk for "jump."

The first baby who jumped tumbled down toward us, hit the net, and went right back up. It was amazing. We stood there gape-mouthed for a second before realizing that the baby had not bounced straight back up. He had drifted away from the building a little and was going to hit the ground unless we moved.

We moved. Fast.

Up he went again. And we moved again.

After his third bounce, when we were getting worried whether he was ever going to come down, he landed right inside an ambulance that was parked nearby. Softly. As safe and as happy as he could be.

Babies are supposed to have an innate fear of falling, but this kid seemed to love it. Someone had to hold him down because he tried to crawl back into the building to do it again.

Another jumped, and we rushed back to the building.

Up he went, again, and again, and again. It was going to be a long day.

Like I said, firemen are tough.

The net is what we use to catch people who have to jump from burning buildings. Unfortunately, some idiot had strung it way too tight.

Typing It In

To make your copy of *Trial by Fire*, drop the Action! cartridge into the computer and threaten to beat up your little brother until he types in Listing 1. As a word of advice, the long lists of numbers are much more easily entered if you have a friend read them to you whilst you type.

continued on page 44

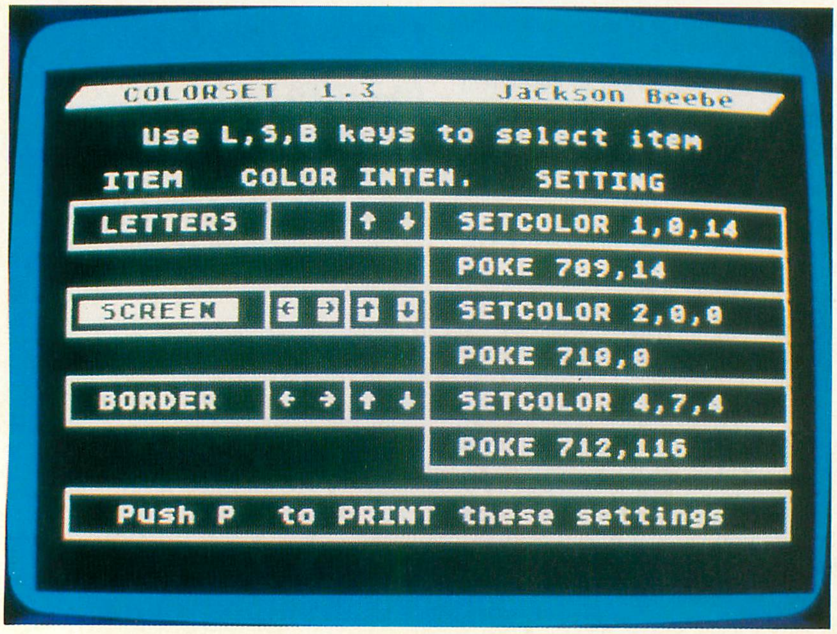
OLOR SET
COLOR SET
COLOR SET

by Jackson Beebe

Have you ever admired the colored text and screens in other people's programs and wished you could add them to your own BASIC programs? Have you given up trying to figure out Atari SETCOLOR statements and their relationship to POKE statements? Have you spent hours trying to design screens by POKEing in one number after another, writing small notes on the backs of envelopes with a pencil that kept getting lost, only to be dissatisfied with the results at the end?

If you answered "Yes" to any of the above, then *Color Set* is the program for you. Color Set allows you to change the color and intensity of the screen, border or text using the arrow keys. The correct SETCOLOR and POKE statements to produce these colors are constantly updated and displayed on the screen. Experimenting and watching the results soon gives you a working knowledge of the color statements.

Color Set will run on any Atari 8-bit computer. It will produce a printed record of screens you compose or will allow you to jot down settings shown on the screen. To quit the program, hit BREAK, and then clear the screen.



```
COLORSET  1.3          Jackson Beebe
   Use L,S,B keys to select item
 ITEM    COLOR INTEN.    SETTING
 LETTERS        ↑ ↓   SETCOLOR 1,0,14
                      POKE 709,14
 SCREEN   ← →   ↑ ↓   SETCOLOR 2,0,0
                      POKE 710,0
 BORDER   ← →   ↑ ↓   SETCOLOR 4,7,4
                      POKE 712,116
  Push P  to PRINT these settings
```

Graphics Modes

The Atari 800XL has 16 graphics modes, numbered 0-15. Each mode has its own resolution, number of lines, number of characters per line, and number of colors available. When you turn on Atari 8-bit computers, they come up in Graphics 0, their default mode. This mode is a text mode (like 1-2) as opposed to a graphics mode (like 3-15). This is the mode you use when you work in BASIC or with most text-oriented or utility functions. It has 40 characters per line, 24 lines per screen, and 16 colors of background or border, each with eight levels of intensity, ranging from dark to bright. The text will be the same color as the screen background, but vary in intensity against the screen. When the screen and text color are the same intensity, the text is invisible.

SETCOLOR Statements

The intensity of the text, the color and intensity of the screen, and the color and intensity of the border are controlled in Graphics 0 by three SETCOLOR statements.

SETCOLOR 1,0,Y = Text intensity

Y can be any even number from 0-14, to change text from dark to bright.

SETCOLOR 2,X,Y = Screen color and intensity

X can vary from 0 to 15 to change the color of the screen and text, and Y can be any even number from 0-14, to change intensity of screen color from dark to bright.

SETCOLOR 4,X,Y = Border color and intensity

X and Y are the same as when setting the screen colors.

The first digit in the SETCOLOR statement specifies that text, screen or border are to be changed. This digit (in Graphics 0) may have a value of 1, 2, or 4 only. The second digit specifies color, and may vary from 0-15. This holds true for both screen and border. Colors selected by the second digit are as follows:

0 = Gray	8 = Blue
1 = Gold	9 = Light Blue
2 = Orange	10 = Turquoise
3 = Red-Orange	11 = Green-Blue
4 = Pink	12 = Green
5 = Pink-Purple	13 = Yellow-Green
6 = Purple-Blue	14 = Orange-Green
7 = Dark Blue	15 = Light Orange

Note that in the SETCOLOR 1,X,Y statement, the digit X has no effect on the text. This is because the text color is always the same as the screen color. You may type numbers in if you wish, but the program always ignores the second digit in SETCOLOR 1,X,Y. For this reason, I have used zero as the second digit in my program and examples. This will satisfy the BASIC interpreter which needs a digit in that location and will remind you not to bother worrying about it for text.

The third digit controls the intensity of the text, screen or border, and may be any even number from 0-14. Intensities are roughly:

0-2 = Dark
4-8 = Medium
10-14 = Bright

For example, SETCOLOR 1,0,14, SETCOLOR 2,0,0 and SETCOLOR 4,7,4 selects bright white text on a black screen with a dark blue border. SETCOLOR 1,0,14, SETCOLOR 2,3,4 and SETCOLOR 4,7,2 selects white text on a red-orange screen with a dark blue border. SETCOLOR 1,0,8, SETCOLOR 2,12,0 and SETCOLOR 4,3,6 will produce green text on a black background with a red border.

POKE Statements

There are corresponding POKE statements that achieve the same results as the SETCOLOR statements. Once you figure them out, they are easier and quicker to use than SETCOLOR. Their locations are:

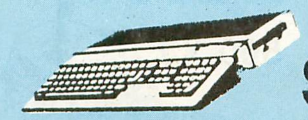
POKE 709 = Text intensity (0-14)

continued on page 41

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

Computer Repeats

Atari 1040ST



$449 NEW

with trade-in of 130XE, 1702 Mon., NP-10, 1200 modem
$CALL for your system

Used

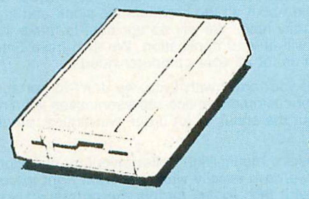
520ST Computer	$335	1040ST Computer	$585
Atari SF354 Drive	$69	65XE Computer	$85
Atari 800-48K	$59	800XL Computer	$69
Hayes Smartmodem	$49	Atari 1030 Modem	$35
1050 Drive	$139	Atari 1025 Printer	$79
Alien Group Voicebox	$39	Okimate 10 w/PIP	$99
ATR-8000 64K, Slaves	$199	Slave Drives from	$35
ICD PR:Connection	$49	Olivetti PR2300 prn	$119
C1802C Monitor	$155	Software/Books from	$1

New

1040ST Plus+ CPU	$719	520ST FM CPU	$499
Atari SF314 Drive	$209	Indus GT-100 Drive	$209
Atari 130XE Computer	$149	SM124 Mono Mon	$159
US Doubler w/DOS	$39	XF-551 Drive	$179
SC1224 Color Mon	$325	Avatex 1200eHC	$79
Atari SX-212 Modem	$89	Modem cable	$15
Avatex 2400HC	$199	1802D Monitor	$189
Star NX-2400 24 Pin	$389	Microline 183 wide	$315
Star NX-1000 144cps	$189	Epyx 500XJ Joystick	$19

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

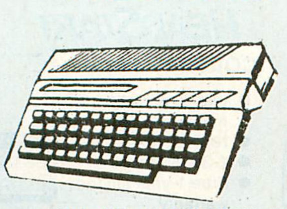
Atari XF551



$49 NEW

with trade-in of130XE, 1050 OR $109 with 1050
$CALL for your system

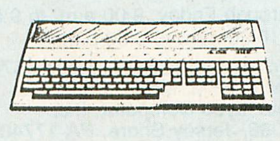
Atari 65XE



$85 NEW

with trade-in of 800XL OR 800-48K
$CALL for your system

Atari 520ST FM
with Double Sided Drive



$239 NEW

with trade-in of 130XE, 1050, 850, 1200 mdm., color mon.
$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!

Technical Support/Questions: 1-303-939-8144
1-800-347-3457

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

VISA MASTERCARD DISCOVER AMERICAN EXPRESS COD

CIRCLE #113 ON READER SERVICE CARD.

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: Monday through Friday, 9:00 a.m. to 9:00 p.m.
Saturday, 10:00 a.m. to 6:00 p.m.

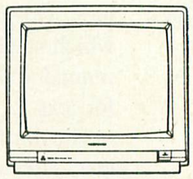
For Customer Service, call 1-717-494-1670,
9AM to 5PM, Mon. - Fri.
Or write: Lyco Computer, Inc.
P.O. Box 5088, Jersey Shore, PA 17740

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.

MAGNAVOX

CM-8502



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

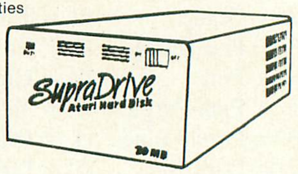
$179.95

ATARI HARD DISK

Super Speed
Super Performance

- 20 Megabyte Drive
- Improves ST Performance
- Includes Hard Disk Utilities

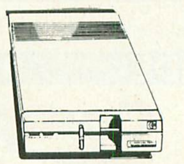
$569.95



INDUS

GTS-100

- Atari ST Drive
- 3.5" DSDD



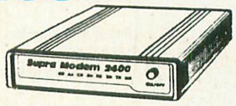
$195.95

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

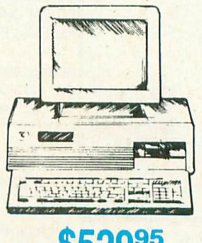
EPYX®

500 XJ

$13.95

8088 XT

- 16 Bit 8088 Intel Running at 4.77/10 MHz
- 640K RAM
- One 360K Floppy, 1 Parallel Port
- Multi I/O (includes 1 Serial Port, 1 Game Port, Clock Calendar, Disk Controller for Two 360K's)
- (8) Eight Expansion I/O Slots
- Jr. AT Case w/Keylock Turbo, Reset LED
- 101 Key Keyboard
- 150 Watt Power Supply
- One Year Warranty on Parts & Labor



$529.95

HEADSTART

COLOR SYSTEM

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

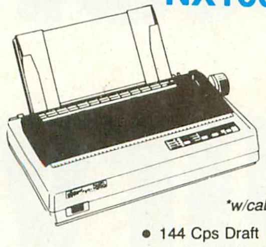
$969.95

• Hi Res color monitor included!

1-800-233-8760

CIRCLE #111 ON READER SERVICE CARD.

star micronics NX1000



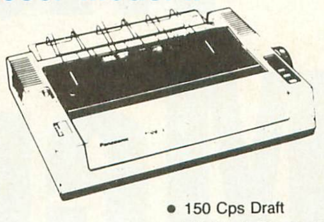
*w/cable purchase

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

$169.95

NX-1000	$169.95*
NX-15	$289.95
NX-1000 Color	$225.95
NR-15	$419.95
NB-15 24 Pin	$669.95
NX-2400	$309.95
NB24-10 24 Pin	$369.95

Panasonic Office Automation 1080i Model II



$149.95*

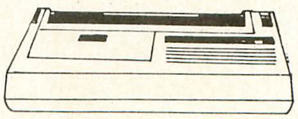
- 150 Cps Draft
- Friction & Tractor Feed Std.
- Bidirectional & Logic Seeking
- NLQ in all Pitches

*quantities limited

1080i Model II	$149.95*
1091i Model II	$195.95*
1092i	$309.95
1592	$375.95
1595	$439.95
1124	$319.95

*quantities limited

OKIDATA Okimate 20



- Color Printer
- 80 cps Draft
- 40 cps NLQ

$129.95

Okimate 20	$129.95
Okimate 20 w/cart	$189.95
180	$219.95
182	$209.95
182+	$225.95
183	$239.95

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

Selkosha

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

Attention Educational Institutions:

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

Epson

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

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 or III...ea.	$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

ATARI ST

Access:	
10th Frame	$22.95
Activision:	
Music Studio	$27.95
Broderbund:	
Superbike Challenge	$11.95
Typhoon Thompson	$21.95
Electronic Arts:	
Hunt for Red October	$32.95
Awesome Arcade Pack	$30.95
Epyx:	
Dive Bomber	$22.95
Impossible Mission 2	$22.95
Winter Games	$11.95
Firebird:	
Universal Military Simulator	$28.95

ATARI ST

Microleague:	
Microleague Baseball	$33.95
Micro. Wrestling	$25.95
Microprose:	
F-15 Strike Eagle	$24.95
Gunship	$28.95
Mindscape:	
Road Runner	$27.95
Indiana Jones and the Temple of Doom	$27.95
Origin:	
Autoduel	$24.95
Ultima IV	$34.95
Strategic Simulations:	
Questron II	$32.95
Heroes of the Lance	$25.95

Monitors

Magnavox:

BM7652	$79.95
BM7622	$79.95
CM8502	$179.95

Interfacing:

Xetec AT	$39.95
Microprint	$39.95
PR Connection	$62.95
Printer Connection	$39.95

DEALER INQUIRIES INVITED

Modems

Avatex:

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

Supra:

300	$119.95*
1200	$119.95*
2400	$129.95

*limited quantities

Avatex™ 2400 Modem $149.95

Diskettes

5-1/4

Disk Notcher	$5.95
Maxell:	
SSDD	$7.95
DSDD	$8.95
Bonus:	
SSDD	$5.95
DSDD	$6.95
SKC:	
DSDD	$6.95
DSHD	$13.95
Generic:	
DSDD	$4.95
Verbatim:	
SSDD	$8.99
DSDD	$11.50

3.5

Maxell:	
SSDD	$11.50
DSDD	$17.95
Bonus:	
SSDD	$10.95
DSDD	$13.95
Verbatim:	
SSDD	$12.95
DSDD	$18.95
SKC:	
SSDD	$9.95
DSDD	$13.99

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

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

Master Memory Map Part VII

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.

Page 3

You probably aren't going to be too thrilled with page 3. Why? It's all about I/O. That means that you may not understand a lot of it, because I/O can get real complicated real fast. Don't worry too much about it, though. BASIC has commands that take care of these locations for you, so you're only reading about these locations for enlightenment. If, on the other hand, you're programming in machine language....

Before we go any further, make sure you've read at least vaguely on I/O. It's not that long, or complicated, but it will give you a nice overview of what everything here is used for.

Input/Output

Input/Output, or I/O as it is more commonly called, is an extremely important part of any computer. Without it, the various parts of the computer wouldn't be able to talk to each other. Such communication isn't limited to disk drives and printers, either. The keyboard, television set, and screen editor must all be able to tell the computer what's going on, and all of this is I/O's responsibility. Unfortunately, because I/O has so much to do, it can be a little complicated. Luckily, complicated doesn't mean difficult in this case, so despite all the memory locations that deal with I/O, the basic concept is relatively simple.

There are three main routines that take care of I/O for a given device ("device" is just a fancy word for the keyboard, printer, or whatever it is we want to talk to). They are shown in Figure A.

Central Input/Output Routine	(CIO)
Device Handler	
Serial Input/Output Routine	(SIO)

Figure A. I/O devices

To help these three routines talk to each other (I told you this got complicated), there are also four "control blocks" in Figure B.

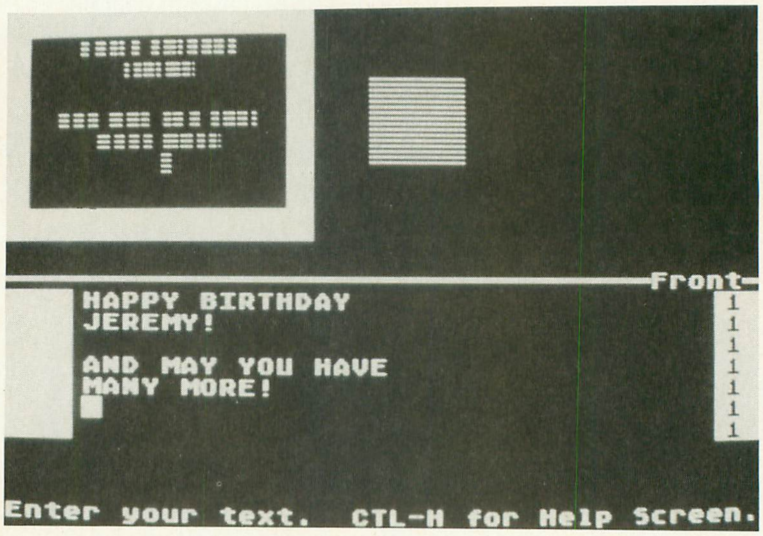
Input/Output Control Block	(IOCB)
Zero-page I/O Control Block	(ZIOCB)
Device Control Block	(DCB)
Command Frame Buffer	(CFB)

Figure B. I/O control blocks

Now that we've got the names straight, let's look at what each does, where it can be found, and how everything is tied together.

IOCB: Actually, it should be IOCBs since there are eight of them. The IOCBs are found

continued on page 54


```
HAPPY BIRTHDAY
JEREMY!

AND MAY YOU HAVE
MANY MORE!
```

Print Power

Hi-Tech Expressions
1700 N.W. 65th Avenue, Suite 9
Plantation, Florida 33313
$14.95

Print Power is a superb sign-, banner-, and card-making program for the 8-bit Atari computers. It comes on two Atari DOS 2.5 single-density format disks (with Commodore 64 versions on the flip side), one for the program and the other for borders, fonts and graphics. It is unprotected and runs great from a RAMdisk or hard drive, under any Atari-compatible DOS, including the new SpartaDOS X cartridge. This ability alone has turned me into a former *Print Shop* user, eliminating the frustrations of an incompatible disk format and protected software that cannot be backed up.

Products from Hi-Tech Expressions have matured significantly over the past couple of years, while the prices have stayed remarkably affordable. Their continued support of the Atari 8-bit computers with high power, low-priced products is certainly a welcome sight in light of the current lack of new productivity software.

Print Power comes with 40 different borders to choose from, which can be used in the creation of signs, letterheads and greeting cards. They are suitable for parties, Christmas, St. Patrick's Day and sporting events, among other more general forms. Borders are chosen from a menu by name. It is easy to page through the names with the up and down arrow keys on the keyboard. (You don't have to press and hold the control key when using the arrow keys.) A graphic printout of these borders is on a separate "Design Tools" reference card, which comes with the software package. You need to keep the card handy, since you will not see a graphical representation of the borders on the display.

(*Note:* The above limitation results in documentation-dependent software. This is a reasonable trade-off for no copy protection. Many companies, such as Electronic Arts, are eliminating disk-based copy protection in favor of documentation dependency. This allows owners of the product to make legitimate backups of the software, while preventing wholesale copying of the software by magnetic means only. Hi-Tech's products are indeed powerful and affordable, certainly worthy of patronage and not piracy.)

There are 60 different graphical icons to choose from, also printed and labeled on the "Design Tools" reference card. Five different graphic sizes may be selected to fill all, half or one quarter of the page. One- or two-inch (wide) sizes may be chosen also, handy for letterheads. Once the graphic is selected, a flashing box is placed on a representation of the card or sign you are currently building. Arrow keys move the box, and the space bar "anchors" a copy in the desired location. Multiple copies of a single graphic can be placed on a single page, but different graphics cannot be loaded for a single card face, sign or letterhead.

In a greeting card, up to four different faces (front, inside left, inside right and back) may

Reviewed by
Matthew J.W. Ratcliff

be defined. On each card face a different border, graphic and body of text (including none) may be defined. Greeting cards may be created in wide, tall and tent formats.

Print Power provides six different fonts for text editing, including Fine Print (a fine typewriter style text), Avante Garde (sans-serif style), Headline (bold courier), Times, Old English and Zapf (a cross between script and italics). While editing your text, many different control-key commands are at your disposal. Special text effects such as Bold, Italic, Outline, Jazz (multiple shadows), Shadowed (outline with shadow) and Raised (outline, italics, with shadow) may be enabled on each line. Each special effect is enabled for an entire line of text; you cannot have two different fonts on the same line. If bold is enabled on a line, the entire line will be printed in bold. Justification can be set to left, center or right. Up to four different fonts can be used in a single design.

Cards, signs and stationary allow three different text sizes, while banners can have eleven text sizes. Banners can be up to 42 pages long! When placing text on a page, the editor is smart enough to know when a line of text overruns the current line. A miniature graphical representation of your page and text are shown at the top of the screen as you enter text in a window at the bottom. If the text is too large to fit, your line becomes filled with "shadowed" characters. At this point characters can be deleted, or the font and text effects (such as size) changed to make the line fit.

The biggest problem with the text editor is that insertions of entire lines must be done manually. You cannot simply press Shift-Insert to make room for another line. In other words, Print Power's editor is "line" rather than "page" oriented. Generally, signs and cards are rather brief and this isn't too much of a problem. Other nice features of the text editor include a help screen and the ability of "masking" text, which provides a white outline of text when overlayed on a graphics background.

Print Power *is unprotected and runs great from a RAMdisk or hard drive.*

Print Power supports a lot of printers. It also includes a printer-driver construction set. Building a custom printer driver requires the assistance of a good printer reference manual and a lot of trial and error. The setup menu provides control of printer selection and testing, as well as memory configuration. The Atari version of Print Power is smart enough to use extra bank-switched memory of XL and XE computers, if enabled. SpartaDOS users can tell Print Power to use only the lower 48K of computer memory, so as not to conflict with the special areas of RAM that SpartaDOS requires.

Print Power may directly use fonts and graphics from the *Sesame Street Print Kit*, and it will modify an *Awardware* disk, enabling you to use its graphics as well. A utility from No Frills Software called *The Converter* will allow the conversion of Print Shop icons to Print Power format. (You can even edit Print Power and other graphics with The Converter, something Print Power doesn't allow.) The Converter will be reviewed in ANALOG soon.

I don't have any real complaints about Print Power. Its limitations are few. It does not provide a scaled graphical representation of icons and borders, as Print Shop does. But this is a fair trade-off, as noted earlier. With Print Power you cannot edit the graphics or fonts. The Converter from No Frills Software solves the graphics editing problem. I would like to see a font creator/editor/converter (load and convert Atari custom fonts) to turbocharge Print Power's capabilities.

As it is, Print Power is a phenomenal value at only $14.95. The Converter will make thousands of Print Shop icons available to Print Power, if you are concerned about making the move over to this product. Sesame Street Print Kit (also $14.95) is a nifty package in its own right, and will expand the graphic, border and font library of Print Power; an excellent product for young children. Print Power will provide many hours of creative fun for you and your family; I highly recommend it.

BITS 'N' PIECES: UPS

by Lee S. Brilliant, M.D.

The neophyte programmer sits hunched over his Atari 800, slowly working out the details of his ultimate graphics display. He has been working for the last eight hours to complete his program in time for the computer fair. So he sits glued to the screen, his trusty 410 recorder by his side, oblivious to everything going on around him, including the dark clouds gathering to the west; and as the time passes, the clouds ominously thicken and approach. Then in a flash and one blink of the lights, all eight hours of programming disappear.

Now the scene is a users'-group meeting where the club president has asked for volunteers to work the AtariFest. By the end of the meeting 20 or so people have entered their names into the computer registration. After at least half of the members have left, someone trips on the power cord and the list is gone.

Do any of these situations sound familiar? If they do then you need a UPS or Uninterruptable Power Supply. This month we will look at the last plug on your computer, the power plug, and we will build a UPS and the first portable Atari.

What is a UPS?

A UPS provides power to your computer and peripherals for as much as several hours, depending on the size of the unit and the demands of your equipment. Such a device is composed of a storage battery, an inverter which creates 60 Hz alternating current from the battery, and a power supply to charge the battery. Generally speaking there are two types: kick-in and continuous.

Look at Figure 1. In the continuous type, 110-volt power is converted to 12 volts in sufficient quantity for the inverter to power the computer all the time. The battery sits in parallel so its energy is consumed only when the main supply fails. Opposed to this is the kick-in type where the battery is kept charged constantly, but the inverter and battery only come online when the AC fails. The major differences are:

1) *Cost.* The continuous system needs to convert enough AC power to 12 volts DC to run the inverter continuously, and so needs a large, expensive power supply. The kick-in type only has to recharge the battery, not replace it.

2) *Switch-over time.* The continuous types provide instantaneous changeover from AC to battery since the inverter is always running, and the battery is never really disconnected. But the kick-in types need to wait to detect the absence of the AC before the battery and inverter turn on. The minimum is half a cycle which is about eight milliseconds. For practical considerations this is about 10 ms, short enough for the computer's power supply to handle.

Now the bad news. Typical 300-watt units cost at least $350 and are available through computer centers or mail-order supply companies. If you own a $5,000 IBM-AT running a $500 database program, a $350 investment for a UPS is reasonable. But when your UPS purchase could buy a computer, disk drive and monitor, it does not. The good news for XL or XE owners is that you can build your own UPS for under $50, one that will keep your computer running for over an hour in case of a power outage and also free your Atari from power lines.

The Heart of the UPS

A standard UPS replaces the 110-volt AC power, but our UPS takes a different approach than those in Figure 1. The new Atari 8-bit computers run off of single five-volt power supplies, so all you need is to provide an uninterrupted supply of five-volt power to your computer, and you are protected from short-term power loss. The easiest way would be to place a five-volt battery across the power plug to the computer. As it turns out, four Nickel Cadmium batteries put out five volts and should suit the bill, as shown in Figure 2.

Unfortunately, the details do not work out this easily. The measured output of four freshly charged NiCads is 5.1 volts, while the output of my XL power supply was 4.8 volts under load (about one ampere). The result is that the batteries continuously lose power into the computer even when the power supply is working. You can place a diode in series with the batteries to block reverse flow and drop the battery voltage to about 4.3 volts. This actually works but is only marginally successful because 4.3 volts is below the operational rating of the computer and may result in some berserk operations. The other problem is that the computer's power supply sucks huge amounts of battery power when you unplug it, so battery life is very short. You can, however, easily provide portable operation this way, using only the NiCads without the power supply or diode.

A better approach is to build our own power supply and design it from scratch to use battery back-up, essentially making a version of a continuous-type UPS. The power supply that comes with your Atari could be modified if its innards were accessible, but I don't recommend this approach. There are at least four different supplies that come with the X series computers, and each one is different. Mine is completely filled with epoxy and, so, unmodifiable. The schematic for our homebrew supply is shown in Figure 3.

T1, D1 and C1 generate about 14 volts of DC at one ampere. This is fed to a voltage regulator, IC1, which drops the output to a steady five volts so long as the supply input is over eight volts. So far we have only duplicated the existing Atari power supply, so the 12-volt battery is added to provide power to the regulator even when the 110-volt source dries up. Power from D1 will tend to flow into the battery, which is only about 12 volts. D2 prevents this backflow and consequent overcharging of the battery. With D2 present, there is no current draw from the battery unless the voltage from the D1 stops, then the transfer of power from the battery begins without any delay. If you leave off T1 and D1 you have a portable five-volt power supply that runs your computer without a plug-in and could open the way for self-contained Atari robots!

Construction

Construction is not difficult but needs a few comments for the beginner. The case should be metal since you are dealing with high voltages and need to provide a heat sink for the regulator. Use an insulating mounting kit and heat-sink grease to attach the voltage regulator so there is good thermal contact but no electrical connection between the regulator and the box. Sand off the paint from the case under the regulator because if there is not adequate heat dissipation for the regulator, it will overheat and shut down, thus defeating the purpose of this project. Don't worry, though, regulators of this type are virtually immune to damage due to overheating or overloading including short circuits. The metal box will get a little warm after running a while and this is normal.

A standard 7805 (fixed at five volts) regulator is not quite powerful enough, so I used a LM317T adjustable regulator because 1) it was laying around my shop, and 2) it is rated at 1.5 amps. The values of R1 and R2 are selected to produce exactly five volts from *my* 317T. You should use a voltmeter to check the output from your supply *before* plugging into your computer. If you put in more than 5.2 volts, you could turn your Atari into a paperweight. If you need to adjust the output of the regulator, change the value of R1—higher to increase voltage, lower to decrease it. Connections to the seven-pin plug are viewed from the plug's pin side, not the cable end.

Any battery can be used so long as it is between nine and 12 volts. You could use ten NiCads, two lantern batteries, an automobile or motorcycle battery or even eight "D"-size alkalines. The type and size of battery is determined by what you want to do and how long you want to run on battery power. Alkaline batteries should run only 30 minutes or so, but the batteries have a long shelf life and need replacement only after use or after three years.

NiCads are readily available in several sizes: both "C" and "D" sizes have a 1.2-amp-hour rating and are readily available. However, they lose power just sitting around, about 10% per month, and so need frequent recharging. Some "D" varieties have four-amp-hour ratings and so will last three times longer. Automotive batteries have massive power ratings but are quite large and can leak acid.

The best overall types are sealed lead-acid batteries or gelled electrolyte batteries (gel-

continued on page 52

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

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

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

Listing 2.
BASIC listing.

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

by Karl E. Wiegers

Karl's Komputerized Kitchen

When last we met, we were discussing how to get our Atari to understand instructions presented to it in normal English sentences—natural-language communication with the computer. The software that gives the computer the ability to at least partially understand natural-language input is called a "parser." The parser, as we discovered, has three principal functions: to break (or parse) the user's input into individual words and numbers; to search for these words in a preset vocabulary list; and to determine if a valid instruction has been entered and take appropriate action.

We chose as an example of natural-language communication a hypothetical kitchen of the future: *Karl's Komputerized Kitchen*. In this dream kitchen, we simply have to tell the computer what ingredients to combine and how to cook them, and the automated kitchen will do the rest. (My fiancée thinks this is the greatest idea since take-out pizza.) While I haven't yet connected my Atari to my Mixmaster, we can at least think about how we'd like to communicate with this komputerized kitchen.

Last time we focused on the second function of the parser, the act of looking up words from the user's command in a vocabulary list. A sample BASIC program was included to build a small vocabulary file with ordinary cooking kinds of words: FLOUR, SUGAR, BAKE, PREHEAT, STIR, and so on. Each word is assigned a number, or "token," to make it easier to handle the program logic when we get to the stage of trying to make sense out of the user's input. Synonyms, such as BUTTER, MARGARINE and SHORTENING, all have the same token. For simplicity, our vocabulary contains only words with all uppercase letters, and only exact matches to vocabulary words will be accepted. This rules out misspellings or plural forms, but it keeps our sample program simpler.

Whereas the core of the previous column was an assembler routine for searching the vocabulary list for a specific word, today's program is written entirely in Atari BASIC. It illustrates the other two functions of a full-parser program. I called the initial step of splitting the user's input into words and numbers a "preprocessor" function. After preprocessing, the machine-language routine we looked at last time is used to look for individual words and assign tokens to them. Step 3 is to evaluate the tokenized input and see if it makes any sense, in the context of what the computer has been programmed to "understand." If we are skillful, the result will be a simple BASIC program that appears to be remarkably intelligent.

Erratum

We begin with a minor correction to Listing 1 from last time. This was program VOCAB.BAS, which created the VOCAB.DAT vocabulary file. In Line 1070, please change the number following the word "DEGREES" to a 33. You may recall that these numbers are the tokens assigned to each vocabulary word. Rerun this program to create a new VOCAB.DAT file before proceeding further. Sorry about that.

Recipe Rules

Before getting too deep into the code,

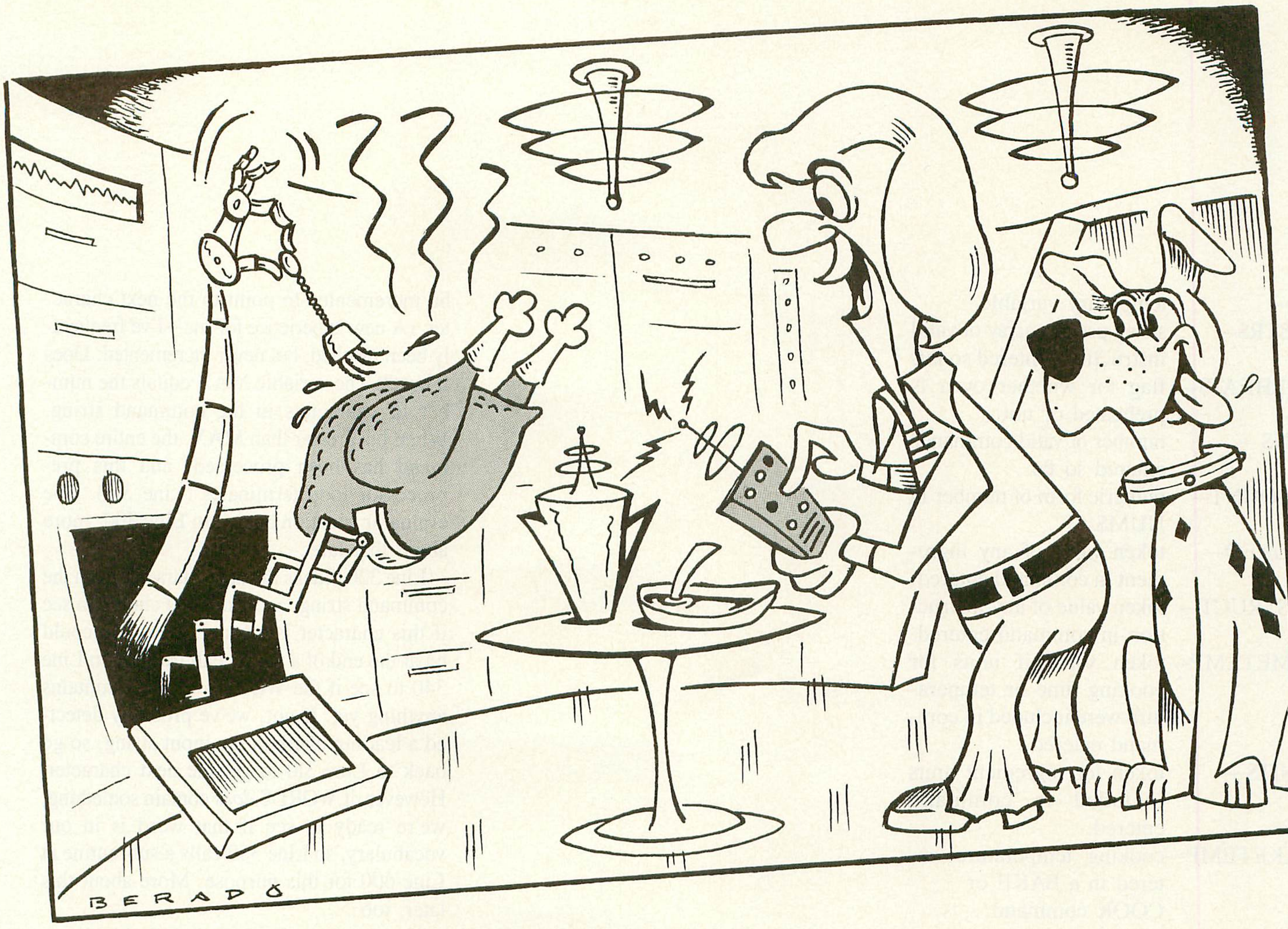


Illustration by John Berado

we'd better establish some ground rules for the komputerized kitchen program. First, I want to limit the number of commands entered before you can throw the pot into the oven to a maximum of ten. As valid commands are entered, I'll save them in a string pseudo-array and write them at the top of the screen before prompting for the next command. You'll have to preheat the oven to the desired temperature before baking your mixture. And the final command will be to bake whatever you have thrown together, no matter how disgusting it may appear.

Our vocabulary is rather limited for a full-blown automated kitchen, but this program should at least illustrate the power of a parser program. A real application program would of course include other useful terms in the vocabulary, such as "HELP" and "QUIT." I've omitted such user aids for conciseness, but don't ever leave these important functions out of an actual program!

As we parse each entered command, any words or numbers identified will be printed on the screen. This way you can watch as the computer tries to make sense out of whatever you tell it to do.

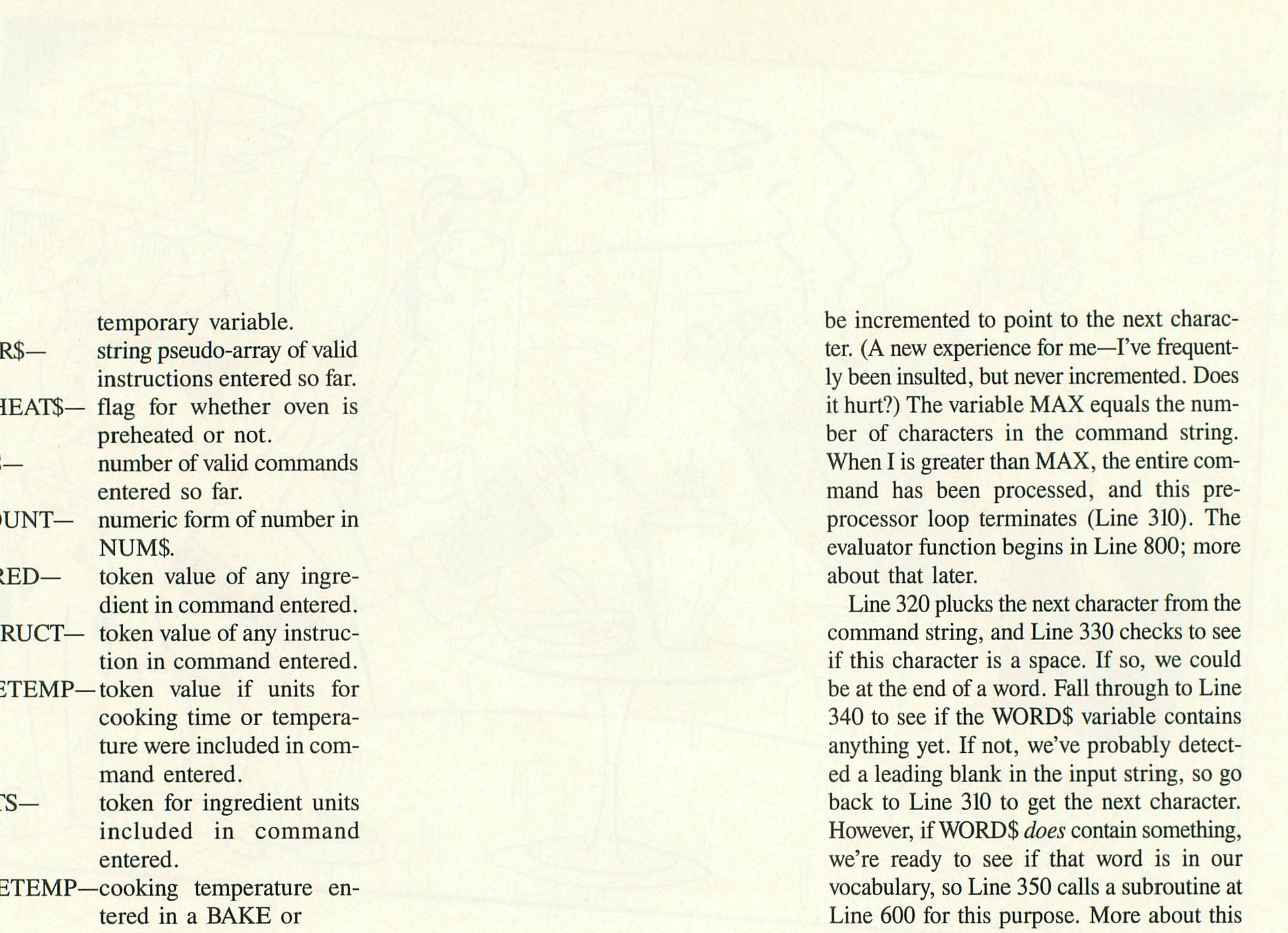
There are four classes of meaningful words that a user could enter in a command in this program. First is the name of an ingredient. The VOCAB.BAS program showed that ingredients have tokens in the numeric range of 0-19. Instructions have tokens in the range of 20-29. Units associated with cooking time or temperature have tokens in the 30s. And units associated with ingredients have tokens in the 40s. Also, each command entered in this particular program must contain a number, to indicate the amount of an ingredient to add, a preheating temperature, or a cooking time.

Warming Up

In brief, our preprocessing function will separate words in the input string by looking for blank (space) characters in the command. As each word is isolated, it will be looked up in the vocabulary list. If the first character in a word is in the ATASCII range for a digit from 0-9, that entire "word" is considered to be a number, and a separate processing routine is used. Note that this method won't handle numbers that begin with either a minus sign (hyphen) or a decimal point (period), but it could be changed easily to accommodate these.

Please turn your attention to the BASIC program in Listing 1. At first glance, this program may appear to contain a lot of spaghetti code, what with all the GOTOs, but I think it's better than it looks. Before we go any further, let's define what some key variables in this program represent.

VOCAB$—	entire vocabulary data string.
TEMP$—	temporary variable for reading from VOCAB.DAT.
WORD$—	target word being searched for in VOCAB$.
PARSED$—	command entered by user after removing unrecognized words.
ML$—	machine-language code for the parser.
IN$—	command string entered by user.
NUM$—	string representation of any number present in the command entered.



A$—	temporary variable.
INSTR$—	string pseudo-array of valid instructions entered so far.
PREHEAT$—	flag for whether oven is preheated or not.
NINS—	number of valid commands entered so far.
AMOUNT—	numeric form of number in NUM$.
INGRED—	token value of any ingredient in command entered.
INSTRUCT—	token value of any instruction in command entered.
TIMETEMP—	token value if units for cooking time or temperature were included in command entered.
UNITS—	token for ingredient units included in command entered.
BAKETEMP—	cooking temperature entered in a BAKE or COOK command.

Lines 10-140 of this program are pretty much the same as Listing 3 from last time. Some string variables are dimensioned, then the vocabulary-searcher machine-language code is read from file PARSER.OBJ (Lines 30-90). Next, the vocabulary data is read into variable VOCAB$ from file VOCAB.DAT (Lines 100-140). Some of the work variables needed are initialized in Line 150.

Lines 160-200 clear the screen and print any valid commands entered so far on the screen. If the oven has already been preheated, Line 205 prints a message to that effect. Lines 210-220 accept the next command from the user. If he just pressed Return without an entry, Line 230 tells him to try it again. Line 240 initializes the values for the four classes of vocabulary entries to zeros. If any words in these categories are found in the command entered, these variables will be set to the corresponding token values for the vocabulary words. The evaluator part of the program will examine the pattern of the tokens stored in these variables when it attempts to interpret the user's command.

Line 300 sets a variable called I to 1. This variable always points to the character we are currently processing from the command entered. After processing each character, I will be incremented to point to the next character. (A new experience for me—I've frequently been insulted, but never incremented. Does it hurt?) The variable MAX equals the number of characters in the command string. When I is greater than MAX, the entire command has been processed, and this preprocessor loop terminates (Line 310). The evaluator function begins in Line 800; more about that later.

I've frequently been insulted, but never incremented. Does it hurt?

Line 320 plucks the next character from the command string, and Line 330 checks to see if this character is a space. If so, we could be at the end of a word. Fall through to Line 340 to see if the WORD$ variable contains anything yet. If not, we've probably detected a leading blank in the input string, so go back to Line 310 to get the next character. However, if WORD$ *does* contain something, we're ready to see if that word is in our vocabulary, so Line 350 calls a subroutine at Line 600 for this purpose. More about this later, too.

If our current character is not a blank, we must already be in the middle of a word. Line 360 tests whether this character is in the ATASCII range for a digit. If so, a subroutine at Line 500 is called to handle it. (We'll get to this in just a bit.) If the current character is a lowercase letter, Line 370 translates it to uppercase. Line 380 adds this character to the end of our WORD$ variable and Line 390 says to go get the next character.

Number processing is dealt with in a subroutine beginning in Line 500. The digit character is added to the end of variable NUM$ and also is printed on the screen in Line 510. Line 520 points to the next character in the input string, and Line 530 tests to see if we have completed the command parsing yet. Line 540 fetches the next character from the command string. Line 550 tests whether this one is also a digit or a decimal point. If so, add it to NUM$ and keep going. If not, we must be at the end of the number, so print a trailing blank on the screen in Line 560.

Lines 570-580 append the string representation of our number to the end of variable PARSED$, which contains the interpreted form of the command entered. Line 590 converts the string form of the number into its numeric equivalent in variable AMOUNT.

This number will be used during the evaluation step.

Batter Up. Hit or Miss?

Recall that the parsing process is interrupted whenever we hit either a blank or the end of the command string, provided that the variable WORD$ contains something. Control is then passed to the subroutine at Line 600, where the contents of WORD$ are matched against the vocabulary list in VOCAB$ to see if we "know" the target word. But first, Line 610 extracts the final character from WORD$, and Line 620 discards that character if it is an anticipated punctuation mark. This gets around the problem of the user entering a command ending with a period or other symbol, yet it allows the user the flexibility of using punctuation in a natural kind of way in his commands.

Lines 630-650 should be familiar from last month. POKE the length of the target word into location 1790, call the machine-language vocabulary searcher routine, and get the corresponding token for the word out of location 1791. If the token is zero, Line 660 concludes that we don't know the word and says to proceed with the next character in the command string. Otherwise, we have a hit! Line 670 prints the word on the screen. Lines 680-710 classify the word into one of the four vocabulary categories I mentioned earlier and assign the word's token value to the appropriate variable. Lines 720-740 add the identified word to the PARSED$ variable and return to continue parsing the command string.

Now We're Cooking!

Hey, we're two-thirds of the way home. So far, all we have for our trouble are a few variables whose values correspond to vocabulary-word tokens, and perhaps a string representing a number entered by the user as part of his command. Now we have to see if the pattern of variable values makes any sense. We need some more rules to determine what constitutes a valid command. Try these on for size:

Large vocabularies, more complex command structures and more classes of vocabulary terms all will increase the required sophistication of the evaluator part of the parser.

1. Every command must include an instruction.
2. Every command must include a number.
3. The PREHEAT instruction must be entered before the BAKE instruction.
4. The PREHEAT instruction must specify the oven temperature.
5. The oven temperature must be between 300° and 500°.
6. The BAKE instruction is the final instruction.
7. The BAKE instruction must specify the cooking time.
8. The cooking time must be between five minutes and ten hours.
9. At least one ingredient must be added before baking.
10. If an ingredient is specified, units also must be included.
11. Exception to Rule 10: an ingredient whose token value is in the range 10 to 19 does not require units.

Whew! And these are just the ones that come to mind right away. Clearly, we need a lot of program code to test the numeric values of the key variables (INSTRUCT, UNITS, INGRED, TIMETEMP, and AMOUNT) to look for these conditions. Most real-life programs involving this sort of natural-language communication will have even more rules. Larger vocabularies, more complex command structures, and more classes of vocabulary terms all will increase the required sophistication of the evaluator part of the parser. You'll have to apply all of your error-checking skills when you design this section of your program.

Line 800 processes the final contents of WORD$ after control is transferred here when the end of the command string is reached. The rest of this section is somewhat hamstrung by the unstructured nature of Atari BASIC. For consistency, I used the approach of testing for "not" each error condition, and branching around the error handling code if no problem was encountered.

For example, Line 810 handles Rule 1. If no instruction was found in the parsed command, the value of variable INSTRUCT still will be zero. Otherwise, it will have the token value of the instruction entered. Line 810 says if we *do* have an instruction, go on to Line 830. Otherwise, fall through to Line 820 where the error message is printed. The subroutine at Line 10000 just waits for the user to press Return. After any such error, control returns to Line 160, where the command entry process begins again.

In this way, we keep leapfrogging through the evaluator code, until we figure out if the command entered is okay and what to do about it. A couple of special cases are handled in separate sections of code. The subroutine to handle a BAKE (or COOK) instruction begins at Line 2000, and a subroutine for the PREHEAT instruction is in Lines 2500-2620. By now you should understand why we are using tokens to represent the vocabulary words, rather than doing complex string comparisons for all these tests.

Without going in detail through all the code, I'll just indicate the line numbers where some of the rules are being tested. (The limit of ten commands before baking isn't being enforced, but the program will probably crash if you exceed this.)

Rule 1—Lines 810-820
Rule 2—Lines 830-840
Rule 3—Lines 2030-2040
Rule 4—Lines 2530-2540
Rule 5—Lines 2550-2580
Rule 7—Lines 2050-2060
Rule 8—Lines 2070-2110
Rule 9—Lines 2010-2020
Rule 10—Lines 910-920
Rule 11—Lines 890-900

If the command turns out to be valid, the contents of variable PARSED$ are copied to the appropriate section of the string pseudo-array variable INSTR$ in Lines 950-960. This is so the accumulated instructions can be printed at the top of the screen to remind you of what has been entered already.

And when you finally get your mixture into the oven, the program is over! The all-important END statement appears in Line 2160. You mean that wasn't the first place you looked for it? Oh, well, think of it as an Easter egg hunt.

Anytime you have a computer application that involves rules, concepts or patterns like we're dealing with here, the term "artificial intelligence" is sure to crop up.

Torch Light. Assembler Boot.

In your classic adventure game, which expects simple two-word commands, it's safe to insist that the first word be a verb and the second be a noun. "LIGHT TORCH" makes some sense then, and "TORCH LIGHT" does not. We can use the simplicity of the parser to impose some syntax restrictions on the kinds of commands that can be entered.

However, the parser I've described in these two issues of *Boot Camp* is a little more flexible in terms of the kinds of commands that can be entered. The downside is that we haven't placed any restrictions on syntax; command interpretation depends only upon the presence of keywords. A command like "COCOA TEASPOONS 11 BEAT" is perfectly comprehensible to our parser. It contains an instruction (BEAT), a number (11), an ingredient (COCOA), and units suitable for an ingredient (TEASPOONS). The fact that no one above the age of two would utter such a jumbled sentence doesn't faze the parser one bit. Obviously, a really sophisticated natural-language processor program would include more rules to govern usage and structure, but that's a topic best left to the specialists for now.

The AI Angle

Notice how we had to set up a bunch of rules to determine what constitutes legitimate commands for this application area, which commands had to be entered prior to others, and so on. The assembler and BASIC programs shown here handle all of these tasks in a detailed, brute-force manner, with explicit procedures for every single case. In fact, most traditional computer languages are called "procedural" languages, for this very reason.

Anytime you have a computer application that involves rules, concepts or patterns like we're dealing with here, the term "artificial intelligence" is sure to crop up. Two of the hot, current areas of interest in artificial (computer-based) intelligence are rule-based systems and natural-language processors. Our parser and komputerized kitchen examples clearly fit right into both of those realms. New programming languages like PROLOG let you program the rules directly into the computer, rather than having to explicitly write all the nitty-gritty code to handle each rule the way we did today. However, most of these require some pretty hefty hardware to do a passable job.

But even those of us stuck with our trusty old 8-bit Ataris might be impressed with the flexible human-computer dialogs that can be created using the ideas discussed in these two articles. The speed with which these simple programs interpret your commands is pretty remarkable. If you're ambitious, you might try recoding the pre-processor into assembler to see if you enjoy much of an execution speed increase. And if you're a hardware type, let me know if you can get the computer to do the dishes, too.

LISTING 1: BASIC

```
NG 1 REM Parser program for Karl's Komput
   erized
FT 2 REM Kitchen.  Enter up to 10 instruc
   tions,
FD 3 REM then bake it!
NJ 4 REM
GO 5 REM by Karl E. Wiegers
NL 6 REM
TG 8 REM -------------------------------
NO 9 REM
IY 10 DIM VOCAB$(280),TEMP$(40),WORD$(20)
   ,PARSED$(30)
YP 20 DIM ML$(79),IN$(40),NUM$(6),A$(1),I
   NSTR$(300),PREHEAT$(1)
NL 30 OPEN #2,4,0,"D:PARSER.OBJ"
YT 40 FOR I=1 TO 6:GET #2,A:NEXT I
PF 50 FOR I=1 TO 79
EE 60 GET #2,A
PA 70 ML$(I)=CHR$(A)
IW 80 NEXT I
MA 90 CLOSE #2
ZQ 100 OPEN #2,4,0,"D:VOCAB.DAT"
NU 110 FOR I=0 TO 6
QW 120 INPUT #2,TEMP$
YB 130 VOCAB$(I*40+1)=TEMP$
CA 140 NEXT I:CLOSE #2
MK 150 NINS=0:INSTR$="":PREHEAT$="N":BAKE
   TEMP=0
RR 157 REM
AA 158 REM -----------------------------
RX 159 REM
UK 160 PRINT CHR$(125)
KM 170 IF NINS=0 THEN GOTO 205
JX 180 FOR I=1 TO NINS
LA 190 PRINT INSTR$(30*I-29,30*I)
FS 200 NEXT I
ZG 205 IF PREHEAT$="Y" THEN PRINT "OVEN I
   S PREHEATED TO ";BAKETEMP;" DEGREES"
BJ 210 PRINT :PRINT "Next instruction, pl
   ease:":PRINT
YJ 220 INPUT IN$:PARSED$="":AMOUNT=0
NH 230 IF IN$="" THEN GOTO 210
ER 240 INGRED=0:INSTRUCT=0:TIMETEMP=0:UNI
   TS=0
YE 300 I=1:WORD$="":MAX=LEN(IN$):NUM$=""
EO 310 IF I>MAX THEN GOTO 800
AH 320 A$=IN$(I,I):A=ASC(A$)
KN 330 IF A$<>" " THEN GOTO 360
HQ 340 IF WORD$="" THEN I=I+1:GOTO 310
OY 350 GOSUB 600:I=I+1:WORD$="":GOTO 310
ZD 360 IF A>47 AND A<58 THEN GOSUB 500:GO
   TO 310
NN 370 IF A>96 AND A<123 THEN A$=CHR$(A-3
   2)
RS 380 WORD$(LEN(WORD$)+1)=A$
TG 390 I=I+1:GOTO 310
SF 498 REM
MP 499 REM -----------------------------
SP 500 REM Subroutine to process numbers
LA 501 REM -----------------------------
QW 502 REM
SJ 510 NUM$(LEN(NUM$)+1)=A$:PRINT A$;
QI 520 I=I+1
IT 530 IF I>MAX THEN GOTO 570
AN 540 A$=IN$(I,I):A=ASC(A$)
NF 550 IF (A>47 AND A<58) OR A$="." THEN
   GOTO 510
FY 560 PRINT " ";
ZC 570 PARSED$(LEN(PARSED$)+1)=NUM$
BJ 580 PARSED$(LEN(PARSED$)+1)=" "
QE 590 AMOUNT=VAL(NUM$):RETURN
SG 598 REM
PG 599 REM ------------------------------
   -------
TQ 600 REM Subroutine to search for curre
   nt word
NR 601 REM ------------------------------
   -------
QX 602 REM
YA 610 A$=WORD$(LEN(WORD$))
WU 620 IF A$="." OR A$="," OR A$="?" OR A
   $=";" OR A$="!" OR A$=":" THEN WORD$=W
   ORD$(1,LEN(WORD$)-1):GOTO 610
UF 630 POKE 1790,LEN(WORD$):POKE 1791,0
JJ 640 X=USR(ADR(ML$),ADR(VOCAB$),ADR(WOR
   D$))
IJ 650 X=PEEK(1791)
WS 660 IF X=0 THEN RETURN
QS 670 PRINT WORD$;" ";
DE 680 IF X<20 THEN INGRED=X
IO 690 IF X>19 AND X<30 THEN INSTRUCT=X
HF 700 IF X>29 AND X<40 THEN TIMETEMP=X
IA 710 IF X>39 AND X<50 THEN UNITS=X
OY 720 PARSED$(LEN(PARSED$)+1)=WORD$
BB 730 PARSED$(LEN(PARSED$)+1)=" "
ZL 740 RETURN
RZ 795 REM
AI 796 REM ---------------------------
QJ 797 REM Evaluator Routine
AO 798 REM ---------------------------
SL 799 REM
XI 800 IF WORD$<>"" THEN GOSUB 600
UB 810 IF INSTRUCT>0 THEN GOTO 830
IS 820 ? :? :? "Please include an instruc
   tion.":GOSUB 10000:GOTO 160
MK 830 IF AMOUNT>0 THEN GOTO 850
BI 840 ? :? :? "Please include a numeric
   amount.":GOSUB 10000:GOTO 160
LT 850 IF INSTRUCT=24 THEN GOSUB 2000:GOT
   O 160
RH 860 IF INSTRUCT=23 THEN GOSUB 2500:GOT
   O 160
DI 870 IF INGRED>0 THEN GOTO 890
GI 880 ? :? :? PARSED$;" of what??":GOSUB
    10000:GOTO 160
VU 890 IF INGRED>9 AND UNITS=0 THEN GOTO
   930
RA 900 IF INGRED>9 AND UNITS>0 THEN ? :?
   :? "Eggs don't have units!":GOSUB 1000
   0:GOTO 160
```

```
PQ 910 IF UNITS>0 THEN GOTO 930
KC 920 ? :? :? "Please include some units
   .":GOSUB 10000:GOTO 160
LC 930 ? :? :? "Okay!!"
UA 940 NINS=NINS+1:L=LEN(PARSED$)
CG 950 IF L<30 THEN FOR I=L TO 30:PARSED$
   (I,I)=" ":NEXT I
KL 960 INSTR$(30*NINS-29,30*NINS)=PARSED$
VF 970 GOSUB 10000:GOTO 160
KV 1998 REM
UE 1999 REM -----------------------------
   -
PN 2000 REM Sub to handle bake instructio
   n
RG 2001 REM -----------------------------
   -
IF 2002 REM
WS 2010 IF NINS>0 THEN GOTO 2030
ZP 2020 ? :? :? "You don't have anything
   to bake yet!":GOSUB 10000:RETURN
AM 2030 IF PREHEAT$="Y" THEN GOTO 2050
UG 2040 ? :? :? "You need to preheat the
   oven first.":GOSUB 10000:RETURN
OW 2050 IF TIMETEMP>0 THEN GOTO 2070
WF 2060 ? :? :? "Please include baking ti
   me and units.":GOSUB 10000:RETURN
KM 2070 IF TIMETEMP=31 THEN AMOUNT=AMOUNT
   *60
ZS 2080 IF AMOUNT>=5 THEN GOTO 2100
VH 2090 ? :? :? "Shortest legal cooking t
   ime is 5 min.":GOSUB 10000:RETURN
IZ 2100 IF AMOUNT<=600 THEN GOTO 2120
GL 2110 ? :? :? "Longest legal cooking ti
   me is 10 hrs.":GOSUB 10000:RETURN
CY 2120 ? :? :? "Okay!"
MQ 2130 ? :? "Your concoction is in the o
   ven,"
GU 2140 ? "and this program is now finish
   ed."
HX 2150 GOSUB 10000:PRINT CHR$(125)
FI 2160 END
KM 2498 REM
EN 2499 REM --------------------------
CI 2500 REM Subroutine to preheat oven
CI 2501 REM --------------------------
IP 2502 REM
WI 2510 IF PREHEAT$="N" THEN GOTO 2530
KN 2520 ? :? :? "The oven is already preh
   eated!":GOSUB 10000:RETURN
OL 2530 IF TIMETEMP=33 THEN GOTO 2550
EO 2540 ? :? :? "Include units for prehea
   ting.":GOSUB 10000:RETURN
UI 2550 IF AMOUNT>=300 THEN GOTO 2570
OG 2560 ? :? :? "Minimum temperature is 3
   00 degrees.":GOSUB 10000:RETURN
XG 2570 IF AMOUNT<=500 THEN GOTO 2590
TA 2580 ? :? :? "Maximum temperature is 5
   00 degrees.":GOSUB 10000:RETURN
ET 2590 ? :? :? "Okay!!"
OD 2600 PREHEAT$="Y"
TW 2610 BAKETEMP=AMOUNT
BK 2620 GOSUB 10000:RETURN
RJ 3000 REM Handle unitless ingredients,
   like eggs
LD 9998 REM
RF 9999 REM ---------------------------
UQ 10000 REM Sub to get RETURN press
EX 10001 REM ----------------------------
BM 10002 REM
ZQ 10010 POSITION 7,23
AV 10020 PRINT " Press RETURN to continue
   █";
WI 10030 OPEN #1,4,0,"K:"
DX 10040 GET #1,A
II 10050 IF A<>155 THEN GOTO 10040
QP 10060 CLOSE #1
DZ 10070 RETURN
```

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

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

☐ 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 #________________

MCNYY
DCNYY

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 April 31, 1989. **WATCH FOR IT!**

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/LX/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 W/SPELL....	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 PRICE
(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
STICKEY BEAR NUMBERS	26.95
STICKEY BEAR ABC	26.95
MATH ENCOUNTER	**9.95**

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
BLUE MAX	19.95
CENTIPEDE	14.95
CHICKEN (400/800)....	9.95
CLAIM JUMPER	9.95
CLOUDBURST	9.95
DAVIDS MIDNIGHT MAGIC	19.95
DEFENDER	14.95
DIG DUG	19.95
DONKEY KONG	5.00
DONKEY KNOG JR.	19.95
E.T. PHONE HOME	9.95
FIGHT NIGHT (XL/XE)..	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)...	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
SLIME (400/800)......	9.95
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!

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 GIFTS!
400 (16K) COMPUTER **$29.95** 48K UPGRADE KIT **$25.00**	**600XL COMPUTER** **16K - $49.95** **64K - $59.95**; 820 40 COLUMN PRINTER - **$39.95**	**800 (48K) COMPUTER** **$79.95** INCLUDES BASIC
1020 COLOR PRINTER/PLOTTER **$29.95** 40 COLUMNS WIDE INCLUDES PAPER AND COLOR PEN SET	1010 PROGRAM RECORDER **$29.95**; 410 PROGRAM RECORDER - AS IS **$9.95** (3 for $15)	**1030 MODEM WITH EXPRESS!** **$24.95** 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

Game Design Workshop

By having the object appear behind some things and in front of others, we can easily create a realistic three-dimensional effect.

by Craig Patchett

Now that you have the PMG basics down, it's time to move forward and take a look at some enhancements. The first enhancement that we'll look at is a really easy one to do. What it does is allow you to make each or all of the players wider. Unfortunately, this doesn't mean that you get more dots to use. Instead, the dots that you already have are made wider, so it's really the equivalent of changing graphics modes. In any case, locations 53256, 53257, 53258, and 53259 are used to adjust the width of the players, and location 53260 is used for the missiles. Here's how they're used:

PLAYER n:
POKE 53256+n,0 for normal width
POKE 53256+n,1 for double width
POKE 53256+n,3 for quadruple width

MISSILE 0:
POKE 53260,0 for normal
1 for double
3 for quadruple

MISSILE 1:
POKE 53260,0 for normal
4 for double
12 for quadruple

MISSILE 2:
POKE 53260,0 for normal
16 for double
48 for quadruple

MISSILE 3:
POKE 53260,0 for normal
64 for double
192 for quadruple

In case you're wondering about the missiles, you should pick the value you want for each missile, add those four values together, then POKE the result into location 53260. So, for example, if you wanted Missile 0 to be normal, 1 and 2 to be double, and 3 to be quadruple, you would use the command POKE 53260,212 (0+4+16+192). Try adding the following line to our last spaceship program from last month:

```
195 POKE 53256,1
```

Now change it to:

```
195 POKE 53256,3
```

Finally, get rid of it or change it to:

```
195 POKE 53256,0
```

Now you've seen the effect of changing the width of the players.

Way, way back, I said that PMG was useful for moving an object around the screen without affecting the background. What I didn't mention was that the background could either be behind the object or in front of it. What difference does it make? Well, by having the object appear behind some things and in front of others, we can easily create a realistic three-dimensional effect. The question is, just how do we go about telling the computer what to put in front and what to put behind? It's really quite simple, through the use of location 623. Atari lets you mix things up to four different ways with this location, as summarized below:

GPRIOR (623)

HIGH

PRIORITY	1	2	4	8
P	P0	P0	PF0	PF0
R	P1	P1	PF1	PF1
I	P2	PF0	PF2	P0
O	P3	PF1	PF3	P1
R	PF0	PF2	P0	P2
I	PF1	PF3	P1	P3
T	PF2	P2	P2	PF2
Y	PF3	P3	P3	PF3
	BAK	BAK	BAK	BAK

LOW

P = PLAYER #
PF = PLAYFIELD #
BAK = BACKGROUND

What does all this mean? At the top of the chart you'll see the numbers 1, 2, 4 and 8. These correspond to the values we POKE into location 623. Under these numbers are a list of PFs and Ps, which correspond to the various playfields and players (remember that playfields are anything on the screen that isn't a player or a missile). Anything in this list will appear in front of whatever is below it in the list if the two are on the screen at the same place. So, for example, if we were to POKE 623 with a 1, Player 0 would appear in front of Player 1, which would appear in front of Player 2, which would appear in front of Player 3, which would appear in front of Playfield 0, and so forth.

Time for an example. Let's set location 623 to 8, which puts the players in front of some of the playfield colors, and behind the rest. This will give us a chance to see the three-dimensional effect mentioned earlier. We'll then draw some simple playfield and move our spaceship around in it. Make sure you add the following program lines to the last spaceship program you typed in last month:

```
130 GRAPHICS 19
131 SETCOLOR 0,0,2:SETCOL
OR 1,0,4:SETCOLOR 2,0,6
132 FOR REG=1 TO 3:COLOR
REG:ST=2+10*REG-15*(REG=3
)
133 FOR X=0 TO 4:PLOT ST+
X,4+(REG=3):DRAWTO ST+X,1
9-(REG=3):NEXT X
134 NEXT REG
195 POKE 623,8
```

This kind of effect should get your creative juices running; now you can move spaceships behind planets, or make people appear to walk into houses, with hardly any effort at all!

Here's something that you may want to play around with, but which I won't really go into too deeply. If we POKE 623 with any combination of 1, 2, 4 and 8 (such as 3, 7, etc.), then the computer will get confused every now and then as to what should be in front of what, and will just give up and make the confused region black.

To see an example of this, change Line 195 to:

```
195 POKE 623,15
```

There's even more that can be done with location 623. First of all, we mentioned before that the four missiles can be combined to make a fifth player. All we have to do is position them alongside each other and move them all together, right? Not quite, since each missile is a different color. As you might have guessed, location 623 takes care of this also. We simply add 16 to whatever value we're POKEing into 623 and the four missiles will each take the color of Playfield 3 (as in SETCOLOR 3). But that's not all. You also get this beautiful set of kitchen knives. Oops, wrong spiel!

There is one more feature of location 623, however, which allows us to add two more colors to the screen when using PMG. If we add 32 to the value we're POKEing into location 623, then when Players 0 and 1 overlap, there will be a new color in the parts that overlap. The same is true for Players 2 and 3, but not for any other combinations.

Where do these new colors come from? The Atari does what is called a "logical OR" of the colors of the two players. This just means that it takes the binary representation of each of the two colors and blends them together to get a third. We'll see an example of this later in the chapter, along with a better explanation of exactly what the OR does.

So, what do you think so far? Pretty impressed by PMG? Believe it or not, there's still more to come. I'm sure that by now you've realized how convenient it is to move an object around the screen without having to worry about what's behind or in front of it. But sometimes you need to know whether two or more things are in the same place. For example, what if there were two spaceships on the screen and they ran into each other? We would probably want to know about it so that we could make them bounce off each other. Or what if we were moving people around the screen and they ran into a wall? We wouldn't want them to just walk right through it (unless they were ghosts!). So, what it comes down to is that we need some way of knowing when collisions occur between objects on the screen.

Fortunately, Atari supplies us with a whole bunch of memory locations to do just that. Before we start to look at them, however, we should warn you that they may look familiar. Why? Because we already used them to do other things. How is it that we come to have memory locations that do two different things? Well, these particular memory locations are in a special part of memory called the GTIA chip. The GTIA chip is different than regular memory in that the locations in it have different meanings depending on whether you POKE or PEEK them. That means that we can POKE a location to set a player's width, and then PEEK it to see whether a missile has collided with a player. This can be a little strange at first, but you get used to it after a while.

Sometimes.

Detecting collisions between the various objects on the screen is not the easiest thing to do, although it will seem easy at first.

Before I discourage you though, let's take a look at the locations that are involved. Basically, there are 16 memory locations that keep track of who has hit whom, and one location that clears these 16. Here's a complete list of which does what:

53248:	M0PF—	Detects collisions between Missile 0 and the playfield.
53249:	M1PF—	Detects collisions between Missile 1 and the playfield.
53250:	M2PF—	Detects collisions between Missile 2 and the playfield.
53251:	M3PF—	Detects collisions between Missile 3 and the playfield.
53252:	P0PF—	Detects collisions between Player 0 and the playfield.
53253:	P1PF—	Detects collisions between Player 1 and the playfield.
53254:	P2PF—	Detects collisions between Player 2 and the playfield.
53255:	P3PF—	Detects collisions between Player 3 and the playfield.
53256:	M0PL—	Detects collisions between Missile 0 and the players.
53257:	M1PL—	Detects collisions between Missile 1 and the players.
53258:	M2PL—	Detects collisions between Missile 2 and the players.
53259:	M3PL—	Detects collisions between Missile 3 and the players.
53260:	P0PL—	Detects collisions between Player 0 and other players.
53261:	P1PL—	Detects collisions between Player 1 and other players.
53262:	P2PL—	Detects collisions between Player 2 and other players.
53263:	P3PL—	Detects collisions between Player 3 and other players.
53278:	HITCLR—	Sets all of the above locations to zero (clears all collisions).

Right now you should be wondering how one location can detect up to four possible collisions, since that's what each of the collision locations does. For example, M0PF (location 53248) detects collisions between Missile 0 and the playfield. But, as we know, there are four types of playfields (five if we include the background, which we won't here). How do we use one memory location to record four different collisions? Quite simply, we use the lower four bits, one for each type of playfield. Since a collision has either occurred or not occurred, one bit is all we need for it. This diagram should help clarify things a little:

7 6 5 4	3	2	1	0	Bit #
not used	PF3	PF2	PF1	PF0	
(set to 0)					

(In case you're wondering, PFn stands for Playfield n.)

If one of these four bits is set (equal to one), then the corresponding collision has occurred. So, for example, if Missile 0 had collided with Playfield 2, M0PF would have a binary value of 00000100, which corresponds to a decimal value of four.

This same principle applies to all of the collision registers. For those registers that keep track of collisions with players, the layout is as follows:

7 6 5 4	3	2	1	0	Bit #
not used	P3	P2	P1	P0	

(set to 0)

(Pn, of course, stands for Player n.)

Now let's look at something that tends to trip people up a lot of the time. What would happen if, say, Missile 0 collided with Playfield 2, and then passed over and collided with Playfield 0? Which of the following values would M0PF have?

A 00000101 **B** 00000001 **C** 00000100

The answer is A, because a collision is not cleared until you POKE any value into HITCLR at location 53278. That means that the collision registers will keep track of all collisions that have occurred since the last time a value was stored in HITCLR. So, make sure you remember to POKE HITCLR after you look at any of the collision registers

A little side note before we look at an example. After the collision registers have been cleared, it takes the computer at least a sixtieth of a second to check for all collisions, and sometimes almost a thirtieth of a second. So what, right? Well, if you POKE HITCLR and then immediately check the collision registers, you may not get a proper value. What you want to do is the following everytime you use HITCLR:

```
240 POKE 53278,0:POKE 20,
0
250 IF PEEK(20)<2 THEN 11
0
```

You can use your own line numbers, of course. What these lines do is to clear the real-time clock at location 20, then wait for two-sixtieths of a second to pass before continuing. This makes sure that the collision registers will have the correct values when you go to look at them.

Try adding the following lines to the PMG program we've been putting together to see the collision registers at work. All they do is use P0PF to set the pitch of sound register 0. This makes the sound change as the player passes over different types of playfield.

Sometimes you need to know whether two or more things are in the same place. For example, what if there were two spaceships on the screen and they ran into each other?

Notice how we clear the collision registers after each SOUND statement.

Take out the POKE 53278,0 in Line 240 and see what happens if we don't.

```
230 SOUND 0,10*PEEK(53252
)+50,2,8
240 POKE 53278,0:POKE 20,
0
250 IF PEEK(20)<2 THEN 25
0
260 GOTO 230
```

There is one more technical detail to cover. It has to do with double-line resolution and isn't really used that often, but you may want to know about it anyway. VDELAY, at location 53276, allows you to move a double-line resolution player or missile down one scan line. "So what," you say? "I can do that by moving the data in memory." No, you can't, because the dots in double-line resolution are two scan lines high, not one. So, to help you get around that problem, Atari kindly supplied VDELAY. Each bit in VDELAY controls a player or missile. If a bit is on, then the corresponding player/missile is moved down one scan line. If it's off, then the player/missile is in its regular position.

Here's the way the bits are assigned:

7	6	5	4	3	2	1	0	Bit #
P3	P2	P1	P0	M3	M2	M1	M0	

Let's suppose you want to move a player down one scan line at a time. Here's what you would do:

1. Set the appropriate bit to one.
2. Move the data down one and set the bit back to zero.
3. Go back to Step 1.

To move the player up (one scan line):

1. Move the data up one and set the appropriate bit to one.
2. Set the bit back to zero.
3. Go back to Step 1.

Remember that you would only do this if you were using double-line resolution and wanted to move one scan line at a time instead of two. For absolutely smooth motion, most people can't see the difference.

Now that you know how to handle PMG, we can take a different approach and look at it from the outside. In other words, we can try and answer a few questions about it, such as, "When and where do we use PMG?" Let's do this by considering its advantages and disadvantages. PMG does have both, and it's important to be aware of them. We've already discussed most of the advantages, but I'll summarize them below:

PMG Advantages

1. Independent movement with priorities.
2. Ability to detect collisions.
3. Easy movement and animation.

Believe it or not, that's about it. The biggest advantage, of course, is that PMG allows us to move objects around the screen without disturbing what's already there. The other two major advantages, while still important, are somewhat secondary to this.

Let's move on to the disadvantages:

PMG Disadvantages

1. Only five players at once.
2. Only one color per player.
3. Each player is only eight dots wide.
4. Takes a lot of code to exploit the capabilities.

The big disadvantage is that there are only five players. Why? Because we can easily use two or more players to get more than one color, or to create a wider object. For example, we could draw a face in one player and give it a flesh color, then draw a hat in another player and color it white. If we then position these two players in the same place on the screen, we have a two-color head. Similarly, we can draw half an object in one player, the other half in another, and then position the two side-by-side to get an object that's sixteen dots wide instead of eight.

But there's nothing we can do to get more than five players. Well, that's not quite true. It is possible to use machine language to "cut" the players up so that there are as many as 940 objects at once (each only one dot high though)! The only problem with this is that you can only have five of these objects on each scan line at any given time. If you're interested in how this is done, take a look at the chapter on display-list interrupts.

We've now seen all the good things and all the bad things about PMG, and should be ready to decide when to use it and when not to. The big clue comes from the advantage list. After all, there's no point in going out of your way to do something unless there's an advantage in doing so, right? The main thing that should cause your mind to click and think, "Hey, I should use PMG here," is the need to move something over or under something else. Once you're in this kind of situation, PMG is really the only way out, at least from BASIC. Actually, once you're in a situation where anything needs to be moved, period, assuming it's small enough to fit in a player (or up to five players side by side), use PMG. The other time PMG comes in handy is when you need to add an extra color or two. After all, PMG does come with four extra colors, and nobody said a player had to move!

Would you believe that we're still not through? Although it's true that we've now covered all the technical details, there's a lot more to life than just the technical. Right now you know enough about PMG to get players up on the screen and moving around. Simple players, that is. What if you wanted a player to do more than just move, though? What if, for example, you wanted to make a robot walk through a tunnel instead of just move through it? In other words, how do you go about animating a player?

Actually, it's really not that hard. We've already seen how to animate using characters, by replacing a character with another that's slightly different. We can use the same technique with PMG, by storing different versions in different players and then switching between these versions. That's the easiest way, but it uses a lot of players. Another way is to keep redrawing one player with the different versions. This method is preferable, but it's too slow from BASIC.

So what? Remember the machine language routine MOVMEM from earlier? Well, there's no reason why we can't use it again here. The following program will take a robot and make him walk.

We'll store the player data in strings so we know where to find it (using BASIC's ADR instruction), then use MOVMEM to shift it around.

Anyway, try it out for yourself and then we'll take a closer look at exactly what's going on: Type NEW, then:

```
110 DIM CHARSET$(152)
120 GOSUB 290
130 DIM MOVMEM$(41):FOR X
=1 TO 41:READ A:MOVMEM$(X
,X)=CHR$(A):NEXT X
140 DIM MEMCLR$(36):FOR X
=1 TO 36:READ A:MEMCLR$(X
,X)=CHR$(A):NEXT X
150 PB=PEEK(740)-8:POKE 1
06,PB-8:PMBASE=PB*256:GRA
PHICS 23
155 POS=30
160 SETCOLOR 4,2,10:SETCO
LOR 2,14,2:SETCOLOR 1,14,
4:SETCOLOR 0,14,6
170 X=USR(ADR(MEMCLR$),PM
BASE+768,1279):POKE 54279
,PB:POKE 53277,3:POKE 623
,8:POKE 559,62
180 COLOR 1:PLOT 99,64:DR
AWTO 99,41:DRAWTO 72,41:P
OSITION 72,64:POKE 765,1:
XIO 18,#6,0,0,"S:"
190 COLOR 2:PLOT 98,40:DR
AWTO 85,32:DRAWTO 63,32:P
OSITION 71,40:POKE 765,2:
XIO 18,#6,0,0,"S:"
200 COLOR 3:PLOT 63,33:PO
SITION 63,51:POKE 765,3:X
IO 18,#6,0,0,"S:"
204 POKE 704,15:POKE 705,
15:POKE 706,15:POKE 707,1
5
205 FOR Z=0 TO 2
210 FOR X=0 TO 2:Y=USR(AD
R(MOVMEM$),ADR(CHARSET$)+
8+Z*48+X*16,PMBASE+1146+8
*X,7)
220 Y=USR(ADR(MOVMEM$),AD
R(CHARSET$)+16+Z*48+X*16,
PMBASE+1402+8*X,7):NEXT X
230 POS=POS+2+2*(Z=2):IF
POS>225 THEN POS=30
240 POKE 53248,POS:POKE 5
3249,POS+8:NEXT Z:GOTO 20
5
290 CHARSET$(1,76)="♥♥♥♥♥
♥♥♥♥┘┐ ▘▘▝┐┘♥phTt hP┤ ▖┴┼r
♣→♠HH◢HH◢x┤┘╲ ↑8<<♥@@◆♦p<
<╲▔┬┬┬ ╲◢◆P(h◢P◆♠▖rrr"
300 CHARSET$(77,152)="♣♣→
┘PHHhh◢xc┘┘▖>xp8┴@@@@@@pp
╲▔┬┬ ╲◢◆P(h◢P◆♠▖▖r♣♣♣→┘H
Hhxx◢xc╲╲╲▖▖▖▖◆◆@♥♥@@♥":
RETURN
400 DATA 104,104,133,207,
104,133,206,104,133,209,1
04,133,208,104,170,160,25
5,138,208,2,104,168,177,2
06,145
410 DATA 208,136,192,255,
208,247,230,207,230,209,2
02,224,255,208,233,96
420 DATA 104,104,133,204,
104,133,203,104,170,169,0
,160,255,224,0,208,4,104,
168,169,0,145,203,136,192
430 DATA 255,208,249,230,
204,202,224,255,208,234,9
6
```

It is possible to use machine language to "cut" the players up so that there are as many as 940 objects at once!

The following program will create Lines 290 and 300 for you. RUN it and then move the cursor up to the two lines it prints out and ENTER them. Then get rid of the original program (or LIST Lines 290 and 300 to disk, type NEW, and ENTER them back in).

```
100 DIM CHARSET$(152)
110 FOR CHAR=1 TO 152:REA
D DAT:CHARSET$(CHAR,CHAR)
=CHR$(DAT):NEXT CHAR
120 FOR LINE=0 TO 1
130 START=LINE*76+1:PRINT
 290+10*LINE;" CHARSET$("
;START;",";START+75;")=";
140 PRINT CHR$(34);
150 FOR CHAR=0 TO 75:PRIN
T CHR$(27);CHARSET$(START
+CHAR,START+CHAR);:NEXT C
HAR
160 PRINT CHR$(34)
170 NEXT LINE
1000 DATA 0,0,0,0,0,0,0,0
,0,3,5,11,11,11,5,3,0,240
,232,212,244,132,232,240,
4,9,11,19,19,17
1010 DATA 16,31,16,200,20
0,136,200,200,8,248,1,3,7
,14,28,56,60,60,128,192,1
92,224,96,112,60,60,7,11,
23,23
1020 DATA 23,11,7,8,224,2
08,168,232,8,208,224,16,9
,17,17,17,16,16,31,3,208,
200,200,232,232,8,248,192
,3,3
1030 DATA 15,62,120,112,5
6,24,192,192,192,192,192,
192,240,240,7,11,23,23,23
,11,7,8,224,208,168,232,8
,208,224,16
1040 DATA 9,9,17,16,16,16
,31,3,200,200,232,248,120
,8,248,192,3,7,7,7,15,15,
15,15,224,224,192,128,128
,192,192,0
```

Pretty neat, eh? Let's take a closer look at how it's all done. First of all, the different versions of the robot look something like Figure 1.

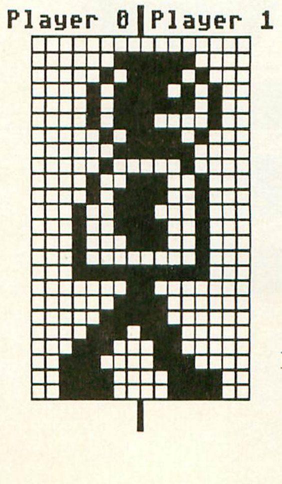
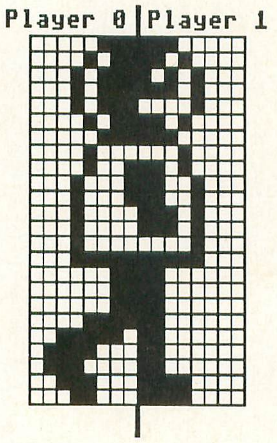
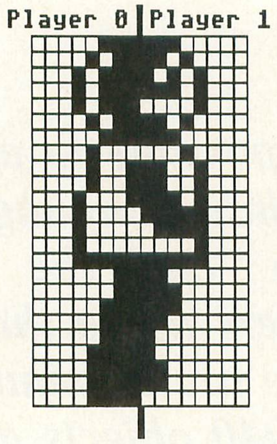


Figure 1





With that in mind, let's break down the program line by line.

110: CHARSET$ will hold the data for our robot versions.

120: First we set up CHARSET$.

130-140: Sets up machine-language strings.

150: Sets up player/missile memory.

155: POS is the horizontal position of the robot.

160-170: These lines should be familiar to you.

180-200: These lines set up a "tunnel" on the screen.

204: Set up the player colors.

205: This loop counts off each version of the robot.

210: We load the robot data in three sections: the head, body and legs. This is to allow us to do multi-color players in the next example. The X loop counts off each section. The USR instruction in this line moves the current section from CHARSET$ to Player 0.

220: This USR instruction moves the other side of the section to Player 1.

230: We now update the horizontal position and move the robot back to the left side of the screen if it has moved off the right.

240: Finally, we position the two players, move onto the next version, and then repeat the process all over again.

290-300: This is the routine to set up CHARSET$.

400-430: This is the data for MOVMEM$ and MEMCLR$.

Not that difficult after all, was it? Now let's try something a little different: multi-color players. We mentioned earlier that we can combine two or more players to get more than one color, but there's actually a way to combine two players to get three colors instead of just two.

How? GPRIOR, location 623, is used to set player/missile priorities; that we already know. We also know that it has other uses. One of these uses is to give the four missiles a common color. But the use that matters now is to create a third color when two players overlap. Specifically, when Players 0 and 1 or 2 and 3 overlap, then the area in which they overlap will be given a third color. To make this happen, simply add 32 to the value you're POKEing into GPRIOR (this is the equivalent of turning on bit 5).

The question remains, though, as to what the third color will be. Actually, it depends on what colors the players are. What the computer will do is OR the two colors together. In English, this means that the computer will look at the binary representations of the two color values and create a third value with the bits turned on wherever at least one of the other two values has a bit turned on. That was English?!

Let's look at a few examples:

01010101 OR 10101010 = 1111111
11010010 OR 10111010 = 1111010

Now let's apply this new knowledge (even if we still don't quite understand it) to our robot friend. We'll divide him up so that the head and body are both in Players 0 and 2, and the body and legs in Players 1 and 3. That way, the bodies will overlap to create the third color. Here are the necessary changes to the previous program:

```
170 X=USR(ADR(MEMCLR$),PMB
ASE+768,1279):POKE 54279,P
B:POKE 53277,3:POKE 623,40
:POKE 559,62
204 POKE 704,154:POKE 705,
101:POKE 706,154:POKE 707,
101
210 FOR X=0 TO 1:Y=USR(ADR
(MOVMEM$),ADR(CHARSET$)+8+
Z*48+X*16,PMBASE+1146+8*X,
7)
220 Y=USR(ADR(MOVMEM$),ADR
(CHARSET$)+16+Z*48+X*16,PM
BASE+1658+8*X,7):NEXT X
226 FOR X=1 TO 2:Y=USR(ADR
(MOVMEM$),ADR(CHARSET$)+8+
Z*48+X*16,PMBASE+1402+8*X,
7)
227 Y=USR(ADR(MOVMEM$),ADR
(CHARSET$)+16+Z*48+X*16,PM
BASE+1914+8*X,7):NEXT X
240 POKE 53248,POS:POKE 53
250,POS+8:POKE 53249,POS:P
OKE 53251,POS+8:NEXT Z:GOT
O 205
```

Notice how the colors tend to flicker somewhat. This is because we're not changing things quite fast enough. We can, however, clean it up a little bit by changing the colors so that there's less of a contrast.

Try these changes:

```
160 SETCOLOR 4,2,0:SETCOL
OR 2,14,4:SETCOLOR 1,14,6
:SETCOLOR 0,14,8
```

Next month, we'll install player/missile graphics into the BASIC Invaders game we were working on. (Remember that game? It's been two months since we've looked at it!) Also, because it's been so long since we've worked on the games, next month's column will include the game's listing up to this point. That'll help refresh your memory or help people catch up who may have missed the first few installments of *Game Design Workshop*.

ATARI 8-BIT POWER

ALPHA SYSTEMS is constantly innovating to provide more power for your 8-bit Ataris

NEW **HACK PACK**
Special OFFER
The Alpha Systems HACK PACK contains all our finest products for making Back-up copies, Analyzing, Understanding and Protecting your Atari programs. It comes complete with Atari Protection Techniques (Book and Disk I), Advanced Protection Techniques (Book and Disk II), The Chipmunk, The Scanalyzer, The Impersonator and Disk Pack 1000. Worth over $150. Get them all for the special price of **Just $99.95**

Atari Software Protection Techniques Vol I & II
These Book and Disk packages detail the most advanced copy protection methods in use today. They guide you through the methods used to create the protection as well as the copying techniques to get around them. They include information on Phreaking • Hacking • On-line security • Black boxes • Self-destructing programs • Pirate bulletin board systems • Logic bombs • New piracy laws • Hardware data keys • Weak sectoring (Phantom, Fuzzy and unstable sectors) • Overfilled tracks • CRC errors • Bank Select cartridges and MUCH, MUCH MORE. The disks include automatic program protectors, Protection Scanners, directory hiding and more.

BOOK I and DISK I	**$24.95**
BOOK II (Advanced protection) and DISK II	**$24.95**
Special Offer, Order both sets for Only	**$39.95**

CHIPMUNK
Automatic Disk Back-Up System. Make perfectly running unprotected back-up copies of hundreds of the most popular Atari programs. Chipmunk's sophisticated programming Automatically finds and **REMOVES copy protection** from most Atari programs. Back-up even heavily protected programs with ease. Finally, a back-up system that needs no special hardware or skills.
(If you need a full list of what Chipmunk copies, call or write for our free catalog) **$34.95**

Scanalyzer Automatically scan & analyze commercial programs. Unlock programming secrets and learn from the masters **$29.95**
Impersonator Cartridge to Disk back up system. Create running back-up copies of any cartridge (up to 16K) **$29.95**

NEW **CHEAT**
Get more from your games with CHEAT. Tired of spending days trying to beat a game? Tired of getting stuck just when you need another life? Cheat is an innovative new product that gives you the chance you need to beat your favorite games. Cheat works with hundreds of Atari games to give you unlimited lives or power. End the frustration and get hours more enjoyment from your games. (Call or write Alpha Systems for our free catalog with a full list of the programs that work with Cheat) **ONLY $24.95**

BASIC TURBOCHARGER
NOW for the first time a BASIC programmer can get the power, flexibility and incredible speed of machine language. BASIC TURBOCHARGER is a **book and disk package** that contains over 150 ready to use machine language routines. Complete instructions show how to add them to your own BASIC programs to get these features and more: • Smooth Scrolling • Player/Missile control • Load & Save Picture files • Sorting and Searching • Special Effects Graphics • Incredible Speed • Much, Much More • Over 150 programs. You've heard of the power of Assembler, now harness it for your own needs. **$24.95**

NEW **PARROT II**
An All New Parrot sound digitizer for your Atari. Parrot II is a sophisticated new hardware device that plugs into your joystick port. Parrot II has two inputs, One for a microphone and one for a powered source such as a tape player, radio or Compact Disk.
The Powerful Parrot II software lets you record sounds into your computer and play them back on any Atari. Parrot II turns your computers keyboard into a musical instrument with nine different sounds covering three octaves each. The sounds can be anything, a dogs bark, a piano, a complete drum set, a symphony or your own voice.
Parrot II lets you modify the sounds on a graphic display to create brand new sounds and special effects. Best of all, the sounds and voices can be put into your own programs that can be used on any standard Atari. Explore the world of digital sound and music. ONLY **$59.95**

Pre-Recorded Sound Disk More pre-recorded sounds for Parrot	**$4.95**
PARROT II Demo Disk (Does not require Parrot to run)	**$5.00**

NEW **POP-N-ROCKER** a fast paced, multi-player trivia game that mixes questions with real songs (digitized with Parrot). Be the first to identify the songs and answer the music trivia questions. *Pop-N-Rocker* comes with three data disks and lets you add new questions so it will never get old. You can use a Parrot Sound digitizer to add new songs too! Use any kind of music from Rock to Classical to Nursery Rhymes. A new concept in entertainment and a perfect add-on for Parrot. **$24.95**

COMPUTEREYES & MAGNIPRINT II +
Turn your computer into a digital portrait studio. This complete package lets you **capture, save & print** digital images from your **Video Camera, VCR or TV.** COMPUTEREYES hardware plugs directly into your joystick ports for easy use. Print your picture on a 6 foot poster. **$119.95**

GIANT WALL SIZED POSTERS.

ComputerEyes camera system
Comes complete with everything above, plus a black and white video camera and connecting cable. **$329.95**
Graphics 9 Software – Add a new dimension to your COMPUTEREYES pictures – captures images in 16 shades of grey. **$12.00**

Magniprint II +
Easily the most powerful print program available today. Print graphics from almost any format in hundreds of shapes, sizes, and shades. Supports **color printing** and lets you create **giant posters.** Magniprint II + lets you stretch and squeeze, invert, add text, adjust shading and much more. Works with EPSON, NEC, Citoh, Panasonic, Gemini, Star, XMM801, and compatible printers. (850 interface or equivalent required). **$24.95**

Graphics Transformer
Now you can combine the most powerful features of all your graphics programs. Create print shop icons from a Koala pad picture, from a photo digitized with ComputerEyes, or any picture file. Graphics Transformer lets you **Shrink, Enlarge and Merge** pictures for unequaled flexibility. **$22.95**

YOUR ATARI COMES ALIVE
SAVE MONEY. Finally an alternative to buying expensive computer add-ons. Your Atari Comes Alive shows you how to **built them yourself.** This 'How-To' **book and disk package** gives you complete step by step instructions and programs needed to built and control these exciting devices and MORE: • Light Pen • Light & Motor Controllers •Alarm Systems • Voice Recognition • Environmental Sensors •Data Decoders • More than 150 pages. **Your Atari Comes Alive** **$24.95**

ALPHA SYSTEMS

24 HOUR HOTLINE 216-374-7469
VISA & MASTERCARD, ORDER BY PHONE, OR SEND MONEY ORDER TO:
ALPHA SYSTEMS 1012 SKYLAND DRIVE MACEDONIA, OH 44056 **FREE BONUS:** DELUXE SPACE GAMES (3 games on a disk) Free with any order of 3 or more items. Include $3.00 shp & hdlg (US Canada) Ohio res. add 5½% sales tax. Foreign orders add $8.00 shp & hldg. Call or write for free catalog. Customer Service Line (216) 467-5665 M-F 9-3.

CIRCLE #115 ON READER SERVICE CARD.

XF551

continued from page 9

the XF551, however.)

Atari was thinking on their feet when they designed the XF551. Not only is it pleasantly quiet and fast, but the layout of the double-sided disk format is very convenient. For example, if you format a double-sided, double-density disk with SpartaDOS, MY-DOS, or even XE-DOS, everything is written to Side 1 of the disk, including the entire disk directory. Once the disk has over 180K of data stored on it, then Side 2 of the disk is written to. The advantage of this is that you can read all of the directory, and all files completely stored on Side 1 on any single-sided drive capable of the same density, such as a US Doubled 1050. (The same holds true for double-sided, single-density disks.)

Writing to Side 2 of the disk in the XF551 has confused a lot of people. If a disk is formatted as double-sided, then forget about it. The DOS will keep track of what is on which side. The disk always goes in the drive face side up. You never flip it to read the backside, nor is it "written backwards" as many people fear. This confusion stems from the fact that we have been flipping disks for so long. If a disk is formatted single-sided, any density, on both sides (by a 1050 or 810 for example), it is the same as two completely separate disks. Each side has a complete directory and file set. If a disk is formatted double-sided, it is one disk made up of two sides, with only one directory for all files on both Sides 1 and 2.

Because the XF551 refuses to format the flip side of a disk, this does not mean the XF551 cannot read and even write the flip side of disks already formatted this way. If you have a huge library of flippies, the XF551 will be able to manage them with no problems.

I have been using my XF551 for about six months now, with no significant problems. I have put SpartaDOS and SDX to good use, getting a full 360K out of every floppy without having to flip it all the time. This is ideal for storing large amounts of data, such as backing up a hard drive. My XF551 has trouble formatting some of my "plain label" diskettes, claiming that as many as two in ten are bad. However, nearly all of those "bad" disks formatted on my IBM PC-AT compatible computer with no problems. (Note: The PC can recognize bad sectors, remove them from the directory map, and still make a completely usable diskette. On the Atari, one bad sector means the entire disk is bad, so far as the format function is concerned.) I have had absolutely no problems formatting brand-name diskettes such as Verbatim and Maxell, however. The "plain label" disks which did format correctly gave me no problems reading and writing later.

If you have the disk version of *Flight Simulator II*, or other commercial programs with similar copy-protection schemes, they will not run on the XF551. The XF551 runs at exactly 300 RPM, while the 810 and 1050 drivers hover close to 288 RPM. Some of the newer, more sophisticated copy-protection schemes use the disk speed, which they expect to be 288, and refuse to run if it varies. It is ironic that Flight Simulator II comes in a cartridge with the XEGS, with the capability of reading FSII scenery disks. However, the newest Atari 8-bit computer, the XEGS, cannot use the newest Atari disk-drive, the XF551, to read them! I have not had any significant problems with this different disk speed, and have not heard of any other programs besides FSII refusing to run in the XF551.

In summary, the new Atari XF551 is fast, quiet, and capable of 360K per diskette—no flips required (or allowed, for that matter). Until DOS-XE comes out, you will need SpartaDOS or MY-DOS to take full advantage of its storage capabilities. The power switch is at the rear, which is a real pain for users with a computer hutch. The XF551 can be fussy about the quality of the diskettes you feed it. You may have occasional problems with copy-protection schemes, but it's doubtful. The "limitation" of not being able to format flip sides of disks is actually a feature to help break you of old habits. Once you begin using double-sided, double-density disks, you will never want to format a flippy again.

I can heartily recommend the XF551. It has no quirks, just a lot of end-user misunderstandings due to lackluster support from Atari. Use the XF551 with SpartaDOS or SDX for a high performance disk-drive package, the perfect upgrade for your system from that ancient 810 disk-drive that's been gasping, wheezing, and clunking along for too many years.

continued on page 13

POKE 710 = Screen color and intensity (0-238)

POKE 712 = Border color and intensity (0-238)

POKE 709 may be any even number from 0-14 to change the text intensity. The formula for the value of POKE 710 and 712 is:

Value = Color * 16+Intensity

For example, SETCOLOR 4,7,4 is equivalent to POKE 712, 116. Both 710 and 712 may range in value from 0-238.

Both statements are used in lines of BASIC code when you want to change a color parameter, as:

120 SETCOLOR 2, 3, 8

or

120 POKE 710, 56

Once set, the colors will stay set until another SETCOLOR or POKE statement is executed, a GRAPHICS 0 command is issued, or the computer is reset or rebooted.

How the Program Works

Lines 10-140 INITIALIZE

The default values for the text, screen and border colors and intensity are set at Line 110. Feel free to alter these to customize your version of this program's default colors. Line 120 dimensions variables for the 35-character printed comment (COM$), the 12-character color (COL$) and the six-character text intensity (LINT$). Line 130 calls the screen-printing menu subroutine at Line 1170, then goes to the LETTERS module to await input.

Lines 150-310 LETTERS module

The LETTERS module reprints the screen-menu lines that say SCREEN and BORDER and their arrows, in regular type, in case it just arrived here from the SCREEN or the BORDER module, then prints LETTERS and its arrows in inverse video. It opens a channel to the keyboard at Line 220 and gets a character. If the input is an up or down arrow, it increments or decrements the variable LINT that holds the text intensity, executes a new SETCOLOR statement, and reprints updated information on the screen.

It uses POSITION statements to specify the exact X and Y coordinates to print at on the screen. Note that when the updated line is shorter (has fewer digits) than the line already on the screen, it avoids leaving the last digit of the old line on the screen by adding one blank character to the end of each print line at 230-260. Forced erasing by hanging extra blanks on the end of lines is a handy household hack. Lines 270 and 280 will send control to the SCREEN or BORDER module, and Line 290 calls the PRINTOUT subroutine. On Return, or if an incorrect choice is entered, Line 300 sends control back to 220 to await more input. Both upper and lower case choices are checked for.

Color Set will produce a printed record of screens you compose or will allow you to jot down settings shown on the screen.

Lines 320-520 SCREEN module

This module accomplishes the same tasks for screen color and intensity.

Lines 530-730 BORDER module

This module updates the border.

Lines 740-860 PRINTOUT module

This prompts for a printed comment, then prints the current settings to a printer using LPRINT statements. Line 810 sets a variable COL equal to the value of the screen color variable SCOL. The program goes to a subroutine at 950 where a color name is assigned to the variable COL$ and printed at the end of the line on return. Line 830 does the same for BORDER. By using the variable COL and setting it equal to SCOL or BCOL, we can use the same subroutine for both modules. When printing is complete, the program goes to Line 1350 to reprint the bottom of the menu, and returns to the module it came from.

Line 880-930 ERROR trapping

Trying to print without having a printer on results in an error, normally crashing a program. Line 780 sets a TRAP statement for Line 890. If the printer is off and an error is generated, the program will go to Line 890. This line uses a PEEK statement to check memory location 195 which holds the error code and checks to see if it's Error 138, "Device did not respond." If so, it prompts you to turn on the printer, goes to Line 1350 to reprint the menu and returns to await input. This error trapping allows recovery whether a printer is present and turned on or not, without crashing the program.

Lines 940-1140 COLOR subroutine

This is a subroutine that assigns a name to the colors and text intensity. When finished, it returns to allow printing the names at the end of the print line.

Lines 1160-1380 Screen menu

These lines print the original screen menu and reprint the bottom portion when Line 1350 is used. Line 1170 begins with a "Clear the screen" command generated by using Esc-Ctrl+Clear. POKE 752,1 turns off the cursor for a neat screen appearance.

Conclusion

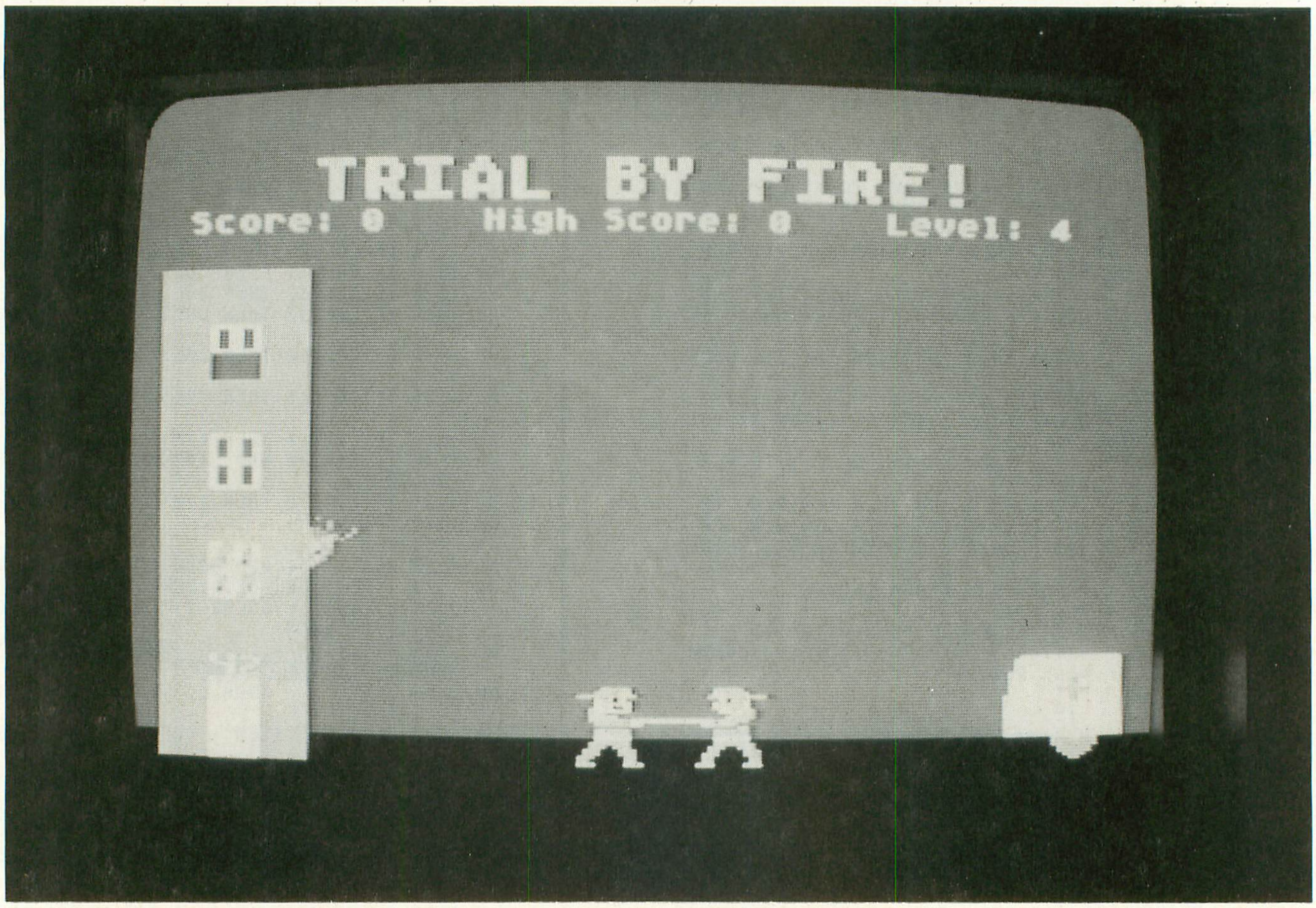
I hope you enjoy Color Set and use it to figure out new screen and text colors for your own programs. You can sit at the keyboard and create a number of attractive screens, labeling them and printing them out for future use. You may add comments like "Green on Black with Blue Border" or "Intro screen for Mailing List 3/21/87" or "Screen # 14" etc. To use them in your programs, simply put in the SETCOLOR or POKE statements as printed. The use of color can add a whole new dimension to your new or existing programs.

Jackson Beebe is a health-care center director in Champaign-Urbana, Illinois. He has owned his 800XL (upgraded to 256K) since the fall of 1984. He teaches Introduction to BASIC *at Parkland College and is President of the Progressive Atari Computing User Group (PAC) of central Illinois. He remains a die-hard fan of BASIC and Atari 8-bit computers.*

LISTING 1: BASIC

```
HY 10 REM ******************************
VA 20 REM *          COLOR SET         *
UO 30 REM *       by Jackson Beebe     *
OH 40 REM *        Copyright 1988      *
ZF 50 REM *     by ANALOG Computing    *
IM 55 REM ******************************
BP 60 REM << Initialize & Screen Menu >>
LV 70 REM < Change Defaults at Line 110 >
XZ 80 REM LINT=Ltr Int    SCOL=Screen Col
HY 90 REM SINT=Scr Int    BCOL=Bdr Col etc
QM 100 REM
PU 110 LINT=0:SCOL=0:SINT=12:BCOL=7:BINT=
   4
RS 120 DIM COM$(35),COL$(12),LINT$(6)
WS 130 GOSUB 1170:REM Print menu
QU 140 REM
BW 150 REM << LETTERS module >>
JK 160 POSITION 2,11:? " SCREEN  |";CHR$(
   27);CHR$(30);" ";CHR$(27);CHR$(31);
BG 170 ? "|";CHR$(27);CHR$(28);" ";CHR$(2
   7);CHR$(29);"|";
LN 180 POSITION 2,15:? " BORDER  |";CHR$(
   27);CHR$(30);" ";CHR$(27);CHR$(31);
BK 190 ? "|";CHR$(27);CHR$(28);" ";CHR$(2
   7);CHR$(29);"|";
EO 200 POSITION 2,7:? " LETTERS |";:POSIT
   ION 16,7:? CHR$(27);CHR$(156);" ";CHR$
   (27);CHR$(157)
RQ 210 REM << Input choice >>
EO 220 CLOSE #1:OPEN #1,4,0,"K:":GET #1,A
XE 230 IF LINT<14 AND A=45 THEN LINT=LINT
   +2:SETCOLOR 1,0,LINT:POSITION 34,7:? L
   INT;" "
NO 240 IF A=45 THEN POSITION 30,9:? LINT;
   " ";:GOTO 160
JT 250 IF LINT>0 AND A=61 THEN LINT=LINT-
   2:SETCOLOR 1,0,LINT:POSITION 34,7:? LI
   NT;" "
AE 260 IF A=61 THEN POSITION 30,9:? LINT;
   " ":GOTO 160
XX 270 IF A=83 OR A=115 THEN 330:REM Scr
LA 280 IF A=66 OR A=98 THEN 540:REM Bdr
SV 290 IF A=80 OR A=112 THEN GOSUB 750:RE
   M Prt
JT 300 GOTO 220:REM Wrong choice
QQ 310 REM
GB 320 REM << SCREEN module >>
MR 330 POSITION 2,7:? " LETTERS ":POSITIO
   N 16,7:? CHR$(27);CHR$(28);" ";CHR$(27
   );CHR$(29)
LH 340 POSITION 2,15:? " BORDER  |";CHR$(
   27);CHR$(30);" ";CHR$(27);CHR$(31);
BE 350 ? "|";CHR$(27);CHR$(28);" ";CHR$(2
   7);CHR$(29);"|";
TC 360 POSITION 2,11:? " SCREEN  |";CHR$(
   27);CHR$(158);" ";CHR$(27);CHR$(159);
QO 370 ? "|";CHR$(27);CHR$(156);" ";CHR$(
   27);CHR$(157);"|"
SF 380 REM << Input choice >>
FD 390 CLOSE #1:OPEN #1,4,0,"K:":GET #1,A
ZJ 400 IF SCOL>0 AND A=43 THEN SCOL=SCOL-
   1:SETCOLOR 2,SCOL,SINT:POSITION 32,11:
   ? SCOL;",";SINT;" "
GI 410 IF A=43 THEN POSITION 30,13:? SCOL
   *16+SINT;" "
QM 420 IF SCOL<14 AND A=42 THEN SCOL=SCOL
   +1:SETCOLOR 2,SCOL,SINT:POSITION 32,11
   :? SCOL;",";SINT;" "
GB 430 IF A=42 THEN POSITION 30,13:? SCOL
   *16+SINT;" "
IO 440 IF SINT<14 AND A=45 THEN SINT=SINT
   +2:SETCOLOR 2,SCOL,SINT:POSITION 32,11
   :? SCOL;",";SINT;" "
HM 450 IF A=45 THEN POSITION 30,13:? SCOL
   *16+SINT;" "
NZ 460 IF SINT>0 AND A=61 THEN SINT=SINT-
   2:SETCOLOR 2,SCOL,SINT:POSITION 32,11:
   ? SCOL;",";SINT;" "
GS 470 IF A=61 THEN POSITION 30,13:? SCOL
   *16+SINT;" "
PU 480 IF A=76 OR A=108 THEN 160:REM Ltr
LE 490 IF A=66 OR A=98 THEN 540:REM Bdr
SG 500 IF A=80 OR A=112 THEN GOSUB 750:RE
   M Prt
NG 510 GOTO 390:REM Wrong choice
QU 520 REM
GK 530 REM << BORDER module >>
MV 540 POSITION 2,7:? " LETTERS ":POSITIO
   N 16,7:? CHR$(27);CHR$(28);" ";CHR$(27
   );CHR$(29)
JM 550 POSITION 2,11:? " SCREEN  |";CHR$(
   27);CHR$(30);" ";CHR$(27);CHR$(31);
BI 560 ? "|";CHR$(27);CHR$(28);" ";CHR$(2
   7);CHR$(29);"|";
VF 570 POSITION 2,15:? " BORDER  |";CHR$(
   27);CHR$(158);" ";CHR$(27);CHR$(159);
NU 580 ? "|";CHR$(27);CHR$(156);" ";CHR$(
   27);CHR$(157);"|";
SJ 590 REM << Input choice >>
EO 600 CLOSE #1:OPEN #1,4,0,"K:":GET #1,A
WA 610 IF BCOL>0 AND A=43 THEN BCOL=BCOL-
```

```
   1:SETCOLOR 4,BCOL,BINT:POSITION 32,15:
   ? BCOL;",";BINT;" "
MG 620 IF A=43 THEN POSITION 30,17:? BCOL
   *16+BINT;" "
JL 630 IF BCOL<14 AND A=42 THEN BCOL=BCOL
   +1:SETCOLOR 4,BCOL,BINT:POSITION 32,15
   :? BCOL;",";BINT;" "
LZ 640 IF A=42 THEN POSITION 30,17:? BCOL
   *16+BINT;" "
BN 650 IF BINT<14 AND A=45 THEN BINT=BINT
   +2:SETCOLOR 4,BCOL,BINT:POSITION 32,15
   :? BCOL;",";BINT;" "
NK 660 IF A=45 THEN POSITION 30,17:? BCOL
   *16+BINT;" "
KQ 670 IF BINT>0 AND A=61 THEN BINT=BINT-
   2:SETCOLOR 4,BCOL,BINT:POSITION 32,15:
   ? BCOL;",";BINT;" "
MQ 680 IF A=61 THEN POSITION 30,17:? BCOL
   *16+BINT;" "
PY 690 IF A=76 OR A=108 THEN 160:REM Ltr
XO 700 IF A=83 OR A=115 THEN 330:REM Scr
SK 710 IF A=80 OR A=112 THEN GOSUB 750:RE
   M Prt
KT 720 GOTO 600:REM Wrong choice
QY 730 REM
AH 740 REM << PRINTOUT Subroutine >>
XY 750 POSITION 3,20:? "Add printed comme
   nt or push RETURN"
LM 760 ? " |
          |"
AM 770 ? " └──────────────────────────────
   ────────┘":POSITION 2,21:INPUT COM$
NB 780 TRAP 890:LPRINT :LPRINT :LPRINT :L
   PRINT "*********************************
   *********"
GB 790 GOSUB 950:LPRINT :LPRINT "
   ";COM$:LPRINT :LPRINT "  LETTERS = SET
   COLOR 1,0,";LINT;" =   ";LINT$
WG 800 LPRINT "                 POKE 709,";LI
   NT
SY 810 COL=SCOL:GOSUB 950:LPRINT :LPRINT
   "  SCREEN  = SETCOLOR 2,";SCOL;",";SIN
   T;" =  ";COL$
MZ 820 LPRINT "                POKE 710,";SCO
   L*16+SINT
CX 830 COL=BCOL:GOSUB 950:LPRINT :LPRINT
   "  BORDER  = SETCOLOR 4,";BCOL;",";BIN
   T;" =  ";COL$
PH 840 LPRINT "                POKE 712,";BCO
   L*16+BINT
FQ 850 LPRINT :LPRINT "******************
   ***********************":LPRINT :LPRINT
    :LPRINT
BQ 860 GOTO 1350:REM Reset
RH 870 REM
KE 880 REM << Error >>
UU 890 IF PEEK(195)=138 THEN POSITION 2,2
   0:? "
      "
NW 900 IF PEEK(195)=138 THEN POSITION 2,2
   1:? "  TURN ON PRINTER AND PUSH RETURN
      "
ET 910 CLOSE #1:OPEN #1,4,0,"K:":GET #1,A
MV 920 TRAP 40000:GOTO 1350:REM Reset
RA 930 REM
OO 940 REM << COLOR Subroutine >>
XO 950 IF COL=0 THEN COL$="GRAY"
JX 960 IF COL=1 THEN COL$="GOLD"
ZX 970 IF COL=2 THEN COL$="ORANGE"
CX 980 IF COL=3 THEN COL$="RED-ORANGE"
XU 990 IF COL=4 THEN COL$="PINK"
FX 1000 IF COL=5 THEN COL$="PINK-PURPLE"
OW 1010 IF COL=6 THEN COL$="PURPLE-BLUE"
XI 1020 IF COL=7 THEN COL$="DARK BLUE"
GC 1030 IF COL=8 THEN COL$="BLUE"
GN 1040 IF COL=9 THEN COL$="LIGHT BLUE"
JD 1050 IF COL=10 THEN COL$="TURQUOISE"
OW 1060 IF COL=11 THEN COL$="GREEN-BLUE"
CM 1070 IF COL=12 THEN COL$="GREEN"
HF 1080 IF COL=13 THEN COL$="YELLOW-GREEN
   "
VV 1090 IF COL=14 THEN COL$="ORANGE-GREEN
   "
PD 1100 IF COL=15 THEN COL$="LIGHT ORANGE
   "
AM 1110 IF LINT<6 THEN LINT$="DARK"
HO 1120 IF LINT>4 AND LINT<10 THEN LINT$=
   "MEDIUM"
UH 1130 IF LINT>8 THEN LINT$="BRIGHT"
AN 1140 RETURN
IN 1150 REM
MS 1160 REM << Screen Menu Subroutine >>
NO 1170 ? "↰":POKE 82,0:POKE 752,1:SETCOL
   OR 1,0,LINT:SETCOLOR 2,SCOL,SINT:SETCO
   LOR 4,BCOL,BINT
WH 1180 POSITION 0,1:? "  ◢  COLORSET  1.3
          Jackson Beebe   ◤"
AW 1190 ? :? "       Use L,S,B keys to sele
   ct item "
VA 1200 ? :? "  ITEM   COLOR INTEN.   SE
   TTING"
OD 1210 ? " ┌────────────┬────┬────┬───────
   ─────────┐"
XS 1220 ? " |           |    |    | SETCOLOR
   1,0,";LINT:POSITION 38,7:? "|"
NK 1230 ? " └────────────┴────┴────┼───────
   ─────────┤"
LS 1240 POSITION 19,9:? "| POKE 709,";LIN
   T;:POSITION 38,9:? "|"
IM 1250 ? " ┌────────────┬────┬────┼───────
   ─────────┤"
NO 1260 ? " |           |    |    | SETCOLOR
   2,";SCOL;",";SINT;:POSITION 38,11:? "|
   "
NW 1270 ? " └────────────┴────┴────┼───────
   ─────────┤"
WB 1280 POSITION 19,13:? "| POKE 710,";SC
   OL*16+SINT;:POSITION 38,13:? "|"
IY 1290 ? " ┌────────────┬────┬────┼───────
   ─────────┤"
QT 1300 ? " |           |    |    | SETCOLOR
   4,";BCOL;",";BINT;:POSITION 38,15:? "|
   "
NG 1310 ? " └────────────┴────┴────┼───────
   ─────────┤"
JD 1320 POSITION 19,17:? "| POKE 712,";BC
   OL*16+BINT;:POSITION 38,17:? "|"
DG 1330 POSITION 19,18:? "└──────────────
   ────┘"
AS 1340 ? " ┌──────────────────────────────
   ────────┐"
BF 1350 POSITION 0,20:? " |  Push P  to P
   RINT these settings   |"
AU 1360 ? " └──────────────────────────────
   ────────┘"
KC 1370 ? "
          "
BD 1380 RETURN
```



continued from page 11

Now that you have entered the whole thing, make sure that your typing is correct with *D:CHECK in Action!*, and then save the file. *Do not* run it before storing it. Some of those annoyingly long lists of numbers are machine-language routines that will send your computer into the electronic equivalent of a coma if typed wrong.

Now that there is a version safely saved, go to the monitor, compile and run the program.

Playing the Game

When the game first starts, you should see a building at the right of the screen with fire coming out of the second-story window, the back end of an ambulance at the right, and two firemen in the middle. A baby should be bouncing on their net for no very good reason.

Press the Select key or move a joystick plugged into Port 1 in any direction to increase the level that you wish to start on. It automatically defaults to Level 1. To begin the game, push Start or the joystick button.

When you start the game, push the joystick to the left. Soon a baby should jump out of the top story of the building. Wait until he bounces off the net that the firemen are holding, then let go of the joystick.

The joystick is self-centering, so to return

RIAL BY FIRE

the firemen to the center position, just let go of the stick. To keep them positioned on either side of the screen you must hold the joystick in that direction. This system of movement takes a little getting used to, so be prepared to be frustrated for the first few games. If you are easily upset, have a relaxing record playing. Anything by Simon and Garfunkel will do.

The baby will now bounce off the net again. Push the joystick to the right. One more bounce and the baby will safely land in the ambulance. (Yes, I realize the improbability of an ambulance with a sunroof, but if it didn't have one, there wouldn't be a game. So there.)

Soon another baby will jump from the window.

This continues until the number of babies you have saved is equal to the present level times three. When this happens the level is increased and you must then rescue a number of babies equal to *that* level times three.

Difficulty also increases as the level goes up. Every even level the babies will not always bounce toward the ambulance. They have a fifty-fifty chance of just going straight up instead of to the right. This will catch you by surprise the first time it happens, especially if you play the game before reading these instructions. Crank Simon and Garfunkel up a notch.

Also, at every third level, the number of babies that can be on the screen at one time is increased by one, up to four. So, on Level 10, there can be four babies on the screen at once and they won't always bounce toward the ambulance. This is generally known as chaos. And after Level 10, they start speeding up.

Every time you successfully bounce a baby toward the ambulance, you get one point. For each baby that you get safely inside the ambulance, you get a bonus of the number of points equal to whatever level you are currently on. Level 1, one point; Level 6, six points.

You can miss bouncing a baby three times before the game ends. After it's over, you will see the score you obtained, the high score, and the level you reached. To return to the title page, press the joystick button.

If you start the next game by just pressing the joystick button or Start, the game will restart on the level that the previous game ended. If you fiddle with Select, a different level can be chosen.

During play, the space bar will turn the pause feature on, and the joystick button will restart the game.

The End

Another baby landed in the ambulance, and I sighed. I'd lost count long ago, but we must have saved hundreds during this fire. Hundreds of little, bouncing lives.

We had to move—another had jumped and one was falling midway between the building and us. We ran and barely caught the one by the building. As he went up, we got under the other one. He bounced and we needed to get back under the first.

We ran.

I tripped.

Greg Knauss is 20 years old and should probably be out looking for a real job like any other respectable young man. Instead, he has decided to write computer programs, listen to The Who and spend time with his girlfriend.

TRIAL BY FIRE

LISTING 1: ACTION

```
; TRIAL BY FIRE
; BY GREG KNAUSS
; COPYRIGHT 1988 BY ANALOG COMPUTING
;
;
;       CHECKSUM DATA
;[DC ED BE 0C B3 A5 3B 0E
; 0C 57 BC E8 86 72 82 30
; A6 95 4A 7E 83 10 CB 94
; C6 BA 44 71 EA A2 10 4F
; 5F F8 73 68 2D 96 B8 B4
; 42 58 CA 9A D4 65 98 ]

CARD I,DL,SC,CH.CH2,DRB.K,SCORE,
     HSCORE,SCR
BYTE S,J,PM,BABY,BABES,HIT,MISS,ADV,
     LVL,UD,BSAV,NXTL,KEY,BRN
CARD ARRAY X(5),Y(5),B(5),Q(5),UP(5)

PROC CHSET()
[0 0 0 0 0 0 0 0
0 0 1 85 2 2 2 5
0 0 0 0 0 0 0 0
0 85 86 85 162 170 160 170
0 64 128 80 0 128 128 128
0 1 2 5 0 2 2 0
0 85 149 85 138 170 170 170
0 0 64 85 128 128 128 80
5 5 5 1 0 1 1 3
64 84 85 85 0 85 85 255
0 0 111 104 0 64 64 192
0 0 255 0 0 0 0 0
0 0 249 41 0 1 1 3
5 21 85 84 1 85 85 255
80 80 64 0 64 64 64 192

1 5 5 21 21 84 80 252
85 85 69 1 1 0 0 0
64 64 64 80 80 80 80 252
1 1 5 5 21 20 20 252
85 85 65 0 0 0 0 0
64 80 80 84 84 20 20 252

255 255 255 255 255 255 255 255
255 195 195 195 195 195 195 255
255 190 190 234 254 254 254 255
255 235 254 254 251 239 234 255
254 255 255 63 63 15 3 0
191 255 255 252 252 240 192 0
3 15 15 15 15 63 63 63
63 63 63 63 255 255 255 255
253 213 213 253 253 253 255 255
127 87 87 127 127 127 255 255
255 255 255 255 255 255 253 253
255 255 255 255 255 255 127 127]

PROC FIRE()
[255 255 255 255 255 253 245 247
0 0 0 1 1 68 64 0
117 85 119 215 213 85 85 85
255 215 211 215 215 199 215 255
85 85 87 87 93 85 87 127
85 119 95 95 127 255 255 255
255 255 247 245 245 213 215 87
223 253 125 117 245 85 85 85
81 81 69 84 84 80 64 0
16 0 1 17 16 17 85 85
245 197 213 215 215 215 87 95
85 85 85 85 85 85 85 85]

PROC FIRE2()
[255 255 255 255 253 253 245 213
0 0 0 0 0 4 16 0
93 125 117 85 93 93 85 85
255 211 215 199 215 199 199 255
87 87 93 85 87 119 95 127
117 119 127 255 255 255 255 255
255 255 255 253 247 213 215 85
221 125 245 245 213 85 85 93
85 84 68 16 16 64 0 0
0 1 1 64 68 68 69 85
245 213 213 213 211 211 87 95
85 85 85 85 85 85 85 85]

PROC BABIES()
[28 56 60 153 255 60 189 255
23 39 62 156 189 246 96 56
216 144 246 255 255 245 144 216
56 96 246 189 156 63 39 19
255 189 60 255 153 60 28 56
28 6 111 189 57 124 228 232
27 9 175 255 255 111 9 27
200 228 252 57 189 111 6 28
28 56 60 153 255 60 189 255
0 0 7 6 31 62 62 59
0 0 7 6 31 62 60 118
0 0 7 6 31 62 60 236
0 0 7 6 31 62 60 118]

PROC DLIST()
[112 112 70 0 0 7 130 16 68 0 0
4 4 4 4 4 4 4 4 4 4 4 4 132 4 4 4 132
4 65 DLIST]

PROC DLI()
[72 238 0 5 173 0 5 141 10
212 201 1 208 17 169 0 141 9 212 169
15 141 24 208 169 44 141 23 208 104
64 201 2 208 7 169 86 141 22 208 104
64 169 0 141 0 5 169 0 141 26 208
104 64]

PROC PLRVBI()
[162 3 189 244 6 240 89 56 221 240 6
240 83 141 254 6 106 141 255 6 142
253 6 24 169 0 109 253 6 24 109 252
6 133 204 133 206 189 240 6 133 203
173 254 6 133 205 189 248 6 170 232
46 255 6 144 16 168 177 203 145 205
169 0 145 203 136 202 208 244 76 87
6 160 0 177 203 145 205 169 0 145
203 200 202 208 244 174 253 6 173
254 6 157 240 6 189 236 6 240 48 133
203 24 138 141 253 6 109 235 6 133
204 24 173 253 6 109 252 6 133 206
189 240 6 133 205 189 248 6 170 160
0 177 203 145 205 200 202 208 248
174 253 6 169 0 157 236 6 202 48 3
76 2 6 76 98 228 0 0]

PROC YPOS()
[175 156 131 123 115 107 99 91 82 74
70 66 62 58 56 55 54 54 55 54 56 58
62 66 70 74 82 91 99 107 115 123 131
156 175 180]

PROC CLRS()
[44 246 72 56 76 15 146 8 146]

PROC BURN()
BRN=8-BRN POKE(1715,CH/256-BRN)
POKE(708,86+BRN/4) RETURN

PROC WAIT()
FOR K=1 TO 1000 DO OD RETURN

PROC SETUP()
```

```
GRAPHICS(0) POKE(82,0) POKE(752,1)
POKE(559,0)
CH=(PEEK(106)-8)*256
MOVEBLOCK(CH,CHSET,8*34)
MOVEBLOCK(CH+8*34,FIRE,13*8)
CH2=(PEEK(106)-16)*256
MOVEBLOCK(CH2,CHSET,8*34)
MOVEBLOCK(CH2+8*34,FIRE2,13*8)

DL=PEEKC(88) SC=(PEEK(106)-40)*256
POKEC(DLIST+3,DL) POKEC(DLIST+9,SC)
POSITION(23,0)
PRINT("trial by fire|") ZERO(SC,3072)
FOR I=0 TO 2 DO POKEC(88,SC+1024*I)
POKE(82,35) POSITION(35,15)
PRINT(";5?@5<5=>555555  98")
POKE(82,0) POSITION(0,1)
PRINTE("555555") PRINTE("555555")
PRINTE("556655") PRINTE("55  55")
PRINTE("555555") PRINTE("555555")
PRINTE("556655") PRINTE("556655")
PRINTE("555555") PRINTE("55BHIDKC")
PRINTE("55LEMFJ") PRINTE("55ELFG")
PRINTE("555555") PRINTE("555555")
PRINTE("557855") PRINTE("555555")
PRINTE("555555") PRINTE("555555")
POKE(82,8+I*9) POSITION(8+I*9,16)
PRINTE("!#$  %&'") PRINTE("()*++,-.")
PRINTE("/01  234") OD POKEC(88,DL)
MOVEBLOCK(1700,DLI,55)
POKE(1715,CH/256)
MOVEBLOCK(1536,PLRVBI,160)
PM=PEEK(106)-56 DRB=PM*256+1
ZERO(DRB+1024,1024) POKE(1788,PM+4)
POKE(53277,3) POKE(54279,PM)
POKE(1771,PM) SETBLOCK(1784,4,8)
SETBLOCK(1772,4,1) FOR I=0 TO 3 DO
MOVEBLOCK(DRB+I*256,BABIES,104) OD
MOVEBLOCK(704,CLRS,9)
HSCORE=0 POKEC(560,DLIST)
POKEC(512,1700) POKE(1280,0)
POKE(54286,192) POKEC(548,PLRVBI)
POKE(559,62) RETURN

PROC MAIN()
SETUP() LVL=1 DO
POKE(1772,1) POSITION(2,1)
POKEC(DLIST+9,SC+1024)
POKE(53248,128) POKE(1780,180)
PRINT("      Copyright 1987 Magnum")
PRINT(" Opus         ") BRN=0
DO FOR I=0 TO 32 DO
J-PEEK(I+YPOS) POKE(1780,J+13)
WAIT() BURN()
KEY=PEEK(53279) IF KEY=5 OR
 STICK(0)<>15 THEN LVL==+1
IF LVL>9 THEN LVL=1 FI POSITION(7,1)
PRINT("          Level: ") PRINTB(LVL)
PRINT("          ") FI
J=STRIG(0) POKE(77,0)
IF KEY=6 OR J=0 THEN EXIT FI OD
UNTIL KEY=6 OR J=0 OD POKE(53248,0)
SCORE=0 SCR=0 BSAV=0 MISS=0 ADV=0
BRN=0 NXTL=3*LVL BABES=LVL/3+1
UD=1-LVL MOD 2 POSITION(7,1)
PRINT("     Get ready...        ")
FOR I=1 TO 100 DO BURN() WAIT() OD
POSITION(2,1) PRINT("Misses: 0")
PRINT("   Score: 0      Level: ")
PRINTB(LVL)

DO
FOR I=1 TO 4 DO X(I)=0 Y(I)=0 B(I)=0
Q(I)=0 UP(I)=1 OD HIT=0
DO POKE(77,0) SNDRST() S=STICK(0)
IF S=11 THEN J=0 ELSEIF S=7 THEN J=2
 ELSE J=1 FI
POKEC(DLIST+9,SC+J*1024) BURN()
IF ADV=1 AND BSAV MOD NXTL=0 THEN
 LVL==+1 POSITION(36,1) PRINTB(LVL)
 NXTL=LVL*3 BSAV=0 ADV=2 UD=0 FI
IF LVL MOD 2=0 AND ADV=2 THEN UD=1 FI
IF LVL MOD 3=0 AND ADV=2 THEN
 BABES==+1 ADV=0 FI
IF LVL MOD 2=0 THEN ADV=0 FI
IF LVL>10 THEN UD=1-LVL MOD 2 BABES=4
 FI
IF SCR>SCORE THEN SCORE=SCR
 POSITION(22,1) PRINTC(SCORE) FI
FOR BABY=1 TO BABES DO
POKE(1772+BABY-1,Q(BABY)/4*8+1)
POKE(1780+BABY-1,Y(BABY)+8)
POKE(53248+BABY-1,X(BABY))
IF B(BABY)=1 THEN X(BABY)==+UP(BABY)
 Q(BABY)==+1
 Y(BABY)=PEEK(YPOS+Q(BABY))
IF Y(BABY)>175 AND X(BABY)=92 AND
 J<>0 THEN HIT=1 FI
IF Y(BABY)>175 AND X(BABY)=127 AND
 J<>1 THEN HIT=1 FI
IF Y(BABY)>175 AND X(BABY)=162 AND
 J<>2 THEN HIT=1 FI
IF Y(BABY)>175 AND HIT=0 THEN
 Q(BABY)=0 SOUND(0,200,10,10)
 UP(BABY)=1 IF UD=1 THEN
 UP(BABY)=RAND(2) FI SCR==+UP(BABY)
 FI
IF X(BABY)>195 THEN X(BABY)=0
 Y(BABY)=0 Q(BABY)=0 B(BABY)=0
 BSAV==+1 ADV=1 SCR==+LVL FI FI
IF RAND(25)<1 AND B(BABY)=0 AND
 (Q(1)<5 OR Q(1)>15) AND
 (Q(2)<5 OR Q(2)>15) THEN IF
 (Q(3)<5 OR Q(3)>15) AND
 (Q(4)<5 OR Q(4)>15) THEN X(BABY)=67
 Q(BABY)=10 B(BABY)=1 S=RAND(4)
 K=PEEK(CLRS+S) POKE(703+BABY,K) FI
 FI
 OD
IF LVL<11 THEN WAIT() ELSE
 FOR K=1 TO 2000-LVL*100 DO OD FI
IF PEEK(764)=33 THEN  DO POKE(77,0)
 SNDRST() UNTIL STRIG(0)=0 OD
 POKE(764,255) FI
UNTIL HIT<>0 OD

FOR BABY=1 TO BABES DO
IF Y(BABY)<176 THEN
 POKE(53248+BABY-1,0) FI
IF Y(BABY)>175 THEN
 POKE(1780+BABY-1,215) J=BABY
 FOR I=1 TO 1000 DO OD FI OD
MISS==+1 POSITION(10,1) PRINTB(MISS)
SOUND(0,200,8,12) FOR I=1 TO 60 DO
BURN() WAIT() IF I=3 THEN SNDRST() FI
OD BABY=9 FOR I=X(J) TO 255 DO BURN()
POKE(1772+J-1,BABY*8+1) BABY==+1
IF BABY=12 THEN BABY=9 FI
POKE(53248+J-1,I) WAIT()
POKE(764,255) OD IF MISS=3 THEN EXIT
 FI
OD

IF SCORE>HSCORE THEN HSCORE=SCORE FI
POSITION(1,1) PRINT("                      ")
PRINT("                              ")
POSITION(1,1) PRINT("Score: ")
PRINTC(SCORE) POSITION(13,1)
PRINT("High Score: ") PRINTC(HSCORE)
POSITION(30,1) PRINT("Level: ")
PRINTB(LVL) DO BURN() WAIT() UNTIL
STRIG(0)=0 OD FOR K=1 TO 10000 DO OD
POSITION(1,1) PRINT("                      ")
PRINT("                              ")
OD
```

Database DELPHI

by Michael A. Banks (KZIN)

You can customize DELPHI to suit the requirements of your terminal software and your personal needs.

Ah, winter! The time of renewal and preparation for spring. The time of kicking up the thermostat, kicking back and catching up on all the indoor activities you stayed away from during the summer. (Florida and Southern California readers excepted, of course! But, gee—isn't all that warm weather a drag?)

Those of you who aren't away on cruises or holidays in warmer climes are probably spending a lot more time with your computers than usual, mainly because there's nothing to do outside (assuming you've already shoveled the snow). If you're spending more time with your computer, chances are you're spending more time on DELPHI—in which case you'll appreciate the discussion at hand: how to customize DELPHI for more efficient use.

As You Like It

If you're like most computer users, you've probably customized your software, adjusting its parameters to match your system's configuration and your personal tastes. Undoubtedly, your communications software reflects your choice of screen colors, terminal emulation and other elements in how it interacts with you and how it works behind the scenes.

You can also customize DELPHI to suit the requirements of your terminal software and your personal needs. It takes only a little time and planning, and you can do it all from within ANALOG's Atari SIG.

The key to customizing DELPHI is the "Set Preferences" selection on ANALOG's Atari SIG menu. Type SET at the ANALOG〉 prompt, and you'll see this menu:

```
Preferences Menu:

Name Change
Editor Preference
Topic Selection
Settings (Profile)
HELP
Exit

PREFERENCES>(Name,Edit,Top,Set,Help,Exit)
```

The "Name Change" selection allows you to change the name you entered when you joined the SIG. (As noted in a previous column, this name is visible to others via the Entry Log.)

"Editor Preference" enables you to select the default editor (EDT or Oldie) that you'll use in Workspace and Forum within the SIG. (While you are in the SIG, this setting supercedes any editor settings you may have made elsewhere.)

"Topic Selections" leads to a menu from which you can change topics that are accessible to you in the Forum and databases.

"Settings (Profile)" is a direct pipeline to the SETTINGS system that you'll find as a selection in the Using DELPHI area off the DELPHI Main Menu. This is a rather complex system, which I'll explain here.

When you select the SETTINGS system (either here, or by typing US SET at the DELPHI Main Menu), you'll see this menu:

```
SETTINGS Menu:

BUSY-Mode                   PROMPT-Mode
DEFAULT-Menu                SET-High-bit
DOWNLOAD-Line-terminators   SLASH-Term-settings
ECHO-Mode                   TERMINAL-Type
EDITOR                      TIMEOUT
FILE-TRANSFERS              UTILITIES
KERMIT-SETTINGS             WIDTH (Columns)
LENGTH (Lines/page)         XMODEM-SETTINGS
NETWORK-PARAMETERS          HELP
PASSWORD (Change)           EXIT

SETTINGS>What would you like to set?
```

Here's an item-by-item explanation of each selection on the menu. The defaults for most of the items are what you will want to use (either because they work best, or because you selected them the first time you signed on to DELPHI). I've put a star by those items that you are most likely to want to change. (Note DEFAULT Menu in particular).

BUSY-Mode: This setting disables or enables pages and "New Mail" alerts, as well as one-line "sends." (These features are normally on.)

★ *DEFAULT Menu*: If you never touch any other settings, this is one default you should change.

Your Default Menu is the first menu you see when you sign on to DELPHI. Normally, this is the DELPHI Main Menu, but you can change it to ANALOG's Atari SIG, which means that you will bypass the Main and Groups & Clubs menus at logon and go directly to the Atari SIG.

To make the Atari SIG your default menu, simply type DEFAULT at the SETTINGS menu, then follow the prompts and enter GR AT as your new default menu. Every time you sign on thereafter, the first thing you'll see is the Atari SIG menu (and this will save you some valuable time).

DOWNLOAD-Line-terminators: This determines what will be sent as line terminator during text transfers. The choices are carriage return, linefeed, or both.

ECHO-mode: The echo-mode toggle is useful if your communications software happens to run in full duplex, or if for any reason you don't want DELPHI to echo what you type.

Changing your password frequently is important, and it's a quick and easy process.

★ *EDITOR*: This sets the editor—EDT or Oldie—that you'll use in the Forum, Mail and Workspace.

★ *FILE-TRANSFERS*: This selection leads to a sub-menu at which you can set your default file-transfer method.

KERMIT-SETTINGS: Kermit settings are changed at this menu.

★ *LENGTH (Lines/page)*: LENGTH sets the number of lines of text DELPHI will display before pausing with a "More?" prompt. (If you want a nonstop text display, set your length to zero.)

NETWORK-PARAMETERS: Here you set network and terminal configurations. This one's a bit complicated, but a menu is provided from which you can select pad parameters involving how the connecting network communicates with you.

★ *PASSWORD (Change)*: Changing your password frequently is important, and this is the place to do it. It's a quick and easy process; just follow the prompts.

★ *PROMPT-Mode*: As you become more skilled and knowledgeable in using DELPHI, you will find that you don't need to see full menus. DELPHI's designers were aware of this fact, and have arranged things so that you can "turn off" menu display with this setting. When you select PROMPT-Mode, you can select any of these prompt settings (examples of the prompts you'll see follow each):

1. Brief — SET>
2. Verbose — SET> (Brief, Verbose, Menu, Exit)
3. Menu — (The full menu)

This is another time-saver, because with the Brief or Verbose setting you will not have to wait for menus to scroll by as you move from one area to another.

SET-High-bit: This setting controls whether the high bit of each byte is set to 1 or left unchanged during an Xmodem or Kermit downloads. In general, this should be left unchanged.

SLASH-Term-settings: This selection leads to a subsystem that you can use to set certain terminal characteristics (i.e., how DELPHI communicates with you). Type /TERM at any prompt to see a list of these characteristics.

TERMINAL-Type: Here is where you set terminal emulation (VT52 or VT100). Certain features, like screen blanking before Forum messages are displayed, are enabled when you use terminal emulation.

TIMEOUT: The timeout setting is the number of minutes that DELPHI will remain connected with you if there is no input—useful if you are called away from your computer while you are online, and forget that you are online.

UTILITIES: This selection allows you to perform advanced settings operations.

★ *WIDTH (Columns)*: This sets the number of columns (letters) of text DELPHI displays on a line before moving to a new line. The range is 16 to 132.

★ *XMODEM-SETTINGS*: Here you can set the way DELPHI handles Xmodem file transfers. You can set the Xmodem error-checking mode, timeout period, number of retries, and line terminators for text transfers.

★ *HELP*: An obvious selection, HELP provides help with the SETTINGS system.

EXIT: Use EXIT (or Ctrl-Z) to exit this menu. (Note that Ctrl-Z or Ctrl-C cancels any selection, as long as you haven't made any changes permanent by pressing Return.)

By the way, many of these items are accessible, too, via "slash" commands (commands preceded by a /). Type /HELP at any prompt other than the Conference prompt to see a list of settings you can change.

And one final word about the Set Preferences menu: The HELP selection leads to a help sub-menu at which you can select any of the topics on the Set Preferences menu. When you select a topic, you have access to one or more articles on that topic.

That's it: everything you need to know about customizing DELPHI to work with you and make your sessions on DELPHI faster and easier.

ANALOG's Atari SIG Databases

Database offerings are growing like grass in the Spring, and you'll find programs from the most recent issues of ANALOG in the Current Issue database. And don't forget to check the Recent Arrivals database for new programs, text files and data files not yet moved to 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.

Make the DELPHI Connection!

As a reader of *ANALOG Computing*, you are entitled to take advantage of a special DELPHI membership offer. For only $19.95 plus postage and handling ($30 off the standard membership price!), you will receive a lifetime subscription to DELPHI, a copy of the 500-page *DELPHI: The Official Guide* by Michael A. Banks and a credit equal to one free evening hour at standard connect rates. Almost anyone worldwide can access DELPHI (using Tymnet, Telenet or other networking services) via a local phone call. Make the DELPHI connection by signing up today!

To join DELPHI:
1. Dial 617-576-0862 with any terminal or PC and modem (at 2400 bps, dial 576-2981).
2. At the Username prompt, type JOINDELPHI.
3. At the Password prompt enter ANALOG.

For more information, call DELPHI Member Services at 1-800-544-4005, or at 617-491-3393 from within Massachusetts or from outside the U.S.

DELPHI is a service of General Videotex Corporation of Cambridge, Massachusetts.

The Conference

Tuesday. 10 P.M., EST. The Atari Users' Group real-time conference.Be there—or be an obtuse rectangle.

(To join a real-time 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; 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: using DELPHI's online, interactive Help system. Until then, use DELPHI's alternate interactive help system: *DELPHI: The Official Guide*. 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 talk about custom software, custom cars and science fiction (among other things) by sending E-mail to membername KZIN on DELPHI.*

ANALOG READER SURVEY

In order to better tailor ANALOG to the needs of its readers, we ask that each of you please take a couple of moments to fill out the questionnaire below and send it to us at the address shown before March 15, 1989. If you don't want to remove this page from your magazine, it's okay to use a photocopy or to jot your answers on a separate piece of paper. Your assistance will be greatly appreciated. Thank you.

Please check the appropriate responses:

—EQUIPMENT OWNED:
- ☐ **130XE**
- ☐ **800XL**
- ☐ **600XL**
- ☐ **1200XL**
- ☐ **400/800**
- ☐ **ST**
- ☐ **Disk drive**
- ☐ **Cassette drive**
- ☐ **Modem**
- ☐ **Printer**

—COMPUTING EXPERIENCE (GENERAL):
- ☐ **Novice**
- ☐ **Intermediate**
- ☐ **Expert**

—COMPUTING EXPERIENCE (PROGRAMMING):
- ☐ **Novice**
- ☐ **Intermediate**
- ☐ **Expert**

—LANGUAGES OF INTEREST:
- ☐ **BASIC**
- ☐ **Assembly**
- ☐ **Action!**
- ☐ **C**
- ☐ **Logo**
- ☐ **None**

—WHAT TYPES OF PROGRAMS DO YOU LIKE?
- ☐ **Games**
- ☐ **Utilities**
- ☐ **Programming aids**
- ☐ **Home use**
- ☐ **Business use**
- ☐ **Educational**
- ☐ **Graphics**
- ☐ **Sound**

—DO YOU THINK ANALOG SHOULD PRINT PROGRAMS WHOSE LISTINGS ARE UNUSUALLY LONG IF THE QUALITY OF THE PROGRAM WARRANTS IT?
- ☐ **Never**
- ☐ **Sometimes**
- ☐ **Usually**
- ☐ **Always**

—WHAT TYPES OF ARTICLES ARE YOU INTERESTED IN?
- ☐ **Programming tutorials**
- ☐ **Reviews**
- ☐ **General interest**
- ☐ **Programs**
- ☐ **ST coverage**
- ☐ **Show reports**

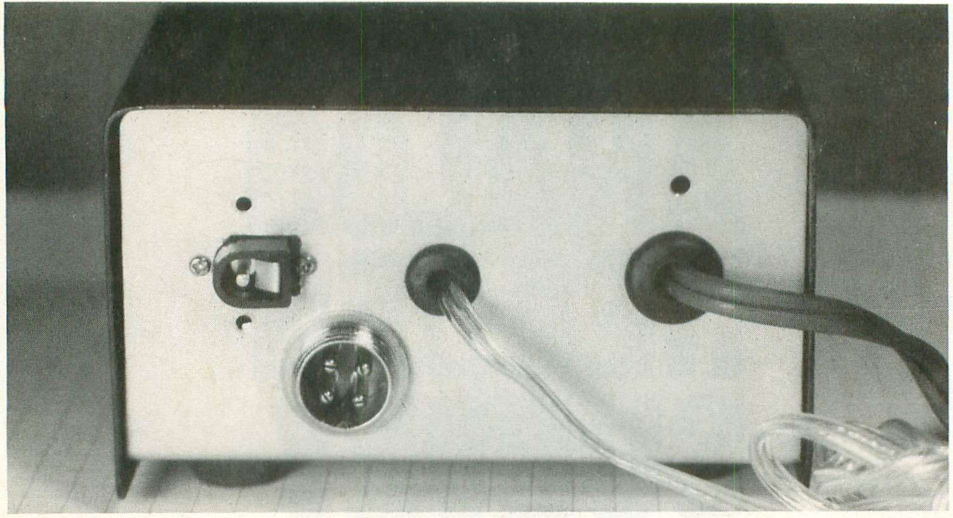
—HOW DO YOU FEEL ABOUT THE TECHNICAL LEVEL OF ANALOG?
- ☐ **Too advanced**
- ☐ **Just right**
- ☐ **Too easy**

—ARE YOU INTERESTED IN THE TYPE-IN PROGRAMS?
- ☐ **Not at all**
- ☐ **Somewhat**
- ☐ **For the most part**
- ☐ **Definitely**

— ARE YOU INTERESTED IN THE ASSEMBLY-LANGUAGE LISTINGS?
- ☐ **Not at all**
- ☐ **Somewhat**
- ☐ **For the most part**
- ☐ **Definitely**

SEND COMPLETED SURVEYS TO:

ANALOG Computing
P.O. Box 1413-M.O.
Manchester, CT 06040-1413.



continued from page 21

cells). These types are available in six- and 12-volt sizes in ratings of one to ten amp-hours. I used a 2.2-amp-hour battery which came with my videotape recorder, but Sears sells such batteries as do most electronics houses. Like NiCads, gel-cells need recharging every so often.

If you notice, nowhere do I provide circuitry for a battery charger. I assume that you will provide your own charger, depending on the batteries used. Dry cells are not rechargeable, NiCads need to be charged separately in their own charger, gel-cells use a different type of charger, while lead-acid types can use a standard automotive unit. So whatever battery you use, get the proper charger to go with it. S1b can be wired so the battery connects to its recharger while the power supply is switched off.

Connections on the Board

The 1050 disk drives, and 400/800 computers, do not use five-volt DC inputs. Instead, they need nine volts AC, which is difficult to produce from a battery. While it is possible to generate nine volts AC from 12 volts DC, it requires you to build your own transformer and inverter, and I feel this is beyond the expertise of the average experimenter. Since most power outages are of short duration, having one to two hours of power to your computer should handle the problem without needing to run the disk drive. For portable operation you can buy just an inverter without the UPS function. The most commonly sold inverters are Tripp-Lite inverters made by the Tripp Co. in Chicago, Ill. A 200-watt model PV-200 costs about $79, and is available from Sears and Jameco. Also, some electronic houses can get them for you.

But if you really need to have portable power to a disk drive or want to have the option of saving data to disk before your batteries go dead then never fear; there are simple ways. You can put about 18 volts DC into the power plug (polarity is critical), and the disk drive will run. I don't recommend this method because it puts additional load on the five-volt regulator and can cause thermal shutdown over long periods on battery. This also means you need a separate 18-volt battery for the disk drive besides the 12-volt one for the computer.

Another approach is easy to do but requires you to open the case of your 1050 drive after it is out of warranty (sorry 810 owners, I don't have the info to know if this works on your drives). First remove all cables and disks from the drive. Place it on its top, and remove the four screws in the deep wells. Now turn it right-side-up, and catch the screws as they fall out. Remove the top cover by lifting it *up* at the rear then pulling toward the rear. Now refer to Figure 4. This is a drawing of the rear portion of the circuit board as viewed from the top and also a schematic of the disk-drive power supply. You need not make any modifications or remove the board. Instead use three mini test clips with two-foot lengths of 22-gauge or larger wire to attach to CR18, CR16, and CR19 (A, B, and C respectively) as shown on the diagrams.

The drive has two power supplies, the first one being a five-volt supply almost identical to the one we built. Number 2 is a voltage doubler which sends about 18 volts to the 12-volt regulator. The connection to CR18 is at the input to the five-volt regulator, CR16 at the 12-volt regulator and CR19 is ground. For this reason, applying 18 volts to the power-input plug will provide about 16 volts to point A and 14 volts to point B, enough to run the drive. Normal input to these points are 10 and 18, respectively, so the excess voltage at A can cause excessive heat on the five-volt regulator. Bring the wires out through a ventilation slot and close up the case. Now you can connect the three wires to the battery, using P3, as shown in Figure 5.

You can use the first circuit for full-time portable operation or the second, which is a modification of Figure 3, for UPS backup

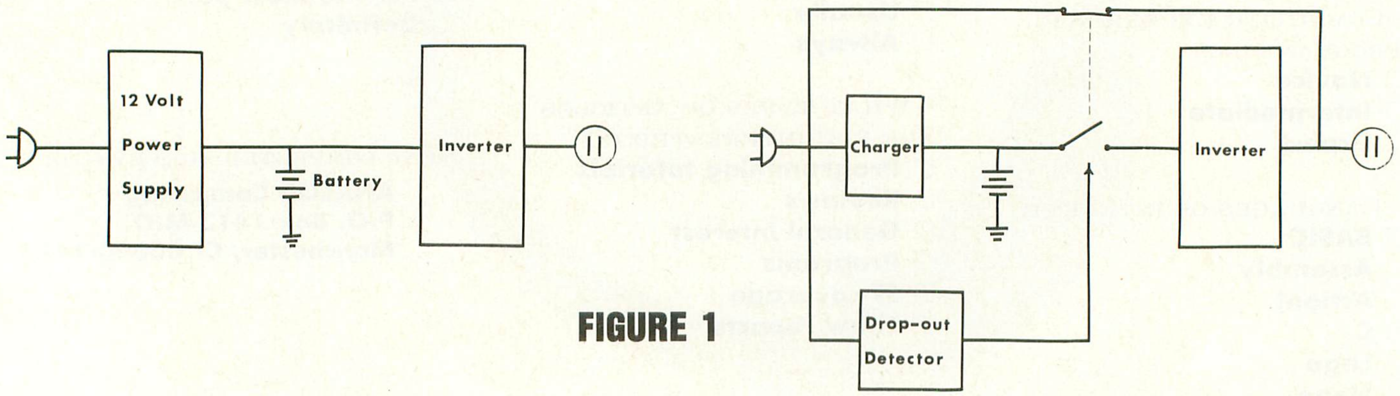


FIGURE 1

operation. Be careful not to apply the normal nine volts AC power and the battery power at the same time. No damage would occur for short durations, but running for long periods could hurt the drive power supply and/or the battery. Using the same battery to run both the computer and the disk drive consumes about 2.5 amps while the drive is running and about 1.75 when it is idle. With my 2.2-amp-hour battery, I can now run about one hour and format and write about ten full disks. That should be plenty of time to get offline in a power outage.

Using Figure 5b, power transition to battery is instantaneous for the computer but merely fast for the disk drive. As long as there is AC power, RY1 stays pulled in and the disk drive works normally. When the AC drops out, the relay switches in the battery. The on/off switch in the drive won't turn off the drive on battery but does disconnect the nine-volt supply. Disk speed will probably be off, and you might want to check it. Mine runs between one and three RPM too slow, but this still is good enough to work.

Be sure to use write verify! Some programs and DOS versions do not verify writes and so if your speed is off, you may get bad writes and not know it until it is too late!

Additional Notes

The 800 has the identical power supply to the disk drive except it has an additional -5 volt supply. This requirement cannot be met with one battery. You would either need to use two sets of batteries or resort to using AC. For this you can either use a power inverter or purchase a special product to do the job. Part #DCONV2 from All Electronics is

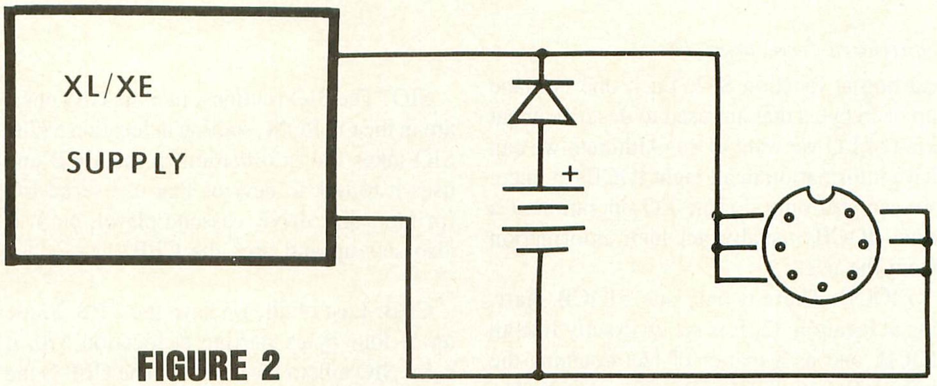


FIGURE 2

a +/− 5V converter and would do the job of producing the five volts from the single battery. But this would have to be installed in the power-supply board and would be a difficult approach; I don't recommend it unless you know what you are doing.

Even though you can run your computer and your disk drive on batteries, unless you can see what you are doing, you are dead in the water. I am not aware of any 12-volt composite monitors, but you can buy small portable televisions which run on batteries, most of which will run directly off 12 volts. So I leave it to your imagination to solve this problem. So now you can build your own version of the portable computer that Atari never made by using the basic circuitry in Figure 3. Make the three-amp upgrade using the components listed in the parts list, and with the additional power available you can connect the computer, disk drive and TV to our UPS, connecting to the junction of D1 and D2. A 130 XE, 1050 drive, a micro TV and this power supply can fit into a large attaché or small suitcase. Your outfit can now run full time off either one single plug-in supply, gel-cell battery, or substitute a cigarette lighter plug for the battery and run off car power!

So if you find a need to have inexpensive protection against blackouts and brownouts or you want to have portable Atari power, then this project is for you. No more will you worry about people tripping on cords or lightning-induced power flickers.

PARTS LIST

B1	12-volt battery. See text: 10 "C" 1.2-amp-hour NiCads (RS 23-124) with charger (23-132) or 10 "D" 4-amp-hour NiCads (RS 23-140) with charger (23-138) or 8 "D" alkaline cells (RS 23-550) or 2 lantern batteries (RS 23-016) or 1 12-volt lantern battery (DSE S-3242) or 1 12-volt 1.2-amp-hour gel-cell (DSE S-3315) or

continued on page 64

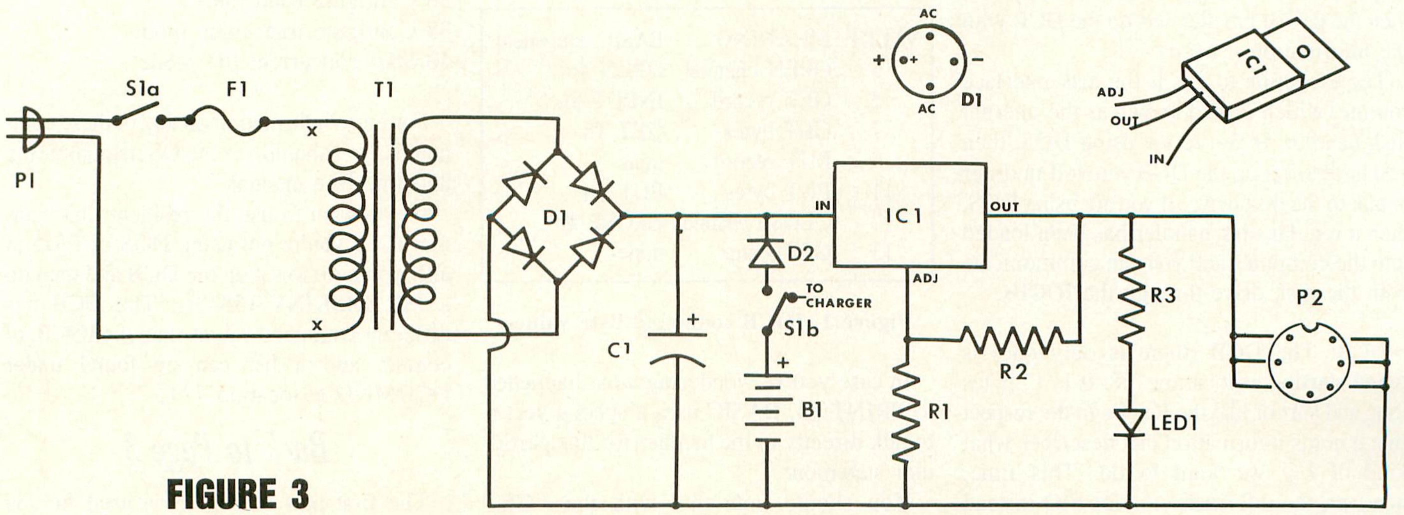


FIGURE 3

continued from page 17

starting at location 832. Each one is made up of 16 bytes that are used to describe what kind of I/O we want to do. Although we can have information in all eight IOCBs at once, we can still only do one I/O operation at a time. IOCBs usually get their information from the user.

ZIOCB: There is only one ZIOCB, starting at location 32. It is set up exactly like an IOCB, and as a matter of fact contains the information for the IOCB that is currently being used. Why is it necessary? Because page 0 is faster than regular memory, and we want to do I/O as quickly as possible. Since only one IOCB can be used at any one time, it would be a waste of precious page 0 memory to put all eight IOCBs in page 0. CIO transfers the information from the IOCB to the ZIOCB.

CIO: The CIO routine can be found in the OS ROM, starting at location 58534. CIO takes the information in the IOCB that is currently being used and stores it in the ZIOCB. It then uses that information along with HATABS (794) to figure out which device handler is needed and then passes control to the device handler.

Device Handler: Again, it should be device handlers, since there is one for each device. The device handlers can be anywhere in memory, with HATABS containing a list of where to find them. A device handler does one of two things. If the device in question is the keyboard, the screen or the screen editor, then the device handler takes care of the I/O itself. If it's something like the disk drive or printer that's plugged into the computer, then the device handler sets up the DCB with the information it needs.

The exception to this is the disk interface routine, which is also known as the internal disk handler. If you're not using DOS, then you have to set up the DCB yourself in order to talk to the disk drive. If you are using DOS, then a regular disk handler has been loaded into the computer and you can communicate with the disk drive through the IOCBs.

DCB: The DCB (there is only one) is found starting at location 768. It is 12 bytes long and sort of like the IOCBs in the respect that it holds information that describes what kind of I/O we want to do. This time, however, the information is for SIO instead of CIO.

SIO: The SIO routines, like the CIO ones, are in the OS ROM, starting at location 59716. SIO takes the information in the DCB and uses it to talk to devices that use serial I/O (printer, disk drive, cassette player, etc.). It also sets up and uses the CFB.

CFB: Last of all, we have the CFB. Made up of four bytes starting at location 570, it helps SIO talk to the devices. The CFB is the one part of the I/O system that you should not mess around with yourself.

Let's summarize by looking at the flow of information. The user sets up an IOCB and calls CIO. CIO takes the information in the IOCB, transfers it to the ZIOCB, figures out what device handler is needed and transfers control to that handler. From there, the handler takes care of internal devices, or sets up the DCB and calls SIO if we need to communicate with a serial (external) device. Finally, SIO takes the information in the DCB, sets up the CFB and does the I/O. After the I/O has been completed, whether by the handler or SIO, control is transferred back to the user. As a programmer, you can skip any of these steps with the exception of SIO. You should *not* skip all the way to the CFB.

BASIC programmers may want to look at the excellent article in issue 13 of ANALOG Computing for ways to go straight to CIO.

IOCB Command Byte Values

The third byte in each IOCB is the command byte, called IOCMD. This is the byte that tells CIO what kind of I/O we want to do. It can have the values in Figure 1.

VALUE	MEANING	BASIC equivalent
3	OPEN channel	OPEN #n
5	GET record	INPUT #n
7	GET bytes	GET #n
9	PUT record	none
11	PUT bytes	PUT #n
12	CLOSE channel	CLOSE #n
13	GET status	none

Figure 1. IOCB command byte values

In case you're wondering what happened to PRINT #n, BASIC uses a special vector to talk directly to the handler for this particular statement.

You should note that with the "GET bytes" and "PUT bytes" routines, BASIC only allows you to GET and PUT one byte at a time.

If you access CIO directly, however, you can GET or PUT a whole buffer. With this in mind, you may ask what the difference is between GETting or PUTting a buffer and GETting or PUTting a record. If there are no carriage returns (End-Of-Lines or EOLs) in the buffer then there is no difference. A record, however, ends when either the end of the buffer is reached or an EOL encountered.

CIO takes care of all of these commands. Some of the devices also have their own special commands, which the corresponding device handler takes care of. Here's a list of those commands, which you can access using BASIC's XIO command:

Display Handler

17 DRAW line
18 FILL

Disk File Manager

32 RENAME file
33 DELETE file
35 LOCK file
36 UNLOCK file
37 NOTE
38 POINT
254 FORMAT disk

RS232 Handler

32 Force short block
34 CONTROL DTR, RTS, XMT
36 Configure baud rate
38 Configure translation mode
40 Start concurrent I/O mode

For more information on any of these commands, you should see the OS manual or the 850 Interface manual.

If you want to use the resident disk handler (i.e., you're not using DOS or FMS at all), you have to set up the DCB and then do a JSR DISKINV (58451). The DCB has different command values than the IOCB, of course, and a list can be found under DCOMND at location 770.

Bac' to Page 3

The first part of page 3 is used for the "device handlers." As the name implies, device handlers are used to handle I/O to the various devices. What devices can we have? The screen (S:), the screen editor (E:), the keyboard (K:), the cassette player (C:), the disk drive (D:), the printer (P:), and the RS232 ports on the 850 interface (R:); all of these handlers, which are just machine language routines, are a part of the OS, with the exception of the RS232 handler. The RS232 handler is stored inside the 850 interface, and gets transferred over to the Atari when you turn on the system.

Locations 768 through 780 make up the Device Control Block (DCB). To use the DCB you must set it with the appropriate values and then JSR DSKINV (58451) for disk I/O, or JSR SIOV (58457) for other device I/O.

DDEVIC
768 0300

Three of the devices, S:; E:, and K:, are a part of the computer. The others are all outside the computer, and we therefore need to have some way of talking to them. The "serial bus" takes care of that (the cords you use to connect the devices together are the visible part of the serial bus). But, since you can have more than one device hooked up to the serial bus, you need some way of telling the bus which device you want to talk to. Each device is therefore assigned a number (think of it as a phone number), and the handler gives DDEVIC the number of the device it wants to talk to. *Do not change DDEVIC yourself.*

Here are the numbers for the various devices in Figure 2.

Disk Drive	49	($31)
Printer 1	64	($40)
Printer 2	79	($4F)
RS232 Port	80	($50)
Cassette	96	($60)

Figure 2. DDEVIC chart

DUNIT
769 0301

We can have up to four disk drives and RS232 ports. DUNIT holds the number of the disk drive, printer or RS232 port we want. DUNIT gets added to DDEVIC, and the result stored in CDEVIC at location 570. CDEVIC is then used during the actual I/O.

If you are using DOS, then a regular disk handler has been loaded into the computer and you can communicate with the disk drive through the IOCBs.

DCOMND
770 0302

Once it has got the number of the device we want to talk to, the handler has to know what it should tell the device to do. For that we have another bunch of numbers, this time for the various commands in Figure 3.

Get Sector	82	($52)
Put Sector (with verify)	87	($57)
Put Sector (w/o verify)	80	($50)
Get Status	83	($53)
Format Disk	33	($21)
Download	32	($20)
Read Address	84	($54)
Read Spin	81	($51)
Motor On	85	($55)
Verify Sector	86	($56)

Figure 3. DCOMND chart

This is one of those tables that gives you the confidence that you know what's going on, until you get about halfway down. The first five commands are probably the only ones you'll ever run into, so don't worry too much about it.

DDEVIC gets transferred over to CDEVIC (570) for use by SIO.

DSTATS
771 0303

Two uses for DSTATS. First of all, after an I/O operation is complete, it holds the status of the operation. A zero means that everything went OK. See the OS manual for the meaning of non-zero values.

Before the I/O operation, DSTATS tells SIO how data is going to be transferred, using bits six and seven as in Figure 4.

00------	($00) means no data will be transferred in this operation.
01------	($40) means data is going to be read from the device.
10------	($80) means data is going to be written to the device.
11------	($00) is not a valid combination.

Figure 4. DSTATS chart

DBUFLO, DBUFHI
772,773 0304,0305

This is a pointer to the buffer that will be used to store the data that is to be sent or received during I/O(!). It's set by the handler to the system buffer at CASBUF (1021) unless you tell the handler differently.

If a GET STATUS command is given, then DBUFLO/HI is set to point to DVSTAT (746).

If you're communicating with SIO directly, you should make sure you set DBUFLO/HI yourself.

DTIMLO
774 0306

DTIMLO is the time-out value for the device being used and is set by the handler. You will recall from our other run-ins with time-outs that a value of 60 here represents 64 seconds.

DTIMLO is initialized to 31.

DUNUSE
775 0307

Another unused byte (warning, warning!).

DBYTLO, DBYTHI
776, 777 0308, 0309

This location specifies the number of data bytes that are to be read to, or written from, the buffer during I/O. It is also used by the FORMAT command to store the number of bad sectors.

The values in DBUFLO/HI and DBYTLO/HI are added together after the I/O is over, and the results stored in BFENLO/HI (52,53).

Just in case you thought the OS was perfect, there's a bug that messes things up if the last byte in the buffer is in an address that ends in $FF (such as $41FF, $32FF, etc). Be careful about this.

DAUX1, DAUX2
778, 779 030A, 030B

These are used to provide information that is unique to the specific device (a sector number, for example). Their values are transferred to CAUX1 and CAUX2 at locations 572 and 573.

The next 14 locations (780 to 793) have various SIO uses.

TIMER1
780, 781 030C, 030D

TIMER1 is the initial timer value for the baud rate. What does that mean? Back at CBAULDL/H (750, 751) we discovered what a baud rate is and how the OS constantly adjusts it during I/O. We found out that an alternating bit pattern is read and timed in order to figure out the correct rate. TIMER1 stores the time at the beginning of this pattern, and TIMER2 below stores the time at the end of it. The difference in these times is then used to figure out the new baud rate.

The first byte of both TIMER1 and TIMER2 is the value of VCOUNT (54283) at the time, and the second is the value of RTCLOK+2 (20).

ADDCOR
782 030E

ADDCOR is an "addition-correction flag" used in the baud-rate calculations. Those quotation marks mean that you'll never need to know what it means.

CASFLG
783 030F

Part of the SIO routine is not needed for cassette I/O, so CASFLG is used to warn SIO that cassette I/O is being done. A value of zero means regular SIO, 255 means cassette.

TIMER2
784, 785 0310, 0311

This is the final timer value for baud-rate. See TIMER1 for a complete description.

TEMP1
786, 787 0312, 0313

TEMP1 is used as a temporary storage location for the difference in the TIMER1/2 values during the baud-rate calculation.

TEMP2
788 0314

Supposedly another temporary storage location of some sort, but according to the OS listing it isn't used.

TEMP3
789 0315

Another temporary storage location that is used, but for nothing particularly important.

SAVIO
790 0316

Back to setting the baud rate. Remember the alternating bit pattern (see TIMER1 if not)? SAVIO is used to check the serial port SKSTAT (53775) to see if the next bit has come in yet. That's all.

TIMFLG
791 0317

This is a flag to indicate that the cassette player has timed out (taken a snooze). If it's equal to one, we're OK. If it's equal to zero, then we're in time-out territory.

For the cassette player to time-out, a data byte must not be found within the given time period (which can vary). This usually indicates that the baud rate was wrong, assuming that you remembered to connect the cassette player, plug it in, put in the program tape and press "Play"!

STACKP
792 0318

Remember the stack at page 1? When SIO first gets going, it stores the value of the stack pointer in STACKP. That way, if somebody presses BREAK before it's done, it can restore the stack pointer and return to where it was called from.

TSTAT
793 0319

This is used to temporarily hold the value of STATUS (48) during I/O.

HATABS
794-831 031A-033F

We now know a little about what handlers do, but where do we find them? Obviously HATABS is going to have something to do with it, but before I tell you what, let's talk a little more about handlers.

Each handler is made up of a bunch of routines that perform different I/O functions. These functions are shown in Figure 5.

OPEN device
CLOSE device
GET BYTE from device
PUT BYTE to device
GET STATUS of device
SPECIAL
INITIALIZE device

Figure 5. I/O functions chart

Since SIO is going to need to know where each of these routines is, it's useful to keep the address of each routine in a table. We'll only need the initialization routine once, so we'll put a JMP instruction in the table right before the initialization address. Finally, we'll call this table the "handler entry point" table, which makes sense if you think about it.

Okay, so now we have a handler entry point table for each of our handlers. Now we need a table of the addresses of these tables (aren't computers fun?). This is where HATABS comes in. Each entry in HATABS consists of the ATASCII value of the one character device name ("C", "D", "K", etc.), followed by the address for the handler entry point table for that device. So, keep in mind that even though HATABS is called the "handler address table," it is actually the handler entry point table address table!

When you first turn on the computer, five entries are automatically set up in HATABS. They are for the printer, cassette player, screen editor, screen and keyboard handlers, in that order. If a disk drive is hooked up and turned on, then the entry for the disk handler is added. Finally, if the 850 Interface is hooked up and on, the entry for the RS-232 handler is added after that for the disk. The addresses for the handler address table of each of these are shown in Figure 6.

"P"	58416
"C"	58432
"E"	58368
"S"	58384
"K"	58400
"D"	1995
"R"	varies

Figure 6. Addresses for handler address table

The address for "R" varies depending on whether you have a disk drive hooked up and, if so, what kind of DOS you are using. PRINT PEEK(813)+256*PEEK(814) will give you the address for your particular setup.

You can use the preceding addresses to take a look at the handler entry point tables. The addresses in these tables are in the same order that the routines were listed (OPEN, CLOSE, etc.). Don't forget that each address is two bytes long with the exception of the last one, which includes a JMP instruction (76).

HATABS is 38 bytes long, which means there is room for 12 three-byte entries (the last two bytes are set to zero and ignored). Even if you are using the disk and RS-232 handlers, that still leaves five entries free. These entries are initially set to zeros, but they're free for your use if you want to write your own handler.

Since the task of writing your own handler is one that most people won't really get into, I'm not going to go into any more detail on it here. If you're interested, *De Re Atari* and the OS manual should provide all the information you need.

One more thing you'll need to know. CIO searches for a handler address from the end of the table up to the beginning. This means that if you write your own screen handler, for example, CIO will use it instead of the original one.

Input/Output Control Blocks (IOCBs)

Back at locations 32 through 47, we ran across something called the Zero-page Input/Output Control Block (ZIOCB). The ZIOCB gets its values from one of the eight Input/Output Control Blocks (IOCBs) located at locations 832 through 959. Basically, the IOCBs are nothing more than a bundle of information used to communicate between the user and the handlers, BASIC usually takes care of them for you in commands like OPEN, PLOT, LPRINT, and so on (all the BASIC I/O commands).

Each IOCB is 16 bytes long, and those bytes are named and used as follows:

ICHID (one byte): This is an offset into HATABS (794 through 831) that points to the name of the device that the IOCB is OPENed for. For example, try the following:

```
100 IOCB1=848:HATABS=794
110 OPEN #1,4,0,"K:"
120 INDEX=PEEK(IOCB1)
130 PRINT "You just OPENe
d device ";CHR$(PEEK(HATA
BS+INDEX));":"
```

ICHID is set by the OS.

ICDNO (one byte): This is the device number. One for D1, two for R2, etc. It is also set by the OS.

ICCOM (one byte): This is the command that specifies what kind of I/O operation we are going to be doing. It is set by the user.

ICSTA (one byte): This is the status of the last I/O operation.

ICBAL/H (two bytes): This is either the address of the data buffer, or the address of the user's filename (depending on the command).

ICPTL/H (two bytes): This is the address minus one of the put-one-byte routine for the device being used. If the IOCB isn't OPEN, then it points to CIO's error routine for an illegal put.

ICBLL/H (two bytes): This is the number of bytes that still have to be transferred. Note that under some circumstances not all bytes *will* be transferred.

ICAX1 (one byte): This is an auxiliary byte (and is also called AUX1), meaning that it helps out ICCOM in specifying what is to be done. It is usually used to modify the OPEN command, but you can use it for your own handlers. With the OPEN command, it is the first value after the IOCB number (#n is the IOCB number), with the bit meanings in Figure 7.

-------1	(1)	append
------1-	(2)	directory
-----1--	(4)	read
----1---	(8)	write
--1-----	(32)	OPEN screen without erasing screen memory

Figure 7. ICAX1 bit meanings

Some combinations are not allowed on some devices. For example, OPEN#1,12,0, "D:TEST" would open a disk file called TEST, and let you read *and* write to that file. This wouldn't work on a cassette file though.

ICAX2 (one byte): This is the second auxiliary byte, and is also called AUX2. There is no common use for this or any of the other auxiliary bytes; their use depends on the handler. For example, if AUX2 is equal to 128, the cassette handler will put shorter gaps between the records on the tape when it writes data.

ICAX3/4 (two bytes): These auxiliary bytes are used by BASIC's NOTE and POINT commands to keep track of the sector number. They are *not* also called AUX3 and AUX4.

ICAX5 (one byte): This is the fifth auxiliary byte and is also used by NOTE and POINT as the number of the byte within the sector.

ICAX6 (one byte): Okay, enough of the "this is" garbage. I won't even insult your intelligence by telling you it's the sixth auxiliary byte. It has no specific use.

You can use the IOCBs directly by POKEing the values you want into them and then doing a JSR SIOV (58454). See SIOV for more details.

Note that the descriptions for the ZIOCB (32-47) are worded differently from the preceding descriptions, so be sure to read them as well for a better understanding.

IOCB0		
832-847	0340-034F	

This is, obviously, IOCB zero. If you're using the screen editor, then don't use IOCB0; that's what the screen editor uses. If you *are* using the screen editor though, you can do neat things by telling the IOCB to send the data somewhere else, like the printer. Try this if you have a printer:

```
100 GRAPHICS 0
110 PRINT "Now we're on t
he screen"
120 POKE 838,166:POKE 839
,238
130 PRINT "Now we're on t
he printer"
140 POKE 838,163:POKE 839
,246
150 PRINT "Now we're back
 on the screen"
```

What we're doing here is changing ICPTL/H to point to the printer's put-one-byte routine rather than the screen editor's. Note that this *doesn't* turn your computer into a typewriter. The screen editor isn't responsible for putting characters you type on the screen; it only works for things the computer prints on the screen.

Another neat thing you can do to the screen editor is give AUX1(842) a value of 13. This tells it to "append," which doesn't really make any sense. What it does, though, is act as though you were continually pressing the Return key. This lets you write a program that will change itself. You simply print some program instructions on the screen, position the cursor on the line of the first instruction, and POKE 842 with 13 to start the computer generating Returns, which reads in each line. When the computer POKEs 842 with a 12, the process stops and everything is back to normal. While you do this, the lines of code will appear on the screen, but you can make them invisible by setting the color of the letters to the color of the screen. Try this:

```
10 ? CHR$(125):LIST 30
20 FOR I=1 TO 1000:NEXT I
30 ? " THIS IS LINE 30"
40 FOR I=1 TO 1000:NEXT I
50 ? CHR$(125)
60 POSITION 2,10:? "  30
?";CHR$(34);" NOW LINE 30
 SAYS SOMETHING ELSE"
70 POSITION 0,2
80 POKE 842,13
90 POSITION 2,17:? "CONT"
100 POSITION 0,2:STOP
110 POKE 842,12
120 ? CHR$(125):? "NOW LE
T'S SEE WHAT LINE 30 SAYS
"
130 ? "THE PROGRAM MODIFI
ED ITSELF!"
140 LIST 30
```

Keep in mind that if you try and delete the lines that change location 842, you'll confuse the heck out of the computer and it will just keep on "pressing" Return forever!

The screen editor, and therefore IOCB zero, is used in graphics mode 0 and in other graphics modes that use text windows. Since IOCB zero is dedicated to the screen editor, however, you should stay away from it even if you're not using the text editor.

IOCB zero is not closed by a NEW, RUN or LOAD command. All the others are.

IOCB1	
848-863	0350-035F

IOCB one.

IOCB2	
864-879	0360-036F

IOCB two.

IOCB3	
880-895	0370-037F

IOCB three.

IOCB4;	
896-911	0380-038F

IOCB four.

IOCB5	
912-927	0390-039F;

IOCB five.

IOCB6	
928-943	03A0-03AF

IOCB six. If you're in a mode other than zero, then IOCB six is used for the screen (IOCB is used for the text window).

IOCB7	
944-959	03B0-03BF

IOCB seven is used by BASIC for I/O to the printer, disk drive and cassette. That means that this is a pointer to the buffer that will be used to store the data that is to be sent or received during I/O (!). It's set by the handler to the system buffer at CASBUF (1021) unless you tell the handler differently.

If a GET STATUS command is given, then DBUFLPO/HI is set to point to DVSTAT (746).

If you're communicating with SIO directly, you should make sure you set DBUFLO/HI yourself.

PRNBUF	
960-999	**03C0-03E7**

This is the print buffer, 40 bytes long, used in sending data to a printer. See PBPNT (29; $001D) and PBUFSZ (30; $001E) for details on how this works.

Forty bytes, as you may be aware, is somewhat shorter than most printer lines (most have 80 character lines). The OS can usually handle this, but sometimes it runs into problems. Semi-colons and commas at the end of LPRINT statements especially tend to mess it up. Several sources briefly mention that the Atari can deal with an 80 column printer if you call it "P2." If this

is true, then you have to OPEN a special IOCB for it and therefore couldn't use LPRINT (you'd have to use PUT and the likes). You're probably better off just to put up with the quirks.

Noname
1000-1020 03E8-03FC

These bytes are marked as being spare, but again, be careful about using them.

CASBUF
1021-1151 03FD-047F

This is the cassette buffer, which is where the cassette handler reads and writes data from and to. It's also used to hold the first disk record when a disk is booted (the OS doesn't know where to put the disk file in memory until it gets a chance to look at this record; see BOOTAD [578, 579]).

A cassette record is made up of 132 bytes. Only 128 of these are actual data; the other four help out the cassette handler. How? I'm glad you asked. The first two bytes, as we learned at CBAUDL/H (750, 751), are used to help the handler figure out the correct baud rate. The third byte tells the handler how much data is in the file. It can have the following values:

A value of 250 means that there are less than 128 bytes of *meaningful* data (there will still be 128 bytes, but some of them toward the end will be zeros). The 128th data byte will give the actual number of meaningful bytes.

A value of 252 means that all 128 bytes are important.

A value of 254 means that this is the last record in the file and all 128 bytes will be equal to zero.

The next 128 bytes are the actual data. Notice that they will be stored in CASBUF starting at 1024 and ending at 1151.

But wait, that was only 131 bytes and you said there were 132. Where does the 132nd go if we already filled the buffer? The last byte in a cassette record is the checksum, which is used to make sure that the rest of the data was read correctly. It gets stored at CHKSUM (49), and you should take a look at CHKSUM for a more detailed description of how a checksum works.

Take a look at BPTR (61) and BLIM (650) for more information on the way the buffer is used.

Locations 1152 through 1791 are not used by the OS. Most of them are, however, used by BASIC and/or the floating point package. Only the locations in page 6 (1536 to 1791) are not used by either. See page 6 for more information.

SYNSTK
1152-1405 0480-057D

This is BASIC's syntax stack. Unfortunately, since there doesn't seem to be any information on what it's for, I can't explain it to you (I never claimed to be perfect). I suspect, however, that it's used during the tokenization of the BASIC program, since BASIC has the runtime stack (see RUNSTK [142,143]) to use when the program is actually running.

LBPR1
1406 057E

LBUFF prefix 1. Again, no information on this one.

LBPR2
1407 057F

LBUFF prefix 2, also not explained anywhere.

LBUFF
1408-1535 0580-05FF

Before I go on, a few words on locations like these. Atari was very nice in releasing the OS listing; a lot of other computer companies don't. Atari did not, however, because of legal restrictions, extend that niceness to BASIC and the floating point package. Therefore, locations that are used by these two are very difficult to explain. The useful locations *have* been documented, so they can be understood and used by yourself. Ones like these, however, are somewhat obscure, so that you should never have to use them. In other words, don't feel that you're not getting something you'll need.

Now that I've freed myself from the responsibility of properly explaining these locations, I'll actually give you some information on this one. This is a buffer used in converting floating point values to ATASCII values. It's pointed to by INBUFF at locations 243 and 244. Now INBUFF supposedly points to the buffer used to convert ATASCII to floating point, so I suspect that LBUFF swings both ways.

LBUFF is also referred to as the "input line buffer," which implies that this is where a BASIC line is stored when you first type it in.

Location 1535 is the last byte in the buffer and so it is also called LBFEND. Notice that the next three locations are all within LBUFF.

PLYARG
1504-1509 05E0-05E5

Polynomial arguments. Sure!

FPSCR
1510-1515 05E6-05EB

This is like a work area for the floating point package.

FPSCR1
1516-1535 05EC-05FF

The same, only bigger.

Page 6

Locations 1536 through 1791 are normally not used by the OS, BASIC or the floating point package. That leaves them free for your use. (Page 6 is a good place to store a machine language routine.) Now I did say that they are "normally" not used. That means that they're not completely safe. If you try and INPUT more than 128 bytes during I/O, then the extra bytes are stored in page 6. That means one of two things. Either don't INPUT more than 128 bytes at a time, or only use the second half of page 6 (locations 1664 through 1791). These locations are absolutely guaranteed not to be used by anything no matter what.

Please notice that I only said the OS, BASIC and the floating point package wouldn't use page 6. If you are using another language, it might, so check the documentation that comes with it.

Page 7, 8, 9. . . .

If you're not using a disk drive, then location 1792 is the beginning of free memory. If you're using BASIC, "free memory" doesn't mean memory for you to use; it means memory for BASIC to use. There's a difference between the two, and you should go back to locations 128 through 145 if you don't know what it is.

If you are using a disk drive, then the

locations from 1792 to the address stored in MEMLO (743, 744) are used by DOS and the File Management System (FMS). The value of MEMLO will depend on the version of DOS that you're using, and also on a couple of other things that will be mentioned next. Use PRINT PEEK (743) +PEEK(744)*256 to find out the value for your particular setup.

This column is designed to teach you about your Atari and not about DOS, so I'm not going to go into any detail about how DOS works or what the locations are for. If you're interested in DOS, take a look at COMPUTE's book *Inside Atari DOS*.

Free RAM

The memory area from the address pointed to by MEMLO (743, 744) up to that pointed to by RAMTOP (106) is free RAM. That doesn't mean you didn't pay for it; it means that it is unused. If you are using BASIC, then your program uses the memory from the address pointed to by MEMLO up to the address pointed to by MEMTOP (144, 145).

As mentioned already, the value of MEMLO will depend on whether or not you are using disk (and the RS-232 handler, which takes up another 1728 bytes). The value in RAMTOP depends on how much total memory you have. The various values it can have are listed in Figure 8. Don't forget that the value in RAMTOP is the high byte of the address (the number of pages).

MEMORY	RAMTOP	BYTES
8K	32	8192
16K	64	16384
24K	96	24576
32K	128	32768
40K	160	40960*
48K	192	49152*

Figure 8. RAMTOP chart

*These values depend on whether or not any cartridges are in place. See the sections on cartridges.

Cartridge B (right cartridge)

The cartridges are a strange breed. They contain their own 8K of ROM, yet they feel the need to shove 8K of RAM out of the way in order to run. The right cartridge gives notice to locations 32768 through 40959 ($8000 through $9FFF). This means that if you have 40K of RAM or more, you'll lose 8K of it. Note that since the 800 is the only Atari that has a right cartridge slot, few companies have cartridges for the slot.

If a right cartridge is present, TRAMSZ at location 6 gets set to one during powerup.

Cartridge A (left cartridge)

Okay, all Atari computers have a left cartridge slot. Since it may be the only slot, it makes more sense to refer to it as cartridge A.

Cartridge A takes up memory locations 40960 through 49151 ($A000 through $BFFF). This will only affect you if you have 48K of RAM, since these locations are the last 8K.

The last six bytes in a cartridge provide the information that the OS needs in order to run the cartridge. Thus:

49146, 49147 ($BFFA, $BFFB) holds the starting (run) address of the cartridge.

49148 ($BFFC) equals zero if a cartridge is plugged in, and doesn't if one isn't.

49149 ($BFFD) tells the OS how to get the cartridge going. If bit 0 is set, then the OS boots the disk before it runs the cartridge. If bit 2 is set, then the cartridge is initialized but not run (if it's not set then it gets run).

49150, 49151 ($BFFE, $BFFF) holds the initialization address of the cartridge.

Note that these addresses are all for cartridge A. For cartridge B, just subtract 8192 from each address.

If cartridge A is present, TSTDAT at location 7 gets set to one during powerup.

If you're using BASIC, then BASIC is cartridge A. Because this book is designed to teach you about your Atari, and not about the languages that can be used with it, I'm not going to give you a detailed listing of all the locations in BASIC. Don't feel as though you're missing out on something great, however; there's very little in there that would be useful to you. The OS listing does mention four routines, however, so I will mention those.

SIN	
48551	BDA7

This routine calculates the sine of the number in floating point register zero (FR0). You should take a look at FR0 (212 to 217) and RADFLG (251) if you're going to try to use it. It might also help to disassemble the code for the routine to get an idea of what's going on.

COS	
48561	BDB1

This routine calculates the cosine of FR0.

ATAN	
48759	BE77

The arctangent of FR0.

SQR	
48869	BEE5

And lastly, the square root of FR0. Note that the carry is significant in all of these operations, in that it will be set if an error occurs during the operation.

SAVE MONEY ON
ATARI 800/XL/XE SOFTWARE

* Atari Public Domain & Shareware Software
* Over 250 Theme Disks! Every disk is Guaranteed!
* Games! Graphics! Educational! Music! Utilities! Home & Business!
* Fast dependable world-wide service!

Send for your FREE descriptive Catalog.

BELLCOM
P.O.Box 1043-G
Peterborough, Ontario
Canada K9J 7A5

CIRCLE #116 ON READER SERVICE CARD.

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

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 NAME______________________

______________________ EXP________ ADDRESS______________________

SIGNATURE______________________ CITY____________ STATE________ ZIP________

Your first issue will arrive in 6 to 8 weeks.
WATCH FOR IT!!

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

Offer expires April 30, 1989 CJOYA

U T I L I T Y

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" elsewhere in this issue.

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
```

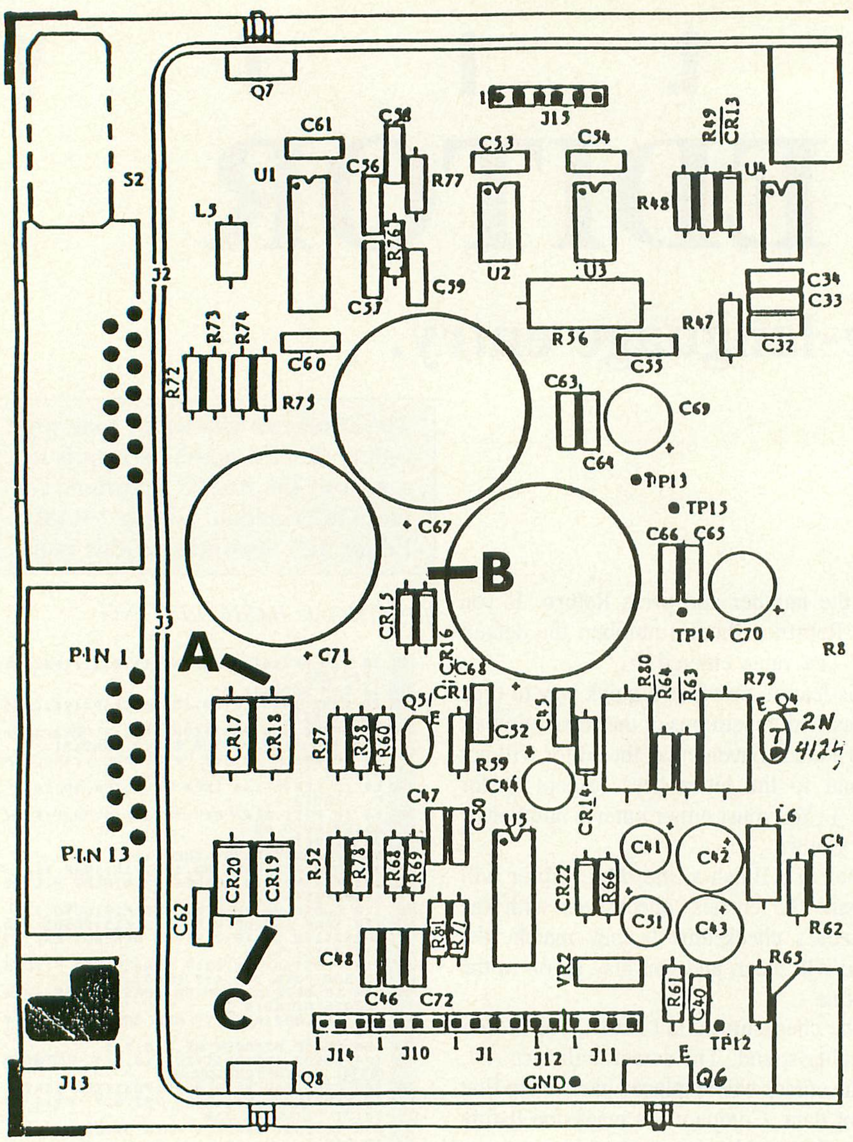
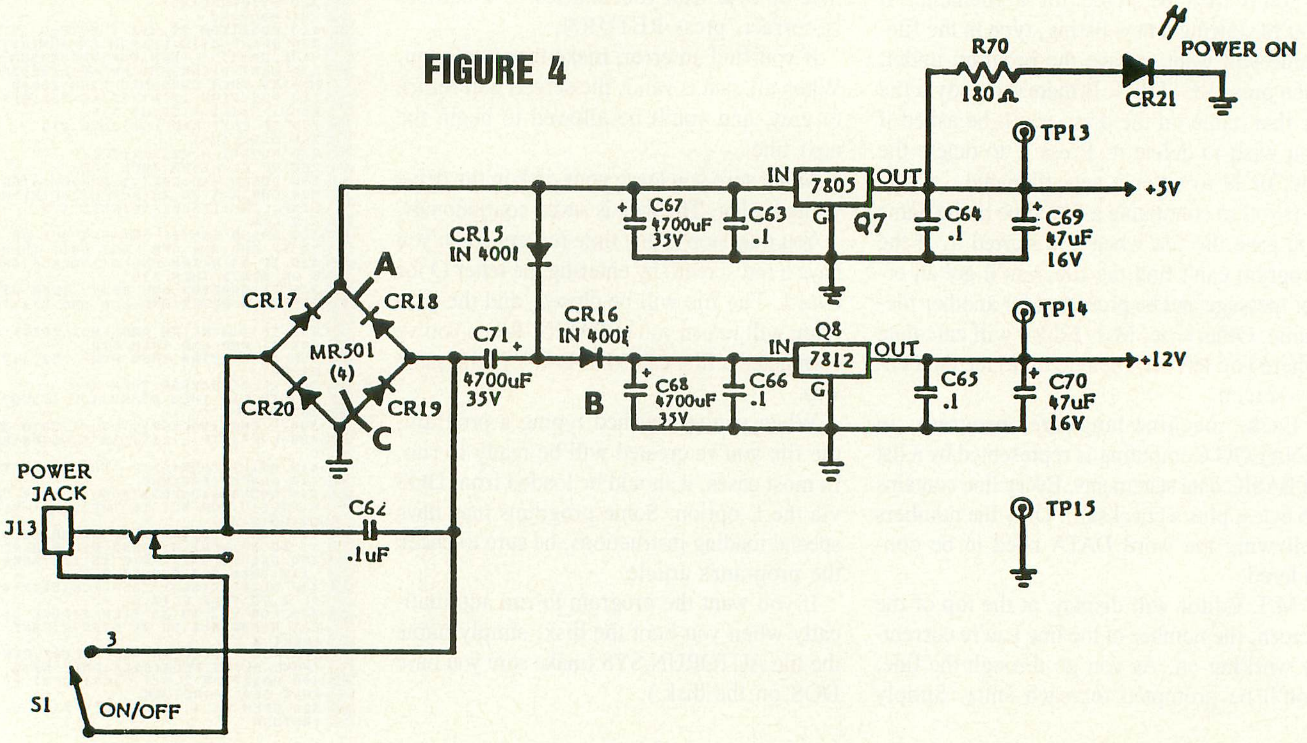
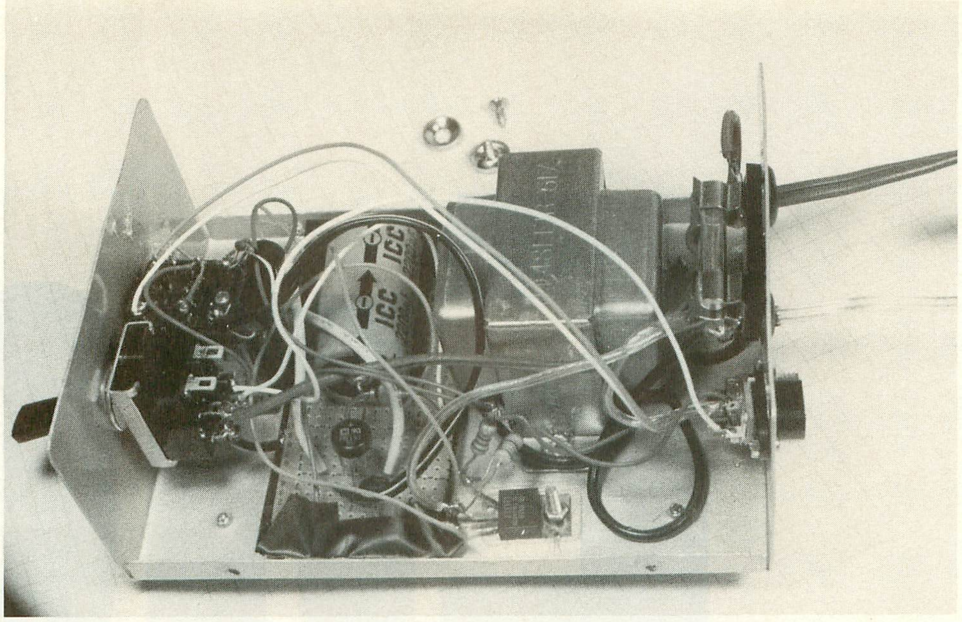


FIGURE 4



continued from page 53

	1 12-volt 3-amp-hour gel-cell (DSE S-3320) Use charger (DSE M-9521)
C1	2200 uf capacitor (RS 272-1020)
D1	1.5-amp bridge rectifier (RS 276-1151)
D2	3-amp blocking diode (RS 276-1141)
F1	1-amp fuse (RS 270-1273) holder (RS 270-739)
IC1	LM317T adjustable regulator (RS 276-1778) mounting kit (RS 276-1373) heat-sink grease (RS 276-1372)
LED1	Red LED (RS 276-041)
P1	Power cord (RS 278-1255)
P2	7-pin DIN plug, not at Radio Shack but available at many electronics stores. (ALL #DIN-7M)
R1	560-ohm, 1/2 watt resistor (RS 271-020)
R2	180-ohm, 1/2 watt resistor (RS 271-014)
R3	330-ohm, 1/2 watt resistor (RS 271-017)
S1	DPDT switch, 6-amp rating (RS 275-652)
T1	12-volt, 1.2-amp transformer (RS 273-1352)
Misc	Metal cabinet (RS 270-252) Strain reliefs (RS 278-1636) 4 feet of 18-gauge wire for computer power cable or another P1 with plug cut off.



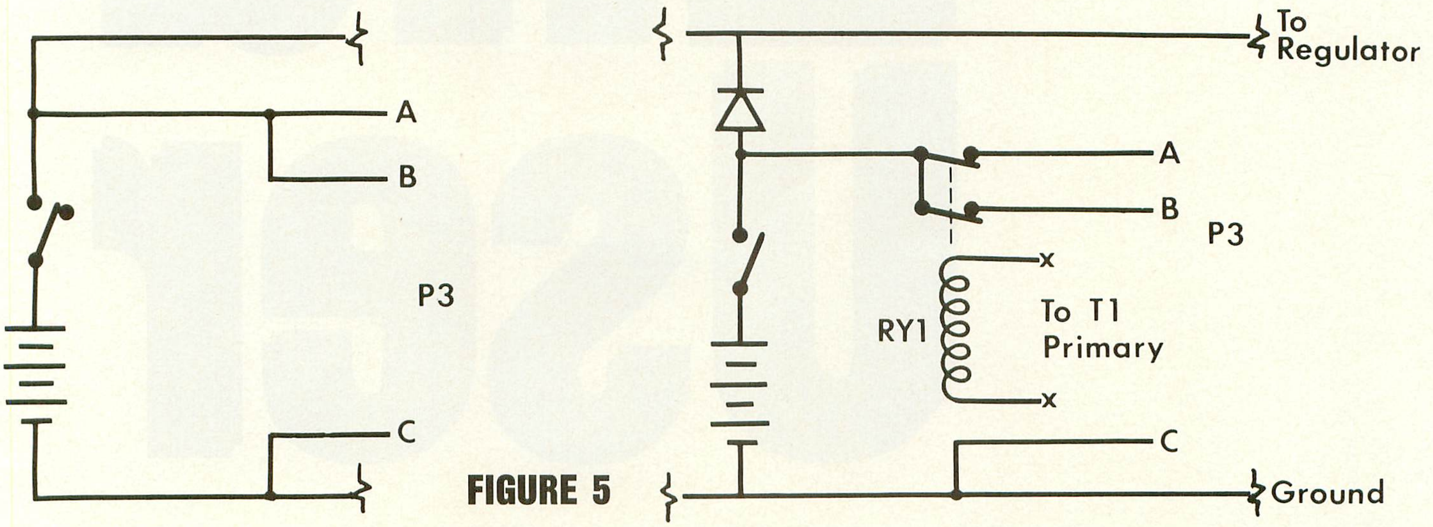


FIGURE 5

Power plug for battery pack (RS 274-1565 and 274-1567)

For 3-amp modifications, substitute:

D1	4-amp bridge rectifier (RS 276-1146)
T1	3-amp transformer (RS 273-1511)

Additional parts for disk-drive power adaptor:

	5-pin DIN plug (RS 274-003)
	jack (RS 274-005)
RY1	Relay, 110 volt (RS 275-220)
	3 2-foot lengths of 20-gauge wire (RS 278-1293)
	3 mini test clips (RS 270-372)

Power inverter: TrippLite PV-200
Jameco PV-200
Sears 28-b-71522

NOTE: this device uses 3.5 to 18 amps and requires a large battery such as an automotive battery.

SOURCES

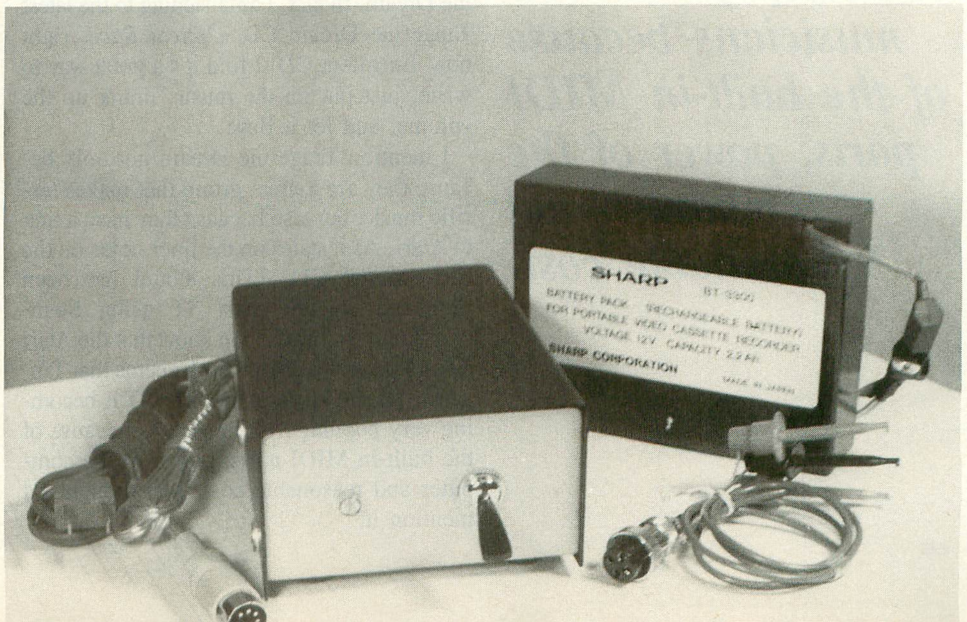
All Electronics (ALL)
P.O. Box 20406
Los Angeles, Ca. 90006

Dick Smith Electronics (DSE)
P.O. Box 8021
Redwood City, Ca. 94063

Jameco Electronics
1355 Shoreway Rd.
Belmont, Ca. 94002

Radio Shack (RS)
Check phone book for local listing.

Sears and Roebuck
Check phone book for local listing.


SHARP
BT-3300
BATTERY PACK · RECHARGEABLE BATTERY
FOR PORTABLE VIDEO CASSETTE RECORDER
VOLTAGE 12V CAPACITY 2.2 AH
SHARP CORPORATION
MADE IN JAPAN

End User

by Arthur Leyenberger

The ST is becoming very popular with musicians because of the built-in MIDI ports, power of the computer and reasonable cost.

What kind of music do you like? Hmmm? I love electronic music: Kraftwerk, Philip Glass, Wendy Carlos, Tomita, Vangelis, Giorgio Moroder and, of course, Tangerine Dream. In fact, I am listening to the latest Tangerine Dream CD, *Optical Race*, right now. Excellent CD! I find it's a great way to write, just put on the music, crank up the volume, and let it flow.

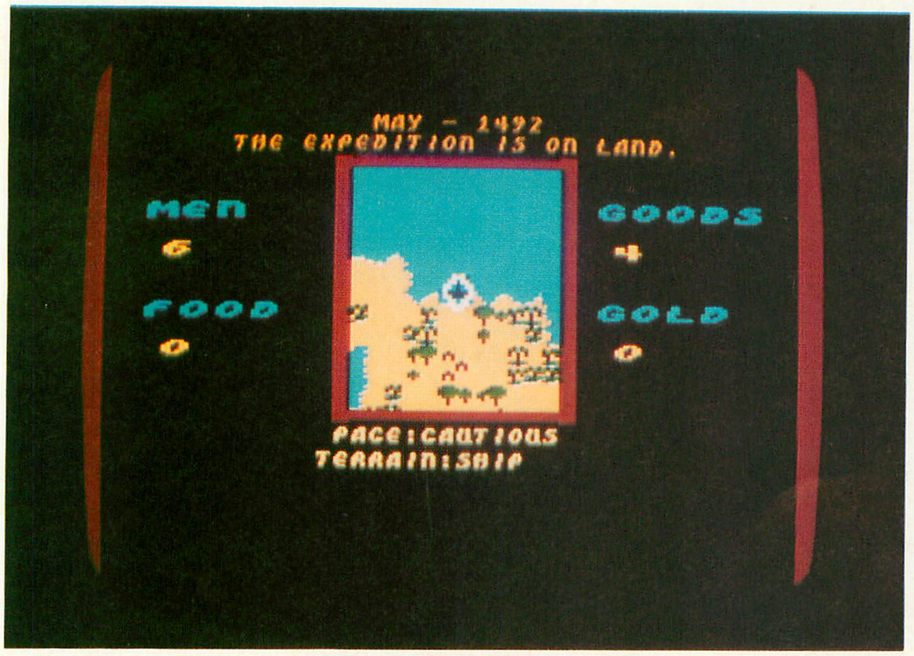
I mention Tangerine Dream not only because they are a great group that makes terrific music, but also because they have a link to Atari. As it states on the liner notes on the back of the CD, "This album has been produced on the Atari ST using Steinberg/Jones Software." We know that the Atari ST is a powerful computer. It seems that Tangerine Dream knows it too. The ST is becoming very popular with musicians because of the built-in MIDI ports, power of the computer and reasonable cost. Just thought I'd mention it.

The Arthur Leyenberger Memorial Museum of Computers & Technology

Welcome to the museum. Well, it's not actually a *real* museum but it comes pretty close. You see, it's my basement, and it's filled with Atari computers, software and memorabilia (I hate that word) from close to seven years as an Atari enthusiast. Yes, I *am* an Atari enthusiast and continue to be one. Let me take you on a tour, but don't worry; there is no admission charge.

Where shall we start? How about this: It's a disk labeled *Data Perfect* by LJK Enterprises. This is a real blast from the past. Back in 1981 or thereabouts, there were several database programs for the Atari 800. The two most popular titles were *File Manager 800* by Synapse and Data Perfect.

File Manager, if I recall correctly, came out first and was quite powerful for its time.


MAY - 1492
THE EXPEDITION IS ON LAND.
MEN 6
GOODS 4
FOOD 0
GOLD 0
PACE:CAUTIOUS
TERRAIN:SHIP

Seven Cities of Gold

But it didn't have report-writing capability. The only way to retrieve information was to do it interactively. Then came Data Perfect. This program was very sophisticated for its time, especially on an Atari 8-bit computer.

Trouble was, the manual was downright terrible. I think it belongs in the "Poor Documentation Hall of Fame." Anyway, I remember I was editing and publishing a user-group newsletter at the time, and I wanted to keep track of the mailing list. The list was already created with File Manager 800, but I wanted the report-writing capability of Data Perfect.

So I go through a lengthy conversion process and then begin the long journey towards Data Perfect report-writing self-actualization. To end a long story, I never did achieve the results I was looking for. I spent an entire weekend, Saturday *and* Sunday (damn the chores) trying to learn how to use the report-writing capability of this blasted program. I never did! From that point on, I put the program back on the shelf and refused to even open the manual ever again.

As bad as Data Perfect was in that respect, *Letter Perfect*, LJK's word processor was clearly the best in the league. I did all of my writing for the club newsletter using that program. Further, I used Letter Perfect exclusively when I wrote for *Creative Computing* for a year.

One of the best things about the program was that it worked with the Bit-3 80-column board. Using a composite monochrome monitor and the 80-column version of Letter Perfect, I was using a system similar to that of the "big rig" Apples and CP/M machines. I still have the Letter Perfect disk, manual and some 8-bit disks with articles.

Ah, the Bit-3 80-column board. This baby sat in the last slot of the Atari 800, which meant you had to have a 32K memory card in the second slot if you wanted a full 48K. With the Bit-3, a Mosaic 32K board (I wonder where *they* are these days), an Atari 16K memory card and an Omnimon system card, my system ran hotter than my '67 BMW with the bad head gasket. I had to keep the cover off the computer to prevent a meltdown within a half-hour. Man, those *were* the days.

The "Journalism Exhibit"

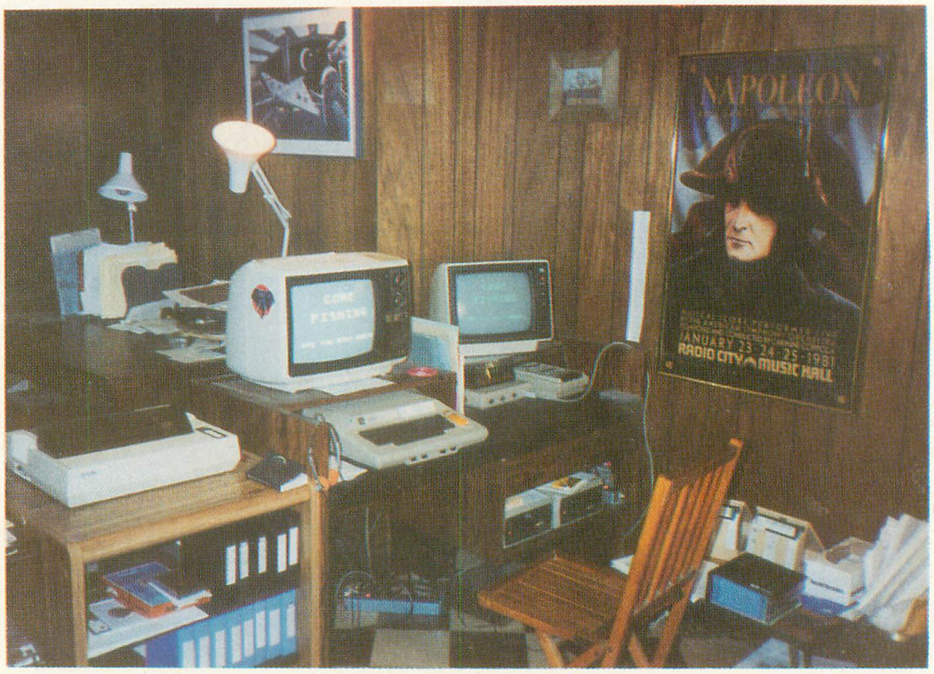
In this exhibit of the museum, we have some rare and notable computer magazines. Here is the first issue of *Byte—The Small Systems Journal*, as it was billed at the time. The issue is dated September, 1975 and carried a cover price of a buck and a half. The cover stories seem almost quaint by today's standards: "Which Microprocessor For You?," "Cassette Interface—Your Key To Inexpensive Bulk Memory" (I learned the hard way about cassette storage with my finicky Atari 410 cassette drive), and "Recycling Used ICs." It was clear that, in those days, *Byte* was a hacker's magazine.

Leafing through that first issue, it appears that half of the ads were for test equipment, power supplies and integrated circuits. There is an ad for the Altair 8800 Computer Kit, a $1,000 system with 4K of memory, serial and parallel interfaces, and cassette interface. I had almost forgotten that these early computers did not use keyboards or monitors. You had to attach a teletype if you wanted to enter programs from a keyboard. We have sure come a long way since 1975.

The "journalism exhibit" also includes many first issues of magazines long gone, such as *ROM*, *OnComputing*, *Creative Computing*, *Personal Computing*, *SCCS Interface* and *Peoples' Computers*. There are also some first issues of still-existing mags such as ANALOG (of course), *Amiga World* (what a fluff piece), *Publish* and *CD-ROM*, among others. A thesis could easily be written on the social psychology of individual computing, just by reading the information in these primary sources.

No serious museum is complete without key issues of *InfoWorld* magazine. Here is one dated February 27, 1984 with the headline "Is Atari Going Down the Tubes?" There is a picture of James Morgan on the cover (who?). In addition to the somewhat ironic cover story, there are some other neat and/or prophetic articles, too. How about a small article discussing the departure of Jack Tramiel and other executives from Commodore. Here's one: "Mattel Drops Intellivision." Interesting reading for the serious video-game historian. Another one: "TI Leaves Home Market." Still another: "Eastern OKs Use of Portable Computers in Flight." This issue is chock full of interesting stuff.

Later that year, in the *InfoWorld* dated August 6, 1984, the cover story is "Tramiel Moves In; Can He Save Atari?" Another great collection of articles and news, historically speaking. Finally, take a look at the January 28, 1985 issue, just six months



"In the Beginning"

later. The cover story: "Atari Unveils Its 'Jackintosh.'" The years 1983, 1984 and 1985 were hectic ones. I wonder how we survived it all?

Companies stopped using copy protection for all but game software once they realized that the pirates were always just one step behind.

The Hardware Room

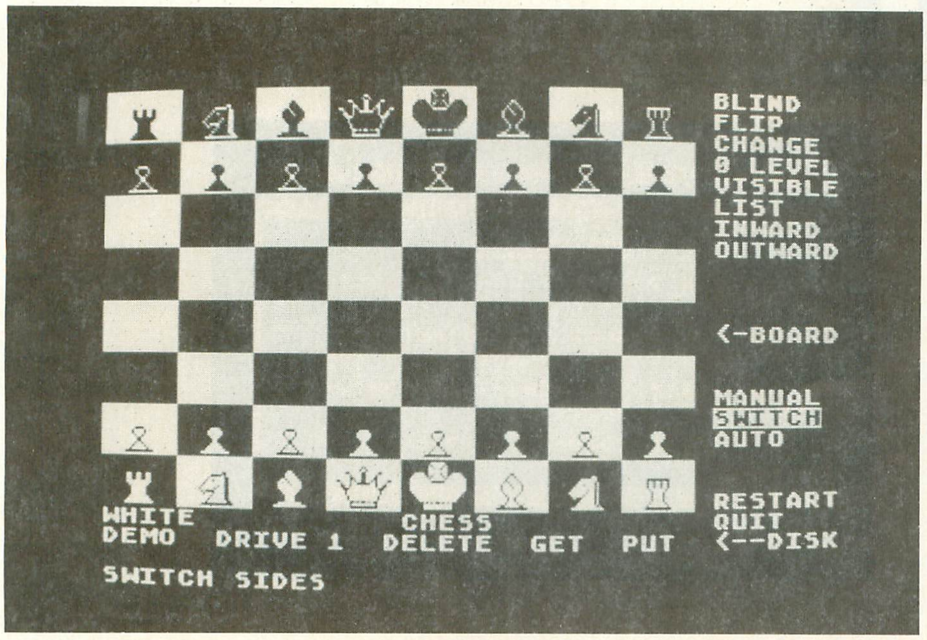
In the hardware department, I'd like to share a few rare items that archaeologists may someday be unable to explain. First, there is my Nibble Notcher. This little fellow gave me years of service, notching out the left side of my 8-bit floppy disks. This allowed me to use the backside of the disk in my 810 disk drive.

Controversy surrounded this technique, with one side arguing that the backside wasn't a reliable surface for data and that the opposite rotating of the flipped disk could somehow cause drive damage. The supporters of the technique, and myself, were seeking a solution to the high cost of floppy disks at the time. Come to think of it, I never lost any data by using the backside of a disk in that manner.

Oh yes, here is my dongle. Actually it is a Batteries Included dongle. This little widget was inserted into the second joystick port of the 800 in order to run *Paper Clip*, the best of the third-generation 8-bit word processors. The idea of copy protection, in whatever form, brings back some fond and not-so-fond memories. As it turned out, companies stopped using copy protection for all but game software once they realized that the pirates (read: software thieves) were always just one step behind.

Here is another interesting hardware item: a Blaster. This is a device that is inserted in between your joystick and the computer. It has a rotary control on it to allow you to speed up the number of pulses of the fire button. At the highest setting, every time you pressed the joystick button about 20 to 30 impulses were sent to the computer. What a way to improve your high score standings in 8-bit shoot-'em-up video games.

Over in the corner is Andy the Robot. Do you remember Andy? One of Nolan Bushnell's (Atari's founder) many companies was Axlon. They started off in the early 1980s making memory boards for the 800 computer. Then they branched out into toys and other stuff. Andy was connected by a tether to one of the joystick ports and could be controlled by a program running on the 8-bit computer. It was a simple control mechanism—left, right, blink eyes, forward and backward—but it was one of the first robot-controlled devices for the Atari computer. Andy is silently keeping me company, even today, as I write. Say "Hi" to the



Chess

folks at home, Andy.

Another robot that keeps me company is Andy's friend, Verbot. Verbot is a voice-controlled robot made by Tomy. You would program voice commands into a remote microphone and then Verbot would respond to your verbal instructions. I thought this was neat enough by itself, but I took it a step further. I wrote a BASIC program that produced eight unique sounds, each produced by pressing a key on the 800 keyboard.

I used this method to program Verbot by placing the microphone next to the monitor speaker and pressing the various keys on the Atari. It was pretty neat to be able to put Verbot through his paces under my keyboard control. If you're interested, I wrote about it in the August, 1985 *End User.* That column even includes the BASIC program. Have fun.

Space does not permit reminiscing about some of the other hardware laying around. Things like the Atari Light Pen, a handful of drawing tablets, the MIDI-Mate 8-bit interface, the collection of Hewlett-Packard calculators dating back to 1974, and an assortment of patch cables and AC adapters of which I'll probably never know their origin, will have to wait until your next visit to the museum. Gee, if we had more time, I really wanted to tell you about the Relax biofeedback system. Oh well.

Software Central

I could spend days talking about the multitudes of software I have kept all these years. There is my prize collection of every Electronic Arts 8-bit game. Some really great stuff like *Pinball Construction Set*, *M.U.L.E.*, *Seven Cities of Gold* and more than a dozen more. Then there is the three-volume set by Odesta of *Chess*, *Checkers* and *Odin* (a Reversi-type game). These three 8-bit games are some of the finest ever produced by *anyone*. They have excellent documentation, packaging, game play and game options. If you wanted to study human-computer interaction or software engineering, this would be the place to start.

Uncategorical Department

Another one of the more important museum collections is the entire set of Atari press kits since 1983. You want memories? You want intrigue? You want *history*. Reading these press kits is like looking at the medical history of Atari. Products that never were, such as Mind Link and the 8-bit CP/M Box are here for the viewing. Such rare conceptions as the 260ST, the portable 8-bit computer, the Atari Educational Dept.—all fantasies of a company trying to find its soul (well, maybe)—are part of the permanent record in these archives. By golly, the last six years hold a lot of excitement and memories.

Cleaning Up

I didn't discuss half of the junk... er...valuable mementos I have laying around. (Have you ever seen a Shanner Planner?) To do so, I would probably need an entire issue of ANALOG. Anyway, these things sure have a lot of memories. It is hard sometimes to recall those early times when I first had my Atari computer. There was a certain excitement in learning to use the machine, programming in BASIC, trying to get the cassette recorder to work and purchasing a new software title.

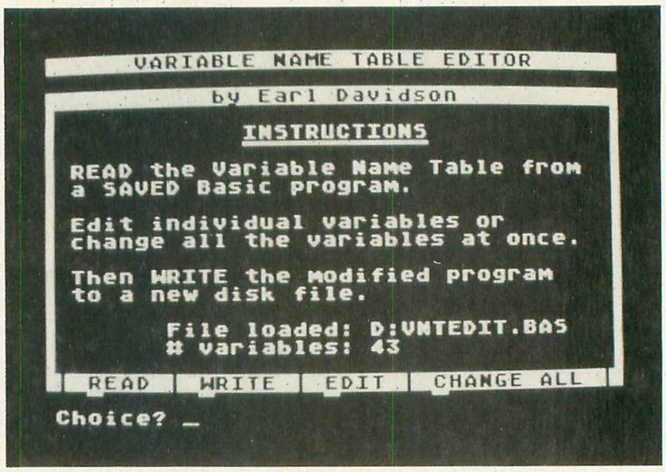
The notion of having my own computer was a thrill. Each new thing I learned to do with it seemed like a major discovery and accomplishment. I couldn't wait to read everything I could get my hands on that discussed this marvelous new device. It certainly was a time for learning and immense satisfaction. In a way, I have lost my computer innocence.

You may have some of this stuff collecting dust in your basement, too. Don't throw it out! It may be valuable some day. Who knows, 20 years from now (or sooner) these programs, hardware items and other knick-knacks may be true collectibles, perhaps as popular a hobby as collecting and trading baseball cards are today.

It has been fun sharing this journey back in time. You know, I guess that is one of the things that has not changed. Although I know much more about computers and software than I did long ago, and maybe there is less excitement with the release of new hardware and software products, I still enjoy sharing information about Atari computers. I'm fortunate that I have this forum every month to do just that.

As they say in the commercials, thank you for your support. See you next time, friend.

VARIABLE NAME TABLE EDITOR


```
VARIABLE NAME TABLE EDITOR
by Earl Davidson
INSTRUCTIONS

READ the Variable Name Table from
a SAVED Basic program.

Edit individual variables or
change all the variables at once.

Then WRITE the modified program
to a new disk file.

     File loaded: D:VNTEDIT.BAS
     # variables: 43

 READ | WRITE | EDIT | CHANGE ALL

Choice? _
```

by Earl Davidson

I have a tendency to change my mind about variable names as I write programs. However, changing the name of a variable which is used many times throughout a program can be a pain, since BASIC provides no command to do this for us, and there is no search-and-replace function. I wrote the *Variable Name Table Editor* to give me an easier way to change variable names. It reads the header and Variable Name Table (VNT) from a SAVEd file, allows each variable name to be edited, then writes a new file to disk.

As an Atari BASIC program is entered, each text line we type with the BASIC editor goes through a process often called pre-compiling. BASIC converts the line into a tokenized format which can be interpreted during program execution much faster than the text line we enter could be interpreted. The tokenized version of the line is placed into the Statement Table (ST). Other tables are built during this pre-compile process: the Variable Name Table (VNT), the Variable Value Table (VVT), and the String/Array Table.

If the line entered contains a new variable, its name is added to the VNT, and information about the variable is placed into the VVT. If the variable is a string or array, it is added to the String/Array Table. In the tokenized version of the program line, the variable is represented by a one-byte token. This value of this byte is the position of the variable in the VNT, plus 128 ($80). A BASIC program may have only 128 variables, and their tokens range sequentially from 128 to 255 ($80 to $FF).

The VNT is maintained by BASIC for the convenience of the programmer. It is used only by the LIST statement. When the program, or a portion of it, is listed, the variable token is used as an index into the VNT and the variable name is displayed. The variable name may be any length. It must consist of uppercase letters and numbers and must begin with a letter. This gives us considerable freedom to use descriptive variable names so that our programs are easier to read.

When a program is written to disk using the LIST command, the file is exactly the same as if the program is listed to the screen. When it is later entered with the ENTER command, BASIC pre-compiles each line and builds the various tables just as if it were being typed in.

However, when a program is written to disk using the SAVE command, the file consists of a 14-byte header and the various tables that have been built. The Statement Table is saved in its tokenized format. When the file is later entered with the LOAD command, the header portion of the file is used by the operating system as an index to the various tables and the tables are simply read and placed into the appropriate place in RAM.

While we are entering a program, if we change a variable's name, the new name is placed into the VNT. The original name remains in the table, even if it is no longer used by the program. The only way to clear the original name and free up RAM and variable space is to LIST the program to disk, type NEW, and ENTER it back into memory, which creates a new VNT using only the variable names encountered in the file.

Using VNT Editor

When VNT Editor is run, a menu of four items is presented: Read file, Write file, Edit Variable Names, and Change All. First you must read the file you wish to modify. When

Changing the name of a variable which is used many times throughout a program can be a pain, so I wrote Variable Name Table Editor to make it easier.

editing the variable names, you should be careful to use valid names if you wish to be able to edit the program later. Although BASIC places no restriction on the length of variable names, VNT Editor limits you to 30 characters per name which should be sufficient. Replacing a short name with a long one can also cause problems later if the new name results in program lines that are longer than the BASIC editor will accept. Once you have completed your changes, write the file out to a new filename. The new file may then be LOADed as usual.

VNT Editor does not force you to use valid variable names nor does it check for duplicate names. There may be occasions where you do not wish to use valid names. If your program uses all of RAM and you need a few more bytes you might wish to change all variable names to one letter names. BASIC will allow only 26 (A to Z) and considers the dollar sign on strings and the parentheses on arrays as part of the variable name. The Change All option allows you to change all variables' names to the same one-character name,which could save you several hundred bytes in a large program.

If you have a program which you do not want other users to modify for some reason, you may change variable names to special characters. The resulting file will run properly, but if a line containing an invalid variable name is edited,an error will result which cannot be corrected. Try changing all variable names to a string of 30 question marks. The resulting file will run correctly, but when LISTed, will appear very strange.

If you create a file with invalid or duplicate variable names, be sure to keep a copy of your original file for future use.

Program Explanation

VNT Editor is straightforward, with no tricks or complicated routines. Remarks are not referenced, so you may omit them if you like.

The Read File routine begins on Line 810. Line 830 reads the 14-byte header for the file. Line 840 checks the first two bytes. They should both be decimal zero if the file is a SAVEd BASIC program. The third and fourth bytes contain an offset (from zero) from the end of the header to the beginning of the VNT+256. VNTSTART is set to this value in Line 850. The offset to the end of the VNT is one less than the start of the VVT. Line 860 assigns this value to VNTEND.

The seventh and eighth bytes of the header contain the offset to the beginning of the VVT+256. VVTSTART is set to this value in Line 870. The offset to the end of the VVT is one less than the start of the ST. Line 880 assigns this value to VVTEND. The ninth and tenth bytes contain the offset to the beginning of the ST+256. Line 890 assigns this value to STSTART. The 13th and 14th bytes contain the offset to the end of the ST+256. Line 900 assigns this value to STEND.

The fifth, sixth, 11th and 12th bytes of the header are not used by BASIC or VNT Editor.

The VNT, as determined by VNTSTART AND VNTEND, is read and placed in the string VNT$. The length of VNT$ is stored in VNTLEN1 for use later. After editing the variable names, a new file is written by the routine beginning at Line 1010. Line 1040 checks to insure the destination filename is not the same as the source filename. The current length of VNT$ is stored in VNTLEN2 in Line 1060, and the difference between VNTLEN1 and VNTLEN2 is stored in CHANGE in Line 1070. The offsets found in HEADER$ are updated by the value of CHANGE in Lines 1080-1160.

Line 1180 writes the HEADER$ to the new file. Line 1190 writes the new VNT$ to the new file. Line 1210 reads the old header and VNT from the old file and discards them. Line 1230 is a loop which reads a byte from the old file and writes it to the new file until the end of the old file is reached.

Other sections of the program should be self-explanatory.

VNT Editor will provide you with a new way to view and manipulate the VNT of your BASIC programs. The extra control and flexibility will be convenient on many programs and could be very helpful on larger programs using a large number of variables.

Earl Davidson has been an avid Atari enthusiast for many years. He is an active member of the Atari Users Group of Albany, GA, and as one of the owners of SoSoft, he wrote the user's manual for InSyst!, *a small business inventory program for 8-bit Ataris.*

LISTING 1: BASIC

```
HY 10 REM ******************************
RK 20 REM * VARIABLE NAME TABLE EDITOR *
RU 30 REM *       by Earl Davidson     *
OH 40 REM *        Copyright 1988      *
SM 50 REM *    BY ANALOG COMPUTING     *
ID 60 REM ******************************
BN 65 REM
CO 70 GRAPHICS 0:POKE 710,0:POKE 709,0:PO
   KE 752,1:GOTO 1340
BF 80 REM
IA 90 REM *** GET KEY FROM KEYBOARD
QM 100 REM
ZE 110 ? "_←";:GET #1,KEY:KEY$=CHR$(KEY):
   RETURN
QQ 120 REM
NA 130 REM *** CLEAR LAST THREE LINES
QU 140 REM
QM 150 FOR J=21 TO 23:POSITION 1,J:? BL$;
   :NEXT J:RETURN
QY 160 REM
SR 170 REM *** BORDER
RC 180 REM
FX 190 FOR I=3 TO 17:POSITION 1,I:? BORDE
   R$:NEXT I:RETURN
QN 200 REM
RA 210 REM *** GET FILESPEC
QR 220 REM
GD 230 POSITION 2,21:? "ENTER FILESPEC: "
   ;
QV 240 REM
ZC 250 REM *** GET LINE
QZ 260 REM
ON 270 WORK$="":CHNUM=1
PY 280 GOSUB 110:KEY$=CHR$(KEY):IF KEY=15
   5 THEN ? " ";:RETURN
IZ 290 IF KEY=27 THEN CHNUM=0:WORK$="":?
   " ":RETURN
JL 300 IF KEY>127 THEN KEY=KEY-128:KEY$=C
   HR$(KEY)
QR 310 IF KEY=30 OR KEY=126 THEN IF CHNUM
   >2 THEN CHNUM=CHNUM-1:? " ←";CHR$(126)
   ;:GOTO 280
EU 320 IF KEY=30 OR KEY=126 THEN IF CHNUM
   =2 THEN CHNUM=1:WORK$="":? " ←";CHR$(1
   26);:GOTO 280
JS 330 IF KEY>26 AND KEY<32 OR KEY>124 TH
   EN ? CHR$(253);:GOTO 280
YB 340 IF CHNUM<MAXL THEN ? KEY$;:WORK$(C
   HNUM,CHNUM)=KEY$:CHNUM=CHNUM+1:GOTO 28
   0
MT 350 ? CHR$(253);:GOTO 280
RA 360 REM
GL 370 REM *** MENU FOR EDIT ROUTINE
RE 380 REM
OJ 390 IF SOURCE$="NONE" THEN GOSUB 1720:
   GOTO 1520
YG 400 POSITION 14,1:? "Page ";PAGE+1;" o
   f ";INT(NUMVAR/15)+1;" "
WJ 410 POSITION 1,2:? " V # VARIABLE NAME
                                ""
UX 420 GOSUB 190
ML 430 POSITION 1,18:? "    | EDIT | MENU |
    PREVIOUS | NEXT |    "
GH 440 POSITION 1,19:? "       _      _
   _      _"
UN 450 GOSUB 540
LO 460 POSITION 2,20:? "Choice?
      ";:POSITION 10,20
EB 470 GOSUB 110:? KEY$;
AE 480 IF KEY$="E" THEN ? "dit";:GOTO 610
NE 490 IF KEY$="N" THEN ? "ext";:PAGE=PAG
   E+(PAGE<MAXPAGE):GOSUB 540:GOTO 460
LN 500 IF KEY$="P" THEN ? "evious";:PAGE=
   PAGE-(PAGE>0):GOSUB 540:GOTO 460
FG 510 IF KEY$="M" THEN ? "enu";:GOTO 142
   0
XD 520 IF KEY=27 OR KEY=155 THEN POSITION
    10,20:? "Menu";:GOTO 1420
PI 530 GOTO 460
XE 540 POSITION 19,1:? PAGE+1
NT 550 POSITION 2,3:LONUM=1+PAGE*15:HINUM
   =15+PAGE*15:BLANK=0:IF HINUM>NUMVAR TH
   EN BLANK=HINUM-NUMVAR:HINUM=NUMVAR
JL 560 FOR I=LONUM TO HINUM:WORK$=BL$:LM=
   1:IF I<100 THEN LM=2:IF I<10 THEN LM=3
SE 570 VAR$=VNT$(VNUM(I),VNUM(I+1)-1):VAR
   L=LEN(VAR$):CH=ASC(VAR$(VARL)):CH=CH-1
   28:VAR$(VARL)=CHR$(CH)
XA 580 WORK$(LM,3)=STR$(I):WORK$(5,4+LEN(
   VAR$))=VAR$:? WORK$:NEXT I
FO 590 IF BLANK THEN FOR I=1 TO BLANK:? B
   L$:NEXT I
ZC 600 RETURN
NJ 610 POSITION 2,20:? "Edit variable #?
     ":POSITION 19,20
HU 620 MAXL=4:GOSUB 270:IF WORK$="" THEN
   GOSUB 150:GOTO 460
EU 630 TRAP 640:I=VAL(WORK$):IF I>0 AND I
   <1+NUMVAR THEN GOTO 650
SE 640 ? CHR$(253):POSITION 2,21:? "INVAL
   ID NUMBER!":GOSUB 1650:GOSUB 150:GOTO
   610
EH 650 TRAP 40000:POSITION 1,21:? " OLD:"
   ;
SD 660 VAR$=VNT$(VNUM(I),VNUM(I+1)-1):VAR
   L=LEN(VAR$):CH=ASC(VAR$(VARL)):CH=CH-1
   28:VAR$(VARL)=CHR$(CH)
XN 670 WORK$=BL$(1,30):WORK$(1,LEN(VAR$))
   =VAR$:? WORK$
MW 680 POSITION 1,22:? " NEW:";:MAXL=31:G
   OSUB 270:IF WORK$="" THEN GOTO 770
IK 690 VARL1=LEN(WORK$):WORK$(VARL1+1)=BL
   $(VARL1+1,32):POSITION 6,I-(PAGE*15)+2
   :? WORK$;
EB 700 WORK$=WORK$(1,VARL1):CH=ASC(WORK$(
   VARL1)):CH=CH+128
CE 710 WORK$(VARL1)=CHR$(CH)
RQ 720 CHANGE=VARL1-VARL
WM 730 IF I=NUMVAR THEN VNT$=VNT$(1,(VNUM
   (I)-1)):VNT$(VNUM(I))=WORK$:GOTO 760
UQ 740 VNT1$=VNT$(VNUM(I+1),LEN(VNT$))
DI 750 VNT$(VNUM(I))=WORK$:VNT$(LEN(VNT$)
   +1)=VNT1$
BP 760 FOR J=I+1 TO NUMVAR+1:VNUM(J)=VNUM
   (J)+CHANGE:NEXT J
HP 770 GOSUB 150:GOTO 460
RI 780 REM
TY 790 REM *** READ FILE ROUTINE
QT 800 REM
GH 810 TRAP 1690:VNT$="":SOURCE$="":NUMVA
   R=0:MAXL=16:GOSUB 230:IF WORK$="" THEN
    GOSUB 150:SOURCE$="NONE":GOTO 1520
DK 820 SOURCE$=WORK$:POSITION 2,21:? "
      READING";:CLOSE #2:OPEN #2,4,0,SOU
   RCE$:I=1
FO 830 FOR I=1 TO 14:GET #2,A:HEADER$(I,I
   )=CHR$(A):NEXT I
RN 840 IF HEADER$(1,1)<>CHR$(0) AND HEADE
   R$(2,2)<>CHR$(0) THEN GOTO 1730
DL 850 VNTSTART=(ASC(HEADER$(3,3))+ASC(HE
   ADER$(4,4))*256)-256
JS 860 VNTEND=(ASC(HEADER$(7,7))+ASC(HEAD
   ER$(8,8))*256)-256-1
CF 870 VVTSTART=(ASC(HEADER$(7,7))+ASC(HE
   ADER$(8,8))*256)-256
DD 880 VVTEND=(ASC(HEADER$(9,9))+ASC(HEAD
   ER$(10,10))*256)-256-1
```

```
HZ 890 STSTART=(ASC(HEADER$(9,9))+ASC(HEA
   DER$(10,10))*256)-256
QB 900 STEND=(ASC(HEADER$(13,13))+ASC(HEA
   DER$(14,14))*256)-256
JS 910 FOR I=(VNTSTART+1) TO VNTEND:GET #
   2,A:VNT$(I,I)=CHR$(A):NEXT I:CLOSE #2
PU 920 VNTLEN1=LEN(VNT$):REM ORIGINAL LEN
   GTH OF VNT$
FZ 930 VNUM(1)=1:I=2
AN 940 FOR J=1 TO VNTLEN1:IF ASC(VNT$(J,J
   ))>127 THEN VNUM(I)=J+1:I=I+1
GT 950 NEXT J
TR 960 NUMVAR=I-2:MAXPAGE=INT(NUMVAR/15):
   PAGE=0
WS 970 GOSUB 150:GOTO 1520
RK 980 REM
TU 990 REM *** WRITE FILE ROUTINE
HW 1000 REM
WB 1010 IF SOURCE$="NONE" THEN GOSUB 1720
   :GOTO 1520
GJ 1020 MAXL=16:GOSUB 230:DESTINATION$=WO
   RK$
DS 1030 IF WORK$="" THEN GOSUB 150:GOTO 1
   520
QD 1040 IF SOURCE$=DESTINATION$ THEN GOSU
   B 1740:GOTO 1010
FR 1050 POSITION 2,21:? "        WRITING";
EU 1060 VNTLEN2=LEN(VNT$):REM ENDING LENG
   TH OF VNT$
DQ 1070 CHANGE=VNTLEN2-VNTLEN1
FV 1080 VVTSTART=VVTSTART+CHANGE
PC 1090 VVTSTARTHI=INT(VVTSTART/256):HEAD
   ER$(8,8)=CHR$(VVTSTARTHI+1)
OH 1100 VVTSTARTLO=VVTSTART-(VVTSTARTHI*2
   56):HEADER$(7,7)=CHR$(VVTSTARTLO)
LR 1110 STSTART=STSTART+CHANGE
DU 1120 STSTARTHI=INT(STSTART/256):HEADER
   $(10,10)=CHR$(STSTARTHI+1)
MS 1130 STSTARTLO=STSTART-(STSTARTHI*256)
   :HEADER$(9,9)=CHR$(STSTARTLO)
ES 1140 STEND=STEND+CHANGE
FW 1150 STENDHI=INT(STEND/256):HEADER$(14
   ,14)=CHR$(STENDHI+1)
QI 1160 STENDLO=STEND-(STENDHI*256):HEADE
   R$(13,13)=CHR$(STENDLO)
IK 1170 TRAP 1700:CLOSE #3:OPEN #3,8,0,DE
   STINATION$
OP 1180 FOR I=1 TO 14:A=ASC(HEADER$(I,I))
   :PUT #3,A:NEXT I:REM WRITE NEW HEADER
YL 1190 FOR I=1 TO LEN(VNT$):A=ASC(VNT$(I
   ,I)):PUT #3,A:NEXT I:REM WRITE NEW VNT
WB 1200 CLOSE #2:OPEN #2,4,0,SOURCE$
MU 1210 FOR I=1 TO VNTLEN1+14:GET #2,A:NE
   XT I:REM THROW AWAY OLD VNT AND HEADER
OE 1220 TRAP 1240
RF 1230 GET #2,A:PUT #3,A:GOTO 1230:REM R
   EAD REMAINDER OF FILE AND WRITE IT TO
   NEW FILE
XC 1240 CLOSE #2:CLOSE #3:GOSUB 150:GOTO
   1520
CZ 1250 POSITION 2,20:? "Change All to (<
   C/R> to EXIT): ":POSITION 5,21:MAXL=30
   :GOSUB 270
EF 1260 IF WORK$="" THEN GOSUB 150:GOTO 1
   520
XH 1270 A=ASC(WORK$(LEN(WORK$))):A=A+128:
   WORK$(LEN(WORK$))=CHR$(A)
DG 1280 VNT$="":VNT$=WORK$:VNUM(2)=LEN(WO
   RK$)+1:FOR I=2 TO NUMVAR:VNT$(LEN(VNT$
   )+1)=WORK$:VNUM(I+1)=LEN(VNT$)+1
FU 1290 NEXT I
QU 1300 POSITION 5,21:? BL$:GOTO 1520
IF 1310 REM
TS 1320 REM *** INITIALIZATION
IL 1330 REM
WE 1340 DIM KEY$(1),VNT$(3840),VNT1$(3840
   ),HEADER$(14),BL$(36),SOURCE$(15),DEST
   INATION$(15),WORK$(40),VNUM(128)
QX 1350 DIM VAR$(30),BORDER$(38)
BS 1360 BL$=" ":BL$(36)=BL$:BL$(2)=BL$
KV 1370 SOURCE$="NONE":BORDER$=CHR$(153):
   BORDER$(2)=BL$:BORDER$(38,38)=CHR$(25)
   :FIRST=1
XE 1380 OPEN #1,4,0,"K:":REM OPEN KEYBOAR
   D
JD 1390 REM
JW 1400 REM *** MAIN MENU
IH 1410 REM
FQ 1420 TRAP 40000:POKE 752,1:POSITION 1,
   0:? "         VARIABLE NAME TABLE EDITOR
         "
XW 1430 POSITION 1,1:? BL$;
HT 1440 POSITION 1,2:? "               by Ear
   l Davidson               "
AB 1450 GOSUB 190
GD 1460 POSITION 1,18:? "|  READ  |  WRITE
   |  EDIT  |  CHANGE ALL  | "
SH 1470 POSITION 1,19:? "       -       -
     -       -         "
PX 1480 POKE 82,3:POSITION 14,4:? "INSTRU
   CTIONS":POSITION 14,5:? "____________"
FD 1490 ? "READ the Variable Name Table f
   rom":? "a SAVED Basic program."
TL 1500 ? :? "Edit individual variables o
   r":? "change all the variables at once
   ."
YT 1510 ? :? "Then WRITE the modified pro
   gram":? "to a new disk file.":POKE 82,
   2
WB 1520 TRAP 1420:POSITION 9,15:? "File l
   oaded: ";SOURCE$;BL$(1,15-LEN(SOURCE$)
   );
MF 1530 POSITION 9,16:? "# variables: ";N
   UMVAR;BL$(1,5);
VE 1540 POKE 709,12
LS 1550 POSITION 2,20:? "Choice? ";BL$(1,
   24);:POSITION 10,20:IF FIRST THEN FIRS
   T=0:KEY$="R":? KEY$;:GOTO 1570
VA 1560 GOSUB 110:POSITION 10,20:? KEY$;
EV 1570 IF KEY$="R" THEN ? "ead File":GOT
   O 810
OT 1580 IF KEY$="E" THEN ? "dit VNT":GOTO
    390
JQ 1590 IF KEY$="W" THEN ? "rite File":GO
   TO 1010
EX 1600 IF KEY$="C" THEN ? "hange All":GO
   TO 1250
OU 1610 ? CHR$(253);:GOTO 1550
IO 1620 REM
XF 1630 REM *** WAIT LOOP
IU 1640 REM
AM 1650 FOR J=1 TO 150:NEXT J:RETURN
JA 1660 REM
FP 1670 REM *** DISK ERROR ROUTINES
JG 1680 REM
FI 1690 SOURCE$="NONE":NUMVAR=0
BI 1700 CLOSE #2:CLOSE #3:POSITION 2,22:?
    "ERROR: ";PEEK(195);" AT LINE ";PEEK(
   186)+256*PEEK(187)
XJ 1710 FOR I=101 TO 255 STEP 2:SOUND 0,I
   ,10,10:NEXT I:SOUND 0,0,0,0:GOSUB 150:
   GOTO 1520
OX 1720 POSITION 2,22:? CHR$(253);"NO FIL
   E LOADED!";:GOSUB 1650:GOSUB 150:RETUR
   N
ZM 1730 POSITION 2,22:? CHR$(253);"NOT SA
   VED BASIC!":GOSUB 1650:GOSUB 150:SOURC
   E$="NONE":NUMVAR=0:GOTO 1520
RM 1740 POSITION 2,22:? CHR$(253);"DESTIN
   ATION CANNOT BE":? "SAME AS SOURCE!";:
   GOSUB 1650:GOSUB 150:RETURN
```

STAR RIDER

by Paul Lay

Star Rider is an all-machine-code, 3-D arcade-action space game for any Atari 8-bit computer with a minimum 32K RAM, controlled by a joystick in Port 1.

Introduction

It is many centuries into our future, and all the countries of the Earth are currently united in an extensive colonization program involving the creation of new civilizations on distant planets. However, the entire program

is under threat from the Xylons, a massive alien empire that for years has been trying to gain supremacy of the universe.

It has taken you many years of intense training, but you've finally graduated as a Star Rider. Armed with the latest in photon weaponry, your craft orbits one of the colonization planets, and you are all that stands between this planet's destruction by the Xylons.

Typing it in

To create your copy of Star Rider, type in Listing 1 using the M/L Editor found elsewhere in this issue. Create the game file under the filename STARRIDE.OBJ, or, if you want the program to load automatically, use the filename AUTORUN.SYS.

Keyboard controls

START: Begin game at Level 1.

1,2,3,4,5: Begin game at Levels 10, 20, 30, 40, 50, respectively.

C: Continue game from previous level.

T: Toggle vertical control.

SPACEBAR: Pause/resume game.

ESC: Abort game.

Game play

When you start a game, you will be presented with a cockpit view from your spacecraft. In the viewport you can see the planet surface below with the attacking Xylon crafts appearing from over the horizon. You will also see your photon-gun sights in the viewport. These are controlled by a joystick in Port 1, and photons are fired by hitting the trigger. Note two points: First, the left photon is always fired first, and the right photon is only fired if the left's already been fired! Second, the vertical control of the sights can be toggled by the T key.

At the bottom of the screen, you can see the control panel displaying your score, shields and the attack wave which you are on. For every Xylon craft which you destroy, five points are added to your score, and your shields are increased by an amount depending upon the attack wave (more for higher waves). Note that your shields cannot increase above 100%. However, if a Xylon craft is not destroyed, and instead shoots past you, deflecting off your shields, then 4% is deducted from your shields. If your shields fall below 0%, the game ends.

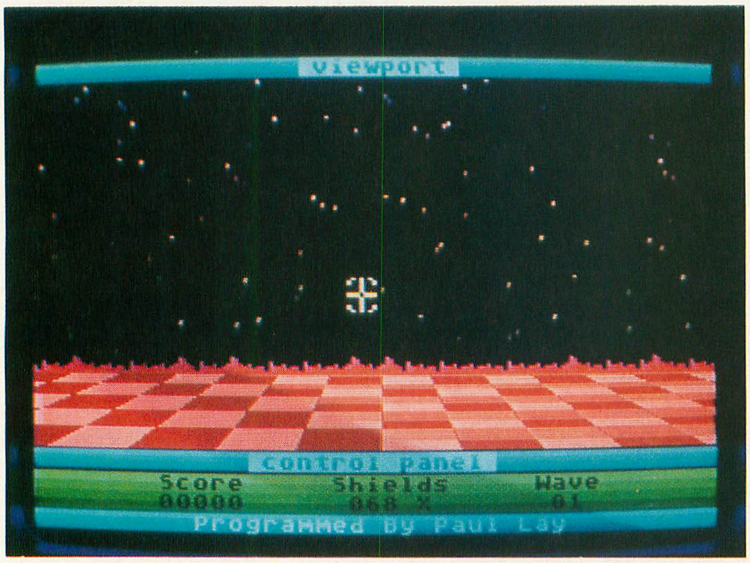
When the game is over, if your score is high enough, you will be able to enter your name on the high-score board. Moving the joystick left and right will move the cursor in the appropriate direction, and moving the joystick up and down will move through the characters. When you have finished entering your name, press the trigger. Note that on subsequent high scores, the previously entered name will appear (in order to save having to enter the same name over again).

The program

Full use is made of the excellent graphics features of the Atari, utilizing player/missile graphics, custom display lists, GTIA modes, display-list interrupts, multiple character sets, load memory scans and so on.

The Xylon ships are drawn in 16-luminance mode, GTIA 9, which gives an excellent metallic effect. Actually, character graphics are used behind GTIA, giving characters of resolution two by eight in 16 luminances. The scrolling terrain is drawn in high-resolution ANTIC F (animated as part of the DLI), and the control panel is done in ANTIC 2. Players are used for the photons and sights, and a single missile is used for the scrolling stars (in both the game and title screen). The metallic text effects on the control panel and on the title screen are done by altering color register luminances from light to dark. The sound on the title screen uses 15KHz clocking and a high-pass filter in Channel 1. Finally note that no OS routines are used at all.

STARIDER



LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,253,31,198,61,76,215
,38,0,0,0,0,0,0,0,6598
1010 DATA 0,0,0,0,0,48,243,255,255,0,4
8,60,60,255,255,255,837
1020 DATA 255,0,0,0,12,12,207,207,255,
0,0,0,11,10,138,121,1096
1030 DATA 105,0,12,212,221,228,238,244
,255,204,201,215,213,231,233,255,814
1040 DATA 249,204,156,125,93,126,158,2
55,159,0,192,77,221,78,238,79,298
1050 DATA 255,0,0,0,176,160,168,151,15
0,89,89,80,96,112,128,144,6748
1060 DATA 153,244,15,4,14,0,0,0,0,247,
245,231,233,221,13,0,6087
1070 DATA 0,127,95,126,158,221,208,0,0
,79,240,64,224,0,0,0,2795
1080 DATA 0,149,149,5,6,7,8,9,153,0,0,
0,0,12,13,174,6569
1090 DATA 159,0,0,10,187,204,13,238,15
,0,0,160,187,204,208,238,9713
1100 DATA 240,0,0,0,0,192,208,234,249,
143,126,128,144,160,187,0,9330
1110 DATA 0,255,13,204,11,0,0,0,0,255,
208,204,176,0,0,0,2104
1120 DATA 0,248,231,8,9,10,187,0,0,0,1
3,0,159,128,157,160,2672
1130 DATA 187,204,221,238,255,238,221,
204,0,0,208,0,249,8,217,10,8274
1140 DATA 187,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,1327
1150 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1150
1160 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1160
1170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1170
1180 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1180
1190 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1190
1200 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1200
1210 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1210
1220 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1220
1230 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1230
1240 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1240
1250 DATA 0,0,0,0,0,0,0,0,0,131,129,13
0,129,129,131,129,3051
1260 DATA 129,130,131,129,130,129,131,
129,131,129,129,130,129,130,131,129,89
05
1270 DATA 131,129,129,129,131,129,129,
129,130,129,131,129,131,130,129,131,89
29
1280 DATA 129,255,255,0,0,3,255,255,25
5,0,0,1,255,255,255,0,9544
1290 DATA 0,1,255,255,255,0,0,1,255,25
5,255,0,0,1,255,255,9929
1300 DATA 255,0,0,1,255,255,255,128,0,
255,252,0,0,15,255,255,610
1310 DATA 254,0,0,7,255,255,254,0,0,1,
255,255,255,0,0,1,5381
1320 DATA 255,255,255,128,0,0,255,255,
255,192,0,0,127,255,255,240,4288
1330 DATA 0,255,224,0,0,127,255,255,24
8,0,0,15,255,255,252,0,176
1340 DATA 0,3,255,255,255,0,0,0,255,25
5,255,192,0,0,63,255,9385
1350 DATA 255,240,0,0,31,255,255,254,0
,255,0,0,1,255,255,255,1625
1360 DATA 224,0,0,63,255,255,248,0,0,3
,255,255,255,0,0,0,5587
1370 DATA 255,255,255,224,0,0,31,255,2
55,252,0,0,3,255,255,255,2382
1380 DATA 224,248,0,0,7,255,255,255,12
8,0,0,127,255,255,240,0,651
1390 DATA 0,7,255,255,255,0,0,0,127,25
5,255,240,0,0,7,255,8027
1400 DATA 255,255,0,0,0,255,255,255,25
2,192,0,0,63,255,255,254,3986
1410 DATA 0,0,0,255,255,255,224,0,0,7,
255,255,255,0,0,0,6053
1420 DATA 127,255,255,248,0,0,3,255,25
5,255,192,0,0,31,255,255,1171
1430 DATA 255,0,0,0,255,255,255,248,0,
0,3,255,255,255,192,0,1117
1440 DATA 0,15,255,255,255,0,0,0,63,25
5,255,252,0,0,1,255,7571
1450 DATA 255,255,240,0,0,7,255,255,25
5,0,0,7,255,255,255,224,3475
1460 DATA 0,0,7,255,255,255,128,0,0,15
,255,255,255,0,0,0,5532
1470 DATA 63,255,255,254,0,0,0,127,255
,255,252,0,0,0,255,255,362
1480 DATA 255,0,0,31,255,255,255,128,0
,0,31,255,255,255,0,0,7759
1490 DATA 0,31,255,255,255,0,0,0,31,25
5,255,255,0,0,0,63,4314
1500 DATA 255,255,255,0,0,0,63,255,255
,0,0,255,255,255,254,0,1561
1510 DATA 0,0,63,255,255,254,0,0,0,31,
255,255,255,0,0,0,5008
1520 DATA 31,255,255,255,128,0,0,15,25
5,255,255,192,0,0,7,255,8745
1530 DATA 255,0,3,255,255,255,248,0,0,
0,255,255,255,252,0,0,63
1540 DATA 0,63,255,255,255,0,0,0,15,25
5,255,255,192,0,0,7,5884
1550 DATA 255,255,255,240,0,0,1,255,25
5,0,31,255,255,255,224,0,2028
1560 DATA 0,1,255,255,255,248,0,0,0,63
,255,255,255,0,0,0,5920
1570 DATA 15,255,255,255,224,0,0,1,255
,255,255,252,0,0,0,63,6690
1580 DATA 255,0,127,255,255,255,128,0,
0,7,255,255,255,240,0,0,9547
1590 DATA 0,127,255,255,255,0,0,0,7,25
5,255,255,240,0,0,0,6502
1600 DATA 255,255,255,255,0,0,0,15,255
,1,255,255,255,254,0,0,9311
1610 DATA 0,15,255,255,255,224,0,0,0,1
27,255,255,255,0,0,0,6494
1620 DATA 7,255,255,255,248,0,0,0,63,2
55,255,255,192,0,0,1,6656
1630 DATA 255,15,255,255,255,248,0,0,0
,63,255,255,255,192,0,0,8961
1640 DATA 0,255,255,255,255,0,0,0,3,25
5,255,255,252,0,0,0,6928
1650 DATA 31,255,255,255,240,0,0,0,127
,63,255,255,255,224,0,0,9265
1660 DATA 0,127,255,255,255,128,0,0,0,
255,255,255,255,0,0,0,7472
1670 DATA 3,255,255,255,254,0,0,0,7,25
5,255,255,248,0,0,0,6940
1680 DATA 15,255,255,255,255,192,0,0,0
,255,255,255,255,128,0,0,9939
1690 DATA 1,255,255,255,255,0,0,0,1,25
5,255,255,254,0,0,0,6987
1700 DATA 3,255,255,255,254,0,0,0,3,25
5,255,255,255,0,0,0,7025
```

```
1710 DATA 3,255,255,255,255,0,0,0,1,25
5,255,255,255,0,0,0,7022
1720 DATA 1,255,255,255,255,0,0,0,1,25
5,255,255,255,128,0,0,8822
1730 DATA 0,255,255,255,252,0,0,0,7,25
5,255,255,254,0,0,0,7065
1740 DATA 3,255,255,255,255,0,0,0,0,25
5,255,255,255,128,0,0,8835
1750 DATA 0,127,255,255,255,224,0,0,0,
255,255,255,240,0,0,0,7943
1760 DATA 31,255,255,255,252,0,0,0,3,2
55,255,255,255,0,0,0,7103
1770 DATA 0,255,255,255,255,192,0,0,0,
63,255,255,255,248,0,0,9774
1780 DATA 0,255,255,255,192,0,0,0,63,2
55,255,255,248,0,0,0,7241
1790 DATA 7,255,255,255,255,0,0,0,0,12
7,255,255,255,224,0,0,8953
1800 DATA 0,15,255,255,255,254,0,0,0,2
55,255,255,0,0,0,0,4829
1810 DATA 255,255,255,255,240,0,0,0,7,
255,255,255,255,0,0,0,7353
1820 DATA 0,127,255,255,255,240,0,0,0,
7,255,255,255,255,128,0,1314
1830 DATA 0,255,255,252,0,0,0,1,255,25
5,255,255,224,0,0,0,7743
1840 DATA 15,255,255,255,255,0,0,0,0,6
3,255,255,255,248,0,0,8707
1850 DATA 0,1,255,255,255,255,224,0,0,
255,255,240,0,0,0,7,6357
1860 DATA 255,255,255,255,192,0,0,0,15
,255,255,255,255,0,0,0,7235
1870 DATA 0,63,255,255,255,252,0,0,0,0
,255,255,255,255,248,0,3038
1880 DATA 0,255,255,192,0,0,0,15,255,2
55,255,255,128,0,0,0,6417
1890 DATA 31,255,255,255,255,0,0,0,0,3
1,255,255,255,254,0,0,8537
1900 DATA 0,0,63,255,255,255,254,0,0,2
55,255,0,0,0,0,63,4055
1910 DATA 255,255,255,255,0,0,0,0,31,2
55,255,255,255,0,0,0,6469
1920 DATA 0,31,255,255,255,255,0,0,0,0
,31,255,255,255,255,128,2731
1930 DATA 0,255,252,0,0,0,0,127,255,25
5,255,254,0,0,0,0,4910
1940 DATA 63,255,255,255,255,0,0,0,0,1
5,255,255,255,255,128,0,393
1950 DATA 0,0,7,255,255,255,255,224,0,
255,240,0,0,0,0,255,8643
1960 DATA 255,255,255,252,0,0,0,0,63,2
55,255,255,255,0,0,0,6795
1970 DATA 0,15,255,255,255,255,192,0,0
,0,3,255,255,255,255,248,5705
1980 DATA 0,255,192,0,0,0,3,255,255,25
5,255,248,0,0,0,0,5753
1990 DATA 127,255,255,255,255,0,0,0,0,
7,255,255,255,255,224,0,1867
2000 DATA 0,0,1,255,255,255,255,254,0,
255,0,0,0,0,7,255,6380
2010 DATA 255,255,255,240,0,0,0,0,127,
255,255,255,255,0,0,0,7373
2020 DATA 0,7,255,255,255,255,240,0,0,
0,0,127,255,255,255,255,4618
2030 DATA 128,252,0,0,0,0,31,255,255,2
55,255,224,0,0,0,0,5257
2040 DATA 255,255,255,255,255,0,0,0,0,
3,255,255,255,255,248,0,2365
2050 DATA 0,0,0,63,255,255,255,255,224
,48,48,112,240,66,104,41,9560
2060 DATA 66,0,116,2,2,2,2,2,2,2,2,2,2
,2,2,2,2734
2070 DATA 68,144,33,79,184,33,15,15,15
,15,15,15,15,15,15,15,5684
2080 DATA 15,15,15,15,15,15,15,15,15,1
5,15,15,15,15,15,15,4120
2090 DATA 15,15,15,15,0,66,144,41,0,2,
2,0,2,65,144,38,7718
2100 DATA 169,90,133,2,169,56,133,3,16
9,2,133,9,169,0,141,68,3504
2110 DATA 2,141,235,0,141,236,0,133,21
2,169,1,141,226,0,169,3,7106
2120 DATA 141,15,210,141,29,208,169,72
,141,8,2,169,48,141,9,2,2801
2130 DATA 169,64,141,14,210,169,120,14
1,7,212,169,128,141,14,212,169,429
2140 DATA 62,141,0,212,32,120,49,32,9,
52,32,80,48,32,55,40,9261
2150 DATA 169,0,141,0,212,169,144,141,
2,212,169,38,141,3,212,169,9164
2160 DATA 95,141,0,2,169,40,141,1,2,16
9,222,141,19,208,169,60,7121
2170 DATA 141,20,208,169,140,141,21,20
8,32,152,54,162,4,189,10,40,4842
2180 DATA 157,22,208,202,16,247,32,107
,49,169,0,141,254,0,141,255,9775
2190 DATA 0,141,8,210,162,2,157,208,0,
202,16,250,169,255,141,244,3903
2200 DATA 0,141,249,0,141,230,0,141,23
2,0,169,120,141,242,0,141,9306
2210 DATA 243,0,169,16,141,243,41,141,
244,41,162,4,157,231,41,202,160
2220 DATA 16,250,169,17,141,242,41,169
,4,141,15,40,32,55,40,169,3688
2230 DATA 62,141,0,212,173,226,0,141,2
27,0,32,43,40,32,61,40,2205
2240 DATA 32,55,40,32,84,43,32,15,46,3
2,61,40,32,55,40,32,7835
2250 DATA 67,40,32,58,44,32,206,46,32,
61,40,32,55,40,32,15,8664
2260 DATA 46,32,177,47,32,183,48,32,61
,40,32,55,40,32,67,40,9513
2270 DATA 32,58,44,32,206,46,32,61,40,
173,227,0,208,194,32,55,6063
2280 DATA 40,32,18,40,173,226,0,201,25
5,240,169,24,105,1,141,226,379
2290 DATA 0,208,161,0,12,0,70,0,0,68,7
4,162,1,189,254,41,4302
2300 DATA 24,105,1,201,26,240,4,157,25
4,41,96,169,16,157,254,41,8847
2310 DATA 202,16,234,96,162,0,32,18,43
,232,236,226,0,208,247,96,976
2320 DATA 173,11,212,208,251,96,173,11
,212,240,251,96,174,15,40,232,2118
2330 DATA 224,5,208,16,173,16,40,72,17
3,17,40,141,16,40,104,141,3512
2340 DATA 17,40,162,1,142,15,40,96,72,
138,72,152,72,162,139,142,6980
2350 DATA 10,212,142,24,208,202,224,13
1,208,245,169,65,162,32,141,10,9964
2360 DATA 212,141,27,208,142,9,212,173
,111,120,72,162,110,189,0,120,8330
2370 DATA 141,10,212,141,4,208,157,1,1
20,142,18,208,202,208,238,104,2072
2380 DATA 141,1,120,169,0,141,10,212,1
41,4,208,169,1,141,10,212,7325
2390 DATA 141,27,208,162,8,142,10,212,
202,208,250,173,16,40,172,17,8859
2400 DATA 40,141,23,208,140,24,208,174
,15,40,142,10,212,202,208,250,2236
2410 DATA 141,24,208,140,23,208,162,5,
142,10,212,202,208,250,141,23,1141
2420 DATA 208,140,24,208,162,6,142,10,
212,202,208,250,141,24,208,140,2477
2430 DATA 23,208,162,7,142,10,212,202,
208,250,141,23,208,140,24,208,1804
2440 DATA 162,8,142,10,212,202,208,250
,174,15,40,224,4,240,14,141,9534
2450 DATA 24,208,140,23,208,142,10,212
,232,224,4,208,248,169,12,141,1954
2460 DATA 10,212,141,23,208,162,139,16
9,224,141,10,212,142,24,208,141,1384
2470 DATA 9,212,202,142,10,212,142,24,
208,202,224,131,208,245,169,0,3182
2480 DATA 141,10,212,141,23,208,162,20
3,142,10,212,142,24,208,142,10,8890
2490 DATA 212,202,224,195,208,242,169,
12,141,10,212,141,23,208,162,139,1587
2500 DATA 142,10,212,142,24,208,202,22
4,131,208,245,104,168,104,170,104,3496
2510 DATA 64,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,2574
2520 DATA 214,246,233,229,247,240,239,
242,244,194,0,0,0,0,0,0,5261
2530 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,2530
2540 DATA 0,0,0,0,0,214,227,239,238,24
4,242,239,236,128,240,225,9497
2550 DATA 238,229,236,194,0,0,0,0,0,0,
0,0,0,0,0,0,4730
2560 DATA 0,0,0,0,0,0,0,0,51,99,111,11
4,101,0,0,0,7911
2570 DATA 0,0,51,104,105,101,108,100,1
15,0,0,0,0,0,55,97,9238
2580 DATA 118,101,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,2900
2590 DATA 16,16,16,16,16,0,0,0,0,0,0,1
7,16,16,0,5,3546
2600 DATA 0,0,0,0,0,0,0,16,17,0,0,0,0,
0,0,0,2881
2610 DATA 0,0,0,0,0,0,0,0,0,48,114,1
11,103,114,97,653
```

```
2620 DATA 109,109,101,100,0,34,121,0,4
8,97,117,108,0,44,97,121,2693
2630 DATA 0,0,0,0,0,0,0,0,0,138,72,169
,0,133,205,165,4407
2640 DATA 241,10,38,205,10,38,205,10,3
8,205,133,206,166,205,134,207,2305
2650 DATA 10,38,205,10,38,205,24,101,2
06,133,204,165,205,101,207,133,2507
2660 DATA 205,165,204,24,101,240,133,2
04,144,2,230,205,165,204,24,105,1758
2670 DATA 0,133,204,165,205,105,116,13
3,205,104,170,96,32,48,42,169,8068
2680 DATA 9,24,109,255,0,160,5,145,204
,56,233,1,136,16,248,169,9626
2690 DATA 15,24,109,255,0,160,45,145,2
04,56,233,1,136,192,39,208,9875
2700 DATA 246,96,32,48,42,169,19,24,10
9,255,0,160,3,145,204,56,6451
2710 DATA 233,1,136,16,248,169,23,24,1
09,255,0,160,43,145,204,56,8020
2720 DATA 233,1,136,192,39,208,246,96,
32,48,42,169,26,24,109,255,7711
2730 DATA 0,160,2,145,204,56,233,1,136
,16,248,96,169,0,72,32,5684
2740 DATA 48,42,104,160,5,145,204,136,
16,251,160,45,145,204,136,192,2042
2750 DATA 39,208,249,96,169,0,72,32,48
,42,104,160,3,145,204,136,7162
2760 DATA 16,251,160,43,145,204,136,19
2,39,208,249,96,169,0,72,32,8478
2770 DATA 48,42,104,160,2,145,204,136,
16,251,96,173,10,210,41,63,7729
2780 DATA 201,35,16,247,157,199,61,169
,14,157,199,62,169,255,157,199,3780
2790 DATA 63,169,32,157,199,64,173,10,
210,41,127,157,199,65,96,208,431
2800 DATA 3,76,3,43,201,1,208,3,76,235
,42,76,211,42,208,3,6534
2810 DATA 76,191,42,201,1,208,3,76,153
,42,76,115,42,162,0,142,5179
2820 DATA 231,0,189,199,64,201,3,48,98
,41,16,208,29,189,199,64,7341
2830 DATA 41,32,240,15,222,199,65,208,
10,169,0,157,199,64,169,9,7964
2840 DATA 157,199,65,232,236,226,0,208
,217,96,222,199,65,48,31,169,1147
2850 DATA 23,141,255,0,189,199,61,133,
240,189,199,62,133,241,189,199,5655
2860 DATA 64,41,3,32,69,43,169,0,141,2
55,0,76,122,43,189,199,7867
2870 DATA 61,133,240,189,199,62,133,24
1,189,199,64,41,3,32,54,43,5771
2880 DATA 169,3,157,199,64,206,227,0,7
6,122,43,189,199,61,133,240,1388
2890 DATA 189,199,62,133,241,189,199,6
4,32,54,43,222,199,65,208,29,9485
2900 DATA 169,9,157,199,65,254,199,64,
189,199,64,201,3,48,14,32,6348
2910 DATA 106,47,169,4,141,230,0,32,18
,43,76,122,43,189,199,62,6048
2920 DATA 24,125,199,63,157,199,62,133
,241,141,231,0,208,5,169,1,8965
2930 DATA 157,199,63,173,10,210,201,25
,176,10,189,199,61,240,22,222,1469
2940 DATA 199,61,16,17,173,10,210,201,
25,176,10,189,199,61,201,34,8743
2950 DATA 240,3,254,199,61,189,199,61,
133,240,189,199,64,32,69,43,9141
2960 DATA 76,122,43,173,16,208,208,26,
173,254,0,240,24,169,0,141,9004
2970 DATA 254,0,173,244,0,16,3,76,19,4
5,173,249,0,16,6,76,2486
2980 DATA 137,45,141,254,0,160,15,174,
243,0,169,0,157,0,125,232,8777
2990 DATA 136,16,249,173,0,211,41,15,1
33,204,174,235,0,240,18,162,463
3000 DATA 0,134,239,74,38,239,74,38,23
9,165,204,41,252,5,239,133,2323
3010 DATA 204,70,204,176,9,173,243,0,5
6,233,8,141,243,0,70,204,9541
3020 DATA 176,9,173,243,0,24,105,8,141
,243,0,70,204,176,9,173,8206
3030 DATA 242,0,56,233,8,141,242,0,70,
204,176,9,173,242,0,24,7687
3040 DATA 105,8,141,242,0,173,242,0,20
1,48,176,7,169,48,141,242,449
3050 DATA 0,16,9,201,201,144,5,169,200
,141,242,0,173,243,0,201,1908
3060 DATA 40,176,7,169,40,141,243,0,16
,9,201,177,144,5,169,176,8758
3070 DATA 141,243,0,160,15,174,243,0,1
85,3,45,157,0,125,232,136,8637
3080 DATA 16,246,173,242,0,141,1,208,1
41,208,0,96,0,84,146,146,7795
3090 DATA 16,16,16,238,238,16,16,16,14
6,146,84,0,165,242,56,233,9463
3100 DATA 48,74,74,74,141,245,0,169,17
6,56,229,243,74,74,74,141,284
3110 DATA 246,0,169,8,141,244,0,169,48
,141,247,0,169,176,141,248,2719
3120 DATA 0,169,4,141,232,0,76,109,45,
173,244,0,16,1,96,160,5639
3130 DATA 15,174,248,0,169,0,157,0,126
,232,136,16,249,173,247,0,687
3140 DATA 24,109,245,0,141,247,0,173,2
48,0,56,237,246,0,141,248,2661
3150 DATA 0,206,244,0,48,27,160,15,174
,248,0,185,255,45,45,10,6982
3160 DATA 210,157,0,126,232,136,16,243
,173,247,0,141,2,208,141,209,2336
3170 DATA 0,96,169,200,56,229,242,74,7
4,74,141,250,0,169,176,56,468
3180 DATA 229,243,74,74,74,141,251,0,1
69,8,141,249,0,169,200,141,1148
3190 DATA 252,0,169,176,141,253,0,169,
4,141,232,0,76,227,45,173,9835
3200 DATA 249,0,16,1,96,160,15,174,253
,0,169,0,157,0,127,232,8232
3210 DATA 136,16,249,173,252,0,56,237,
250,0,141,252,0,173,253,0,1407
3220 DATA 56,237,251,0,141,253,0,206,2
49,0,48,27,160,15,174,253,415
3230 DATA 0,185,255,45,45,10,210,157,0
,127,232,136,16,243,173,252,3247
3240 DATA 0,141,3,208,141,210,0,96,24,
60,126,126,126,255,255,255,3923
3250 DATA 255,255,255,126,126,126,60,2
4,32,96,46,32,151,46,32,64,3531
3260 DATA 45,76,182,45,56,233,40,74,74
,74,141,225,0,173,224,0,8172
3270 DATA 56,233,48,74,74,56,253,199,6
1,24,105,1,48,35,72,189,5475
3280 DATA 199,64,168,104,217,90,46,16,
24,173,225,0,56,253,199,62,9270
3290 DATA 24,105,1,48,12,217,93,46,16,
7,169,8,141,233,0,56,4260
3300 DATA 96,24,96,3,4,6,1,2,2,165,244
,240,1,96,162,0,4842
3310 DATA 189,199,64,201,3,48,7,232,23
6,226,0,208,243,96,165,247,4911
3320 DATA 133,224,165,248,32,27,46,144
,20,32,233,46,32,70,47,189,5924
3330 DATA 199,64,9,16,157,199,64,169,8
,157,199,65,96,76,110,46,6836
3340 DATA 165,249,240,1,96,162,0,189,1
99,64,201,3,48,7,232,236,347
3350 DATA 226,0,208,243,96,165,252,133
,224,165,253,32,27,46,144,20,9778
3360 DATA 32,233,46,32,70,47,189,199,6
4,9,16,157,199,64,169,8,6543
3370 DATA 157,199,65,96,76,165,46,162,
0,173,10,210,41,3,168,189,7971
3380 DATA 32,32,57,229,46,157,216,32,2
32,224,182,208,236,96,0,15,981
3390 DATA 240,255,173,242,41,201,17,24
0,73,165,226,56,233,1,41,248,2168
3400 DATA 74,74,74,201,4,48,2,169,3,16
8,185,62,47,141,228,0,6813
3410 DATA 185,66,47,141,229,0,173,244,
41,24,109,228,0,201,26,48,7256
3420 DATA 6,238,229,0,56,233,10,141,24
4,41,173,243,41,24,109,229,1058
3430 DATA 0,201,26,48,14,169,17,141,24
2,41,169,16,141,243,41,141,9178
3440 DATA 244,41,96,141,243,41,96,2,5,
0,0,0,0,1,2,160,9416
3450 DATA 4,185,231,41,24,105,5,201,26
,48,19,56,233,10,153,231,7829
3460 DATA 41,136,48,14,185,231,41,24,1
05,1,201,26,16,237,153,231,9758
3470 DATA 41,96,96,160,2,185,242,41,56
,233,4,201,16,16,19,24,4164
3480 DATA 105,10,153,242,41,136,48,14,
185,242,41,56,233,1,201,16,8023
3490 DATA 48,237,153,242,41,96,169,16,
141,242,41,141,243,41,141,244,3115
3500 DATA 41,32,196,54,32,74,49,162,3,
169,0,157,64,48,157,68,5200
3510 DATA 48,202,16,247,162,0,154,76,3
3,39,174,230,0,48,27,189,6956
3520 DATA 21,48,141,144,38,189,26,48,1
41,0,210,141,64,48,189,31,6632
3530 DATA 48,141,1,210,141,68,48,206,2
30,0,173,231,0,141,2,210,9909
3540 DATA 141,65,48,141,3,210,141,69,4
8,174,232,0,48,21,189,36,6386
```

```
3550 DATA 48,141,4,210,141,66,48,189,4
1,48,141,5,210,141,70,48,6663
3560 DATA 206,232,0,174,233,0,48,21,18
9,46,48,141,6,210,141,67,7181
3570 DATA 48,189,55,48,141,7,210,141,7
1,48,206,233,0,96,48,32,6455
3580 DATA 64,16,80,0,128,176,208,224,0
,130,132,134,136,0,160,140,9628
3590 DATA 120,48,0,138,140,142,198,0,2
40,230,220,210,200,190,180,170,7376
3600 DATA 0,136,138,140,140,142,142,14
2,140,0,0,0,0,0,0,0,9788
3610 DATA 0,173,9,210,141,234,0,104,54
,32,120,52,32,6,55,169,4633
3620 DATA 255,133,234,173,31,208,201,6
,208,4,162,0,240,34,165,234,1902
3630 DATA 201,45,208,11,173,235,0,73,2
55,141,235,0,76,86,48,201,9866
3640 DATA 18,240,30,162,5,221,159,48,2
40,6,202,208,248,76,90,48,1068
3650 DATA 189,165,48,133,226,189,171,4
8,141,255,41,189,177,48,141,254,4360
3660 DATA 41,169,255,133,234,76,74,49,
255,31,30,26,24,29,1,10,2012
3670 DATA 20,30,40,50,17,16,16,16,16,1
6,16,17,18,19,20,21,6311
3680 DATA 165,234,201,28,208,3,76,152,
47,201,45,208,11,165,235,73,404
3690 DATA 255,133,235,169,255,133,234,
96,201,33,208,118,32,74,49,32,8613
3700 DATA 55,40,169,133,141,2,212,169,
49,141,3,212,169,64,141,27,8495
3710 DATA 208,169,224,141,26,208,162,2
,169,0,157,1,208,202,16,250,1052
3720 DATA 169,255,133,234,169,5,141,8,
210,169,166,141,1,210,141,5,9906
3730 DATA 210,169,128,141,0,210,169,64
,141,4,210,32,192,49,165,234,1585
3740 DATA 201,33,208,247,32,55,40,169,
144,141,2,212,169,38,141,3,7905
3750 DATA 212,169,1,141,27,208,169,0,1
41,26,208,162,2,189,208,0,8986
3760 DATA 157,1,208,202,16,247,169,255
,133,234,169,0,141,8,210,32,1139
3770 DATA 85,49,96,162,7,169,0,157,0,2
10,202,16,250,96,162,3,8780
3780 DATA 160,6,189,64,48,153,0,210,18
9,68,48,153,1,210,202,136,517
3790 DATA 136,16,239,96,169,0,133,222,
169,116,133,223,162,4,76,138,941
3800 DATA 54,169,0,133,222,169,104,133
,223,162,8,76,138,54,112,112,9289
3810 DATA 112,79,0,104,15,15,15,15,15,
15,15,15,15,15,15,15,6386
3820 DATA 15,15,15,15,15,15,15,15,15,1
5,15,15,15,15,15,15,5860
3830 DATA 15,15,15,15,15,15,15,15,15,1
5,15,15,15,15,15,15,5870
3840 DATA 15,15,15,15,15,15,65,133,49,
32,55,40,162,13,189,81,3939
3850 DATA 50,133,237,189,95,50,133,238
,189,157,50,141,226,49,189,171,3951
3860 DATA 50,141,227,49,138,72,162,0,1
60,0,189,255,255,145,237,165,5444
3870 DATA 237,24,105,40,133,237,144,2,
230,238,232,224,24,208,235,104,5844
3880 DATA 170,202,16,202,162,13,189,66
,50,157,67,50,202,208,247,166,3305
3890 DATA 236,224,48,208,4,162,0,134,2
36,189,109,50,141,67,50,230,628
3900 DATA 236,162,13,189,67,50,48,17,1
89,81,50,24,125,67,50,157,5536
3910 DATA 81,50,144,20,254,95,50,208,1
5,189,81,50,24,125,67,50,5840
3920 DATA 157,81,50,176,3,222,95,50,20
2,16,216,96,0,0,0,0,3011
3930 DATA 0,0,0,0,0,0,0,0,0,0,61,62,63
,64,65,66,9091
3940 DATA 67,68,69,70,71,72,73,74,106,
106,106,106,106,106,106,106,7120
3950 DATA 106,106,106,106,106,106,0,0,
0,40,40,40,40,216,216,216,7736
3960 DATA 216,0,0,40,40,40,80,80,120,1
36,176,176,216,216,216,0,1536
3970 DATA 0,0,40,80,120,136,176,216,0,
0,40,40,40,40,40,40,2026
3980 DATA 136,176,216,0,0,0,185,209,23
3,1,25,49,73,97,121,145,7495
3990 DATA 169,193,217,241,50,50,50,51,
51,51,51,51,51,51,51,51,2568
4000 DATA 51,51,0,0,0,0,136,153,170,18
7,204,221,238,255,255,238,8922
4010 DATA 221,204,187,170,153,0,0,0,0,
0,0,0,0,0,136,153,1133
4020 DATA 170,187,0,13,14,0,15,14,13,1
2,187,170,153,0,0,0,1226
4030 DATA 0,0,0,0,0,0,136,153,170,187,
204,13,14,255,255,238,3391
4040 DATA 221,204,187,170,153,0,0,0,0,
0,0,0,0,0,136,153,1163
4050 DATA 170,187,192,13,14,0,15,14,13
,12,187,170,153,0,0,0,1832
4060 DATA 0,0,0,0,0,0,136,153,170,187,
204,13,14,15,15,14,2877
4070 DATA 13,12,187,170,153,0,0,0,0,0,
0,0,0,0,136,153,601
4080 DATA 170,187,12,13,14,15,15,14,13
,0,187,170,153,0,0,0,1292
4090 DATA 0,0,0,0,0,0,136,153,170,187,
12,13,14,15,15,14,795
4100 DATA 13,12,187,170,153,0,0,0,0,0,
0,0,0,0,136,153,631
4110 DATA 170,187,0,13,14,0,255,238,22
1,0,187,170,153,0,0,0,6540
4120 DATA 0,0,0,0,0,0,136,153,170,187,
12,221,238,15,15,14,6233
4130 DATA 13,12,187,170,153,0,0,0,0,0,
0,0,0,0,136,153,661
4140 DATA 170,187,0,13,14,0,15,14,13,0
,187,170,153,0,0,0,1226
4150 DATA 0,0,0,0,0,0,136,153,170,187,
12,221,238,255,255,238,6807
4160 DATA 221,12,187,170,153,0,0,0,0,0
,0,0,0,0,136,153,899
4170 DATA 170,187,0,13,14,15,15,14,13,
0,187,170,153,0,0,0,1346
4180 DATA 0,0,0,0,0,0,136,153,170,187,
204,13,14,15,15,14,2997
4190 DATA 13,204,187,170,153,0,0,0,0,0
,0,0,0,0,136,153,1105
4200 DATA 170,187,204,221,238,255,255,
238,221,204,187,170,153,0,0,0,2764
4210 DATA 0,0,169,0,133,222,169,80,133
,223,162,11,32,138,54,162,9628
4220 DATA 5,189,96,52,133,220,189,102,
52,133,221,189,108,52,133,222,3399
4230 DATA 169,114,52,133,223,138,72,20
1,4,48,4,162,15,16,2,162,4935
4240 DATA 39,160,5,177,220,145,222,136
,16,249,165,220,24,105,6,133,1023
4250 DATA 220,144,2,230,221,165,222,24
,105,40,133,222,144,2,230,223,3915
4260 DATA 202,16,222,104,170,202,16,18
5,96,71,55,39,23,7,103,57,4731
4270 DATA 58,59,60,61,61,88,14,20,106,
23,29,80,80,80,80,87,3176
4280 DATA 87,32,74,49,32,55,40,169,201
,141,2,212,169,52,141,3,7844
4290 DATA 212,169,210,141,0,2,169,53,1
41,1,2,169,208,141,24,208,9348
4300 DATA 169,140,141,25,208,169,5,141
,8,210,169,170,141,1,210,141,1813
4310 DATA 3,210,169,166,141,5,210,141,
7,210,169,0,141,219,0,169,862
4320 DATA 39,141,220,0,169,1,141,195,5
4,32,152,54,169,0,141,195,9257
4330 DATA 54,96,112,112,240,79,0,80,15
,15,15,15,15,15,15,15,9174
4340 DATA 15,15,15,15,15,15,15,15,15,1
5,15,15,15,15,15,15,6380
4350 DATA 15,15,15,15,15,15,15,15,15,1
5,15,15,15,15,15,15,6390
4360 DATA 15,15,15,15,15,15,15,15,15,1
5,15,15,15,15,15,15,6400
4370 DATA 15,15,15,15,112,66,30,53,112
,6,112,6,112,6,112,6,1798
4380 DATA 112,6,112,2,65,201,52,0,0,0,
0,0,0,0,0,0,6743
4390 DATA 0,0,0,0,0,0,40,41,39,40,0,51
,35,47,50,37,8816
4400 DATA 51,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,4451
4410 DATA 0,0,97,110,97,108,111,103,0,
0,0,16,17,16,16,16,9008
4420 DATA 0,0,0,0,0,0,97,110,97,108,11
1,103,0,0,0,16,645
4430 DATA 16,21,16,16,0,0,0,0,0,0,97,1
10,97,108,111,103,3073
4440 DATA 0,0,0,16,16,18,21,16,0,0,0,0
,0,0,97,110,8182
4450 DATA 97,108,111,103,0,0,0,16,16,1
7,16,16,0,0,0,0,6318
4460 DATA 0,0,97,110,97,108,111,103,0,
0,0,16,16,16,21,16,9120
4470 DATA 0,0,0,0,0,0,0,0,0,0,0,0,48,1
14,111,103,3
4480 DATA 114,97,109,109,101,100,0,34,
121,0,48,97,117,108,0,44,3446
4490 DATA 97,121,0,0,0,0,0,0,0,0,72,
138,72,162,224,4509
4500 DATA 169,65,141,10,212,142,26,208
,141,27,208,162,60,189,0,120,137
4510 DATA 141,10,212,141,4,208,142,18,
208,222,0,120,138,74,144,3,8847
4520 DATA 222,0,120,202,208,231,142,10
,212,142,26,208,169,1,141,27,278
4530 DATA 208,162,109,138,10,141,10,21
2,141,23,208,24,105,64,141,22,7406
4540 DATA 208,189,60,120,141,4,208,142
,18,208,222,60,120,138,74,144,1417
4550 DATA 3,222,60,120,202,208,220,142
,4,208,206,219,0,16,40,174,1209
4560 DATA 220,0,224,40,208,2,162,0,189
,98,54,141,0,210,24,105,7745
4570 DATA 1,141,2,210,74,141,4,210,24,
105,1,141,6,210,232,142,362
4580 DATA 220,0,169,7,141,219,0,104,17
0,104,64,72,110,74,111,72,7607
4590 DATA 110,74,111,64,126,66,127,64,
126,66,127,72,110,74,111,72,7202
4600 DATA 110,74,111,64,126,66,127,64,
126,66,127,64,74,84,94,62,6373
4610 DATA 72,82,92,160,0,152,145,222,2
00,208,251,230,223,202,208,246,1649
4620 DATA 96,169,0,133,222,169,125,133
,223,162,3,32,138,54,162,0,8673
4630 DATA 173,10,210,172,195,54,208,8,
201,47,144,244,201,207,176,240,7742
4640 DATA 157,0,120,169,1,157,0,123,23
2,208,229,96,0,169,200,133,3097
4650 DATA 211,141,10,212,173,10,210,41
,112,141,144,38,160,64,162,3,8656
4660 DATA 173,10,210,157,4,208,157,18,
208,202,16,244,141,12,208,141,2995
4670 DATA 17,208,41,7,170,173,10,210,4
1,239,157,0,210,136,208,222,4684
4680 DATA 198,211,208,205,169,48,141,1
44,38,169,0,141,12,208,96,165,888
4690 DATA 212,240,46,169,82,133,213,16
9,53,133,214,162,0,160,0,177,1424
4700 DATA 213,217,231,41,144,24,208,5,
200,192,5,208,242,165,213,24,3870
4710 DATA 105,20,133,213,144,2,230,214
,232,224,5,208,224,96,76,59,3379
4720 DATA 55,230,212,96,134,217,169,13
0,133,213,169,53,133,214,169,4,3596
4730 DATA 56,229,217,170,240,36,165,21
3,24,105,20,133,215,165,214,105,3927
4740 DATA 0,133,216,160,19,177,213,145
,215,136,16,249,165,213,56,233,6256
4750 DATA 20,133,213,176,2,198,214,202
,16,218,169,73,133,213,169,53,3844
4760 DATA 133,214,165,217,10,10,72,24,
101,213,133,213,144,2,230,214,3322
4770 DATA 104,10,10,24,101,213,133,213
,144,2,230,214,160,7,185,82,2117
4780 DATA 56,170,189,45,56,145,213,136
,16,244,165,213,24,105,9,133,652
4790 DATA 215,165,214,105,0,133,216,16
0,4,185,231,41,145,215,136,16,2097
4800 DATA 248,160,0,32,27,56,169,255,1
45,213,32,27,56,185,82,56,8745
4810 DATA 170,189,45,56,145,213,173,16
,208,240,67,173,0,211,41,15,9953
4820 DATA 201,7,208,9,200,192,8,48,218
,160,0,16,214,201,11,208,1130
4830 DATA 7,136,16,207,160,7,16,203,20
1,13,208,12,185,82,56,56,8223
4840 DATA 233,1,16,20,169,36,16,16,201
,14,208,183,185,82,56,24,7714
4850 DATA 105,1,201,37,48,2,169,0,153,
82,56,76,186,55,173,16,6907
4860 DATA 208,240,251,96,32,55,40,32,6
1,40,32,55,40,32,61,40,2195
4870 DATA 32,55,40,76,61,40,0,97,98,99
,100,101,102,103,104,105,6949
4880 DATA 106,107,108,109,110,111,112,
113,114,115,116,117,118,119,120,121,65
6
4890 DATA 122,80,81,82,83,84,85,86,87,
88,89,0,0,0,0,0,587
4900 DATA 0,0,0,169,0,141,0,212,141,14
,210,141,14,212,141,8,8922
4910 DATA 210,169,3,141,15,210,169,169
,141,2,212,169,56,141,3,212,1689
4920 DATA 169,120,141,9,212,169,12,141
,23,208,169,0,141,24,208,169,1213
4930 DATA 35,141,0,212,169,14,141,1,21
0,141,3,210,162,1,142,0,8122
4940 DATA 210,232,142,2,210,162,127,17
3,10,210,157,0,120,202,16,247,2840
4950 DATA 48,243,66,231,56,2,66,231,56
,2,66,231,56,2,66,231,8672
4960 DATA 56,2,66,231,56,2,66,231,56,2
,66,231,56,2,66,231,8208
4970 DATA 56,2,66,231,56,2,66,231,56,2
,66,231,56,2,66,231,8218
4980 DATA 56,2,66,231,56,2,66,231,56,2
,66,231,56,65,169,56,7855
4990 DATA 0,1,2,3,4,5,6,7,8,9,10,11,12
,13,14,15,6350
5000 DATA 0,1,2,3,4,5,6,7,8,9,10,11,12
,13,14,15,6360
5010 DATA 0,1,2,3,4,5,6,7,8,9,10,11,12
,13,14,15,6370
5020 DATA 8,9,10,11,12,13,14,15,0,1,2,
3,4,5,6,7,5868
5030 DATA 8,9,10,11,12,13,14,15,0,1,2,
3,4,5,6,7,5878
5040 DATA 8,9,10,11,12,13,14,15,0,1,2,
3,4,5,6,7,5888
5050 DATA 15,255,255,255,255,80,255,25
5,255,255,255,245,255,255,255,255,8320
5060 DATA 255,245,255,255,255,255,255,
245,238,238,238,238,238,230,238,238,78
28
5070 DATA 238,238,238,230,238,238,238,
238,238,230,238,238,238,238,238,230,71
98
5080 DATA 221,221,221,221,221,215,221,
221,221,221,221,215,221,221,112,13,65
5090 DATA 221,215,221,221,112,13,221,2
15,204,204,128,0,0,0,204,204,2801
5100 DATA 128,0,0,0,204,204,128,0,0,0,
204,204,128,0,0,0,4724
5110 DATA 187,187,187,187,187,144,187,
187,187,187,187,185,187,187,187,187,26
0
5120 DATA 187,185,187,187,187,187,187,
185,170,170,170,170,170,170,170,170,88
32
5130 DATA 170,170,170,170,170,170,170,
170,170,170,10,170,170,170,170,170,649
0
5140 DATA 0,0,0,9,153,155,0,0,0,9,153,
155,0,0,0,9,648
5150 DATA 153,155,0,0,0,9,153,155,136,
136,192,8,136,140,136,136,714
5160 DATA 192,8,136,140,136,136,136,13
6,136,140,136,136,136,136,136,140,3576
5170 DATA 119,119,119,119,119,126,119,
119,119,119,119,126,119,119,119,119,14
80
5180 DATA 119,126,119,119,119,119,119,
126,102,102,102,102,102,111,102,102,98
60
5190 DATA 102,102,102,111,102,102,102,
102,102,111,6,102,102,102,102,240,340
5200 DATA 255,255,255,255,255,246,255,
255,255,255,255,246,255,255,255,255,97
18
5210 DATA 255,246,255,255,255,255,255,
246,238,238,238,238,238,231,238,238,80
02
5220 DATA 238,238,238,231,238,238,238,
238,238,231,238,238,238,238,238,231,73
78
5230 DATA 221,221,221,221,221,216,221,
221,221,221,221,216,221,128,221,216,38
14
5240 DATA 13,216,0,0,221,216,0,0,0,0,2
04,201,0,0,0,0,2742
5250 DATA 204,201,0,0,0,0,204,201,0,0,
0,0,204,201,0,0,4358
5260 DATA 0,0,187,186,0,0,0,0,187,186,
0,0,0,0,187,186,5889
5270 DATA 0,0,0,0,187,186,0,0,0,0,170,
171,0,0,0,0,1243
5280 DATA 170,171,0,0,0,0,170,171,0,0,
0,0,170,171,0,0,2954
5290 DATA 0,0,153,156,0,0,0,0,153,156,
0,0,0,0,153,156,4101
5300 DATA 0,0,0,0,153,156,0,0,0,0,136,
141,0,0,0,0,189
5310 DATA 136,141,0,0,0,0,136,141,0,0,
0,0,136,141,0,0,1550
5320 DATA 0,0,119,126,0,0,0,0,119,126,
0,0,0,0,119,126,2313
5330 DATA 0,0,0,0,119,126,0,0,0,0,102,
111,0,0,0,0,9135
5340 DATA 102,111,0,0,0,0,102,111,0,0,
0,0,102,111,0,0,146
5350 DATA 15,255,255,255,255,96,15,255
,255,255,255,96,255,255,255,255,5248
5360 DATA 255,246,255,255,255,255,255,
246,238,238,238,238,238,231,238,238,81
52
5370 DATA 238,238,238,231,238,238,238,
238,238,231,238,238,238,238,238,231,75
28
5380 DATA 221,221,221,221,221,216,221,
221,221,221,221,216,221,221,128,13,623
5390 DATA 221,216,221,221,128,13,221,2
16,204,204,144,12,204,201,204,204,8977
5400 DATA 144,12,204,201,204,204,144,1
2,204,201,204,204,144,12,204,201,7186
5410 DATA 187,187,160,11,187,186,187,1
87,160,11,187,186,187,187,160,11,4815
5420 DATA 187,186,187,187,160,11,187,1
86,170,170,170,170,170,171,170,170,796
5
5430 DATA 170,170,170,171,170,170,170,
170,170,171,170,170,170,170,170,171,85
80
5440 DATA 153,153,192,9,153,156,153,15
3,192,9,153,156,153,153,192,9,3035
5450 DATA 153,156,153,153,192,9,153,15
6,136,136,208,8,136,141,136,136,3245
5460 DATA 208,8,136,141,136,136,208,8,
136,141,136,136,208,8,136,141,2546
5470 DATA 119,119,224,7,119,126,119,11
9,224,7,119,126,119,119,224,7,1255
5480 DATA 119,126,119,119,224,7,119,12
6,102,102,240,6,102,111,102,102,379
5490 DATA 240,6,102,111,102,102,240,6,
102,111,102,102,240,6,102,111,226
5500 DATA 255,255,255,255,255,96,255,2
55,255,255,255,96,255,255,255,255,7318
5510 DATA 255,246,255,255,255,255,255,
246,238,238,238,238,238,231,238,238,83
02
5520 DATA 238,238,238,231,238,238,238,
238,238,231,238,238,238,238,238,231,76
78
5530 DATA 221,221,221,221,221,216,221,
221,221,221,221,216,221,221,128,13,773
5540 DATA 221,216,221,221,128,13,221,2
16,204,204,144,12,204,201,204,204,9127
5550 DATA 144,12,204,201,204,204,144,1
2,204,201,204,204,144,12,204,201,7336
5560 DATA 187,187,160,11,187,186,187,1
87,160,11,187,186,187,187,160,11,4965
5570 DATA 187,186,187,187,160,11,187,1
86,170,170,170,170,170,176,170,170,818
5
5580 DATA 170,170,171,0,170,170,170,17
0,171,0,170,170,170,170,170,176,6428
5590 DATA 153,153,192,9,153,156,153,15
3,192,9,153,156,153,153,192,9,3185
5600 DATA 153,156,153,153,192,9,153,15
6,136,136,208,8,136,141,136,136,3395
5610 DATA 208,8,136,141,136,136,208,8,
136,141,136,136,208,8,136,141,2696
5620 DATA 119,119,224,7,119,126,119,11
9,224,7,119,126,119,119,224,7,1405
5630 DATA 119,126,119,119,224,7,119,12
6,102,102,240,6,102,111,102,102,529
5640 DATA 240,6,102,111,102,102,240,6,
102,111,102,102,240,6,102,111,376
5650 DATA 8,136,136,136,136,136,153,15
3,153,153,153,153,170,170,170,170,6959
5660 DATA 170,170,187,187,187,187,187,
187,192,12,192,0,192,12,208,208,5413
5670 DATA 221,13,208,208,224,224,238,1
4,224,224,240,15,255,15,240,240,9656
5680 DATA 240,240,255,15,240,240,224,2
24,236,14,224,224,208,208,221,13,9798
5690 DATA 208,208,192,192,192,0,192,12
,187,187,187,187,187,187,170,170,8231
5700 DATA 170,170,170,170,153,153,153,
153,153,153,8,136,136,136,136,136,3893
5710 DATA 136,136,136,136,0,0,153,153,
153,153,144,0,170,170,170,170,3716
5720 DATA 160,0,187,187,187,187,176,0,
192,0,192,12,192,0,208,221,3614
5730 DATA 208,208,208,0,224,238,224,22
4,224,0,240,15,240,15,240,0,4652
5740 DATA 240,255,240,240,240,0,224,23
8,224,224,224,0,208,221,208,208,1808
5750 DATA 208,0,192,0,192,192,192,0,18
7,187,187,187,176,0,170,170,5402
5760 DATA 170,170,160,0,153,153,153,15
3,144,0,136,136,136,136,0,0,8824
5770 DATA 226,2,227,2,253,31,0,0,0,0,0
,0,0,0,0,0,8140
```

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

```
THE A.N.A.L.O.G. #69 DISKETTE CONTAINS 20
MAGAZINE FILES.  THEY ARE LISTED BELOW.

SIDE 1:

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

TRIAL    .OBJ  ML       (#3)   TRIAL BY FIRE
VNTEDIT  .BAS  BASIC    LOAD   VNT EDITOR
COLORSET.BAS   BASIC    LOAD   COLOR SET
STARRIDE.OBJ   ML       (#3)   STAR RIDER
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
MLEDITOR.BAS   BASIC    LOAD   M/L EDITOR
EDITORII.LST   BASIC    ENTER  BASIC EDITOR II

SIDE 2:

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

TRIAL    .ACT  ACTION!  (#1)   TRIAL BY FIRE SOURCE
PROG     .SRC  ASSEMBLY        STAR RIDER SOURCE, P1
PROG2    .SRC  ASSEMBLY        STAR RIDER SOURCE, P2
CHSET    .SRC  ASSEMBLY        STAR RIDER SOURCE, P3
TERRAIN  .SRC  ASSEMBLY        STAR RIDER SOURCE, P4
LETTERS  .SRC  ASSEMBLY        STAR RIDER SOURCE, P5
BOOTCAMP.BAS   BASIC    LOAD   BOOT CAMP, LISTING 1
PARSER   .OBJ                  BOOT CAMP DATA FILE
VOCAB    .DAT                  BOOT CAMP DATA FILE

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.
```

When you want to talk Atari

XL/XE HARDWARE

CMO SPECIAL

Atari 800XL **$69.99**

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	28.99
US Doubler	28.99

MODEMS

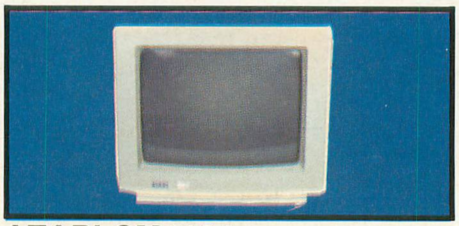
Atari	
SX212 300/1200 (ST)	79.99
XMM301	46.99
Anchor	
VM520 300/1200 ST Dir. Con	109.00
Avatex	
1200 HC	89.99
2400	169.00
Supra	
2400 Baud XL/XE or ST	159.00
2400 Baud (no software)	139.00

MONITORS

Magnavox	
CM 8505 14" Composite/RGB/TTL	189.00

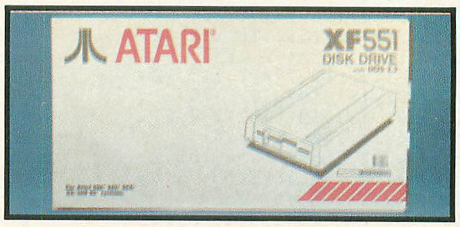
ST HARDWARE

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)	179.00
Supra	
FD-10 10MB Removable Floppy	859.00
30 Meg Hard Drive (ST)	649.00

CALL FOR DISKETTE SPECIAL

PRINTERS

Atari 1020 Printer 40 Col. Color **$19.99**

Atari	
1027 Letter Quality XL/XE	69.99
XDM121 LQ (XL/XE)	189.00
XM-M801 XL/XE Dot Matrix	189.00
Brother	
M-1109 100 cps Dot Matrix	169.00
M-1509 180 cps Dot Matrix	359.00
HR-20 22 cps Daisywheel	349.00
Citizen	
120D 120 cps Dot Matrix	159.00
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

CMO

Your Source for Hardware, Software & Peripherals

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

SPECIALS XL/XE

#AAB822P 822 Printer Paper	$2.49
#AA14746 T.V. Switch Box	2.49
#AA4010 Tic-Tac-Toe	4.99
#AA4011 Star Raiders	3.99
#AA4012 Missile Command	3.99
#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	8.99
#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)
$2.99 ea or 5 for $13.99

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

Rocklyn	
Gorf	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	
Fight Night	19.99
Atari	
Atariwriter Plus	35.99
Broderbund	
Printshop	26.99
Datasoft	
Alternate Reality (City)	23.99
221 Baker St.	20.99
Electronic Arts	
Pinball Construction	15.99
Firebird	
Silicon Dreams	15.99
Jewels of Darkness	15.99
Microprose	
Top Gunner	14.99
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	
Hardball	21.99
Activision	
Hacker II/Music Studio (ea.)	28.99
Antic	
Flash	21.99
Avant Garde	
PC Ditto	59.99
Batteries Included	
Degas Elite	37.99

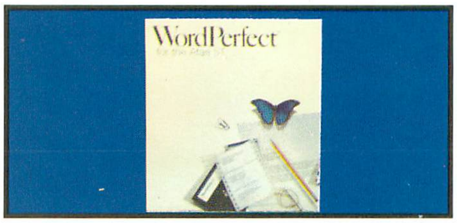
Cygnus Starfleet I **$32.99**

ST SOFTWARE

Data East	
Speed Buggy	25.99
Epyx	
Auto Duel	31.99
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	
High Roller	31.99
Mark Williams	
C	119.00
Paradox	
Wanderer (3D)	24.99

ST SOFTWARE

Progressive Computer	
Graphic Artist 1.5	109.00
Psygnosis	
Barbarian	25.99
Soft Logik Corp.	
Publishing Partner	54.99
Strategic Simulation	
Questron II	35.99
Timeworks	
Swiftcalc/Wordwriter (ea.)	43.99
Desktop Publisher	79.99
Unison World	
Printmaster Plus	25.99



Word Perfect **$159**

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 #114 ON READER SERVICE CARD.

B702

INSIDE THIS ISSUE:

MORE

MASTER MEMORY MAP

PLUS

BOOT CAMP

DATABASE DELPHI