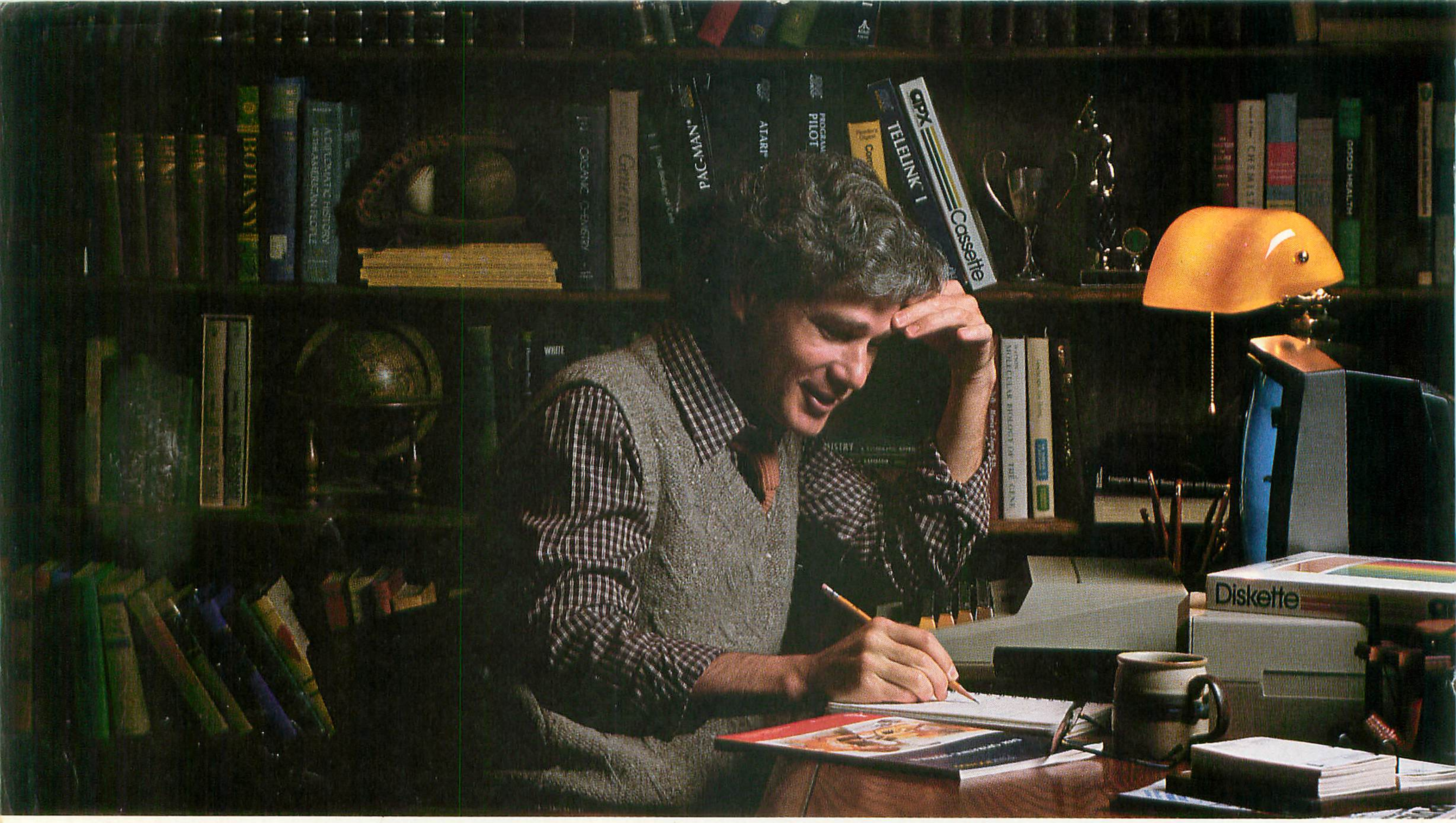
Antic™

The ATARI Resource

BONUS GAME

SOUND & MUSIC

October/November 1982
Vol. 1, No. 4
$2.50



MORE THAN EVER, ATARI HOME COMPUTERS ARE SPEAKING YOUR LANGUAGE.

With more program languages than ever to choose from, you now have more opportunities than ever to utilize the amazing capabilities of the ATARI 800™ Home Computer.

Whether you're a beginning programmer, or at the forefront of the art, you'll find an ATARI programming language that can make your task quicker and simpler than ever before. Just look at what ATARI has to offer:

ATARI Microsoft BASIC—Now we offer the industry standard, the most powerful Microsoft BASIC yet. With simple commands, it allows you to take advantage of unique ATARI hardware features such as our well-known player/missile graphics. For ease of programming, it includes n-dimensional arrays for numerics and string handling. And importantly, conversion procedures are simple.

ATARI Macro Assembler—Faster and more powerful than any ATARI language before, the ATARI Macro Assembler also allows you to access more memory space. And it's excellent for I/O interface and manipulation of such features as: player/missile graphics, sound registers and peripherals. In addition, the macro processor and "include" file library features speed-up program development considerably.

Fig-FORTH*—For specialized programming needs, such as educational or game applications, ATARI Fig-FORTH is uniquely effective. Fig-FORTH combines power and simplicity in an efficient 10K size, with characteristics of an interpreter and the speed of machine language code.

ATARI BASIC—An affordable and easy to use BASIC that requires only 8K of memory. It allows you to take advantage of the spectacular ATARI graphics and sound capabilities. And its immediate mode error messages greatly simplify debugging.

ATARI Assembler Editor—An excellent tool to assist the assembler-programmer in creating, editing and debugging assembly programs.

PILOT—ATARI PILOT is an exceptional learning language, with built-in "turtle" graphics to let you create spectacular designs and pictures with very short programs. Simple one or two-letter commands allow you to create a dialogue with the computer. And a single "match" command can perform complex text evaluation and pattern-matching instantly.

ATARI Pascal*—An excellent high-level language for teaching structured programming, and for developing and maintaining programs. In addition to offering all the features of the ISO Pascal standard, ATARI Pascal offers unique extensions that allow you to take advantage of ATARI graphics and sound capabilities.

ATARI is constantly developing new ways to help you get more out of your ATARI 800 Home Computer. So watch for more innovative and exciting programming languages from ATARI in the future.

For more information, write to ATARI, Inc., Dept. C27Z, P.O. Box 16525, Denver, CO 80216.

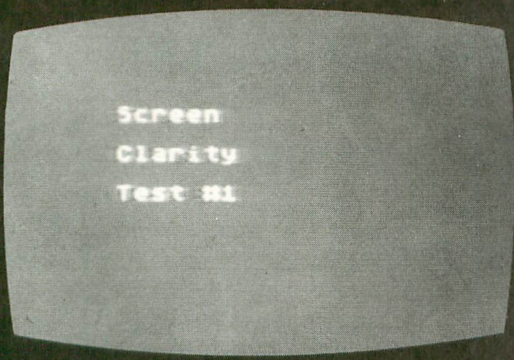
ATARI®
A Warner Communications Company

*Available from the ATARI Program Exchange

ATARI HOME COMPUTERS
We've Brought The Computer Age Home.™

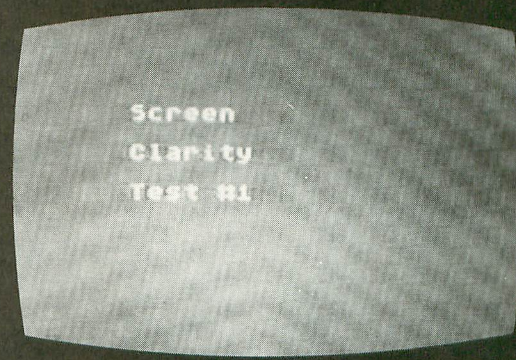
THE MOSAIC RAM SYSTEMS FOR ATARI*

CLEARLY THE BEST



MOSAIC 32K RAM

THE SCREEN CLARITY TEST



THE COMPETITION

YOU CAN SEE THE MOSAIC DIFFERENCE

WHAT THE EXPERTS HAD TO SAY

A.N.A.L.O.G. 400/800 MAGAZINE said in a 32K RAM board comparison test: "The Mosaic 32K RAM showed no sign of interference and gave the best screen clarity" and "Mosaic uses what we feel are the highest quality components which may improve reliability".

INTERFACE AGE said after replacing the Atari memory board with a Mosaic 32K RAM: "Once in place there is no noticeable change in screen clarity" and "in view of its excellent performance it should be a serious choice for those Atari owners intending to expand their memory."

Each Mosaic RAM board gives you more than just the best screen clarity but also the best in reliability, flexibility, and compatibility. The Mosaic RAM systems offer you the best in every way — these features prove it.

- Works in both the Atari 400* & 800* computers.
- 4 year guarantee.
- Complete instructions.
- Test cycled 24 hours for reliability.
- Gold edge connectors for better reliability.
- Quick no-solder installation.
- Full flexible memory configuration.
- Can be used with 8K, 16K and future products.
- Designed to take advantage of Atari 800's superior bus structure.
- Designed for inter-board communication in Atari 800.
- Allows Atari 800 to have 2 slots open for future expansion.
- Always the best components used for superior screen clarity and reliability.
- Low power design for safety and reliability.
- Available companion board ($5) to allow running The Mosaic RAM systems independent of other boards.

THE MOSAIC RAM SYSTEMS FOR THE SERIOUS ATARI OWNER.

THE MOSAIC 32K RAM

For the serious Atari owner. This is the 32K RAM board you've been hearing about. It has every feature you could want from a RAM board and more features than any other board offers. Each Mosaic 32K RAM comes with complete instructions so in a few minutes you will have expanded your Atari 800 to 48K RAM. The Mosaic 32K RAM works as well in the Atari 400, but we suggest the NEW Mosaic Expander.

THE MOSAIC EXPANDER

This is the most effective way you can expand to 32K RAM for your Atari 400 computer. And at almost the 16K price! The revolutionary Mosaic Expander is a memory board with 16K RAM in place and open slots to add 16K more from the Atari 16K board that comes with your Atari 400. Each board comes with complete instructions so in a few minutes you will have expanded to 32K RAM.

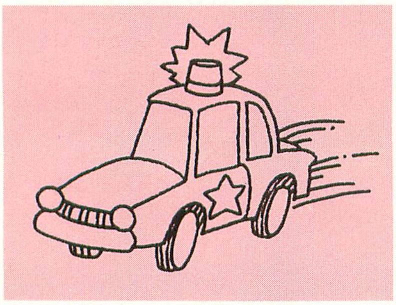
For your nearest Mosaic dealer call: toll free 1-800-547-2807.

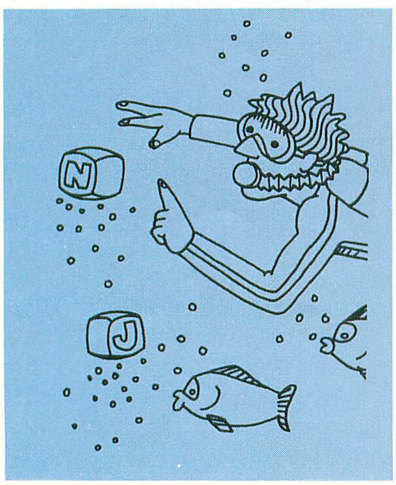
*Trademark of Atari, Inc.

MOSAIC
ELECTRONICS

P.O. Box 708 Oregon City, Oregon 97045
503/655-9574, 1-800 547-2807

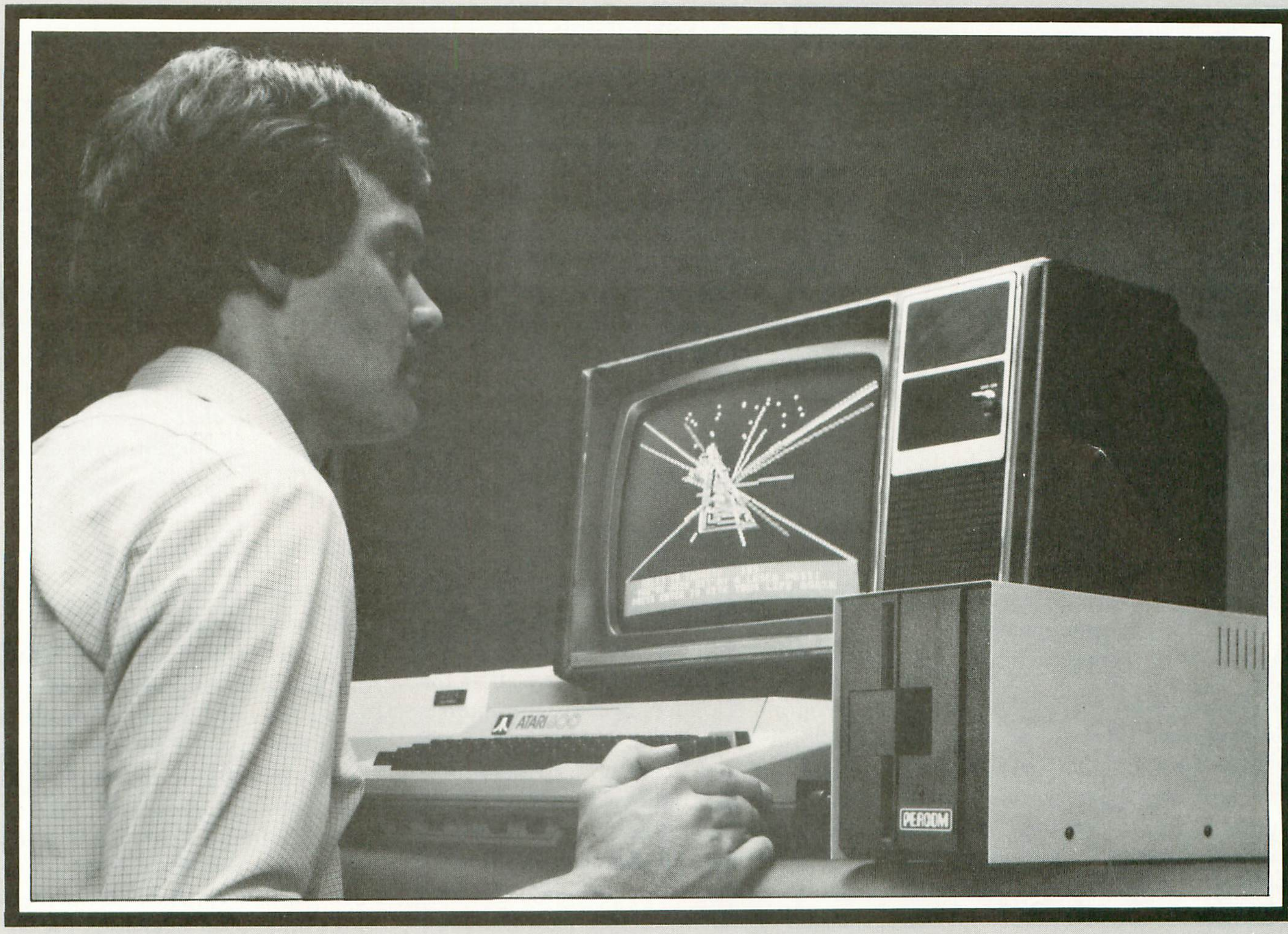
Antic™
The ATARI® Resource OCT/NOV 1982 Volume 1, Number 4





FEATURES

DEPARTMENTS



Introducing the PERCOM Alternative to ATARI Disk Storage

Your Atari 800 is the finest home computer on the market. Now you can own a floppy disk system that measures up — an RFD mini-disk storage system from Percom.

At Percom we've been making disk storage systems since 1977.

Our designs are proven, our quality is well known. And we back our dealers with service know-how. Expect more from Percom. You won't be disappointed.

- Operate in either single- or double-density storage mode using Atari DOS 2.OS. In double-density you can store almost 184 Kbytes (formatted) on one side of a 40-track diskette.
- Connect your Percom RFD first-drive system directly to your computer or connect into your system through your Atari 810 Disk Drive.
- Add an RFD first-drive system with its versatile four-drive controller, then connect as many as three more low-cost RFD add-on drives.
- Write application programs that can query and set up your system to operate a different type drive at each cable position — that can even change configuration as the program executes.
- Get quality and state-of-the-art capability at competitive prices. Percom first-drive RFD systems are priced from only $799, first add-on drive is only $459. Cables included.

Watch for announcement of a new, powerful, easy-to-use disk-operating system for your Percom-equipped Atari 800 computer.

Minimum system requirements — are an Atari 800 computer with 24-Kbytes of RAM and compatible video display system; Atari's disk-operating system (ver 2.OS) and owner's manual.

For the best thing next to your computer, see your Atari dealer about a Percom RFD floppy disk storage system. For the name of your nearest dealer, call Percom toll-free 1-800-527-1222.

PERCOM

PERCOM DATA COMPANY, INC.

11220 PAGEMILL RD. • DALLAS, TX 75243 • (214) 340-7081

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

Yes . . . I'd like to know more about Percom RFD disk drives for my Atari 800 Computer. Rush me free literature.

Send to: **PERCOM DATA COMPANY, Inc.,** Dept. 18-A01
11220 Pagemill Road, Dallas, Texas 75243

name ____________________

street ____________________

city ____________________ state ______

zip ____________ phone number ____________

I plan to add a hard disk system to my Atari: ☐ yes ☐ no.

MAIL TODAY!

TOP-OF-FORM

In the chart accompanying the survey of printers (ANTIC Issue #3) you show the Atari 825 as having "top of form" capability. I don't think it does. It doesn't respond to the standard "control +L" form feed, nor does it have a manual form feed control.

This deficiency would cause us to choose a different printer, perhaps the Epson or the NEC, which do look competent.

Harold Barton,
Engineer
Ford Aerospace & Communications
Newport Beach, CA

You are correct. There is no top-of-form command. We also should point out that the NEC 8203A-C will not work with the Atari Word Processor program without altering that program, a task not easily accomplished by an amateur. The version of DataSoft's Text Wizard that will work with the NEC should be available by October, 1982.—ANTIC ED

TRACE RACE

I want a "trace" routine for my ATARI. I understand that **Basic A+** has one, but right now that product is more than I need.

Would you put this in I/O Board to see if any of your readers can come up with one? I'd like to have features like "list branch commands only," and "show loops once, with times looped."

Walter Varner
Palo Alto, CA

Our first contest! Send all entries to TRACE RACE, care of ANTIC. Winner gets a free copy of **Basic A+**, *courtesy of* **Optimised Systems Software**. *All entries must be in no later than January 31, 1983.*—ANTIC ED

LET'S GET IT RIGHT

De Re ATARI is an excellent book, but the authors should have consulted a Latin teacher. "De re" means "of the thing," as in **De Re Metallica** by Agricola, which translates "of the metallic thing." "Re" is in the ablative case (singular, feminine). "Metallica" is an adjective, so ATARI should be adjectivized, as "De Re ATARICA" (feminine), or better, "De Rerum ATARII (masculine, plural), which translates as "of the things about ATARIes."

D. Grau
Forest Hills, N.Y.

MUSIC WHILE YOU WAIT

How can I record music, or other audio, on a cassette so that it plays when the cassette is loading? **PDI** does it, for example, on **Pumpkin Stand**.

I already know how to POKE 53018 on and off. This works great *after* the program is loaded, because the audio is recorded on a section of tape where no data is found. I am under the impression that the audio **with** data must be recorded after the data because a CSAVE erases all channels on the tape. I also believe that normally recorded audio signals interefere with the computer's ability to receive a CLOAD.

I notice that **PDI**'s music-while-loading plays on the left speaker of a stereo system. How do they record on only one channel?

Robert Cash
Pekin, IL

John Victor, President of PDI, explains in this issue. Also note our Tape Topics *section in this and future issues, for related information.*—ANTIC ED

LOOSENING LINES

While typing in your games, I've noticed several lines got so long they wouldn't fit. Line 672 of **Deathstar** has 119 characters, and line 745 has 117, exceeding the limit of ATARI's logical line. **Pac Invaders** also has that problem at lines 370, 390 and 430. It's not always clear where to break a line without affecting the program. Please advise.

Donald Harrier
Tallmadge, OH

Sorry about that. We will try to avoid tight lines in the future, but you can also correct by moving the margin two spaces left with a POKE 82,0 in immediate mode. Also, using any legal abbreviations (F. for FOR, N. for NEXT) and eliminating spaces will save space. BASIC will expand abbreviations for you when it interprets the line. In fact, that's usually why the listed line is longer than allowed.
—ANTIC ED

LICKIN' CHICKEN

Space Invaders invades my space.
I lose at **Asteroids** too.
I never get far in **Caverns of Mars**
And **Star Raiders** makes me blue.
I made the third screen of **Apple Panic**
When the "cores" finally did me in.
I guess I should give up computer games
And go back to games like Gin.
But I then found a game that I can win;
Yes, a game that I can beat!
I can get that **Chicken** across the road
And I don't even have to cheat.
So, I guess there's a game for each of us;
A game for our way and style.
You go ahead and smash the **Galaxians**.
I'll just dodge cars for a while.

Guy Hurt
Lansing, MI

MORE CHICKEN

I think I made an improvement to Stan Ocker's **Chicken** game (ANTIC #1). I found during play that if the chicken gets hit by the bottom part of a car, the chicken is moved up, but not clear of the car. So the chicken gets hit again by the same car.

I thought this was unfair and frustrating, so I altered the program as follows:

```
535 POKE 1560, PEEK(1560)-33:
A=USR(LD,0,PM):
IF DIF>1 THEN DIF=DIF-1
```

This change moves you up a full lane, clearing you of the car you hit.

Mike Colvin
Sacramento, CA

RESTON SOFTWARE PRESENTS

Capital Children's Museum

Expand the boundaries of your imagination and creativity with PAINT—your brush is the Atari's joystick and your canvas is the monitor. Book/software: $39.95

For the Atari 800/48K. Atari 400/16K version coming soon!

Reston Publishing Company, Inc. A Prentice-Hall Company 11480 Sunset Hills Road Reston, VA 22090 (800)-336-0338

ANTIC
Publishing

Editor & Publisher
James Capparell

Assistant Editor
Robert DeWitt

Contributing Editors
Jon Loveless
Ken Harms
Carl Evans

Technical Consultant
Jerry White

Art Director
Marni Tapscott

Production Assistants
Mona Borger
Linda Tapscott

Typesetting
Terrific Graphics

Circulation Manager
Les Torok

Software Librarian
Dave Duberman

Contributing Writers
Dave Plotkin
Gordon Smith
Linda Schreiber
Adrian Dery
Ed Rotberg
Herb Moore
Clyde Spencer
Pete Goodeve
John Victor

Contributing Illustrator
John Musgrove

NOTE: ATARI is a trademark of Atari, Inc. ATARI 400 and ATARI 800 are trademarks of Atari, Inc. All references should be so noted.

October / November, 1982, Volume 1, Number 4
ANTIC— *The ATARI Resource is published six times per year by* **ANTIC** *Publishing. Editorial offices are located at 297 Missouri Street, San Francisco, CA 94107. Phone is (415) 864-0886. See page 78 for subscription information.*

Central European Distributor:
Ing. W. Hofacker GMBH
Tegernseerstrasse 18
D-8150 Holzkirchen
08024 7331

Australian Distributor:
Futuretronics
79-81 Levanswell Rd.
Moorabbin, Vic. 3189
Australia
03 555 9544

ANTIC *The ATARI Resource (ISSN applied for) is published six times per year for $15.00 by* **ANTIC** *Publishing Co., 297 Missouri St., San Francisco, Ca. 94107. Second Class Postage pending at San Francisco, Ca. and additional mailing offices. POSTMASTER: Send address change to* **ANTIC** *297 Missouri St. San Francisco, Ca.*

EDITORIAL

With our fourth issue now history and the long wait for subscription fulfillment accomplished, I thought it would be appropriate to tell you something of our goals and editorial direction.

Since our first issue was released back in April, we've tripled the number of copies printed and doubled the number of pages. Obviously, ANTIC is rapidly establishing it's place at the side of your computers. Our paid circulation suggests we are the most widely read independent magazine for the ATARI owner, and we thank you for your confidence and support. We have had the opportunity to discuss ATARI with users world wide. Your dedication and enthusiasm reinforces the notion that we are active participants in a discovery that rivals the appearance of fire or the wheel.

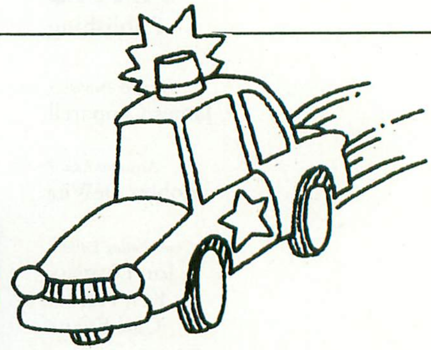
Active participants need information, accurate and up-to-date. Being long-time residents of the "silicon valley" area has provided us important professional and personal contacts which help bring you this information. Atari corporate headquarters is right down the road, and of course we try to bring you a look inside whenever possible.

Most of our writers are well-qualified old hands at the ATARI. Some are authors of significant pieces of commercial software, like Jerry White. Jerry will now be serving as our Technical Consultant, expect listings to improve and bugs to diminish. Carl Evans, president of Vervan Software, will be bringing you a new department, Tape Topics. You can expect more articles for the beginner. Watch for a new series by Bob Albrecht, starting next issue. Linda Schreiber will continue to provide excellent examples of teaching games. We also expect to print excerpts from her new graphics book published by Tab Books. I should also mention the excellent quality of work we are getting from our new users. For those of you who would like to write for ANTIC, but don't know where to begin, send for our new Author's Guide.

We continue looking for new ideas and encourage your participation in our magazine. I / O Board is very important in this regard. We appreciate your criticisms, and while your favorite ideas may not see print, I assure you we give serious consideration to all. Expect continued improvement in our design and art and perhaps some surprises here too. We encourage any artists or cartoonists to consider ANTIC.

The ATARI community is growing fast. ATARI is the number-one-selling home computer, and sales show no sign of slacking. I predict there will be half a million ATARI computers in homes by the end of 1983. I would like to think that each ATARI owner can benefit from ANTIC, we are The ATARI Resource. In fact my attitude is if you own an ATARI then you should be reading ANTIC. Spread the word — see you next issue.

Jim Capparell
Editor / Publisher
ANTIC — The ATARI Resource



a SOUND introduction

by JIM CAPPARELL

Many new users have not realized the tremendous potential for music and sound hidden in their ATARI computers. After all, a computer that can produce phaser noise or let you hear Indianapolis cars race down the straightaway, by altering a few simple commands, should be capable of more.

The following applies to both the 400 and 800 and is completely memory independent.

Sound on the ATARI is really made possible by the same technology that brought you hand-held calculators. I'm talking about the integrated circuit. In this case a special integrated circuit was designed and named POKEY (Pots and Keys). Every ATARI built has this special chip and therefore can play music and generate interesting sounds.

You might think of POKEY as a barber shop quartet, since there are four voices available. Each voice can be turned up loud or so low it can barely be heard. Each barber (voice) can "sing" or sound 255 different notes or pitches. Some of these are so similar your ear can't distinguish the differences. Among them are several that correspond to the musical scale (see Table 1). Each voice can be made to sound a pure tone — as if you were to whistle the note — or distort the tone. Distortion is one way of taking a familiar note and making it sound like a growl, hiss or rumble.

Let's put this in the context of the standard ATARI BASIC statements.

SOUND A,B,C,D is the general format to generate sound, where:

A = Voice, one of the four barbers. A can equal any value from 0 to 3.
B = Pitch or note. This can equal any number from 1 to 255. The higher the value the lower the note.
C = Distortion. Any even number from 0 to 14. Ten gives the purest tone with least distortion.
D = Volume. Any number from 1 to 15 is legal. A zero turns sound off.

That seems pretty easy, and so it is. Try this! SOUND 0,121,10,8 [press return]. This will cause the first barber (his number is zero) to sing middle C with as little distortion as possible. Now vary the volume; try a 4 and then a 14. Eight is a good volume value when more than one barber is singing. Experiment with Distortion; change the 10 to a 4, then a 14. Restore the sound statement as it is above. Now, add a second barber.

SOUND 1,72,10,8 This voice sings the note A above C.

SOUND 2,45,10,8 This voice sings the note F.

SOUND 3,193,10,8 This sings E below middle C.

TABLE 1

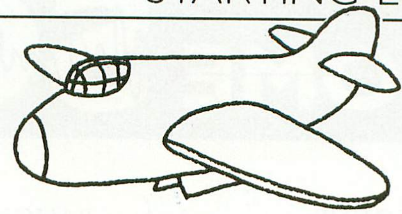
E	193	C	121 Middle C
F	182	D	108
G	162	E	96
A	144	F	91
B	128		

Turn off each barber's voice by making the corresponding volume 0.

To turn off all voices, type END.

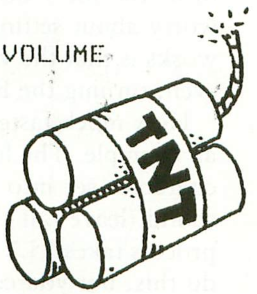
The legal abbreviation for the SOUND command is SO.; try it and save typing.

The following sounds should be experimented with. They are presented to get the wheels turning. I'm sure you can all do much better.



Our first sound is an explosion. Change the value of DUR in line 30. Experiment with volume changes in line 90.

```
10 REM EXPLOSION
20 REM DUR=LENGTH OF EFFECT,1-10
30 DUR=6
40 PITCH=20:GOSUB 80
50 SOUND 1,0,0,0:SOUND 2,0,0,0
60 GOTO 30
70 REM *** SUBROUTINE ***
80 SOUND 2,75,8,15
90 ICR=0.79+DUR/100
100 V1=15:V2=15:V3=15:REM VOLUME
110 SOUND 0,PITCH,8,V1
120 SOUND 2,PITCH+20,8,V2
130 SOUND 2,PITCH+50,8,V3
140 V1=V1*ICR
150 V2=V2*(ICR+0.05)
160 V3=V3*(ICR+0.08)
170 IF V3>1 THEN 110
180 SOUND 0,0,0,0:RETURN
```



Sound number two is a familiar siren. Change the DUR value in line 30. Try varying the step size in line 60.

```
10 REM SIREN
20 REM DUR=TIME IN SECONDS
30 DUR=10
40 LO=50:HI=35:STP=-1
50 FOR TIME=1 TO DUR
60 FOR PITCH=LO TO HI STEP STP
70 SOUND 0,PITCH,10,14
80 FOR WAIT=1 TO 15:NEXT WAIT
90 NEXT PITCH
100 XX=LO:LO=HI:HI=XX:STP=-STP
110 NEXT TIME
120 SOUND 0,0,0,0:GOTO 30
```

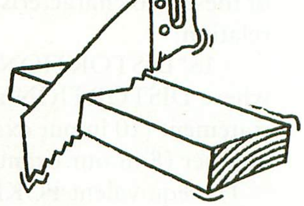
Sound number three is a European variation of the siren. Run it, you'll hear the difference. Experiment with the LO and HI values in line 40.

```
10 REM EUROPEAN SIREN
20 REM DUR=SECONDS RUN
30 DUR=5
40 LO=57:HI=45:PITCH=HI
50 FOR TIME=0 TO DUR*2
60 SOUND 0,PITCH,10,14
70 FOR WAIT=1 TO 180:NEXT WAIT
80 PITCH=LO:LO=HI:HI=PITCH
90 NEXT TIME
100 SOUND 0,0,0,0:GOTO 30
```

Sound four is the whistle and explosion of a falling bomb. Try to determine what makes the whistle sound and what part of the program makes the explosion sound.

```
10 REM WHISTLE & BOMB
20 REM DUR=LENGTH OF EFFECT
30 DUR=5
40 V1=4:FOR PITCH=30 TO 75
50 SOUND 0,PITCH,10,V1
60 SOUND 1,PITCH+3,10,V1*0.7
70 FOR WAIT=1 TO DUR*3:NEXT WAIT
80 V1=V1*1.03:NEXT PITCH
90 SOUND 2,35,8,12
100 V1=15:V2=15:V3=15
110 PITCH=DUR+5:ICR=0.79+DUR/100
120 SOUND 0,PITCH,8,V1
130 SOUND 1,PITCH+20,8,V2
140 SOUND 2,PITCH+50,8,V3
150 V1=V1*ICR
160 V2=V2*(ICR+0.05)
170 V3=V3*(ICR+0.08)
180 IF V3>1 THEN 120
190 SOUND 0,0,0,0:SOUND 1,0,0,0
200 SOUND 2,0,0,0:GOTO 30
```

Sawing wood is sound five. Try changing the pitch and volume. Also eliminate the wait in line 180.



```
20 REM SAWING WOOD
30 REM DUR=SECONDS RUN
40 DUR=8
50 FOR TIME=1 TO DUR
60 ST=6:VL=12:GOSUB 90
70 ST=8:VL=8:GOSUB 90
80 NEXT TIME:RETURN
90 FOR PITCH=ST+5 TO ST STEP -1
100 GOSUB 160:NEXT PITCH
110 FOR PITCH=ST TO ST+5
120 GOSUB 170:NEXT PITCH
130 SOUND 0,0,0,0:SOUND 1,0,0,0
140 FOR WAIT=1 TO 25:NEXT WAIT
150 GOTO 40
160 SOUND 0,PITCH,2,VL
170 SOUND 1,PITCH,8,VL*0.7
180 WAIT=(WAIT/5)*5:RETURN
```

There are many opportunities for the experimenter using the sound command. Perhaps a program using the joystick to vary pitch or distortion would make your experimentation easier. Random notes and harmonies can be very interesting. Look up and use the Random command in your BASIC Reference Manual. If you should write something interesting let us know, ANTIC is always looking for new, interesting and helpful material.

SOME SOUND ADVICE

by DAVE PLOTKIN

The SOUND statement in Atari BASIC is very powerful. Its ability to modify tone, distortion, and volume for each of four voices has been put to good use elsewhere in this issue. One of the problems with the SOUND statement is that using it extensively slows down program execution. While this is true of BASIC statements in general, with the SOUND statement there is an easy alternative — SOUND registers. SOUND registers are memory locations which control properties (tone, distortion and volume) of the ATARI's sound.

Memory Location	Function
53760	Tone of Voice 1 (SOUND 0)
53761	Distortion and Volume of Voice 1
53762	Tone of Voice 2 (SOUND 1)
53763	Distortion and Volume of Voice 2
53764	Tone of Voice 3 (SOUND 2)
53765	Distortion and Volume of Voice 3
53766	Tone of Voice 4 (SOUND 3)
53767	Distortion and Volume of Voice 4
53768	Tone "clock" control

The even-numbered memory locations (53760, 62, 64, 66) control the TONE; or which *note* the ATARI will play. This is identical to the second number in a SOUND statement. For example, to get the same tone as SOUND 0, 100, 10, 8 you would POKE 53760, 100. This specifies Voice 0, note 100. But what about distortion and volume? The odd-numbered memory locations (53761, 63, 65, 67) take care of these two characteristics for each voice via the following relation:

16*DISTORTION + VOLUME

where DISTORTION is the third number in the SOUND statement (10 in our example) and VOLUME is the fourth number (8 in our example).

The equivalent POKE in our example is 16* (10) + 8 = 168, and you would specify POKE 53761, 168. Try it. Type in: POKE 53760, 100:POKE 53761, 168 [RET]. The other pairs of registers work the same way. You can turn off the note by specifying zero in either TONE or DISTORTION-and-VOLUME registers.

Memory location 53768 is an interesting one. The ATARI maintains two internal "clocks" which it uses to measure the frequency of the sound wave it generates. The two clocks run at different speeds. Switching clocks changes the frequency (and thus the tone) of the sound. Bit 1 of memory location 53768 controls which "clock" the ATARI uses to produce its sound. Normally Bit 1 is off, and the ATARI's sounds correspond to the tables in the reference manual. Turning Bit 1 on (POKE 53768, 1) selects the slower clock, and alters the tone produced upward. Toggling Bit 1 off and on will switch all four voices up and down for a pretty good "alarm" effect. Note that the loop:

FOR N = 0 TO 255:POKE 53768, N: NEXT N

turns the Bit 1 off and on very nicely without having to worry about setting and resetting the bit. The reason this works is that the values jump back and forth from odd to even, turning the Bit 1 on and off.)

How much faster is POKE than SOUND? Well, let's try an example. The following program downloads the ROM character set into RAM so it can be modified. With no sound (leave out the SOUND statements in line 30), this process takes 15.7 seconds. There are much faster ways to do this, but you can use this method until you feel confident. Fifteen seconds is a long time to sit looking at a computer doing nothing visible. Most people start getting nervous and wondering if "Atari Lockup" has struck again. Let's add some sound to assure the user that something is happening.

```
10 POKE 106,PEEK(106)–4:POKE 53761,168:POKE
   53763,168:GRAPHICS 0
20 CHBASE = PEEK(106):OLDCH = 57344:NWCH =
   CHBASE*256
30 FOR X = 0 TO 1024:C = PEEK (OLDCH + X):POKE
   NWCH + X,C:SOUND 0,C,10,4:SOUND 1,X,10,4
40 NEXT X
```

Downloading the character set now takes some 25 seconds. If we try the following instead, substituting POKEs, the character set loads in about 20 seconds.

```
10 POKE 106,PEEK(106)–4:POKE 53761,168:POKE
   53763,168:GRAPHICS 0
20 CHBASE = PEEK(106):OLDCH = 57344:NWCH =
   CHBASE*256
30 FOR X = 0 to 1024:C = PEEK(OLDCH + X):POKE
   NWCH + X,C:POKE 53760,C:POKE 53762,140
40 NEXT X
```

Note that X, which varies from 0 to 1024, can be used as an input to the SOUND statement — each time it rolls over a multiple of 255, it starts over at 0 (thus 256 is 0, as is 513 and 769). This is *not* true of the POKE statement, so a constant was used. Doing a calculation to keep everything in range (such as POKE 53762, X/4) slows things down still further (about 28 seconds) and isn't a good idea.

Finally, various sources give the equations that relate tone to the internal clocks and note frequency. While these equations are beyond the scope of this article, they can be useful to music computer enthusiasts.

ATARI INSTITUTE TEACHES MUSIC

MUSIC MUSIC

by HERB MOORE

"We'll begin this morning's session with a fishing report. The lake has all kinds of fish in it. I did pretty well using artificial lures, but there were a couple of fellows using worms and they were doing even better."

In this relaxed fashion, Dr. Fred T. Hoffstetter, Director of the University of Delaware's Office of Computer-Based Instruction, addressed the students at the "Computer Applications to Music" seminar sponsored by the Atari Institute for Educational Action Research. He was telling them about another of the many activities available to them during their two-week stay this summer at Clear Water Estate in the Catskill Mountains of New York.

Fred finds himself in the unusual position, for a teacher, of trying to intice his students away from their class work for volley ball games, canoe rides, etc. But then, these were hardly typical students. Most of the participants in this workshop are teachers themselves. To be precise, they are mostly music educators at the high school and college level. Many of them were taking this class for credit through the University of Delaware, and most of them wanted to take as much advantage as possible of the equipment available to them in the lab.

Herb Moore is a musician, teacher, and co-author of "ATARI Sound and Graphics" (with Judy Lower and Bob Albrecht), published by John Wiley & Sons.

The lab provided an ATARI system for each student, consisting of an 800 computer, a Disk Drive, and a CRT. There were also several printers available so students could print listings of their programs. Or, if they wanted to use the University of Delaware's PLATO computer system, there was a terminal with which to do so. With this system, students are able to experiment with an ear training program called GUIDO which was developed by Dr. Hoffstetter. Also available in the lab, was a Synclaviar digital synthesizer with which students could create and store their own instrument sounds, and play them back using an organ style keyboard.

So there is ample inspiration for class projects. And these projects are what seem to be keeping the students from volley ball and fishing. A look at, and listen to, some of these projects makes it difficult to believe that, less than two weeks previous, most of these students had never used a computer, let alone programmed one to create sound and graphics.

One student program has the computer randomly generate a chord of up to four notes. The user then tries to identify the chord by entering it's name. If the answer is correct, another chord is generated. If the answer is incorrect, the player gets another try. If missed three times in a row the chord is played again and its name is displayed on the screen.

Another student program gives a screen display of two sine waves slightly out of phase with each other, and generates two slightly different tones to demonstrate the beat frequencies that occur in the sound. The user then tries to match the tones by moving a joystick. Yet another program gives a screen display of guitar chords with their correct fingerings.

One of the few nonprofessionals in the class happened to be a high school student from Old Greenwich, Connecticut. He designed a program that allows you to generate a note sequence which is continually played while you vary pitch, speed, and loudness with a joystick.

There are two or three programs being worked on by members of the class which are intended to help students become familiar with the positioning of notes on the staff. There were also projects showing where the notes are on the piano keyboard, and a colorful display of *Solfegio*, the "Do, Re, Mi" method of teaching notes. Another unusual project showed different positions for dance choreography on the screen.

Although most of these projects are not the kind of refined products you would expect to buy in a computer store, they clearly demonstrate just how quickly a motivated person can learn to program the ATARI computer. The chief language used in the course was BASIC, but other languages, such as Atari PILOT and a version of LOGO were also demonstrated.

One interlude, which saved a few braincells by temporarily diverting students, was a day spent with Tim Gallwey, the author of *The Inner Game of Tennis*. Since the students were mostly music teachers, the group worked on applying the "inner game" techniques to the teaching of music. Many insights into one's own music were discovered.

This was the first event sponsored by the Atari Institute for Educational Action Research that focused completely on music. Judging from the reactions of the participants, the seminar was a success.

According to Ted M. Kahn, Executive Director of the Atari Institute, its goals are to support through grants of equipment and minor funding projects engaged in *action research*. Action research is defined as having an orientation towards:

- social action as integral part of the research project
- improvement in learning and teaching techniques
- significant impact on social equity
- the promotion of lifelong learning

Dr. Kahn feels that the outcome of projects supported by the Institute should help advance the state-of-the-art of educational technology.

The "Computer Applications to Music" course clearly revealed a number of unique ways in which the ATARI computer can be integrated into the music teaching process. Many of the participants in the class will undoubtedly continue to refine their projects, and all of them will surely play a more active part in the development of computer-based music teaching.

UPGRADES AVAILABLE

The ATARI Home Computer Division currently offers four upgrades on the Home Computer product line. Each of these upgrades is explained below.

GTIA Upgrade

The GTIA (General Television Interface Adaptor) chip offers enhanced graphics capabilities over the CTIA chip, which was the original graphics chip used in the 400 and 800 computers. The GTIA offers 12 different graphics modes, 16 colors, and 16 intensities. The GTIA is fully compatible with software written for the CTIA since it has a superset of the CTIA's capabilities. The CTIA was the standard chip with all 400 and 800 computers manufactured prior to November, 1981.

Atari's Regional Service Centers will perform this upgrade on out-of-warranty-units for $62.52 ($22.52 in parts, $40.00 labor).

810 ROM C

An 810 drive with ROM C and DOS 2.0 will offer improved performance in read and write operations between the 810 and the 400 or 800. ROM C causes diskettes to be formatted with an improved sector layout which is more efficient than that used by earlier 810 control ROM's.

ROM C is automatically included in drives manufactured after October, 1981. Atari's Regional Service Centers will perform the upgrade on out-of-warranty units for $53.56 ($8.56 in parts, $45.00 in labor). 810 units still in the 90-day warranty period will be upgraded free of charge, on request.

Operating System Version B

The OS ROM's in the 400 and 800 are available in a revision B form, which provide a higher level of system performance by improving the operating system peripheral I/O control routines. OS version B eliminates annoying pauses in disk and printer operations that sometimes occurred with OS version A. OS version B is compatible with DOS 2.

OS version B ROM's have been automatically included in 400 and 800 computers manufactured since November, 1981. Atari's Regional Service Centers will perform the upgrade on out-of-warranty units for $49.78 ($30.00 in labor, $19.78 in parts). 400's and 800's still in the warranty period will be upgraded free of charge.

810 Data Separator Board

The Data Separator Board for the 810 improves the drive's ability to distinguish between data pulses and clock pulses on the disk. This is necessary in part because of the variations in the characteristics of different diskettes. The data separator lowers the chance of a misread from the disk.

Data separator boards have been a standard part of all drives produced since October, 1981. Older drives that are out-of-warranty may be upgraded at an Atari Regional Service Center for a charge of $135.61 ($40.00 labor, $95.61 parts).

The HCD Factory Authorized Service Centers will also perform these upgrades on request, although their prices may differ from the RSC's. Any 810 still in the warranty period will be upgraded free of charge.

BENCHMARKING THE FASTCHIP

by CLYDE SPENCER

The FASTCHIP, from Newell Industries, replaces the existing math package chip in the Operating System (OS) card. This chip substantially decreases the execution time of arithmetic operations. To determine how much faster our ATARI is with the new chip installed we need to talk a little about benchmarks.

Benchmarks are simple programs that compare specific operations in a computer. They may be designed to test a particular ability, such as the time required to perform an addition or multiplication, or they may be designed to test all operations. Probably the most usable benchmarks are those which attempt to simulate average operating conditions.

There is no general agreement as to what constitutes average conditions, let alone what is a standard benchmark. Here are two benchmarks, with similar running times, that I have some familiarity with.

The first benchmark was originally suggested by Greg Dolkus, formerly of the Homebrew Computer Club. This short program exercises your Central Processing Unit (CPU) by calculating the first prime number larger than one million:

```
10 FOR I=1000001 TO 1000003 STEP 2
20 FOR D=3 to SQR (I) STEP 2
30 IF I/D=INT(I/D) THEN 60
40 NEXT D
50 PRINT I
60 NEXT I
```

The correct answer is 1000003; not all computers offer enough precision to give the correct answer. The ATARI does. Greg had reported the following times for various machines:

Apple II	5 seconds
Commodore Pet	6 seconds
OSI	6 seconds
TRS-80	12 seconds

A significant increase in speed can be accomplished with the ATARI by turning the ANTIC chip off. This can be done by POKEing a 0 in decimal location 559. The ANTIC chip can be reactivated with a 34 POKEd into 559. Because of the different speed options available in the ATARI, I will list several benchmark times.

Atari 400/800	(ANTIC ON, with original OS)	15 seconds
Atari 400/800	(ANTIC OFF, with original OS)	11 seconds
Atari 400/800	(ANTIC ON, with FASTCHIP)	9.5 seconds
Atari 400/800	(ANTIC OFF, with FASTCHIP)	6.5 seconds

BASIC A+ by Otpimized Systems Software appeared to run about 1/4 second faster in all cases. Microsfot BASIC was unavailable for comparison.

I recently received a different benchmark by R. Broucke, from the University of Texas at Austin. This program compares both the speed AND precision of most of the major microcomputers by computing the sum of a thousand squares:

```
50 S=0
100 X=0
200 FOR N=1 TO 1000
300 S=S+X*X
400 X=X+0.00123
500 NEXT N
600 PRINT S,X
```

Because of the repeated additions of floating point numbers of the same sign, this program does a good job of testing error propagation. The correct answers are:

S=503.543802149 X=1.23

Broucke states that those computers, such as the ATARI, that use a four byte mantissa for representing floating point numbers, can be expected to run about 25% slower than those that use a three byte mantissa, such as the IBM personal computer. The following abridged table lists some of the more popular computers along with times and answers:

Computer	Time	S	X
TRS color Computer	37	503.543832	1.2300004
Commodore Pet	30	503.543832	1.23000004
Commodore Vic-20	27	503.543832	1.23000004
Apple II	26	503.543832	1.23000004
TRS-80 model II	23	503.545	1.23
Apple III	20	503.545	1.23
Sinclair ZX-81	13.5	503.54383	1.23
Osborne I (MBASIC)	8	503.545	1.23
IBM Personal Computer	7.5	503.545	1.230001
Atari 400/800 (ANTIC ON)	15.5	503.543594	1.23
Atari 400/800 (ANTIC OFF)	10.5	503.543594	1.23
Atari 400/800 (ANTIC ON, with FASTCHIP)	11.5	503.543594	1.23
Atari 400/800 (ANTIC OFF, with FASTCHIP)	8	503.543595	1.23

As you see, I left the best till last. The ATARI tests were done with the 8K cartridge BASIC. BASIC A+ was about 1/4 second faster; Microsoft BASIC was unavailable for comparison. With the FASTCHIP installed and ANTIC turned off, the Atari is as fast as (and more precise than) the other two fastest machines.

Based on these particular benchmarks, one can expect a 23–41% increase in speed with FASTCHIP, the average being a little over 30%.

DATA PERFECT

FOR THE ATARI 400 AND 800 COMPUTERS

YOU MAKE THE COMPARISON

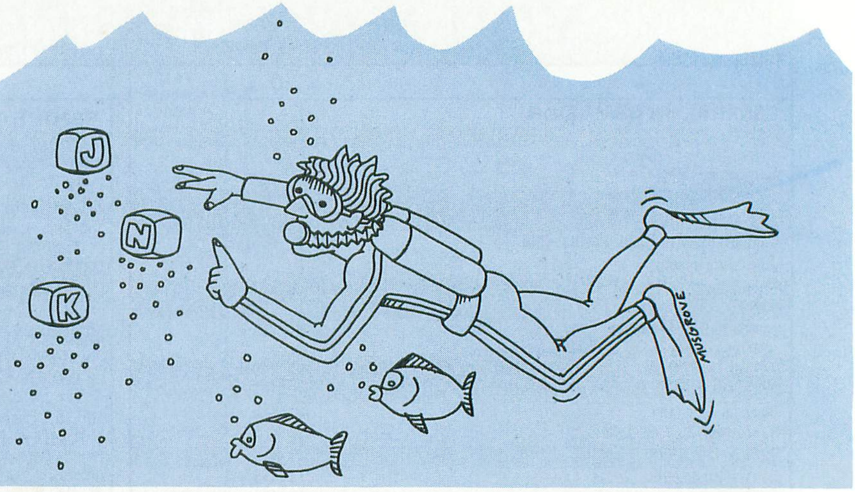
	D.P.	FILE MANAGER "800"
GENERAL INFORMATION		
Cost of Program	$99.95	
Cost of Utilities Program *(Included In Program)*	$00.00	
Cost of Reports Program *(Included In Program)*	$00.00	
Compatible With Letter Perfect (tm) Word Processing	**YES**	
Menu Driven *(Very User Friendly)*	YES	
Complete Documentation *(Manual Tabbed And Indexed)*	YES	
Single Load Program *(No Swapping Of Program Diskette)*	YES	
Machine Language *(Extremely Fast Operation)*	YES	
Can Use Single Disk Drive	YES	
Can Us Multiple Disk Drives	YES	
Ability To Design Screen Mask *(User Designs Arrangement Of Data)*	YES	
Full Keyboard Editing Available *(Delete/Insert A Character; Go To End/Beg. of Line; Fine 'n', TAB, ETC.)*	YES	
Compatible With Bit 3 80-Column Board *(40-Column and 80-Column Version Available)*	YES	
Works With Any Parallel Printer *(Supports Atari 850 Interface)*	YES	
Totals Of Numeric Field *(Return Total And Average Value/Field)*	YES	
Fail Safes Provided For Data Protection	YES	
Error Messages Displayed	YES	
Status Lines For Ease of Use *(Options Always Available For Reference)*	YES	
SEARCHES AND EDITING		
Multiple Searches Allowed On Same Record *(Search On 9 Criteria Per Record)*	YES	
Search On Two Criteria In Same Field *(Up To 4 Fields In Single Record)*	YES	
Wild Card Searches *(And/Or, Include, Character, Or Block)*	YES	
Search On Basis Of Record Number *(Search For An Individual Record)*	YES	
Search On Range Of Data Desired *(Dates, Numbers, Values, Greater Or Less Than, Equal To, etc.)*	YES	
Editing Of Records Individually	YES	
Editing Records Globally *(Verification Allowed)*	YES	
Delete Records Individually *(Verification Allowed)*	YES	
Deleting Records Globally *(Verification Allowed)*	YES	
UTILITIES SECTION		
Add Fields To Existing Data Base	YES	
Delete Fields From Existing Data Base	YES	
Reformat A Data Base *(Copy Format Of Existing Data Base)*	YES	
Make Additional Copies Of Data Base *(Create Data Base For Extended Records)*	YES	
Sort on Multiple Criteria *(Sort On Basis Of 4 Fields In A Sort)*	YES	
Sorts On Multiple Criteria *(Assending Or Descending)*	YES	
Depth Of Sort Can Be Changed *(Designate Number Of Charters Deep To Sort)*	YES	
Merge Information From Other Data Bases *(Merge Standard Text Files)*	YES	
Add Or Delete Fields From Data Base	YES	
Merge Previous Entered Data From Existing File	YES	
Back Up A Data Base *(Make A Back Up Of Current Source Data)*	YES	
Pack A Data Base *(Remove Deleted Records From Disk Storage)*	YES	

	D.P.	FILE MANAGER "800"
REPORT GENERATOR		
Design Report To User Specifications	YES	
Level Breaks Allowed At Users Option *(Up To 4 Level Breaks Per Report)*	YES	
Designate Font To Be Used In Report	YES	
Boldfacing Allowed In A Report *(With Dot Matrix Printer)*	YES	
Mathematical Formulas Allowed In Report *(Example, Field 'x' + Field 'y' = Field 'z')*	YES	
Auto Page Number Allowed In Report	YES	
Auto Date Entering Allowed In Report	YES	
Repeating Characters Allowed	YES	
Optional Level Breaks and Page Breaks When Sort Values Change	YES	
Up To 7 Lines Allowed For Header on Each Report	YES	
Up To 2 Lines Allowed For Detail Information On A Report	YES	
Variable Spacing Allowed Between Data On Items In A Report	YES	
Multiple Fields Allowed In A Report *(Number, Date, Alpha, Formula)*	YES	
Search Criterian Allowed On Report *(Same Criteria As In Editor)*	YES	
Ability To Have "Literal" Data *Printed In A Report*	YES	
Ability To Have "Conditional" Data Printed In A Report	YES	
Use A Default Date Field	YES	
Designate Default Value For Specific Fields	YES	
LABELS REPORT GENERATOR		
Mailing Labels Allowed *(Specifically Designed For Labels)*	YES	
User Designs Data Placement On Label *(One Across Label Design)*	YES	
Multiple Fields Allowed On Label *(Date, Alpha, Numeric, Formula)*	YES	
Repeating Characters Allowed	YES	
Front Designation Allowed	YES	
Print Labels On A Conditional Basis	YES	
Search Criteria Valid On Label *(Same Search Criteria As Editing)*	YES	
MATHEMATICAL ABILITIES		
Basic Math Calculation Addition, Substraction, Multiplication, Division	YES	
Built In Calculator (Automatic) *(Use In Editing, Or Adding Data)*	YES	
Find the Integer Value Of A Numeric Expression	YES	
Find The Log Base 'e' Of 'x'	YES	
Find The Log Base '10' Of 'x'	YES	
Find The Absolute Value Of 'n'	YES	
Exponentential Notation Used	YES	
Find The Square Root Of 'n'	YES	
Formulas Allowed Between Fields [Field × (+ − */) Field y = Field z] [Field × (+ − */) N = Field Y]	YES	
SPECIFICS		
Maximum Number Of Fields Per Record	32	
Maximum Number Of Formulas In A File	16	
Maximum Length Of A Field	127	
Maximum Record Length	511	
Maximum Number Of Level Breaks	4	
Records Per Diskette *(Depends On Length And Number Of Fields)*	VAR.	
Data Bases Allowed On Each Diskette *(Can Be Expanded To Additional Diskettes)*	ONE	
Form Letter Capability *(Compatible With Letter Perfect)*	YES	

LJK ENTERPRIES, INC.

P.O. BOX 10827
ST. LOUIS, MO. 63129
[314] 846-6124

Atari - A Trademark of Atari Inc., Full View 80 a Trade Mark of Bit 3 Computers, Fill Manager 800 A Trademark of Synapse Software, Letter Perfect, Data Perfect A Trademark of LJK Enterprises, Inc.



BUBBLE SORT

by ADRIAN DERY

This is a handy Sort Utility intended to be called from Basic and allows you to sort almost anything that can fit in your computer's memory. The flexibility of the Sort should cover many applications. Records may be any size up to 256 bytes. The sort fields may be any size up to the length of the record. You can sort on as many different fields as you need, and each field can be independently sorted in ascending or descending sequence.

The sorting technique is the traditional Bubble Sort which works by looking through a file of records in memory, and comparing the sort field of each record to the one following it. If any two adjacent records are not in sequence, the sort will exchange the positions of those two records. The sort continues to scan the file until there are no more records to exchange. In this way, records with the higher sort fields get pushed towards the end of the file, and records with the lower sort fields get pushed towards the beginning of the file. All of this takes place in memory so that it appears that the records BUBBLE into place.

The Sort only requires 182 bytes and the machine language is relocatable, therefore you can load and execute this sort anywhere in memory. Although you can put the Sort in any program you like, your file size is going to be limited by available memory. For large files, it is best to write a small Basic program that contains only this Sort, a String large enough to hold your file, and whatever Basic statements it takes to load a file, call the Sort and write out the new sequenced file.

Although the Sort works very fast, its speed can be improved by about 30% by turning Antic off. Just before calling the sort, save the value at PEEK(559) then POKE in a zero. All this does is shut down the screen display, but in so doing, it makes about 30% more CPU cycles available to the Sort. After the sort, POKE the saved valued back into 559 and the screen display will turn back on.

All sort parameters are passed to the Sort in the Basic USR call in the following sequence: 1. Address of the String containing the file. 2. Length of the records. 3. Number of records to be sorted. The next parameters specify the fields to be sorted by; 4.1. Position of the first byte of the field. 4.2. Length of the field. 4.3. '0' for ascending sequence, or '1' for descending sequence. Sort Fields are specified in Major to Minor order. That is, if you want to sort on State, and Zip Code within State, then State is the Major order and should be the first set of Sort Field parameters. The only limitation on the number of sort fields is the number of parameters that fit in the Basic statement calling the Sort.

The program in Listing 2 loads the machine language code for the Sort in Lines 1 to 9. The rest of the program demonstrates one of many techniques that can be used to read an unsequenced file, sort and rewrite a sequenced file. Type and run the program and, at the prompt, enter the first and last names of about 9 friends. The first names will be sorted ascending, the last names will be sorted descending and then displayed on the screen.

```
10 REM SORT UTILITY DEMONS
   TRATION
20 DATA 216,104,56,233,3,1
   33,217,104,133,204;104,
   133,203,104,133,215,104
   ,133,214,104,133,210,10
   4,133,209,162,0
30 DATA 104,104,157,0,1,23
   2,228,217,208,246,56,16
   5,209,233,2,133,209,165
   ,210,233,0,133,210,48,1
   08,165,209,133,211
40 DATA 165,210,133,212,16
   5,204,133,206,133,208,1
   65,203,133,205,24,101,2
   14,133,207,165,208,101,
   215,133,208,160
50 DATA 0,185,0,1,190,2,1,
   134,218,190,1,1,200,200
   ,200,132,216,168,136,17
   7,205,209,207,240,12,16
   5,218,208,4,144
60 DATA 16,176,46,144,44,1
   76,10,200,202,208,234,1
   64,216,196,217,208,210,
   198,211,169,255,197,211
   ,208,6,166,212,240
70 DATA 11,198,212,165,208
   ,133,206,165,207,24,144
   ,172,165,213,240,4,134,
   213,208,148,96,134,213,
   160,0,177,205,170
80 DATA 177,207,145,205,13
   8,145,207,200,196,214,2
   08,241,240,203
90 DIM SORT$(182):FOR I=1
   TO 182
```

```
100 READ A:SORT$(I,I)=CHR$(A):NEXT I
110 REM ------------------------------
120 REM INPUT A FILE TO BE SORTED
130 DIM FILE$(270),NAME$(15)
140 FILE$=" ":FILE$(270)=FILE$
150 FILE$(2)=FILE$
160 GRAPHICS 0
170 ? "ENTER THE NAMES OF 9 FRIENDS"
180 FOR I=0 TO 8:LE=I*30+1
190 ? I+1;" FIRST NAME ";:INPUT NAME$
200 FILE$(LE,LE+14)=NAME$
210 ? I+1;" LAST NAME ";:INPUT NAME$
220 FILE$(LE+15,LE+29)=NAME$
230 NEXT I
240 REM ------------------------------
250 REM PRINT UNSORTED FILE
260 GRAPHICS 0:? "UNSORTED NAME LIST"
270 FOR I=0 TO 8:LE=I*30+1
280 ? FILE$(LE,LE+29)
290 NEXT I
300 REM ------------------------------
310 REM SORT AND PRINT THE FILE
320 ANTIC=PEEK(559):POKE 559,0
330 X=USR(ADR(SORT$),ADR(FILE$),30,9,16,15,1,1,15,0)
340 POKE 559,ANTIC
350 ? :? "SORTED NAME LIST"
360 FOR I=0 TO 8:LE=I*30+1
370 ? FILE$(LE,LE+29)
380 NEXT I
390 END
```

[redacted]

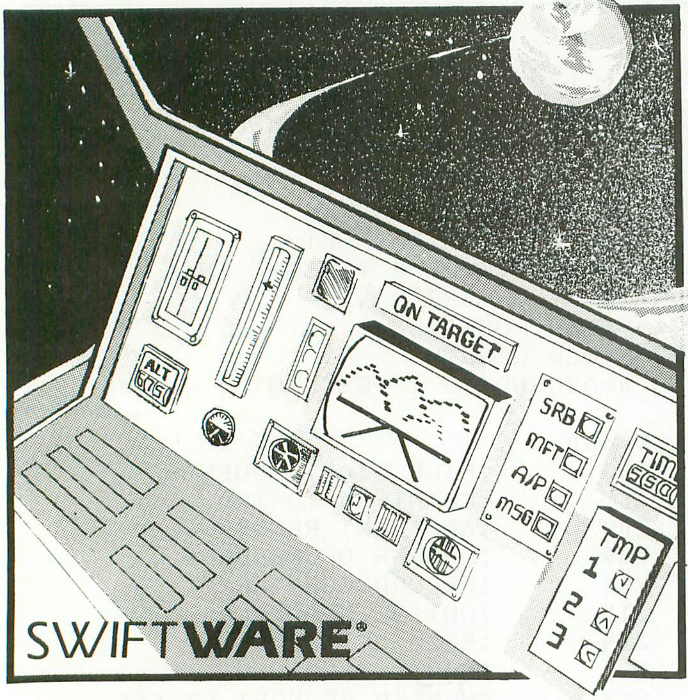
Variable checksum = 170377

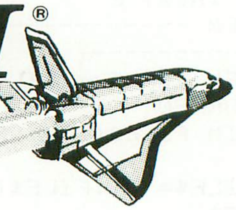
Line num range	Code	Length
10 - 60	SC	578
70 - 180	AM	463
190 - 300	FB	331
310 - 390	QT	269

```
0100 ;UTILITY SORT - CALLED FROM BASIC
0105 ;
0110 ;ENTRY PARAMETERS:
0115 ;
0120 ;   1. FILE ADDRESS
0125 ;   2. RECORD LENGTH <=256 BYTES
0130 ;   3. NUMBER OF RECORDS TO SORT
0135 ;   4. ANY NUMBER OF FIELDS TO SORT IN
0140 ;   MAJOR TO MINOR ORDER
0145 ;      4.1 FIELD POSITION
0150 ;      4.2 FIELD LENGTH
0155 ;      4.3 0=ASCENDING 1=DESCENDING
0160 ;
0165        ORG $0600
0170 FILE   =   203        ;FILE START ADDRESS
0175 PNTR1  =   205        ;POINTERS TO TWO
0180 PNTR2  =   207        ;ADJACENT RECORDS.
0185 RECNBR =   209        ;NUMBER OF RECORDS
0190 SCOUNT =   211        ;RECORD COUNTER
0195 BUBLE  =   213        ;OUT OF SEQUENCE
0200 RECSIZ =   214        ;SIZE OF RECORD
0205 FLDNDX =   216        ;SORT FIELD COUNTER
0210 FLDCNT =   217        ;NUMBER OF SORT FIELDS
0215 SORTAD =   218        ;ASCENDING/DESCENDING
0220 STACK  =   256        ;SAVE SORT FIELDS HERE
0225 ;
0230 ;DETERMINE HOW MANY FIELDS TO SORT
0235        CLD
0240        PLA             ;ALL BUT THE FIRST
0245        SEC             ;THREE PARAMETERS
0250        SBC #3          ;ARE FIELDS TO
0255        STA FLDCNT      ;SORT
0260 ;
0265 ;PICK UP SORT PARAMETERS
0270        PLA             ;FILE START
0275        STA FILE+1      ;ADDRESS
0280        PLA             ;
0285        STA FILE        ;
0290        PLA             ;RECORD LENGTH
0295        STA RECSIZ+1    ;
0300        PLA             ;
0305        STA RECSIZ      ;
0310        PLA             ;NUMBER OF RECORDS
0315        STA RECNBR+1    ;
0320        PLA             ;
0325        STA RECNBR      ;
0330 ;
0335 ;PICK UP FIELDS TO SORT
0340        LDX #0
0345 PICKFIELDS
0350        PLA             ;GET ALL THE SORT
0355        PLA             ;FIELD PARAMETERS FOR
0360        STA STACK,X     ;POSITION, LENGTH
0365        INX             ;AND DIRECTION.
0370        CPX FLDCNT      ;ANY MORE
0375        BNE PICKFIELDS  ;GO GET THEM
0380 ;
0385 ;SET UP NUMBER OF RECORDS TO SORT
```

listing continued on page 21

Fly the SPACE SHUTTLE from your ATARI®





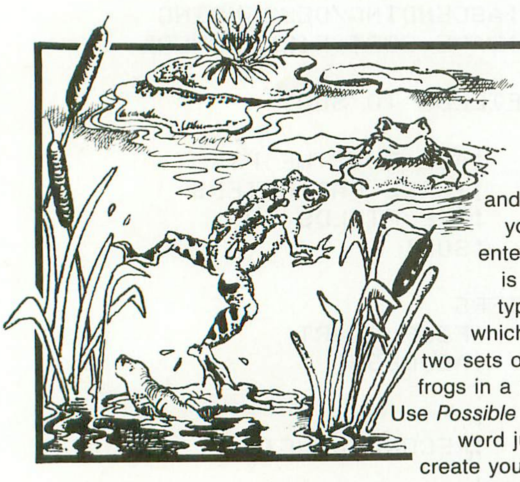
SPACE SHUTTLE

By Paul Kindl

Join the crew of the *Space Shuttle* as they prepare to take the next step into the world of space travel. Take control of the world's first reuseable spaceship, the *Space Shuttle,* and in an accurate full graphic simulation, place yourself in the cockpit. Pilot the Space Shuttle through take-off with booster stage separations, orbit, descent down the glide path and landing to touchdown – complete with a chase plane and scrolling runway visible through the cockpit windscreen. You assume command throughout all phases of the mission aided by complete instrumentation.

Price: $29.95 32K Diskette

ATARI® is a registered trademark of Atari Inc., a Warner Communications, Co.



FUN 'N' GAMES™

Leapfrog, Possible, and Wordgames giving you hours of fun and entertainment. *Leapfrog* is a Chinese-Checker type jumping game in which you try to position two sets of animated jumping frogs in a minimum of moves. Use *Possible* to help descramble word jumble puzzles or to create your own. *Wordgames,* two games in one, contains *Guessit* – a deductive alphabetic reasoning *game for one or two players and Wordjumble* - a multiple word descrambling puzzle with play-on-word hints and mystery answers. Disk version of *Guessit* works with a Votrax Type'N'Talk. A real crowd pleaser. Joystick and printer optional.

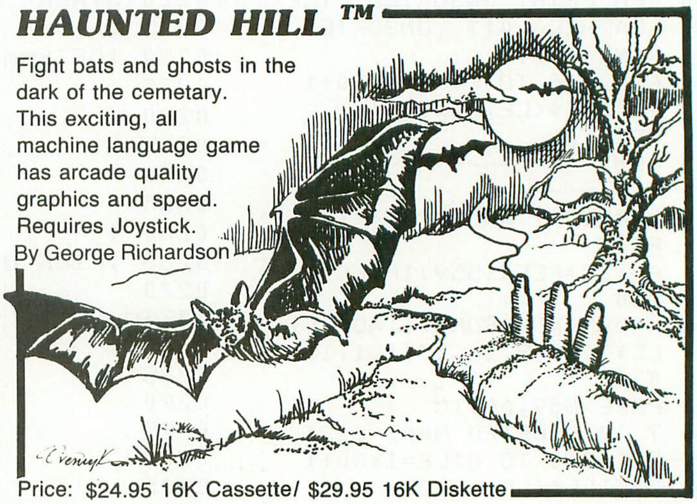
By Jerry White

Price: $17.95 16K Cassette/ $19.95 24K Diskette

HAUNTED HILL™

Fight bats and ghosts in the dark of the cemetary. This exciting, all machine language game has arcade quality graphics and speed. Requires Joystick.

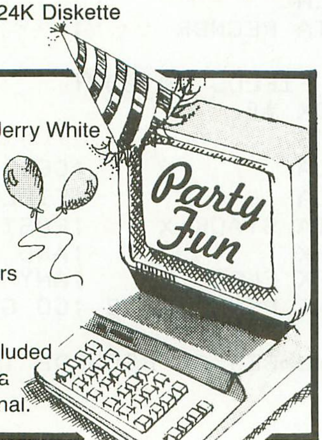
By George Richardson



Price: $24.95 16K Cassette/ $29.95 16K Diskette

TRIVIA TREK™

By Jerry White



Play it for fun, test your knowledge or entertain friends. Fun for the whole family and great at parties. A multiple choice trivia game for one or two players complete with 500 trivia questions in fifty categories and two thousand multiple choice answers. Included is a program for creating your own trivia questions and answers. Joystick optional.

Price: $29.95 32K Diskette

Plus many more

AVAILABLE AT SELECT COMPUTER STORES

MAIL ORDERS: Send check or money order plus $2.50 shipping and handling. N.Y. Residents add 7¼% sales tax.

TELEPHONE ORDERS: (516) 549-9141

Dealer Inquires Invited

Send for FREE catalog

MasterCard

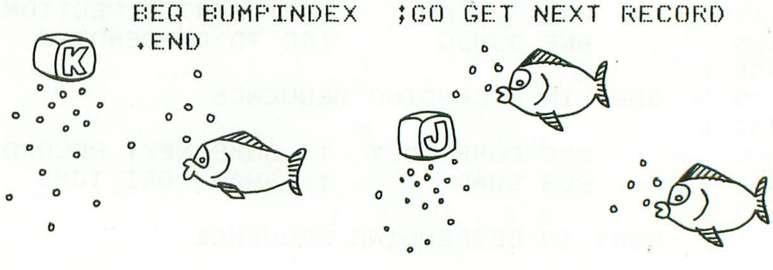
SWIFTY SOFTWARE, INC.
64 Broad Hollow Road
Melville, New York 11747

VISA

```
0390        SEC
0395        LDA RECNBR          ;MUST BE AT LEAST
0400        SBC #2              ;TWO RECORDS TO
0405        STA RECNBR          ;SORT
0410        LDA RECNBR+1        ;
0415        SBC #0              ;
0420        STA RECNBR+1        ;
0425        BMI ENDSORT         ;ELSE GET OUT
0430 ;
0435 ;MAIN LINE SORT LOOP
0440 ;
0445 SORT   LDA RECNBR          ;RESET NUMBER OF
0450        STA SCOUNT          ;RECORDS TO SORT
0455        LDA RECNBR+1        ;
0460        STA SCOUNT+1        ;
0465        LDA FILE+1          ;SET UP POINTERS
0470        STA PNTR1+1         ;FOR THE FIRST
0475        STA PNTR2+1         ;AND
0480        LDA FILE            ;SECOND RECORDS.
0485 BUMPRECORD
0490        STA PNTR1           ;PUT PNTR2
0495        CLC                 ;AHEAD
0500        ADC RECSIZ          ;OF
0505        STA PNTR2           ;PNTR1
0510        LDA PNTR2+1         ;BY
0515        ADC RECSIZ+1        ;ONE
0520        STA PNTR2+1         ;RECORD.
0525 ;
0530 ;SEQUENCE CHECK RECORDS
0535 ;
0540        LDY #0              ;RESET STACK INDEX
0545 NEXTFIELD
0550        LDA STACK,Y         ;FIELD POSITION.
0555        LDX STACK+2,Y       ;SORT DIRECTION
0560        STX SORTAD          ;SAVE IT.
0565        LDX STACK+1,Y       ;FIELD LENGTH.
0570        INY                 ;BUMP
0575        INY                 ;STACK
0580        INY                 ;INDEX
0585        STY FLDNDX          ;AND SAVE IT.
0590        TAY                 ;FIELD POSITION TO Y
0595        DEY                 ;MAKE RELATIVE TO ZERO
0600 SEQCHECK
0605        LDA (PNTR1),Y       ;COMPARE ADJACENT
0610        CMP (PNTR2),Y       ;RECORDS
0615        BEQ SEQNDX          ;= KEEP ON LOOKING
0620        LDA SORTAD          ;GET SORT DIRECTION
0625        BNE DSNDG           ;GO TO DESCENDING
0630 ;
0635 ;  SORT IN ASCENDING SEQUENCE
0640 ;
0645        BCC BUMPINDEX       ;< BUMP NEXT RECORD
0650        BCS SWAP            ;> SWAP POSITIONS
0655 ;
0660 ;  SORT IN DESCENDING SEQUENCE
```

listing continued on next page

```
0665 ;
0670 DSNDG  BCC SWAP         ;< SWAP POSITIONS
0675        BCS BUMPINDEX    ;> BUMP NEXT RECORD
0680 ;
0685 SEQNDX INY              ;CHECK THE LENGTH OF
0690        DEX              ;THE SORT FIELD AND
0695        BNE SEQCHECK     ;KEEP SEQUENCE CHECKING.
0700        LDY FLDNDX       ;ANY MORE FIELDS
0705        CPY FLDCNT       ;TO SORT
0710        BNE NEXTFIELD    ;YES, GO TO IT
0715 ;
0720 ;INDEX THROUGH THE SORT FILE
0725 ;
0730 BUMPINDEX
0735        DEC SCOUNT       ;COUNT DOWN RECORDS
0740        LDA #255         ;AND CHECK FOR
0745        CMP SCOUNT       ;END OF FILE.
0750        BNE NOTEOF       ;
0755        LDX SCOUNT+1     ;
0760        BEQ CKSWAP       ;
0765        DEC SCOUNT+1     ;
0770 NOTEOF LDA PNTR2+1      ;BUMP PNTR2 AND
0775        STA PNTR1+1      ;PNTR1 TO THE
0780        LDA PNTR2        ;NEXT RECORDS.
0785        CLC
0790        BCC BUMPRECORD
0795 ;
0800 ;AT END OF FILE SEE IF A SWAP WAS MADE
0805 ;
0810 CKSWAP LDA BUBLE        ;IF NO RECORDS SWAPPED
0815        BEQ ENDSORT      ;THEN IS END OF SORT,
0820        STX BUBLE        ;ELSE SEQUENCE CHECK
0825        BNE SORT         ;THE FILE AGAIN.
0830 ENDSORT
0835        RTS              ;BACK TO BASIC
0840 ;
0845 ;SWAP RECORDS IF OUT OF SEQUENCE
0850 ;
0855 SWAP   STX BUBLE        ;STILL OUT OF SEQUENCE
0860        LDY #0
0865 SWAPLOP
0870        LDA (PNTR1),Y    ;THIS ROUTINE
0875        TAX              ;EXCHANGES THE
0880        LDA (PNTR2),Y    ;POSITIONS OF TWO
0885        STA (PNTR1),Y    ;OUT OF SEQUENCE
0890        TXA              ;ADJACENT RECORDS
0895        STA (PNTR2),Y    ;
0900        INY
0905        CPY RECSIZ       ;KEEP LOOPING FOR
0910        BNE SWAPLOP      ;THE LENGTH OF RECORD.
0915        BEQ BUMPINDEX    ;GO GET NEXT RECORD
0920        .END
```



ON HAVING A GOOD TIME

by PETE GOODEVE

Most enlightened users know the ATARI computers have a crystal-controlled clock that reports to the CPU with an interrupt each time a vertical TV scan begins. The scan rate is a constant sixty times per second, so this looks like an ideal basis for an accurate time-of-day clock.

But anyone who has tried this will have stumbled across a few obstacles. For a start, crystal-controlled accuracy isn't that good. Run your program for an hour or so and you will find that it's about five seconds slow. Also, an accurate clock should keep running independently of other computer activities. It is not very helpful if, when the program stops, so does the clock. It would be even more disastrous if the clock module code was erased.

What is wrong with our "precise" 60 Hz scan rate? It is not, as you might think, simply sloppy engineering! In fact the frequency is a tightly maintained 59.92 Hz. The reason for this odd value goes back in electronic history.

A twisty tale of time

In the days before color TV, the vertical scan rate was indeed exactly 60 Hz. Color meant that a lot more information now had to be packed into the space originally allocated for black and white transmission, without upsetting older receivers tuned to the same signal. A color frequency with its own set of harmonics and sidebands would, given a chance, insinuate themselves onto the other signals, causing "herringbone" patterns in the picture. These effects were minimized by careful juggling of the various frequencies involved.

Unfortunately, the ATARI gets a little more complicated. Unlike a standard color TV transmission, where the color signal is carefully kept unrelated to the line frequency, the Atari *wants* an exact number of color clocks per line, so that it can generate colors digitally. The color clock itself must adhere to the standard pretty closely, because this is critical for proper color, but now all the other frequencies are divided down from this, ending up with that vertical scan frequency of 59.92 a second (compared with a broadcast TV rate of 59.94).

We have a simple cure for this tardiness. Every time we compute that our time-keeping has slipped one "tick" (i.e. one vertical scan period) out of step, just add an extra one into the count. The proper interval between corrections is almost thirteen seconds. With this correction we get well within the accuracy of the crystal, and should keep time within a couple of seconds per day.

Hanging on to the reins

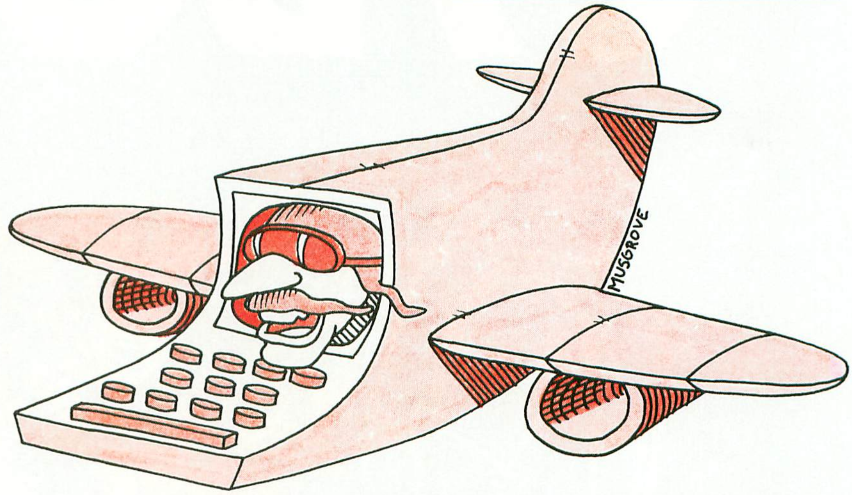
What about keeping the clock running and on time, independent of other activities? The main hangup is that most processing related to the vertical-scan interrupt is inhibited when urgent tasks — like servicing the disk — have to be done. Normally our clock must be treated in the same way. There *is* a critical path for the clock interrupt which is not blocked at these critical times, but if we did all our timekeeping there we would quickly run into serious trouble, by interfering with all our peripheral communications.

The solution is to split our processing, doing only the essential counting of seconds in the unblockable path, and all the rest in the non-critical way. If the main process finds that it has missed some seconds while it was blocked, it just does some extra cycles to catch up. (By the way, if you knowledgeable readers know that the ATARI operating system already has several clocks and timers based on the same interrupt, and are wondering why we don't just use one of them, the reason is that they are all either used for something else, or are cleared by "RESET"; so constructing one of our own is a necessity.)

The ability to postpone updating our clock, that we have gained with this split-processing approach, turns out to be useful in another way. The time that is so diligently kept by our clock has to be read at some point — usually by a BASIC program. The trouble is that more than one number is involved (hours, minutes and seconds). At the speed with which we can do things in BASIC there is a good possibility that by the time the last number is read, the first is no longer valid! To correct this problem we simply add a flag that, when set, freezes the clock. We set this before picking up the time values and

continued on page 65

THE *Musical* PILOT



by KEN HARMS

This article will open the door to string parsing, a powerful way to analyze PILOT strings. Along the way, we'll read and write on the disk/cassette, do some Boolean algebra, change data types and reveal a beautiful PILOT bug. And, oh yes, we'll play four-voice music.

As always, we'll be way "beyond the book." Since it will be getting pretty deep, I'll give page references to Atari's PILOT Primer.

A string is a combination of letters, numbers, symbols, words, etc., "strung together." In PILOT, a "string variable" is made by giving it a name (always beginning with "$") in an A:ccept or C:ompute instruction (pp. 69–76). The book tells how to concatenate ("grow") strings. We'll discuss how to parse ("cut") strings so you can analyze each part of a string. This could be useful for analyzing sentences, riddles, or in this case, for storing data for a program's use (PILOT lacks a "Data" statement).

String parsing relies on the Match String command which produces three pre-named variables, $Left, $Match, and $Right (pp. 41–44, 81–82). Parsing programs work as follows (refer to the Pilot Player listing):

1. Place the string into the "accept buffer" (line 1270).
2. Match on the "separator." In this case, I used the blank as a separator. In line 1280, we skip over the initial blank, which the A:ccept instruction inserts in each string, and M:atch on the second blank. (Note the right arrow in the instruction which doesn't print in front of the "_").
3. Check for the end of string (the JN: in line 1290).
4. Store the remainder of the string (found in $Right) in a safe place (line 1300).
5. Use $LEFT as the parsed word, letter, etc. (lines 1310–1370).
6. Jump back to step 1.

Although this may seem complicated, it's conceptually as easy as BASIC.

To play a C,D,E,F chord for a sixteenth, the Pilot Composer produces a string looking like this: " 1 3 5 6 16 ! ". The first four values are the usual notes (pp. 106–107) for each of the ATARI's four voices. The "16" is the inverse duration of the note (1/16 of a note). The "!" is a "terminator" to tell us that we're out of notes. Our problem: parse it and play it. The *Loop2 routine (lines 1250–1390) cuts the string and sets up variables for each voice and for the PAUSE command. After each Match String, the variables look this way (the underlines represent blanks):

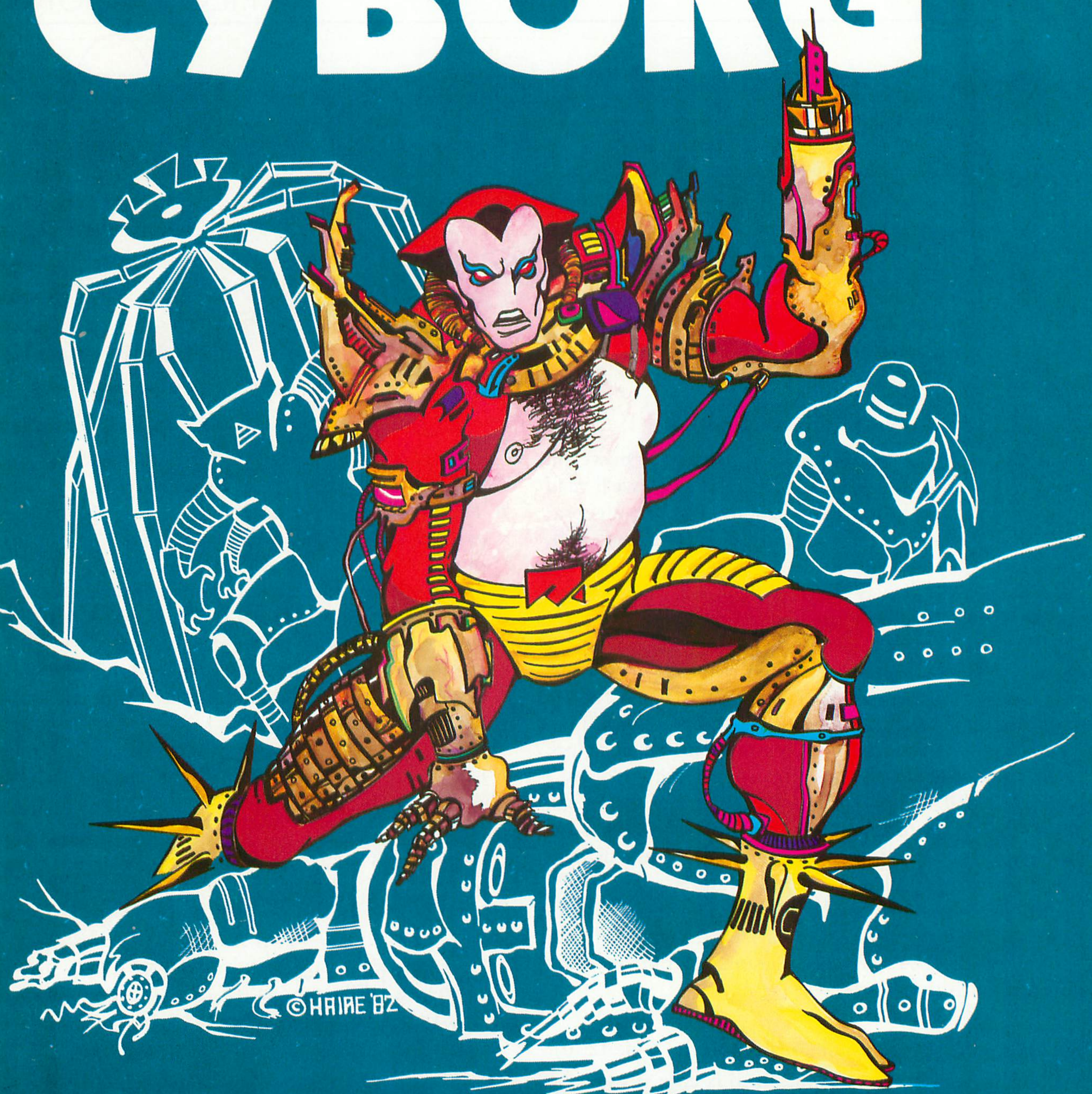
PASS	$PLAYVALUES BEFORE MATCH	$LEFT	$MATCH	$RIGHT	$LEFT USED FOR
0	_1_3_5_6_16_	NULL	NULL	NULL	
1	_1_3_5_6_16_	_1_	____	_3_5_6_16_	#A
2	_3_5_6_16_	_3_	____	_5_6_16_	#B
3	_5_6_16_	_5_	____	_6_16	#C
4	_6_16_	_6_	____	_16_	#D
5	_16_	_16_	____	____	#L
6	____	_16_	____	____	NO MATCH

Simply put, each value marches to the left into the $LEFT bucket and then gets used. Notice that the "no match" in pass six did not change any of the special string variables.

The Pilot Composer parses strings in a similar fashion but on each letter. In this case, the match parsing instruction (line 1200) skips two spaces (the leading blank and the first letter) and M:atches on the next character to put **all** remaining characters in $MATCH (the comma does that). Once the string is split, a simple $LEFT inspection finds the character and then restores the balance of the string. The *TRANSLATE module (lines 1400–1690) performs a similar M:atch to find good notes and durations in $GOODNOTES and $GOODDURATION, and then to translate them into note and duration values. The translation lookup in $NOTEABLE is "fail safe" — it first M:atches on the note followed by "/" and then M:atches on the subsequent ".". This forces the value (a 5, say) into $LEFT. This was required, since at M:atch for 1 or 8 without the "." would have found the value of notes C and

continued on page 27

CYBORG



Three hundred years in the future, the gaming grids are dominated by the human/machine hybrids known as Cyborgs. As the newest achievement of the cybernetic sciences, your final testing demands that you battle, and defeat, the most dangerous robotics in the universe. Humanoid robots, killer hyper-spheres and giant spider creatures test your relays. Up to forty enemies converge on you as you frantically fire your arm pistol. As you clear each arena, your opponents move faster, take more hits, and mutate into additional nightmarish creations.

Cyborg can be played with one or two joysticks. One joystick moves your Cyborg, the other fires your energy weapon. With one joystick, the joystick button fires your pistol. Cyborg has full color mode 7.5 graphics, incredible sound effects, one or two player options, and one or two joystick control. It saves high scores and pauses with the touch of a button. We guarantee that Cyborg is one of the fastest, most exciting games you'll ever play on your Atari, or return it within 14 days for a full refund.

CYBORG Atari 400/800 16K cassette. . . .$29.95 *CYBORG Atari 400/800 16K diskette. . . .$29.95*

Please add $2 00 for first class postage, $4.00 for overseas air mail.

Med Systems Software • PO Box 3558 • Chapel Hill, NC 27514

To order, call: 1-800-334-5470, or see your dealer

G. Of course, I could have designed the string in reverse order — that's an improvement for you.

Let's digress to the music before going on with the programming. The Pilot Music "System" now has two simple programs. Pilot Composer accepts four-note chords composed of the eight basic notes (no sharps or flats), followed by a duration (a whole note, half note, etc.). It checks these data, catches most errors, and rings a "bell" when it's ready for another chord. It won't find short chords, so make sure you enter four notes and a duration, or change the *TRANSLATE module between lines 1670 and 1680. Chords are written to the disk or cassette every 10 chords. This is required since the maximum length of an accept buffer is 254 characters.

The Pilot Player asks for a tempo (how fast to play) and a file of music. It then opens that file and plays the notes stored there.

Back to the Pilot Composer program. Under PILOT (pp. 73), strings are concatenated by naming two strings in a C:ompute (or A:ccept) instruction (e.g.: C:$ONE = $ONE $TWO). If, however, one of the strings is "undefined" because it has never been used before, it has the value of a text literal rather than the value of a string. In the example, if $TWO had the value JOHN but $ONE was undefined, the new value of $ONE would be $ONEJOHN — hardly what we wanted! I avoid this by initializing strings used in this way (see lines 130 & 140).

PILOT input and output (I/O) is handled with READ:, WRITE: and CLOSE: instructions. Each instruction requires a "device name" (a "C:" for cassette or a "D:" for disk) and, for disk, a file name. These are separated from following data by a comma. The data can be text literals, numeric or string variables. In a single file, READ: must be separated from WRITE: by a CLOSE:. You can try this in immediate mode or in a program:

DISK

```
WRITE:D:TEST,ABCD
CLOSE:D:TEST
READ:D:TEST,$STRING
T:$STRING
```

CASSETTE

```
WRITE:C:,ABCD
CLOSE:C:
READ:C:,$STRING
T:$STRING
```

We'll have more on I/O in a future article to discuss a hidden glitch. For now, just do as line 430 does and put all device specifications in a single string.

Keeping a clean screen in a program often requires erasing a line on the screen. It's not so simple in PILOT since the "blank line" string automatically defaults to one character. Lines 750 and 1230 show an easy way; just print a series of blanks followed by a non-printing character such as an arrow. Line 750, for instance, prints the #A followed by a blank and a left arrow. When the line is printed, the right-most character is blanked out, and the left arrow holds the space, but doesn't show. You can type an arrow by hitting the ESC key then holding down the CTRL key while hitting the desired arrow key. Repeat all three strokes for each arrow.

Although the Primer tells us that variables come in two flavors — strings (pp. 69–81) and numerics (pp. 85–92), we never find out how to change one into the other. It's simple but tricky. String variables can be made from numeric variables by C:omputing or A:ccepting them:

```
C:$ONE = #A
A:$ONE = #A
```

A string variable can be turned into a numeric variable ONLY by A:ccepting it:

```
A:#A = $ONE
```

After this instruction, #A will have the numeric value from $ONE; non-numeric data will be disregarded (see the Player program, lines 1310–1350).

Line 1140 in the Player program presents a powerful way to combine "relational operators" to make "conditional statements" (pp. 89–90). Linking conditions with "+" signs creates "logical ors". For instance, line 1140 would be read, "if #T = 256 OR if #T = 128 OR if #T = 64 then J:ump . . .". In other words, if #T equaled any one of the three numbers, the program would find a "true" and J:ump. Neat! But, you can't do it the other way, with a JN: instruction to execute on a "false," because the "N" looks at the M:atch register, not at the conditionals.

You can get "logical ands" by multiplying the conditionals:

```
T(#T = 100)*(#U = 200)*(#V = 50):ALL THREE
```

This statement would be read: "if #T = 100 AND if #U = 200 AND if #V = 50 then T:ype ALL THREE".

At last, the BUG. (A friend says that micros are too small to have bugs. She claims that they have fleas!) Right there on page 31 the Primer tells us that the computer "ignores" remarks. Although that may be accurate in the linguistic sense, it's not so in the operative sense. In line 1150 in the Composer program the remark set off by a "[" MUST be typed without spaces. It seems that the [turns any intervening spaces into significant space and, therefore, part of the accept buffer. Ditto for other commands. I don't know if it's a bug or a flea — I know it's a bear to figure out! (Atari's internal manuals even have it wrong!) Be safe, don't use brackets when in doubt.

listing on page 68

AUDIO WHILE YOU CLOAD

by JOHN VICTOR

"Your mission, Jim, if you choose to accept it . . ."

There is no question that the microcomputer community dislikes computer cassettes — and with good reason. In the early days of computing when hobbyists had no other storage medium, hours of frustration were spent trying to save or load programs from cassettes. When disk storage became available, most hobbyists gladly junked their cassettes. Many manufacturers have quietly stopped supporting their cassette systems.

Unfortunately, this has prejudiced software developers against the use of the Atari cassette system. However, I consider this component one of Atari's strongest points. The Atari system, unlike most others, uses a cassette player made specifically to run on the ATARI. This makes cassette recording and playback much more reliable.

But the strongest point is this: the Atari cassette is recorded in stereo. The digital information for programs is stored on the right track. Sound recorded on the left track is played back through the user's TV set. The existence of the leftside sound track means that recorded voice or music can be played at any time while the computer is on — either during the running of a program or during the loading of a program.

One technique that we use at **PDI** is to put voice instructions on the left sound track to play while a cassette is loading. This means that we do not have to put instructions for using for using the program in the program itself, reducing the memory requirements. At least half of the Atari market consists of 16K ATARI 400 computers. By keeping memory requirements within 16K (and providing programs in cassette format) a software publisher will reach a greater percentage of the Atari market.

The existence of a voice track gives the program user something to do in the time it takes to load the program. This can set the mood for the game itself. In **MOONBASE IO** we use the voice to give the player a "recorded message" from Earthbase control as to the nature of the mission (just like MISSION IMPOSSIBLE). Most of the four and a half minutes it takes to load the program is spent doing something related to playing the game.

To create and use the voice track during the cassette load, several things have to be done. First, the sound that the computer makes during a cassette load has to be turned off. This is done with a POKE 65,0. This can be put in a loader program placed first on the cassette. This loader program will contain a visual display, the POKE 65,0, and a CRUN routine that will automatically load and run the main program.

The following is the CRUN routine. POKE 764,32 will automatically produce a carriage RETURN so that the next program will begin loading. The ASCII values in the REM statement are those for the machine language CRUN routine found in the USR routine. (USR routines are used to run machine language from BASIC.)

```
1000 REM ROUTINE TO CRUN NEX
     T PROGRAM
1005 POKE 65,0
1010 POKE 764,32
1020 X=USR(ADR("":)7H)TH)
     6;)A"))
1030 REM ASC VALUES FOR ABOV
     E STRING
1040 REM 162,253,154,169,183
     ,72,169,84,72,169,4,32,
     182,187,169,255,76,4,18
     7
```

After the first program is loaded and run, instructions will be put on the screen and the next program load started. Any recorded sound in the left channel will now be heard clearly in the TV set. Positioning of the recording is important.

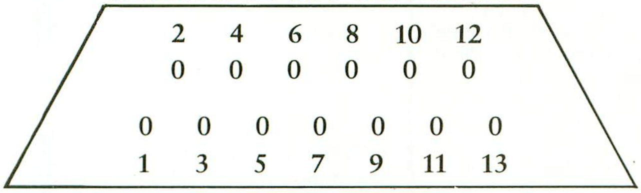
Atari programs have a two-second string of zeros recorded at the end of each cassette program. The programs stop loading two seconds before the recorded program ends. This means that the recorded voice or music can begin just before the first program ends, but must end two seconds before the main program's record track. Otherwise the computer is going to turn off the voice track before it finishes.

The Atari 410 Program Recorder can play back voice and music but cannot be used to record it. This must be done on a stereo tape deck or a reel-to-reel recorder. For the sake of quality, master tapes from which cassettes are going to be manufactured should be made on reel-to-reel recorders ONLY! Cassette recorders do not produce good enough SOUND to be copied. There is just too much speed variation and lack of separation between the two stereo tracks on cassette masters. If the user only wants a few copies, then a stereo tape deck is okay, but this is not acceptable for commercial software producers.

John Victor is President of Program Design, Inc., a software manufacturer whose products for the ATARI computers include MOONBASE IO *and* Pumpkin Stand, *both of which use the techniques described in this article.*

The first step in making the master tape is to record the programs. The ATARI computer makes no provision to connect the ATARI to a stereo recorder, so the programmer will have to rig up something. This is not very difficult. The "data out" and the "ground" pins in the peripheral connector are the ones that send the program signal to the recorder.

SERIAL I / O PORT CONNECTOR



1. Clock In
2. Clock Out
3. Data In to Computer
4. GND
5. Data Out of Computer
6. GND
7. Command
8. Motor Control
9. Proceed
10. +5 / Ready
11. Audio In
12. +12
13. Interrupt

These can be connected to the recorder with a cable that has alligator clips on one side and an RCA connector on the other. A local Radio Shack or audio dealer may have this, or an audio technician can make one. The alligator clips are then connected to the ATARI pins 5 and 6, and the RCA is plugged into the right recording jack of the stereo unit. It's not a bad idea to put tape over the alligator clips to keep them from touching the wrong points.

Before recording, start the computer outputting and set the VU meter on the recorder at between 7 and 5. Also note the reading on the tape counter.

Record the loader program. The computer will lay down 18 seconds of pilot tone before the program is recorded. However, after the program is loaded, the computer will continue to output pilot tone. Listen to the computer for an indication of when the program stops, and immediately shut off the recorder. Next, record the main program. Using the tape counter, keep track of where on the tape the second program is.

The voice (or music) can now be recorded. To record voice, connect a microphone to the left-side "mic" jack. The recorder must be one that will not erase the right track while the left one is being recorded. This can be determined quite simply — there must be a separate record button for each track.

Using the tape counter as a guide, rewind the tape. Then begin recording voice instructions and / or music on the left track. This must be finished 2 seconds before reaching the end of the recorded program (because that is where the computer is going to stop when the program is loading).

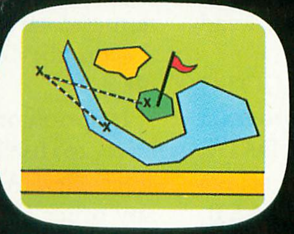
It will also help to have an appropriate graphic on the screen while the main program is being loaded. If directions are being given, the directions might also appear on the screen at the same time.

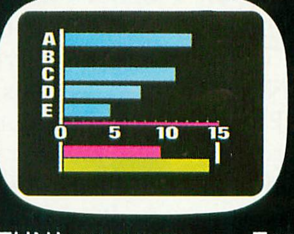
This technique can enhance a program and make it more interesting. It also adds a "professional" touch to cassettes.

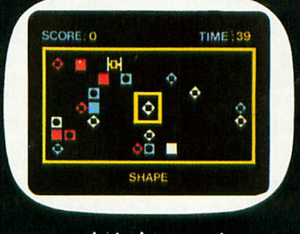
MILLIKEN PUBLISHING COMPANY'S

EduFun!™

Exciting, new educational games for the ATARI*







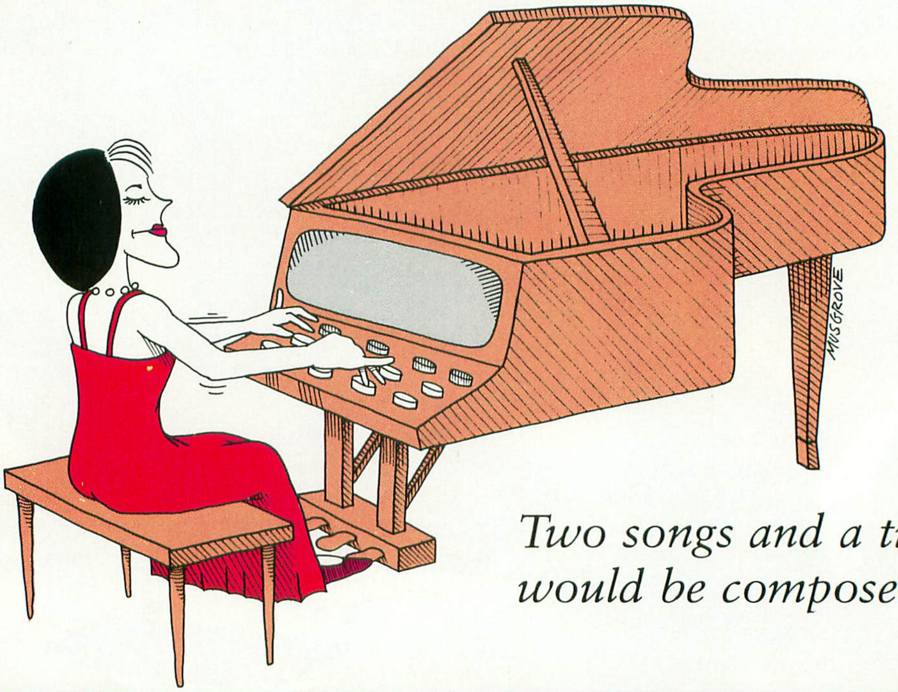
- **MILLIKEN PUBLISHING** . . . the leading publisher of educational software for schools
- **EDU**cational games . . . designed to help and challenge children at home and school
- **FUN!** . . . for the family and classroom

Put it all together and you've got MILLIKEN'S EDUFUN! programs . . . For your edutainment

CONTACT YOUR COMPUTER DEALER OR
MILLIKEN PUBLISHING COMPANY
EduFun!™ Division
1100 Research Blvd., St. Louis, MO 63132

*Available on the Atari 400 and 800 computers. Available soon on the Apple, TI 99/4, and Commodore VIC 20.

MUSIC WITH BASIC

by JERRY WHITE

Two songs and a tutorial for would be composers.

This tutorial and example program demonstrate one of the many ways of playing music using ATARI BASIC. Those of you with no knowledge of music may simply type in the program and follow the instructions on the screen. If you have some knowledge of music, and you'd like further information on how this program works, read on.

The program begins with a GOTO 370. This bypasses the main program loop, subroutines, and song DATA, and brings us to our setup and screen display. Here we specify GRAPHICS 0, set the background color at random, turn off the cursor, set the left margin at 5, set the print tab width at 7, and NP = 0. The numeric variable NP will be used to count the Notes Played. Lines 380–420 display our program description, author name, and user options. POKE 764,255 tells the computer to ignore the last key pressed.

The routine beginning at line 430 and ending at line 460 waits for the user to press a normal video 1, 2, or 3. Nothing will happen until one of these keys is pressed. The checking is done by peeking at location 764 until it contains a 31, 30, or 26. These are the internal keycodes for 1, 2, and 3. By checking the last key pressed, we eliminate the need to press the RETURN key.

Once we have a valid key, we position the cursor at the appropriate option number on the screen, and print that number using inverse video. The numeric variable PLAY is used to store the number of notes we are about to play.

If option one was selected, we do not have to use a RESTORE command since the DATA for this song preceeds any other DATA. If either of the other options has been chosen, we use the RESTORE command to point to the line number where the appropriate DATA begins.

If the number 3 key was pressed, we also must set a flag to indicate a special condition. Since this program reruns itself when a song is over, we set the variable EXIT = 1 in line 450 before GOTO 120 instruction.

Look at line 120. In English, it says that if the number of Notes Played is equal to the number of notes we wanted to PLAY, then go to line 470. Line 470 begins with "IF EXIT". This is the same as saying "IF EXIT <> 0". So IF EXIT = 0, the program falls through to line 480 where we have a RUN command. If EXIT <> 0 then we reset the left margin, turn the cursor back on, tell the user that BASIC has control, and END the program.

Now that we know how the program starts and how it ends, let's see what happens in between. Let's assume you have chosen option number 3. As you pressed the number 3 key, an ASCII 26 was automatically stored in location 764. At line 450 we hit a true condition and highlight the number 3 on the screen, set PLAY = 10, RESTORE 360, set EXIT = 1, and GOTO 120.

The routine from line 120 through 150 is called the main program loop. We haven't played any notes yet so NP = 0 and we fall through to line 130. Here we read two bytes of DATA. This will result in the variable PITCH being set to 91 and DUR being set to 12. Remember, we are reading the DATA that begins in line 360. Also in line 130 we add 1 to NP.

In line 140, we see if PITCH = 0, and if it is, we GOTO our REST routine which begins at line 170. PITCH = 91 so we GOTO our SOUND routine at line 200.

We will POKE the value of DUR into a countdown timer at RAM location 540. Countdown timers count backwards at the rate of 60 per second until zero is reached. In other words, when we POKE 540,DUR, since DUR = 12, exactly 12/60 of a second later, the countdown timer will reach zero. In that same line we calculate the pitches we will use in SOUND registers 1 and 2, and store the value of PITCH + 1 in P1 and PITCH–1 in P2.

At line 210 we turn the tables and set DUR = PEEK(540), and check to see if it is equal to zero. At this point it isn't zero yet, so we continue on to line 220 and see if DUR > 6. Six will be our maximum volume of each of three SOUND commands. In any case, we continue on to execute three SOUND commands, then go back to line 210 and check the value in our countdown timer again. We stay in this loop until we find that our countdown timer has reached zero.

When PEEK(540) = 0, we GOTO line 240 where all sounds are turned off, and we can finally go back to where this whole thing started, line 120.

Remember line 120? That's the main program loop. We have played one note and have nine to go. But what if the PITCH is a 0? When we want no sound for a period of time (a REST), we enter a zero as the pitch, and use the routine beginning at line 170 to rest for the period of time specified by DUR. By the way, 60ths of a second are also known as "jiffies."

By using DUR as the volume value in the SOUND commands, we get a slight decay or decreasing volume at the end of each note. By using two additional SOUND channels, and setting their frequency levels slightly higher and lower than the desired pitch, we achieve a richer, fuller sound.

This program demonstrates only one method of playing music on your computer. BASIC can be used to play true four-part harmony and even display the lyrics of your songs on the screen at the same time. This is demonstrated by Swifty Software's Singalong Sound & Music Tutorial package.

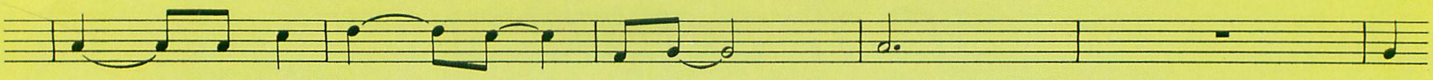
ATARI's Music Composer provides another way to play music and displays musical notes on your screen. Unfortunately, you can't put the Music Composer Cartridge and BASIC in at the same time. But I found a way around that problem too.

P.D.I.'s Music Box will convert your Music Composer files and play them for you using Vertical Blank Assembler Subroutine. This is done while the BASIC cartridge is installed. The best part is that once the music begins, BASIC is at your disposal. You can even write a BASIC program while the music continues to play.

The possibilities provided by your computer's audio channels are almost limitless. Take advantage of this and let us know what you come up with.

```
10 REM ATARI BASIC MUSIC by Jerry White 5/4/82
20 GOTO 310
30 REM MAIN PROGRAM LOOP
40 IF NP=PLAY THEN 420
50 READ PITCH,DUR:NP=NP+1
60 IF PITCH=0 THEN 90
70 GOTO 130
80 REM REST TIME DELAY SUBROUTINE
90 POKE 540,DUR
100 IF PEEK(540)<>0 THEN 100
110 GOTO 40
120 REM PLAY NOTE SUBROUTINE
130 POKE 540,DUR:P1=PITCH+1:P2=PITCH-1
140 DUR=PEEK(540):IF DUR=0 THEN 170
150 IF DUR>6 THEN DUR=6
160 SOUND 0,PITCH,10,DUR:SOUND 1,P1,10,DUR:SOUND 2,P2,10,DUR:GOTO 140
170 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,0,0,0:GOTO 40
180 REM DATA FOR POP GOES THE WEASEL
190 DATA 121,6,91,6,0,6,91,6,81,6,0,6,81,6,72,6,60,6,72,6,91,6,0,6
200 DATA 121,6,91,6,0,6,91,6,81,6,0,6,81,6,72,18,0,6,91,6,0,6
210 DATA 121,6,91,6,0,6,91,6,81,6,0,6,81,6,72,6,60,6,72,6,91,6,0,18
220 DATA 53,12,0,12,81,12,0,6,68,6,72,18,0,6,91,12
230 REM DATA FOR TEN LITTLE INDIANS
240 DATA 121,18,121,6,121,6,121,18,121,6,121,6,96,18,81,6,81,6,96,6,96,6,121,18
250 DATA 108,18,108,6,108,6,108,18,108,6,108,6,128,18,108,6,108,6,128,6,128,6,162,18
260 DATA 121,6,121,6,121,6,121,6,121,18,121,6,121,6,96,18,81,6,81,6,96,6,96,6,121,18
270 DATA 108,18,108,6,108,6,162,6,162,6,162,18,121,48
280 REM DATA FOR EXIT ROUTINE
290 DATA 91,12,0,6,121,6,128,6,121,6,108,24,121,24,0,24,96,24,91,24
300 REM SETUP/DISPLAY/OPTIONS
310 GRAPHICS 0:SETCOLOR 2,RND(0)*16,0:POKE 752,1:POKE 82,5:POKE 201,7:NP=0
320 ? :? :? ,"ATARI BASIC MUSIC"
330 ? :? ," by Jerry White":? :?
340 ? :? "Type 1 for POP GOES THE WEASEL"
350 ? :? "Type 2 for TEN LITTLE INDIANS"
360 ? :? "Type 3 for PROGRAM EXIT";:POKE 764,255
370 IF PEEK(764)=31 THEN POSITION 10,8:? "1";:POKE 764,255:PLAY=43:GOTO 40
380 IF PEEK(764)=30 THEN POSITION 10,10:? "2";:PLAY=44:RESTORE 240:GOTO 40
390 IF PEEK(764)=26 THEN POSITION 10,12:? "3";:POKE 764,255:PLAY=10:RESTORE 290:EXIT=1:GOTO 40
400 GOTO 370
410 REM EXIT/RERUN
420 IF EXIT THEN POKE 82,2:POKE 752,0:? :? :? "BASIC":? "IS";:END
430 RUN
```

Variable checksum = 153160

Line num range	Code	Length
10 - 120	JT	275
130 - 210	NH	521
220 - 310	GO	602
320 - 410	DS	509
420 - 430	RE	76

ZOUNDS!

by ED ROTBERG

Since this issue of ANTIC delves into the mysteries of computer-generated sound, I will share with you some of the inner workings of a major project of mine, the Rotberg Synthesizer. I will have to assume a reasonably high level of programming competency on your part.

The Synthesizer does a pretty good job of shaping POKEY sounds into approximations of real musical instruments. It works much better, in my opinion, than the Atari Music Composer cartridge. The most important reason why is that it can provide "envelopes" for the frequencies, and an amplitude for each note.

The term "envelope" refers to the temporal variation of some aspect of a sound. In this case, the aspects to be varied are frequency and amplitude. The code "ADSR" is the standard way of specifying an amplitude envelope, and the code stands for "Attack, Decay, Sustain and Release."

Figure 1 will give an idea of what these terms mean in the case of a harpsicord-like amplitude envelope. The rest of the article will present an approach to creating such envelopes in a music generating program like the Rotberg Synthesizer.

The whole project started as a gag while I was working at the Atari Coin-Op Division. One of our colleagues, a *disco* freak, was compounding his bad judgment by getting married. Such was the birth of the Synthesizer, which was used to compose our congratulatory lament, the Disco Dirge, written by Dan Pliskin, another ex-Atarian.

Some stubborness prevents me from just listing the program for you. I guess I'd rather lead you to an understanding of how to do it for yourself. I will be referring to various registers in the POKEY chip, and certain functions of the POKEY, but I will in no way describe that chip. Also, I will not be relating any of this to BASIC techniques, which are hopelessly slow for this kind of work. Nor will I discuss any sound editing techniques, but only the means of generating the musical sounds.

There are basically two major classes of sound generation used: static and dynamic. The first consists of nothing more than storing a few values to the various POKEY registers, and sitting back and listening. The capabilities of this approach are quickly exhausted. More useful, and far more interesting, are the dynamic sounds, in which the values stored to the POKEY are constantly changed during the duration of the sound. Three approaches to dynamic sound generation are:

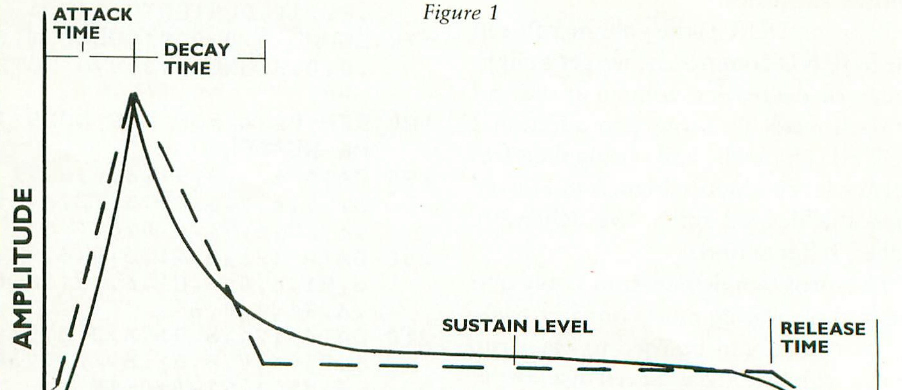
1) **Algorithmic.** A short routine calculates the values to be stored. The possibilities are limited only by the imagination of the programmer.
2) **Table driven.** A short program keeps an index into a lookup table to determine what values are to be stored into POKEY during that time interval. New sounds can be generated very quickly by slopping some new values into the tables.
3) **Interpretive.** A small interpreter program reads instructions and data from a command stream, causing the sounds to be generated by a few preset rules. This method keeps the data tables short, compared to a pure table-driven approach.

Let's go over just what the Synthesizer is capable of. It has the ability to produce sound on all 4 channels of the POKEY simultaneously. The basic unit of sound is called a NOTE, since this program was intended to be primarily a *music* synthesizer, though it is capable of generating a wide variety of sounds. The frequency of the NOTE is specified

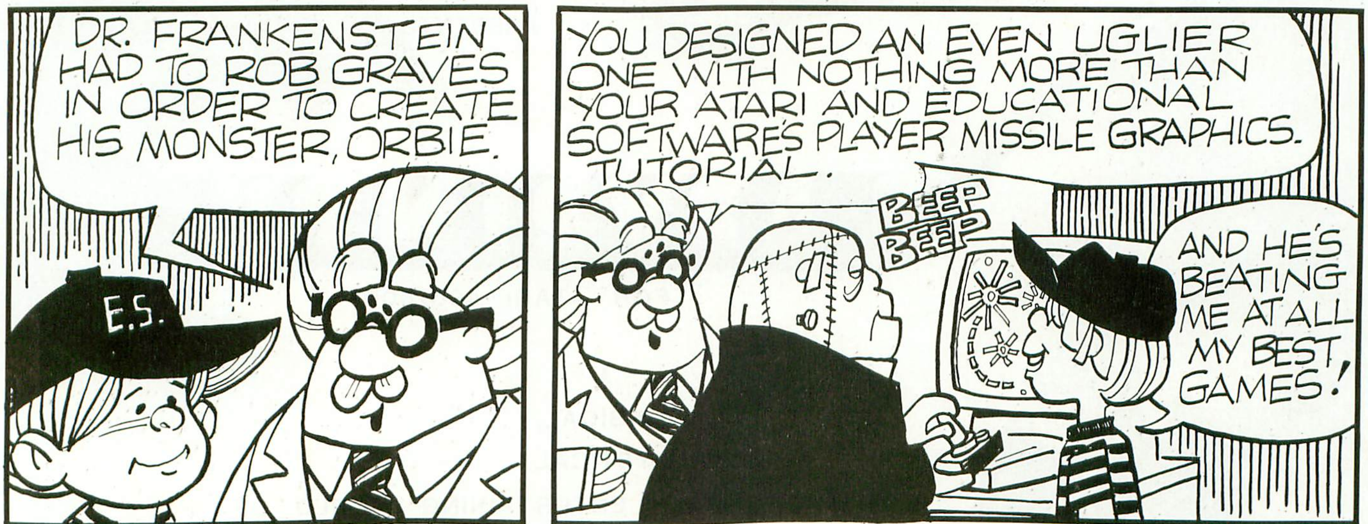
continued on page 37

Ed Rotberg is an Electrical Engineer with many years of computer programming experience. He was with Atari, Inc. from 1979 to 1981 as a software developer and consultant on the ATARI 800 project. Among his programs are the Rotberg Scrolling Marquee and the Rotberg Synthesizer. He helped create sound effects, using the ATARI, for the movie TRON, and is a partner in Videa, Inc., a new electronic entertainment firm in Sunnyvale, CA.

Figure 1



The Adventures of PROFESSOR VON CHIP & ORBIE



TRICKY TUTORIALS (tm)

There are many things that the ATARI computers can do either better, or easier than other small computers. The following series of programs is designed for anyone who is at least familiar with BASIC programming. What each tutorial offers is similar to an extensive magazine article with all discussion in as simple language as possible, plus you get MANY examples already typed in and running. The instruction manuals range from 10 to 50 pages, and some tutorials fill up a complete tape or disk. There is little overlap in what is taught, so anyone wanting to know all they can should buy them all (my banker thanks you). ATARI buys these from us to use in training their own people! Rave reviews have been published in ANTIC, ANALOG, CREATIVE COMPUTING, and even INFOWORLD. You trust INFOWORLD, don't you?

TT #1: DISPLAY LISTS—This program teaches you how to alter the program in the ATARI that controls the format of the screen. Normally, when you say "Graphics 8", the machine responds with a large Graphics 8 area at the top of the screen and a small text area at the bottom. Now, you will be able to mix various Graphics modes on the screen at the same time. The program does all of the difficult things (like counting scan lines). You will quickly be able to use the subroutines included in your own programs.
16K Tape or 24K Disk. **$19.95**

TT #2: HORIZONTAL/VERTICAL SCROLLING—The information you put on the screen, either GRAPHICS or TEXT, can be moved up, down, sideways, or diagonally. We provide the basic methods and leave the rest up to your skill and imagination. Includes 18 examples to get you started, with several using a small machine language subroutine for smoothness.
16K Tape or 24K Disk. **$19.95**

TT #3: PAGE FLIPPING—Now you don't have to redraw the screen every time you change the picture or text. You will learn how to have the computer draw the next screen you want to see while you are still looking at the previous screen, then flip to it instantly. You won't see it being drawn, so a complicated picture can seem to just appear. Depending on your memory size and which graphics or text modes you are using, you can instantly look at up to 50 pages. The basic method takes only 9 lines and the usefulness is infinite.
16K Tape or 24K Disk. **$19.95**

TT #4: BASICS OF ANIMATION—This program shows you how to animate simple shapes (with some sound) using the PRINT and PLOT commands, and it also has a nice little PLAYER/MISSILE GRAPHICS game you can learn from. The P/M example is explained and will get you started on this complicated subject (more fully explained in TT #5). This would be an excellent way to start making your programs come alive on the screen with movement! Recommended for beginning users.
16K Tape or 24K Disk. **$19.95**

TT #5: PLAYER/MISSILE GRAPHICS—Learn to write your own games and other animated applications! The tutorial begins with many small examples that compliment the 50 page manual, then gradually builds up to a complete game where everything you need to know is fully explained. Also included are two machine language utilities that you can use to animate Players with from BASIC. Next we include two of the best editors currently available; one for editing playfield shapes (backgrounds); and one to edit your players, and all in glorious Technicolor!! Everything except the two editors run in 16K Tape or 32K Disk. **$29.95**

TT #6: SOUND AND MUSIC—Unless you have spent many years experimenting with the four voice channels, you will learn a lot from this one! Learn to play standard notes, chords, and whole songs using some simple "tricks". One of the nicest parts are the examples of special sound effects that you can refer to whenever you need a sound for a program or to impress a friend. This program will be of interest to all ages and levels of experience!
16K Tape or 24K Disk. **$19.95**

SPECIAL DISCOUNT

Order the first six tutorials in a 3-ring binder for $99.95, a $30.00 savings!

TT #7: DOS UTILITIES—We at Educational Software have been shocked by some of the prices others are charging to offer you small utilities to help in the use of your Disk Drive. We now offer you all of the following plus explanation as to how each was written, and how to use them: A UNIQUE MENU PROGRAM, AN AUTORUN.SYS BUILDER, DISK INSPECTOR (LOOK AT SECTORS), DISK JACKET PRINTER, AUTOMATIC FORMATTER, RECORD SAVE AND LOAD UTILITY.
32K Disk Only. **$29.95**

MASTER MEMORY MAP (tm)

This book is the most valuable source of information for your ATARI you can buy. It starts out by explaining how to PEEK and POKE values into memory, so that even new computer owners can use many of these "Tricks". Then you are given 32 pages of the memory locations that are the most useful, along with hints on how to use many of the locations. Finally, it includes hints on problems you may be having with the computer and discusses the new Graphics modes 9 to 11. Even ATARI buys this book from us! **$6.95**

CONTACT YOUR LOCAL DEALER or ORDER BELOW

We have other fine programs for ATARI computers.
Write for a catalog.
Send us your programs to sell too!

USER SUBMITTED PROGRAMS

MINI-DATABASE/DIALER—stores and edits up to 8 lines of information such as names & addresses, phone numbers, messages, inventories, or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature: If your file includes phone numbers and your phone company allows touch-tone phone signals, the program will DIAL THE PHONE NUMBER FOR YOU! 16K Tape or 24K Disk. **$24.95**

THE GRAPHICS MACHINE—allows the ATARI to act like more expensive graphics computers using simple commands like line, box, circle, polygon, fill, and savescreen to get a high resolution picture you can save on disk in only five seconds! Many more features! 48K Disk Only. **$19.95**

BOB'S BUSINESS—14 small business type programs accessed from a common menu. 16K Tape or 32K disk. **$14.95**

MINI WORDPROCESSOR—A simple text editor to write, save, and print several pages at a time. 32K Tape or Disk. **$19.95**

KID'S #1—Includes a MATH QUIZ, a children's TREASURE HUNT, and a DIALOGUE program. 16K Tape or 24K Disk. 3 for . . . **$14.95**

KID'S #2—SPELLING BEE, WORD SCRAMBLE, and TOUCH. 16K Tape or 24K Disk. 3 Educational games for . . . **$14.95**

PLAYER PIANO—Turns your keyboard into a mini-piano and more. 24K Tape or 32K Disk. **$14.95**

DOG DAZE—Two cute little doggies race for the fire hydrants, shoot their bones, and just have a lot of fun! A fast action program for all ages. 8K Tape or 16K Disk, in machine language. **$16.95**

GRAPHIC SYMBOL LABELS—for your keyboard to remind you of the built-in Graphics symbols. 2 complete sets for . . . **$2.95**

OUR NEWEST PROGRAMS

*** OUR BEST GAME ***

SPACE GAMES—Our family is being attacked by ALIENS, and only you can save us. A comic book manual will guide you through three games that test your ability in space skills. Includes ALIENS, SURVIVE, and ROBOT ATTACK, and is for all ages. The first two games require 16K for Tape. The last game and all Disk users need 32K. **$24.95**

MATHS FOR FUN—Another ENGLISH import teaching basic math skills. Very colorful and enjoyable to use. For ages 5 to 16. 16K Tape or 24K Disk. **$19.95**

TT #10: SOUND EFFECTS—From laser blasts to ringing phones, this tutorial will show you how to make unique sound effects in all of your programs! 16K Tape or 24K Disk. **$19.95**

MARATHON—This is a unique math quiz for one or two players. You are in a race to move your runner across the screen first! There are four levels of play with five modes of operation for each. The game uses joysticks for all input, so play is easy for young children. This wonderful learning tool is imported from ENGLAND for your learning pleasure. Your kids will never even notice they are playing an EDUCATIONAL program. 16K Tape or 24K Disk. **$19.95**

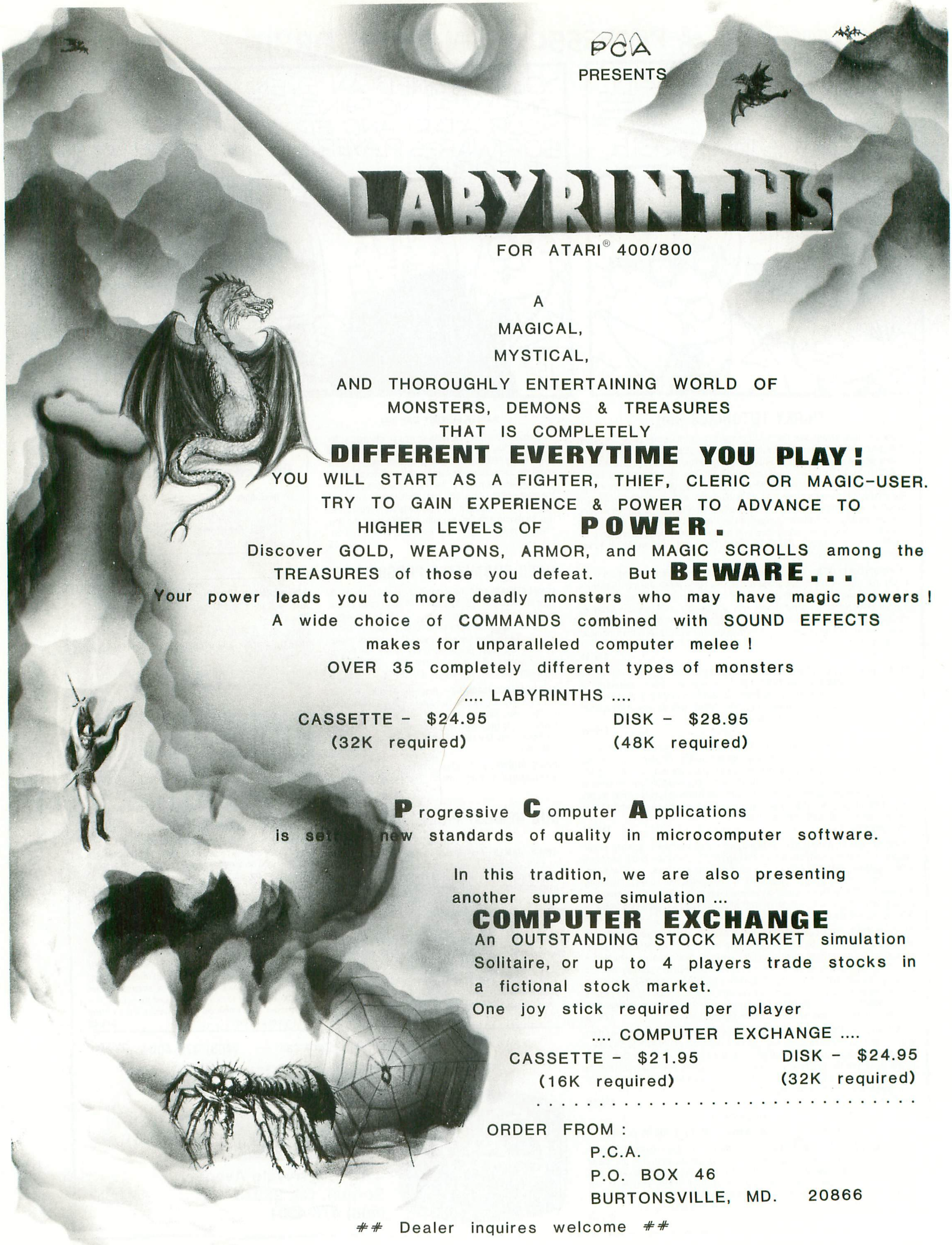
TT #8: CHARACTER GRAPHICS—Character Graphics is the best way to animate your ATARI! Make letters look like space monsters, gunfighters, or a myriad of other shapes. Use our editor to create these multicolor shapes and then we'll show you how to move them around the screen. This tutorial even shows how our Space Games were written! 16K Tape or 24K Disk. **$19.95**

To order COD, VISA or MasterCard call — **(408) 476-4901**

By mail include $3.00 postage (USA only) or $1.50 Memory Map only—
California residents add 6.5% TAX.—Specify Tape or disk.

Educational Software inc.
4565 Cherryvale Avenue
Soquel, Ca. 95073
(408) 476-4901

800-692-9520
Dept. AT-1


PCA
PRESENTS
LABYRINTHS
FOR ATARI® 400/800
A
MAGICAL,
MYSTICAL,
AND THOROUGHLY ENTERTAINING WORLD OF
MONSTERS, DEMONS & TREASURES
THAT IS COMPLETELY
DIFFERENT EVERYTIME YOU PLAY!
YOU WILL START AS A FIGHTER, THIEF, CLERIC OR MAGIC-USER.
TRY TO GAIN EXPERIENCE & POWER TO ADVANCE TO
HIGHER LEVELS OF POWER.
Discover GOLD, WEAPONS, ARMOR, and MAGIC SCROLLS among the
TREASURES of those you defeat. But BEWARE...
Your power leads you to more deadly monsters who may have magic powers !
A wide choice of COMMANDS combined with SOUND EFFECTS
makes for unparalleled computer melee !
OVER 35 completely different types of monsters
.... LABYRINTHS ....
CASSETTE - $24.95
(32K required)
DISK - $28.95
(48K required)
Progressive Computer Applications
is setting new standards of quality in microcomputer software.
In this tradition, we are also presenting
another supreme simulation ...
COMPUTER EXCHANGE
An OUTSTANDING STOCK MARKET simulation
Solitaire, or up to 4 players trade stocks in
a fictional stock market.
One joy stick required per player
.... COMPUTER EXCHANGE ....
CASSETTE - $21.95
(16K required)
DISK - $24.95
(32K required)
ORDER FROM :
P.C.A.
P.O. BOX 46
BURTONSVILLE, MD. 20866
Dealer inquires welcome

by 8 bits which may either be a pointer into a table of frequencies, or the actual frequency itself. This is an implementation decision, and each method has its merits and drawbacks. If the actual frequency is stored, the NOTE must also specify the "noise content or distortion" value to be stored in the control register along with the "sustain" volume for each channel. Each NOTE can specify a 4-bit value for its sustain volume, and can have a duration specified by 16 bits.

This duration is relative to the current TEMPO. The TEMPO is specified by an 8-bit value, which is used as a delay loop counter. The TEMPO can only be changed relative to its current value by a 2's complement add of any 8-bit value. Note that in versions of the Synthesizer that run during the vertical blanking interval, such as the Atari POP Demo program, the TEMPO feature is not implemented, as the timing interval is fixed at 60 hertz. Each channel can specify it's own current ENVELOPE table which controls the attack / decay of either amplitude, frequency, or both. Attack and decay are not specified as rates or times, but rather as a table of digitized amplitudes during the attack / decay period. This period can cover a span of a few milliseconds to a few seconds.

Care must be taken not to wrap either of these values, unless of course that is the intended result. At the present time, "Release" is not implemented. The Synthesizer has the ability to REPEAT a section of music up to 100 (hex) times. These REPEATS may be nested without any restriction except that the total number of REPEATS in a piece of music must not exceed 100 (hex). The Synthesizer can also play PHRASES. I have chosen not to implement the four separately tracking stacks necessary to allow for nesting of PHRASES, although this is certainly simple enough to do. Each PHRASE must specify its own RETURN. In addition, any channel's instruction stream can cause AUDCTL to be changed on the fly. That's about it.

In its current form, THE ROTBERG SYNTHESIZER supports 7 instructions:

1) Repeat
2) Set / change Envelope
3) Set / change AUDCTL Register
4) Play Phrase
5) Return from Phrase
6) Change Tempo
7) Play 1 note

The Synthesizer processes 4 sets of these instructions simultaneously, one for each channel in POKEY. Each instruction stream is made up entirely of these instructions, in addition to a STOP directive that is only valid when encountered in channel 1's instruction stream.

The data structure format for each instruction follows, where each cell represents one byte. All value / ranges are given in hexadecimal.

REPEAT: op-code = FF

FF
nn
ll
hh
ii

FF = *REPEAT op-code*
nn = *repeat count (0 = 100, 1 = NOP, count indicates number of times section is to be played)*
ll = *low byte of address of 1st instruction of section*
hh = *hi byte of address*
ii = *index into ram table for this section's repeat counter*

This instruction has the effect of conditionally repeating a section of the instruction stream a specified number of times. Because each REPEAT instruction has its own loop counter in a RAM table 100 (hex) bytes long, any amount of nesting of these REPEAT instructions is allowed, as long as the total number of REPEATS in any composition is 100 or fewer. Each REPEAT can play its section up to 100 times. This instruction appears *at the end* of the section to be repeated, and refers to the first instruction of that section in its operand field.

SET ENVELOPE: op-code = FE

FE
ll
hh

FE = *SET ENVELOPE op-code*
ll = *low byte of address of envelope table*
hh = *hi byte of address*

This instruction sets the pointer to the current ENVELOPE table for that channel. A SET ENVELOPE instruction MUST precede the first note instruction on any channel. ENVELOPES may be changed at any time.

CHANGE AUDCTL: op-code = FD

FD
cc

FD = *CHANGE AUDCTL op-code*
cc = *new audctl value*

This instruction is used to change AUDCTL on the fly. This represents powerful, dynamic control of the POKEY. It may be used from any channel, but in practice, it is best only altered from one channel within a piece, as AUDCTL can affect ALL channels.

CALL PHRASE: op-code = FC

FC
cc
hh

FC = *CALL PHRASE op-code*
ll = *low byte of address of 1st instruction of phrase*
hh = *hi byte of address*

This instruction will transfer control to a PHRASE which can be "called" any number of times. In the current implementation, there is NO nesting of PHRASE calls (i.e. only 1 level of calling a PHRASE). PHRASES themselves, may therefore use any instructions other than CALL PHRASE, and must terminate with a RETURN instruction. Note, that while possible, it is dangerous to have 2 channels use the same PHRASE, especially if that PHRASE

continued on next page

contains REPEAT instructions.

RETURN FROM PHRASE:
op-code = FB

FB

FB = *RETURN op-code*

This instruction is used to return from a PHRASE.

CHANGE TEMPO

FA
tt

FA = *CHANGE TEMPO op-code*
tt = *2's complement delta change to TEMPO*

This instruction is used to change the current TEMPO by a 2's complement delta value. This instruction can appear in any channel, and obviously affects all channels.

NOTE: op-code = <FA

ca
ff
dd
ee

c = *control nibble (upper nibble of volume)*
a = *sustain volume*
ff = *sustain frequency or pointer to freq. table*
dd = *low byte of 16 bit duration*
ee = *hi byte of 16 bit duration Duration is relative to TEMPO. For convenience, a value of 100 (hex) is usually used to represent a whole note. This means that for long durations, the high byte (ee) of the duration represents a measure count in 4/4 time.*

All instructions not having an op-code for FA or greater are NOTE instructions. ENVELOPES will be applied to all NOTE instructions with one exception! If the first two bytes (ca,ff) are zero, then the NOTE is considered a rest, and no envelope is applied. Note that in processing the instruction stream for each channel, all non-NOTE instructions are processed immediately, until a NOTE instruction is encountered. In other words, all non-NOTE instructions take up NO duration time, and a NOTE instruction MUST be processed for each channel every cycle through the interpreter. Also, when a rest (NOTE ca,ff = 0) of duration zero is encountered in channel 1, it is evaluated as a global STOP instruction, and the piece is over.

Various data structures are used by the interpreter for processing the instruction streams. A brief description of each follows.

PNTR - 8 bytes

Two bytes per channel. This table maintains the current "program counter" for each channel.

NRPT - 8 bytes

Two bytes per channel. This structure contains the duration remaining on the current NOTE of each channel.

RPTBLK - 100 (hex) bytes

There is a one-to-one correspondence between each REPEAT instruction and a unique byte in this table. These bytes contain the counts remaining in each repeat section. When a REPEAT instruction is encountered, this byte is checked. If it is zero, it is then initialized to the value specified in the REPEAT instruction and decremented immediately. If it is non-zero, then it is merely decremented. The interpreter will then execute the REPEAT only if the decrement does not bring the value to zero. Thus, a 1 for a repeat count is an effective NOP, and the repeat count represents the number of times a section is actually played. Obviously, this entire table MUST be erased prior to starting to play a piece.

TREG - 8 bytes

Two bytes per channel. This is a staging area for the values to be stored to all 8 frequency and control registers for the four POKEY voices. Since the processing time for each of the 4 channels in a single interpreter cycle may vary, the POKEY values generated are saved in a holding register until all are calculated, and can be stored to POKEY with a single move loop.

ENVL - 8 bytes

Two bytes per channel. This table maintains the pointer to the current ENVELOPE table.

EINDX - 4 bytes

One byte per channel. This is the current index into the ENVELOPE table. It counts up by 2 from an initial value of 2. The reason for this will become evident in the discussion of the ENVELOPE table itself. EINDX is reset to 2 by the start of each new note.

RTNADR - 8 bytes

Two bytes per channel. This table contains the return address to a main instruction stream from a PHRASE. It is zero when in the main instruction stream so that RETURNS and CALL PHRASES can check for validity. Because these return addresses are not stacked, there is no nesting of PHRASE calls allowed.

ENVELOPE tables -
4 to 100 (hex) bytes

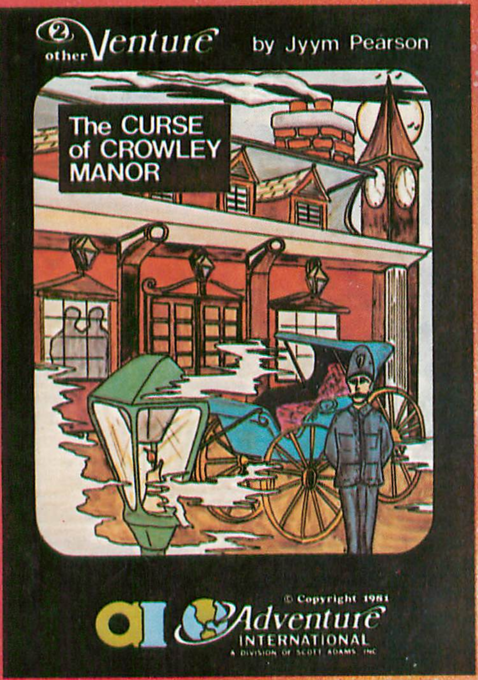
The first byte has the table length, a maximum of FE (hex). The EINDX value is compared against this first byte to determine whether the NOTE value is to be modified by the ENVELOPE, or whether the duration has exceeded the attack/decay period and the sustain values for frequency and amplitude are to be used. Each 2 bytes in the table represent both frequency and amplitude modifiers for one duration count. Since a maximum EINDX of FE is allowed, this means that durations longer than 7F cannot be modified by an envelope past that point. The hi byte of each 2 byte value modifies the amplitude (low nibble only), and the low byte modifies the frequency, both by 2's complement addition.

The remaining data structures used are the instruction streams themselves. There must be one per channel, even if the channel is dormant.

There it is, in the proverbial nutshell. This should be enough to get the more adventuresome of you started.

OTHER WORLDS. OTHER REALITIES.

(FOR EXPERIENCED ADVENTURERS ONLY!)



THE CURSE OF CROWLEY MANOR

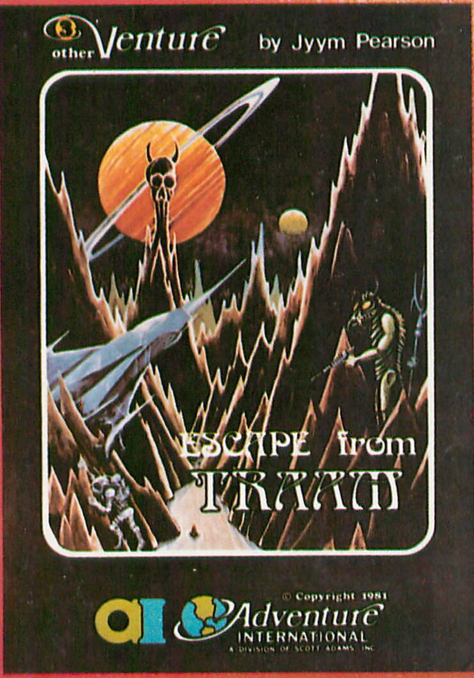
by Jyym Pearson

You've sensed it before: nameless, unreasoning fear. Fear of the unseen — of things that scurry and scrabble through the shadows when the lights have been put out. How then will you fare when you face the Ultimate Evil; how will you possibly survive — THE CURSE OF CROWLEY MANOR?

Jyym Pearson, author of ESCAPE FROM TRAAM and many other fine programs, explores a new dimension in fright with a macabre tale of things that go bump in the night. THE CURSE OF CROWLEY MANOR is an excursion into mind-wrenching terror as you enter the dreaded Crowley Estate in search of answers to some pretty grisly questions. Fear fans take note: This is the stuff that nightmares are made of, so think twice before you approach . . .

THE CURSE OF CROWLEY MANOR features Adventuring the way YOU like it, with plenty of clever plot twists, and a scenario that'll roll your socks up and down your legs! If Adventuring has become a bit tame for you lately, this is one sure-fire cure! THE CURSE OF CROWLEY MANOR will run on any ATARI 400 or 800 with ample memory. We supply the software; you supply the atmosphere. (A well-lit room is optional, but recommended!)

ATARI 16K TAPE 050-0108 $24.95



ESCAPE FROM TRAAM

by Jyym Pearson

As soon as your small space cruiser began losing altitude, you knew you were in for serious trouble. You somehow walked away from a fiery landing that would've spelled certain death for a less fortunate man, but as one may quickly learn, there are fates far worse than a rapid death . . .

You've managed to walk away intact from the awesome Curse of Crowley Manor. So far, so good. But will even your finely tuned Adventuring skills give you the necessary edge to . . . ESCAPE FROM TRAAM?

ESCAPE FROM TRAAM is a detailed Adventure for the ATARI 400 or 800 computer that defies the player to successfully search a bizarre alien world for a means of escape. The perils are many and the opportunities for survival are few. But for the clever Adventurer, a single opportunity is all that might be needed for a speedy exit from this hellish world. Are you up to the challenge?

All of the traditional Adventure elements are in ESCAPE FROM TRAAM, including two-world command input, game save feature, and an extensive vocabulary. Add to this the impact of a dynamic storyline that'll give even the most experienced Adventurer a real workout! ESCAPE FROM TRAAM — a fine addition to any Adventure collection.

ATARI 16K TAPE 050-0109 $24.95

PRICES SUBJECT TO CHANGE

ai Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.

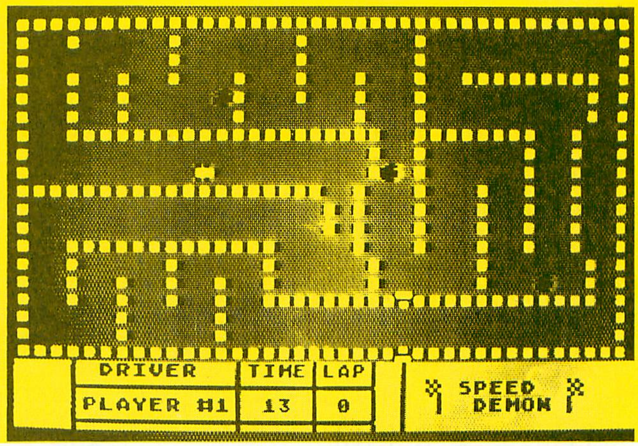
To order, see your local dealer. If he does not have the program, then call 1-800-327-7172 (orders only please) or write for our free catalog.

Published by ADVENTURE INTERNATIONAL
a subsidiary of Scott Adams, Inc.
BOX 3435 • LONGWOOD, FL 32750 • (305) 830-8194

SPEED DEMON

by JOHN MAGDZIARZ

A slick way to circumvent player/missile programming.



SPEED DEMON! is a one-player car-racing game using a joystick in Port One. You are given a high-performance stock car which leaks oil every now and then. At the start your car is warming up at the gate waiting for the count down. In the text window you see a prompt for the skill level you wish. There are two levels, and pressing '1' starts you at the beginners level. Pressing '2' starts you at the pro level, in which your car has a severe oil leak. Once the count reaches zero, "Yer Off!". Your objective is to lap the entire course three times in as little time as possible.

Avoid hitting the bales of hay that line the entire course, or the oil slicks, left behind by your car. These cause your car to spin out. You can only resume driving when your car has regained traction.

To restart the game, press [START] and get ready to burn rubber!!

```
 90 REM SPEED DEMON BY JOHN
     MAGDZIARZ 1982
100 GRAPHICS 1+16
110 POSITION 3,8:? #6;"JOHN
     MAGDZIARZ"
120 POSITION 6,10:? #6;"pre
     sents"
130 POSITION 4,12:? #6;"spe
     ed demon!"
140 FOR LOOP=1 TO 1000:NEXT
     LOOP
150 POSITION 3,16:? #6;"ple
     ase wait...":FOR T=1 TO
     600:NEXT T
160 GOTO 180
170 FOR LOOP=1 TO 10:NEXT L
     OOP:RETURN
180 GRAPHICS 0:POKE 559,0:R
     EM SHUT OFF SCREEN TEMP
     ORARILY
190 OPEN #1,4,0,"K:":REM OP
     EN KEYBOARD FOR DIRECT
     INPUT
200 A=PEEK(106):POKE 106,A-
     5:REM ROOM FOR NEW CHAR
     ACTER SET
210 CP=PEEK(106)+1:POKE 756
     ,CP:REM GIVE NEW LOCATI
     ON OF CHARACTER SET
220 CHAR=CP*256
230 FOR M=0 TO 1023
240 POKE CHAR+M,PEEK(57344+
     M):REM COPY CHARACTER S
     ET FROM ROM TO RAM
250 NEXT M
260 FOR NC=1 TO 9:READ OLD
270 DIF=(OLD+4+64)*8
280 FOR M=0 TO 7:READ LINE:
     REM POKE DATA FOR ALTER
     ED CHARACTERS SET
290 POKE CHAR+DIF+M,LINE:NE
     XT M
300 NEXT NC:REM DATA FOR NE
     W CHARATERS
310 DATA 1,136,136,255,255,
     255,255,136,136
320 DATA 2,60,60,190,190,60
     ,60,190,190
330 DATA 3,34,34,191,191,19
     1,191,34,34
340 DATA 4,190,190,60,60,19
     0,190,60,60
350 DATA 9,0,0,20,20,20,20,
     0,0
360 DATA 10,204,204,51,51,2
     04,204,51,51
370 DATA 11,255,85,255,170,
     60,20,60,0
380 DATA 12,0,60,20,60,170,
     255,85,255
390 DATA 13,8,42,168,42,170
     ,170,168,40
400 GRAPHICS 0:POKE 756,CP:
     POKE 559,0
410 DL=PEEK(560)+PEEK(561)*
     256:REM ALTER DISLPLAY
     LIST
420 POKE DL+3,68
430 FOR I=6 TO 23:POKE DL+I
     ,4:NEXT I
440 POKE DL+28,4
450 FOR I=0 TO 4:READ COL:R
     EM SET SCREEN COLORS, R
     EGISTERS 0-4
460 POKE 708+I,COL:NEXT I
470 DATA 26,0,198,72,5
480 ? "}":DIM HOR$(40),VERT
     $(30):DIM CR1$(4)
490 POKE 559,0:? "}":RESTOR
     E 550:S1=7
500 FOR I2=1 TO 39:HOR$(I2,
     I2)=CHR$(13):NEXT I2
510 FOR I=1 TO 30 STEP 3:VE
     RT$(I,I)=CHR$(13):VERT$
     (I+1,I+1)=CHR$(29):VERT
     $(I+2,I+2)=CHR$(30):NEX
     T I
520 POKE 82,0
530 REM DRAW PLAYFIELD
540 FOR LOOP=1 TO 8:READ I,
```

APOLOGY

The game PAC-INVADERS, published in this department in Issue #3, was the original work of Sheldon Leemon and not Vince Scott as we indicated. Mr. Leemon's version, titled, Outer Space Attack, appeared in Softside Magazine, March 1982.

Mr. Leemon is the author of INSTEDIT, a character editor program from APEX, as well as an upcoming GTIA tutorial from Educational Software.

Softside is a monthly magazine featuring information and programs for users of Apple, TRS-80 and ATARI computers.

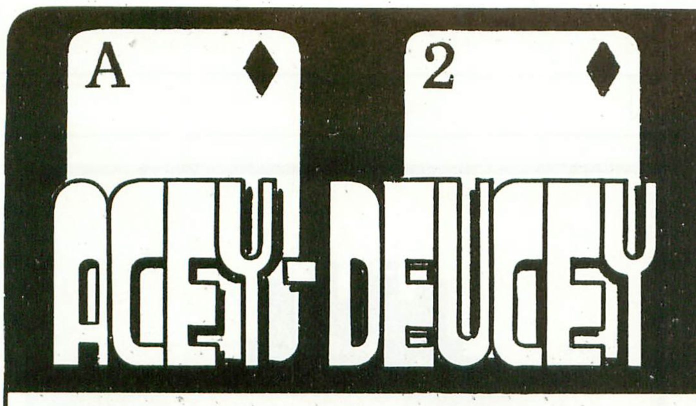
We apologize to all concerned for any misrepresentation or confusion.

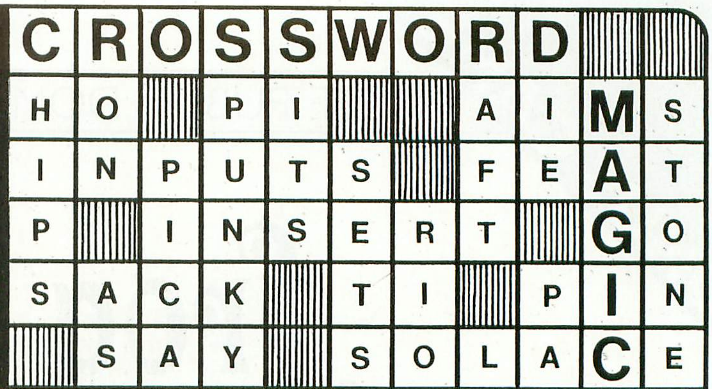
continued on next page

```
    J,K,L:POSITION I,J:? HO
    R$(K,L):NEXT LOOP
550 DATA 0,0,1,39,28,3,1,9,
    3,6,1,20,25,6,1,8,0,9,1
    ,19,3,12,1,13,15,15,1,2
    2,0,18,1,39
560 FOR LOOP=1 TO 27:READ I
    ,J,K,L:POSITION I,J:? V
    ERT$(K,L):NEXT LOOP
570 REM DATA FOR PLAYFIELD
580 DATA 0,0,1,30,0,10,1,22
    ,3,1,1,1,3,3,1,12,6,3,1
    ,9,9,1,1,9
590 DATA 13,3,1,9,17,1,1,12
    ,21,3,1,9,25,1,1,28,25,
    11,1,6,29,9,1,18
600 DATA 32,7,1,18,36,4,1,6
    ,35,6,1,21,36,13,1,6,38
    ,1,1,30
610 DATA 38,11,1,24,19,10,1
    ,6,15,13,1,6,12,14,1,12
    ,9,13,1,9
620 DATA 6,14,1,12,3,13,1,9
    ,22,7,1,6,21,9,1,12,22,
    13,1,6
630 POSITION 3,19:? "| DRIV
    ER  |TIME|LAP|"
640 POSITION 3,20:? "
    "
650 POSITION 3,21:? "|PLAYE
    R #1| 0   | 0 |"
660 POSITION 3,22:? "
    "
670 POSITION 24,19:? CHR$(1
    24);CHR$(29);CHR$(30);C
    HR$(124);CHR$(29);CHR$(
    30);CHR$(124);CHR$(29);
    CHR$(30);CHR$(124)
680 POSITION 26,20:? CHR$(1
    4);" SPEED  ";CHR$(14)
690 POSITION 26,21:? CHR$(2
    );"  DEMON ";CHR$(22)
700 POSITION 24,15:? CHR$(1
    5):POSITION 24,18:? CHR
    $(16)
710 FOR I=1 TO 4:CR1$(I,I)=
    CHR$(I+132):NEXT I:LAP1
    =-1:OK=0:AG=0:TIME1=0:L
    P=1
720 POSITION 23,16:? CR1$(1
    ,1):X1=23:Y1=16
730 POKE 559,34:POKE 752,1:
    POKE 53279,0
740 SOUND 0,170,4,4
750 POSITION 26,19:? "WHAT
    LEVEL?";:GET #1,LEV
760 IF LEV=49 THEN SC=45:GO
    TO 790
770 SC=10
780 IF LEV<>1 AND LEV<>50 T
    HEN 750
790 POSITION 26,19:? "
           "
800 REM
810 GOSUB 1280
820 FOR G=5 TO 1 STEP -1:PO
    SITION 23,21:? G:SOUND
    0,40,10,4:FOR T=1 TO 10
    0:NEXT T:SOUND 0,0,0,0:
    GOSUB 170:NEXT G
830 POSITION 23,21:? " "
840 SOUND 0,85,6,5
850 POKE 20,0:POKE 19,0:REM
     RESET REAL TIME CLOCK
860 REM START OF GAME ROUTI
    NE
870 POSITION 15,21:? PEEK(1
    9):IF PEEK(19)>98 THEN
    480
880 IF PEEK(53279)=6 THEN 4
    90
890 IF OK1=1 THEN 870
900 IF AG=1 THEN 1200
910 S=STICK(0):IF S=15 OR S
    =10 OR S=6 OR S=5 OR S=
    9 THEN S=S1
920 S1=S
930 REM READ JOYSTICK
940 ON S-4 GOTO 960,970,980
    ,0,990,1000,1010,0,1030
    ,1040
950 GOTO 910
960 GOTO 870
970 GOTO 870
980 X1L=X1+1:Y1L=Y1:P1=1:GO
    TO 1070
990 GOTO 870
1000 GOTO 870
1010 X1L=X1-1:Y1L=Y1:P1=3:IF
     (X1=24 OR X1=25) AND (
     Y1=16 OR Y1=17) THEN 87
     0
1020 GOTO 1070
1030 X1L=X1:Y1L=Y1+1:P1=4:GO
     TO 1050
1040 X1L=X1:Y1L=Y1-1:P1=2
1050 IF (S=14 OR S=13) AND (
     (X1=24) AND (Y1=16 OR Y
     =17)) THEN S=7:GOTO 940
1060 REM CHECK COLLISIONS
1070 LOCATE X1L,Y1L,Z1:POSIT
     ION X1L,Y1L:PUT #6,Z1:L
     X=X1:LY=Y1
1080 IF Z1=17 THEN POSITION
     X1,Y1:? " ":X1=X1L:Y1=Y
     1L
1090 IF Z1<>32 THEN 1170
1100 REM MOVE CAR
1110 POSITION X1,Y1:? " ":PO
     SITION X1L,Y1L:? CR1$(P
     1,P1):IF P1<>LP THEN GO
     SUB 1350
1120 LP=P1:X1=X1L:Y1=Y1L
1130 IF X1=24 AND (Y1=16 OR
     Y1=17) THEN LAP1=LAP1+1
     :SOUND 2,50,12,10:GOSUB
     170:POSITION 20,21:? L
     AP1
1140 SOUND 2,0,0,0
1150 IF LAP1=3 THEN POSITION
      X1,Y1:? " ":SOUND 0,0,
     0,0:GOSUB 1290:GOTO 640
1160 GOSUB 1230:GOTO 870
1170 K=INT(RND(0)*3+1)
1180 FOR I=14 TO 0 STEP -2:S
     OUND 1,100,0,I:GOSUB 17
     0:NEXT I
1190 FOR G=1 TO K:FOR I=1 TO
      4:POSITION X1,Y1:? CR1
     $(I,I):AG=1:GOTO 870
1200 NEXT I:NEXT G:AG=0:POSI
     TION X1,Y1:? CR1$(P1,P1
     )
1210 GOTO 910
1220 REM OIL SLICK PLACEMENT
      ROUTINE
1230 I=INT(RND(0)*SC+1)
1240 IF I=7 THEN GOSUB 1260
1250 RETURN
1260 LOCATE LX,LY,Z:IF Z=32
     THEN POSITION LX,LY:? C
     HR$(17):RETURN
1270 POSITION LX,LY:PUT #6,Z
     :RETURN
1280 RETURN
1290 POSITION 26,20:? "GAME
     OVER!":POSITION 25,21:?
     "           "
1300 POKE 53279,0
1310 S2=PEEK(53279):IF S2<>6
      THEN 1310
1320 GOTO 490
1330 RETURN
1340 REM CAR TURNING SOUND
1350 SOUND 2,5,0,8:SOUND 3,2
     0,10,8:FOR X=1 TO 10:NE
     XT X:SOUND 2,0,0,0:SOUN
     D 3,0,0,0
1360 RETURN
```

Variable checksum = 1029818

Line num range	Code	Length
90 - 200	TB	539
210 - 320	KN	432
330 - 440	MV	425
450 - 550	OX	545
560 - 660	RE	543
670 - 730	PS	543
740 - 850	AF	516
860 - 970	WD	406
980 - 1090	BQ	464
1100 - 1190	PL	524
1200 - 1310	FO	353
1320 - 1360	HZ	211





By LARRY SHERMAN

Two variations of an exciting new game with detailed graphics. Challenge your friends or the computer to see who can win the pot or break the bank.

ONLY
$29.95
PLEASE SPECIFY
ATARI 400/800, 16K
DISK OR CASSETTE
APPLE II/II+
48K, 3.3 DOS

By LARRY SHERMAN

Supply words using ANY subject and watch CROSSWORD MAGIC interconnect them with lightening speed. Print professional quality puzzles with any one of 24 graphics printers or play them on the screen.

ONLY
$49.95
PLEASE SPECIFY
ATARI 800, 40K
DISK ONLY
APPLE II/II+
48K, 3.3 DOS

At computer stores or:

L & S COMPUTERWARE
1589 FRASER DRIVE
SUNNYVALE, CA. 94087
(408) 738-3416

VISA/MC...$2.00 shipping
Ca. residents add 6% tax

Artwork by Dick Davies...Apple is a trademark of Apple Computer Inc...Atari is a trademark of Atari Inc.

RAM

For ATARI

48K RAM BOARD FOR THE 400
with Lifetime Warranty

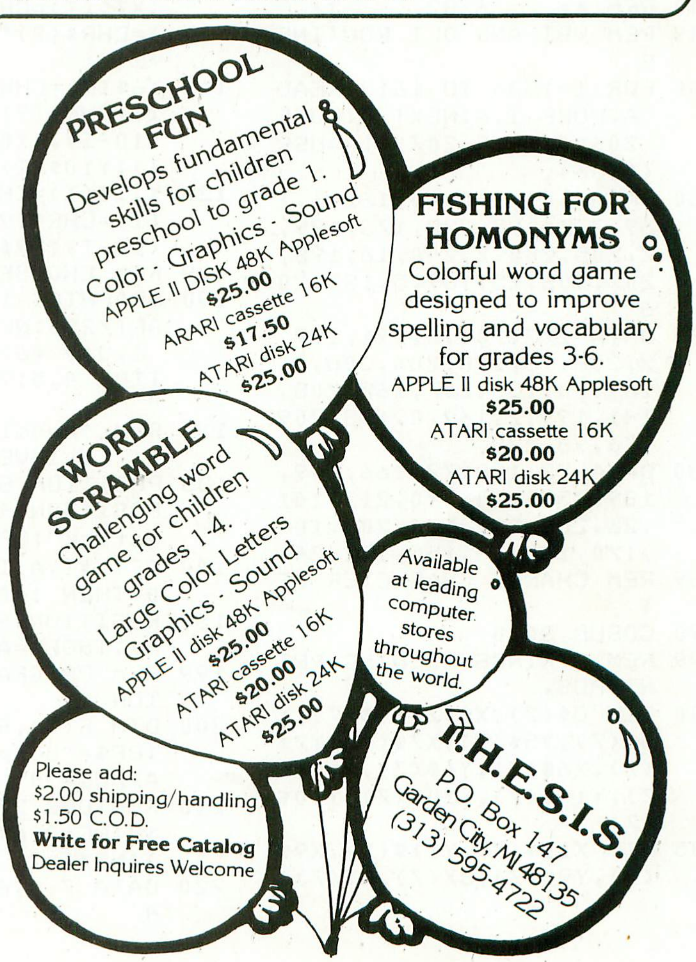
- Highest quality available
- Reduces power consumption
- Reduces heat

48K Board	(400)	**$175**
32K Board	(400/800)	**$ 90**
16K Board	(800)	**$ 60**

FREE SHIPPING ANYWHERE IN U.S.A.

INTEC PERIPHERALS CORP
906 E. Highland Ave.
San Bernardino, CA 92404
VISA (714) 881-1533 MasterCard

ATARI, 400, 800 are Trademarks of ATARI, Inc.



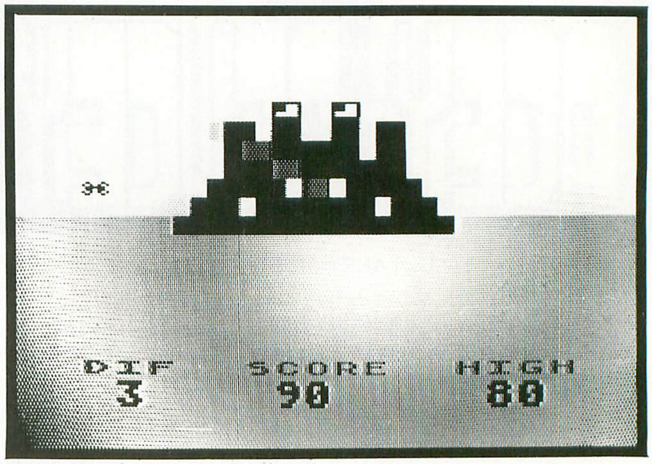
BONUS GAME

IN THE PUBLIC DOMAIN

frog

by STAN OCKERS

Lick up the blinking bugs.



You may be a Prince at heart, but in this game you are only a hungry Frog, sitting in the middle of a Dismal Swamp. Swarms of blinking bugs fly by. Use the joystick and fire-button to stick out your tongue and catch them. But don't let them sting you, their bite is fatal! Your score is related to the number of bugs you eat before getting stung.

There are two levels of difficulty in this game, and paddle controllers will work instead of the joystick. There are several special touches too, including a Vertical Blank Interrupt routine that makes the bugs blink on and off.

```
20 REM **     FROG      **
30 REM ** STAN OCKERS **
40 REM **     6/82      **
49 REM VBI AND DLI ROUTINE
   S
50 FOR I=1536 TO 1613:READ
    A:POKE I,A:NEXT I:POKE
    206,4:POKE 207,4:A=USR
   (1536)
60 DATA 104,160,10,162,6,1
   69,7,76,92,228,173,199,
   2,205,200,2,240,16,198,
   207,208,25,169,5,133,20
   6
70 DATA 173,200,2,141,199,
   2,208,13,198,206,208,9,
   169,10,133,207,165,208,
   141,199,2,169,0,133,209
   ,76,98,228
80 DATA 72,138,72,166,209,
   189,75,6,141,10,212,141
   ,26,208,232,134,209,104
   ,170,104,64,155,152,24
89 REM CHANGE CHARACTER SE
   T
90 GOSUB 3000
99 REM STRINGS HOLDING PRI
   NT POS.
100 DIM C$(7),X$(7),Y$(7),X
   5$(7),Y5$(7),X7$(7),Y7$
   (7),X6$(7),Y6$(7),X14$(
   7),Y14$(7),X10$(7),Y10$
   (7)
105 DIM X11$(7),Y11$(7),X9$
   (7),Y9$(7),BX(7),BY(7)
110 FOR I=1 TO 7:X5$(I)=CHR
   $(8+I):Y5$(I)=CHR$(8+I)
   :X7$(I)=CHR$(8+I):Y7$(I
   )=CHR$(9):X6$(I)=CHR$(8
   +I)
115 Y6$(I)=CHR$(10-I):X14$(
   I)=CHR$(9):Y14$(I)=CHR$
   (10-I):X10$(I)=CHR$(10-
   I):Y10$(I)=CHR$(10-I)
120 X11$(I)=CHR$(10-I):Y11$
   (I)=CHR$(9):X9$(I)=CHR$
   (10-I):Y9$(I)=CHR$(8+I)
129 REM CHOOSE INPUT DEVICE
130 GRAPHICS 18:POKE 756,ST
   ART/256:NEXT I:POSITION
    6,3:? #6;"choose":POSI
   TION 4,5:? #6;"1 paddle
   "
131 REM 'paddle' & 'joystic
   k' IN INVERSE CHAR.
135 POSITION 8,6:? #6;"or":
   POSITION 4,7:? #6;"2 jo
   ystick":OPEN #1,4,0,"K"
140 GET #1,A:IF A<49 OR A>5
   0 THEN 140
150 POSITION 9,9:? #6;CHR$(
   A):INDEV=A-48
199 REM PROGRAM INITIALIZAT
   ION
200 DIM F(7),ST$(10),BP$(4)
   :BP$="burp":MAXWAIT=6:M
   AXBUG=30
210 RESTORE 220:FOR I=1 TO
   10:READ A:ST$(I)=CHR$(A
   ):NEXT I
220 DATA 7,5,6,0,1,3,2,0,0,
   4
280 POKE 208,37:S=1:REM 208
    IS BUG COLOR
285 REM INITIAL DIFFICULTY
   SELECTION
286 GRAPHICS 1:POKE 708,229
   :POKE 712,155:GOSUB 700
   :GOSUB 400
289 REM RESTART GAME
290 FOR I=1 TO 7:F(I)=0:NEX
   T I:SCORE=0:NUMBUG=0:BO
   NUS=50
300 GRAPHICS 1:POKE 756,STA
   RT/256:GOSUB 700:X$=X5$
   :Y$=Y5$:POKE 708,229:PO
   KE 709,54:POKE 710,227:
   POKE 712,155
305 REM DRAW FROG
306 GOSUB 2000:POSITION 8,5
   :GOSUB 905
308 REM MAIN LOOP
309 IF INDEV=2 THEN GOSUB 5
   00:GOTO 320
310 S=INT((PADDLE(0)-45)/20
   ):IF S>7 THEN S=7
312 IF S<1 THEN S=1
314 S=8-S
320 POSITION 8,5:GOSUB 900+
   S
325 IF INDEV=2 AND STRIG(0)
   =0 THEN GOSUB 1000
330 IF INDEV=1 AND PTRIG(0)
   =0 THEN GOSUB 1000
339 I=0:IF WAIT>1 THEN WAIT
   =WAIT-1:GOTO 350
```

```
340 WAIT=MAXWAIT
341 I=I+1:IF I=8 THEN 350
342 IF F(I)=0 THEN 341
345 IF F(I)=2 THEN 347
346 RESTORE 800+I:READ DX,D
    Y,BX,BY:POSITION BX,BY:
    ? #6;CHR$(129):F(I)=2:B
    X(I)=BX:BY(I)=BY:GOTO 3
    41
347 RESTORE 800+I:READ DX,D
    Y:POSITION BX(I),BY(I):
    ? #6;" ":BX(I)=BX(I)+DX
    :BY(I)=BY(I)+DY
348 POSITION BX(I),BY(I):?
    #6;CHR$(129):IF BX(I)>5
     AND BX(I)<13 AND BY(I)
    >5 AND BY(I)<11 THEN 37
    0
349 GOTO 341
350 IF BUG>1 THEN BUG=BUG-1
    :GOTO 360
352 BUG=MAXBUG:IF NUMBUG=7
    THEN 360
354 I=INT(RND(0)*7)+1:IF F(
    I)>0 THEN 354
356 F(I)=1:NUMBUG=NUMBUG+1
360 GOTO 309
369 REM BUG GOT TO FROG
370 POSITION 8,5:? #6;". ."
    :FOR I=1 TO 6:POKE 708,
    38:SOUND 0,100,12,10:FO
    R J=1 TO 50:NEXT J
375 POKE 708,36:SOUND 0,150
    ,12,10:FOR J=1 TO 10:NE
    XT J:NEXT I:SOUND 0,0,0
    ,0
398 GOSUB 400:GOTO 290
399 REM PROMPT FOR NEW GAME
400 POSITION 4,13:? #6;"PRE
    SS start":POSITION 6,14
    :? #6;"TO PLAY":REM USE
     IN INVERSE CHAR. IN TH
    ESE TWO LINES
401 POSITION 4,16:? #6;"PRE
    SS select":POSITION 3,1
    7:? #6;"FOR DIFFICULTY"
402 POKE 53279,8:K=PEEK(532
    79):IF K=6 THEN 420
404 IF K=5 THEN MAXWAIT=MAX
    WAIT-1:MAXBUG=MAXBUG-5:
    IF MAXBUG<5 THEN MAXBUG
    =30:MAXWAIT=6
406 POKE 656,0:POKE 657,3:?
     7-MAXWAIT
408 FOR I=1 TO 100:NEXT I
410 GOTO 402
420 POSITION 5,16:? #6;"
             ":POSITION 3,17:
    ? #6;"                 "
430 IF SCORE>MAXSCORE THEN
    MAXSCORE=SCORE
440 RETURN
499 REM JOYSTICK SUBROUTINE
500 S0=STICK(0):IF S0=15 OR
     S0=13 THEN RETURN
510 S=ASC(ST$(S0-4)):RETURN
599 REM BURP SUBROUTINE
600 POKE 77,0:X=1:Y=0:FOR K
    =1 TO 4:X=X+1:Y=Y+1:POS
    ITION X,Y:? #6;BP$(K,K)
    :NEXT K
620 FOR K=250 TO 150 STEP -
    3:SOUND 0,K,2,10:NEXT K
630 X=1:Y=0:FOR K=1 TO 4:X=
    X+1:Y=Y+1:POSITION X,Y:
    ? #6;" ":NEXT K:RETURN
699 REM CHANGE DISPLAY LIST
700 A=PEEK(560)+256*PEEK(56
    1):POKE A+3,198:POKE A+
    15,134:POKE A+24,134
710 IF PEEK(A)<>66 THEN A=A
    +1:GOTO 710
720 POKE A,71:POKE A+3,7:PO
    KE A+4,65:POKE A+5,PEEK
    (A+7):POKE A+6,PEEK(A+8
    )
722 POKE 512,54:POKE 513,6:
    POKE 54286,192
730 POKE 656,0:POKE 657,3:?
     7-MAXWAIT:POKE 656,0:P
    OKE 657,15:? MAXSCORE
740 POSITION 2,19:? #6;"dif
      score  high":RETURN
800 REM X-Y INCREMENTS & ST
    ARTING POS.
801 DATA 1,-1,0,18
802 DATA 1,0,0,9
803 DATA 1,1,0,0
804 DATA 0,1,9,0
805 DATA -1,1,18,0
806 DATA -1,0,18,9
807 DATA -1,-1,18,18
900 REM PRINT EYES ON FROG
901 X$=X9$:Y$=Y9$:? #6;") )
    ":RETURN
902 X$=X11$:Y$=Y11$:? #6;"(
     (":RETURN
903 X$=X10$:Y$=Y10$:? #6;"#
     #":RETURN
904 X$=X14$:Y$=Y14$:? #6;"$
     $":RETURN
905 X$=X6$:Y$=Y6$:? #6;"% %
    ":RETURN
906 X$=X7$:Y$=Y7$:? #6;"& &
    ":RETURN
907 X$=X5$:Y$=Y5$:? #6;"' '
    ":RETURN
999 REM TONGUE ROUTINE
1000 I=0
1001 I=I+1:IF I=8 THEN 1010
1002 POSITION ASC(X$(I)),ASC
     (Y$(I)):GET #6,A:C$(I,I
     )=CHR$(A)
1004 POSITION ASC(X$(I)),ASC
     (Y$(I)):? #6;CHR$(13):S
     OUND 0,7-I,8,8:IF A=129
      THEN GOTO 1050
1005 GOTO 1001
1010 I=I-1:IF I=0 THEN 1020
1012 POSITION ASC(X$(I)),ASC
     (Y$(I)):? #6;C$(I,I):SO
     UND 0,7-I,8,8:GOTO 1010
1020 IF BURP=1 THEN GOSUB 60
     0:BURP=0
1022 SOUND 0,0,0,0:RETURN
1049 REM GOT A BUG
1050 F(S)=0
1060 SCORE=SCORE+10*(7-MAXWA
     IT):POKE 656,0:POKE 657
     ,8:? SCORE;"  ":NUMBUG=
     NUMBUG-1
1070 IF SCORE>BONUS THEN MAX
     WAIT=MAXWAIT-1:MAXBUG=M
     AXBUG-5:BONUS=BONUS+100
     *(7-MAXWAIT):BURP=1
1072 IF MAXBUG<5 THEN MAXBUG
     =5:MAXWAIT=1
1080 POKE 656,0:POKE 657,3:?
      7-MAXWAIT
1090 C$(I,I)=" ":I=I+1:GOTO
     1010
1999 REM PRINT FROG SUBROUTI
     NE
2000 POSITION 6,6:? #6;"+*-
     -+*":POSITION 6,7:? #6;
     "+*---+*":POSITION 6,8:
     ? #6;"+------*"
2010 POSITION 6,9:? #6;"--*
     +--":POSITION 5,10:? #6
     ;"+-+---*-*"
2020 POSITION 5,11:? #6;"---
     -------":RETURN :REM INV
     ERSE CHAR. IN THIS LINE
2999 REM CHANGE CHAR. SET SU
     BR.
3000 DIM ZZ$(32):RESTORE 301
     0:FOR I=1 TO 32:READ A:
     ZZ$(I)=CHR$(A):NEXT I
3010 DATA 104,104,133,204,10
     4,133,203,104,133,206,1
     04,133,205,162,4,160,0
3020 DATA 177,203,145,205,13
     6,208,249,230,204,230,2
     06,202,208,240,96
3030 POKE 106,PEEK(106)-5:GR
     APHICS 0:START=(PEEK(10
     6)+1)*256:POKE 756,STAR
     T/256:POKE 752,1:? "INI
     TIALIZING ...."
3040 A=USR(ADR(ZZ$),57344,ST
     ART):RESTORE 3100:FOR I
     =START+8 TO START+119:R
     EAD A:POKE I,A:NEXT I
```

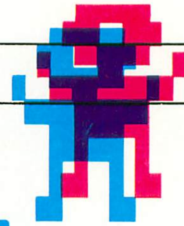
continued on next page

```
3050 RETURN
3100 DATA 0,34,85,62,54,85,3
     4,0,0,102,102,102,0,0,0
     ,0,255,241,241,241,129,
     129,255,255
3110 DATA 255,189,189,189,12
     9,129,255,255,255,143,1
     43,143,129,129,255,255,
     255,129,143,143,143,129
     ,255,255
3120 DATA 255,129,129,143,14
     3,143,255,255,255,129,2
     41,241,241,129,255,255,
     255,129,129,241,241,241
     ,255,255
3130 DATA 240,240,240,240,24
     0,240,240,240,15,15,15,
     15,15,15,15,15,0,0,0,0,
     0,24,24,48
3140 DATA 255,255,255,255,25
     5,255,255,255,255,145,1
     45,255,145,145,255,255
```

Variable checksum = 1738191

Line num range	Code	Length
20 - 90	PQ	511
99 - 120	XE	554
129 - 210	MM	545
220 - 309	PL	509
310 - 346	ZC	473
347 - 370	ZX	516
375 - 404	PR	520
406 - 620	RH	502
630 - 730	JI	520
740 - 902	AD	285
903 - 1010	SR	446
1012 - 1999	TI	515
2000 - 3020	LK	506
3030 - 3130	LW	577
3140 - 3140	FM	69

FROG was first published in the A.C.E. newsletter of Eugene, Oregon. We would like to thank Stan Ockers for an excellent program and Mike Dunn, editor of the A.C.E. newsletter for permission to re-print.



TUNING YOUR ATARI

by LINDA M. SCHREIBER

There's something about music that fascinates kids. Give them a small piano, drums, harmonica, and they will sit for hours creating their own melodies. A few years ago there was a toy piano on the market that contained a tape recorder. This was a big hit with my daughter. Now, she could not only make her own music, but listen to it afterward.

TUNING YOUR ATARI uses this idea. It is a musical game for children. Type it in and run it, and you will see a simple menu. Choice #3 demonstrates the program. Choice #1 allows you to compose a tune, and Choice #2 will play it back. The tones appear to be made by little figures jumping on a bellows.

Above each figure is the letter name of the tone which that bellows will produce. To operate the bellows, press 1, then press any number from 1–8 on the keyboard. Key one corresponds to the low C; eight to high C. When a number is pressed, the character will jump down on the bellows, flapping his arms as the bellows is compressed. Once the tone is played, he bounces back up to his original position. The program can hold up to 100 notes. If your melody is less than 100 notes, press the escape key and the menu will reappear on the screen. Press #2 to hear your melody.

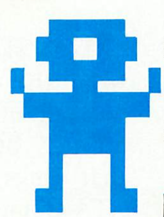
Young children will enjoy this program just to see the characters jump up and down while they are playing the tunes. Slightly older children will enjoy listening to the tunes that they have created. The letters above the characters do not attract attention, but are a subtle reminder of the names of the notes. After a while, children will begin to associate the letters with the tones of the character. Don't be surprised if you hear your child singing 'A-G-F-G-A-A-A'!

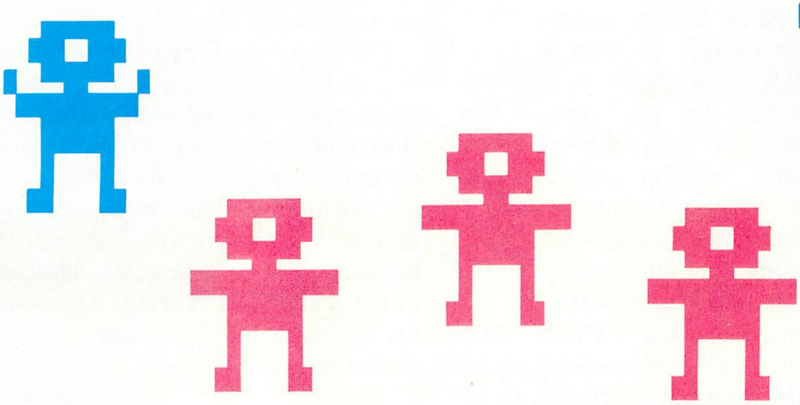
Once again, in this program, we will move the character set out of ROM and into RAM so that we can change some of the characters. In line 70, P1$ should equal h, reverse quotation marks, control D, reverse space, control comma, reverse 1, reverse M, reverse control Q. The characters from K to r are all in reverse. The last character in the string is control period. This string is the machine language subroutine that moves the characters.

Variables Used

P1$	—machine language subroutine
M$	—string holds the melody played
A	—location of the new character set. This value is POKEd into 756 to change to the new character set.
TONE	—line number that starts the tone for the key pressed.
WAIT	—line number for the timing routine.
Q	—no function
CHBS	—first decimal location of the new character set.
X	—no function — used in FOR . . . NEXT loops.
C	—used in READ for new character set, used for value of key pressed, and for position of character.
K	—counter for the note being entered or played.
T	—value of the tone to be played.
TL	—value used in timing loop.
ROUTINE	—the line number that the program goes to when entering the melody, or playing one back.

```
10 REM TUNING YOUR ATARI
20 REM BY L.M.SCHREIBER
30 REM FOR ANTIC OCTOBER 1
982
40 DIM M$(101),P1$(20)
50 GRAPHICS 18:POKE 711,PE
EK(710):POKE 710,100
60 A=PEEK(106)-8:POKE 204,
A:POKE 206,224:REM STOR
E THE BEGINNING OF NEW
& OLD CHARACTER SETS
70 P1$="h" 1MKHPyfNfLJP
r`":TONE=430:WAIT=500:R
EM P1$ IS A MACHINE LAN
GUAGE SUBROUTINE TO MOV
E THE CHARACTER SET
80 Q=USR(ADR(P1$)):CHBS=A*
256:POKE 756,A:REM CHAN
GE TO THE NEW CHARACTER
 SET
90 FOR X=CHBS+8 TO CHBS+71
:READ C:POKE X,C:NEXT X
:REM CHANGE THE CHARACT
ERS FROM ! TO $
100 DATA 0,254,124,254,124,
254,124,254,108,0,254,2
54,124,254,124,254,40,1
08,0,254,254,254,124,25
4
110 DATA 186,40,108,0,254,2
54,254,254,56,108,56,16
,254,56,40,108,0,56,108
,56,146,124,56,40
120 DATA 0,0,56,108,56,16,2
54,56,0,0,0,56,108,56,1
6,124
130 OPEN #2,4,0,"K:":REM OP
EN THE KEYBOARD FOR REA
D
140 POSITION 2,9:? #6;"! !
! ! ! ! ! !":POSITION 2
,8:? #6;"% % % % % % %
%":REM THE ! AND % ARE
THE NEW CHARACTERS
150 POSITION 2,6:? #6;"c d
e f g a b c":REM PLACE
THE TONE NAMES
160 K=0:POKE 710,100:REM RE
STORE THE MENU
170 POSITION 2,0:? #6;"1. P
LAY KEYBOARD"
180 POSITION 2,2:? #6;"2. R
EPEAT MELODY"
190 POSITION 2,4:? #6;"3. P
LAY EXAMPLE"
200 GET #2,C:POKE 710,0:REM
 GET THE KEY PRESSED-RE
MOVE THE MENU
210 IF C>127 THEN C=C-128:P
OKE 694,0:REM INVERSE F
LAG IS ON RESET IT TO N
ORMAL
220 IF C<49 OR C>52 THEN 16
0:REM NOT A NUMBER FROM
 1 TO 4
230 C=C-48:REM GET THEN NUM
BER
240 ON C GOTO 250,540,520,5
60
250 M$="":REM REMOVE CONTEN
TS OF THE STRING
260 ROUTINE=260:K=K+1:IF K=
101 THEN 160:REM ONLY A
CCEPT 100 NOTES
280 GET #2,C:REM GET THE KE
Y PRESSED-RETURN TO MEN
U ON ESCAPE KEY
290 IF C>127 THEN C=C-128:P
OKE 694,0:REM INVERSE F
LAG IS ON RESET IT TO N
ORMAL
300 IF C<49 OR C>56 THEN 16
0:REM NOT A NUMBER FROM
 1 TO 8
310 C=C-48:M$(K,K)=STR$(C):
REM GET THEN NUMBER-PUT
 IT IN THE STRING
320 C=C*2:REM OFFSET IT FOR
 THE PROPER POSITION
330 ON C/2 GOSUB 350,360,37
0,380,390,400,410,420
340 GOTO ROUTINE
350 T=121:GOTO TONE:REM 'C'
360 T=108:GOTO TONE:REM 'D'
370 T=96:GOTO TONE:REM 'E'
380 T=91:GOTO TONE:REM 'F'
390 T=81:GOTO TONE:REM 'G'
400 T=72:GOTO TONE:REM 'A'
410 T=64:GOTO TONE:REM 'B'
420 T=60:REM 'C'
425 REM LINES 430-450 MAKE
THE CHARACTER APPEAR TO
 PUSH DOWN ON THE BELLO
W AND MAKE THE TONE
430 TL=10:POSITION C,8:? #6
;CHR$(134):POSITION C,9
:? #6;CHR$(130):SOUND 0
,T,10,6:GOSUB WAIT
440 POSITION C,8:? #6;CHR$(
135):POSITION C,9:? #6;
CHR$(131):SOUND 0,T,10,
8:GOSUB WAIT
450 POSITION C,8:? #6;CHR$(
136):POSITION C,9:? #6;
CHR$(132):GOSUB WAIT
460 SOUND 0,T,10,10
470 POSITION C,8:? #6;CHR$(
135):POSITION C,9:? #6;
CHR$(131):SOUND 0,T,10,
8:GOSUB WAIT
475 REM LINES 470-490 RETUR
N THE CHARACTER AND BEL
LOW TO THE CORRECT POSI
TION
480 POSITION C,8:? #6;CHR$(
134):POSITION C,9:? #6;
CHR$(162):SOUND 0,T,10,
6:GOSUB WAIT
490 POSITION C,8:? #6;"%":P
OSITION C,9:? #6;"!":SO
UND 0,0,0,0:RETURN
500 FOR X=1 TO TL:NEXT X:RE
TURN :REM TIMING LOOP
510 REM PLAY A SAMPLE TUNE
520 M$="11556654433221"
530 REM ROUTINE TO PLAY BAC
K THE MELODY ENTERED
540 ROUTINE=540:K=K+1:IF K<
=LEN(M$) THEN C=VAL(M$(
K,K)):GOTO 320:REM KEEP
 PLAYING UNTIL THE END
OF THE STRING
550 M$(K,K)="0":C=VAL(M$(K,
K)):M$=M$(1,K-1):GOTO 1
60
560 CLOSE #2:END
```





Variable checksum = 225145

Line num range	Code	Length
10 - 90	QA	529
100 - 150	SI	504
160 - 250	LD	539
260 - 350	YQ	521
360 - 450	SJ	554
460 - 530	TI	537
540 - 560	QP	187

S.A.M.

Don't Ask Computer Software
2265 Westwood Blvd., #B-150
Los Angeles, CA 90064
(213) 397-8811
$59.95 diskette; 32K
Reviewed by Jerry White

Computer speech synthesis has just been revolutionized by a product called SAM, the Software Automatic Mouth. Unlike all the other systems, SAM requires no special hardware. That's right, SAM provides the highest-quality computerized speech currently available for ATARI computers, and does it with software only. All you need is an ATARI 400 or 800 with at least 32K RAM and one disk drive.

Also unlike all the others, SAM is inexpensive. The SAM software on diskette including demonstration programs and a well-written 38 page manual, costs only $59.95. That's a small fraction of the cost of most hardware speech systems currently available.

SAM, when combined with the companion program RECITER, will pronounce nearly 90% of the English language properly. Using RECITER, speech from BASIC is as easy as placing your text into a string called SAM$, and issuing the command A=USR (8199).

If you don't use the RECITER program, you save 6K of RAM, and you can still make SAM pronounce all words properly by using the phonetics system. You can tell SAM to use any of eight different stress factors on each syllable, no more monotone monotony. You are also in control of the pitch of SAM's voice, as well as the rate of speed at which he speaks.

Of course it takes a bit more work to master the phonetic systems, but the manual provides about 1500 example words in normal and phonetic spellings. If you're having any problems, DON'T ASK support is only a telephone call away.

The only drawback other than the RAM requirements is that SAM will blank the screen when he speaks. You can use SAM with the screen left on, but this will cause distortion in SAM's voice. Since computer speech is generally used for short phrases, the blank screen is a small price to pay in most cases.

Adventure type games and educational software are obvious applications for computer speech. I put SAM to work answering my telephone when I'm not available. In order to provide the highest quality speech for my 20 second recording, I used phonetics. The numbers you see in the phonetically spelled words in my sample program, indicate the stress factor. The number 1 would indicate a very emotional stress on the preceeding syllable, while 8 would indicate an extreme pitch-dropping stress. Most punctuation is used to insert a pause. The period will also cause the pitch to fall while the question mark will cause the pitch to rise.

VOICE BOX

The Alien Group
27 West 23rd Street
New York, N.Y. 10010
$169.00 includes diskette
Reviewed by Benton J. Elkins

"ATARI says its the first word," the ad from the Alien Group began. That hooked me enough to read that the VOICE BOX is a speech synthesizer that plugs into the serial port of the ATARI and routes all speech sounds to the television speaker. Synthesis is accomplished with phoneme analysis. I called for information, but Bob Ezzard, an Alien, convinced me to accept COD shipment on 10 day approval.

In a few weeks the VOICE BOX arrived. The disk loaded without problem, and the program instructed me to connect the unit. Immediately a voice came out of the TV, "Please teach me to speak." A backup message of the words was printed on the screen. I was amazed that it worked the first time I tried it!

I entered some words on the keyboard, pressed return, and was amused at the literalness of the pronunciation. I tried some phonetic variations, by changing the spelling of the words. The screen displays the controls used for examining the vowels, consonants and defined words in the program's dictionary files. Everything I tried worked. All that was missing was syllable emphasis.

The VOICE BOX disk contains three dictionaries, 16K and 32K versions of the programs in BASIC, a stripped down list version of the program to be used in adding vocal capability to other programs, and an object code of the phoneme pronouncer.

The BASIC program incorporates a random sentence recitation program using a standard sentence vocabulary, or it invites on input another learned vocabulary which may be saved later. New words or portions may be learned by typing the spelling variation and the phonetic spelling, then connecting the two with an = sign.

The 32K version has a "talking head" whose lips are synchronized to the pronounced syllables. The drawing is sketchy but the effect is impressive. There may be some potential for developing lip reading skills, if the graphics are improved.

There are two pronunciation and one spelling dictionaries. The first dictionary is to be used with the 16K version and the others with the larger version. The spelling dictionary spells out the input strings. Dictionaries contain strings of symbols that translate hexadecimal phoneme equivalent codes to the VOICE BOX itself. The source for the assembly level code is not included on the disk.

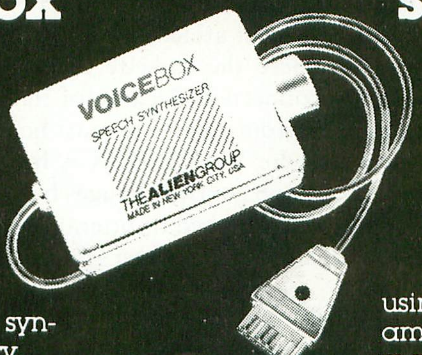
continued on page 52

WIN $5,000 Plus Royalties!*

For the best Talking Game

For the Atari®

Using the VOICE BOX Speech Synthesizer



Now you can make your Atari® 400/800 games and other programs come alive with the VOICE BOX™ by the Alien Group —the first low-cost, smart speech synthesizer with unlimited vocabulary.

Add jokes to your programs. Insults. Compliments. Help messages. Stories. Alien voices. Animal roars. Have your computer talk to the fire department or police in emergencies. To kids. Or blind people. Teach touch typing with immediate spoken feedback. Or just about any other subject — the fun way. Or help a speech-impaired friend communicate . . . the possibilities are limitless.

The VOICE BOX plugs into your Atari's serial port. And talks directly through your TV set. No speaker, amplifier, power supply, special interfaces, or cables needed.

Just select from its simple screen menu. A dictionary with thousands of common words (on diskette or cassette) automatically translates your text into speech. It's that easy.

But don't let its friendliness fool you. The VOICE BOX has all 64 phonemes (basic sounds, like "ah") built in. So you can precisely create any word or sound you can imagine. And store it all on diskette or tape. Names or foreign language words, for example. Or wierd non-human languages.

The VOICE BOX is creative too. It will crack you and your friends up with non-stop random, grammatically correct sentences, using words you specify. It also has an amusing talking face with lip-sync animation — a real crowd-stopper. Best of all, you can call the VOICE BOX from any BASIC program and make your program really hum — literally!

Don't confuse the VOICE BOX with "dumb" speech synthesizers that can't learn new words. Or software-based ones with lower speech quality — and an annoying tendency to blank out the display when they talk. The VOICE BOX is a true breakthrough in speech synthesis. Small wonder thousands of Atari owners have already bought the VOICE BOX.

The VOICE BOX is available now at leading computer stores throughout the world. Or direct from the Alien Group, with 10-day money back guarantee if you're not completely satisfied.

VOICE BOX For Atari. $169.00
16K and 32K versions included
(Specify diskette or cassette).

Enclose check or money order, Visa or Mastercard #. Please include expiration date.

CHECK YOUR LOCAL COMPUTER STORE FOR TALKING "VOICE BOX" VERSIONS OF YOUR FAVORITE GAMES FROM LEADING GAME COMPANIES

*Win a $5,000 prize — plus royalties — for the best Atari 400/800 game using the VOICE BOX. Deadline: May 30, 1983. Write for contest details.

Please mail to: The Alien Group, Department AT-2 , 27 W. 23 St., N.Y., N.Y. 10010
Or call in order to (212) 741-1770

The verbal quality and understandability of the VOICE BOX is not the greatest thing about it, it definitely will not be mistaken for a human voice. "Intelligible" is the operative word to use when describing the characteristics of the VOICE BOX.

The box itself is a 3″ x 4″ x 1½″ box with a large knob on the side and a cable and serial port connector attached. It must attach last in the daisy chain, where the cassette recorder is usually connected. Because of this, all programs from a cassette must be read in before the VOICE BOX is hooked up. The knob control changes both the frequency and speed of utterances. No additional external power is needed.

Here are three program ideas for using the VOICE BOX. They are enhancements to already existing programs that will add a new dimension to them.

1. Talking Menu Program

This program will read the disk and announce the programs on it, then accept input to select one for loading, and indicate any loading errors and recycling if this is the case.

2. Error Diagnostic Program

The program will trap to a verbal indication of what the error was and which sentence, rather than stopping with an error number at the encounter point.

3. Program List Speller using Dictionary 2

This program will accept another program as list input and spell it out for aid in checking & debugging input. It can also be used as an aid by handicapped individuals.

There is also a great potential for teaching children to spell and for an added dimension to games. Overall, I believe, the VOICE BOX is well worth the price tag.

NAUTILUS

Synapse Software
5237 Jacuzzi St., Suite 1
Richmond, CA 94804
$29.95 diskette and cassette
Reviewed by Gordon Miles

Nautilus is a strategy game with an arcade feel. The Nautilus is a submarine that scores points by destroying underwater cities to steal their energy cores. Meanwhile, it must avoid depth charges and other hazards. The Collosus, a surface ship, is the opponent, directed either by the computer or another player.

The graphics in Nautilus are excellent. The submarine has a little propeller at the stern. The oceanscape, with its underwater cities and subterranean passages, is also well done. The realistic tumbling of the depth charges, the predatory helicopter that patrols the surface, the fish and the tenacious limpet mines all make this a visually rich game.

The most innovative graphics feature is the use of a split-screen to depict the two commanders' views. A scoreboard is positioned just above mid-screen. Above the board is the Collosus commander's view on the ocean's surface; below the board is the sub commander's view. Both views scroll independently of one another. The sub commander can be to the far right of the ocean (and deep down); and the Collosus commander can be on the surface to the far left. When the two ships are in the same part of the ocean, they are both realistically shown in both views. The split-screen's main asset is an illusion of quasi-hidden movement which is so essential in a sub / destroyer type of game.

Play begins after choosing the number of players, the time limit, and the playing difficulty. Joysticks allow rapid movement and quick responses especially needed at the higher difficulty levels where the depth charges and homing mines move much faster.

Nautilus is more a strategy game than an arcade game. Destroying and rebuilding of underwater cities is the sole scoring criterion in the game. No sort of hunter/killer scoring is done. This is unfortunate since an arcade game could readily develop at the higher levels where the action is fast. While the sub commander has plenty to do, the Collosus commander just goes from right to left picking up and delivering city repair crews, every once in a while dodging the helicopter, and occasionally, dropping depth charges in its haste from one shore to the other. This can get tedious in a short time. Most decisions stem from whether to continue dealing (repair/destroy) with the cities or where and when to delay your opponent — the difficulty is found in striking the right balance between these options.

Nautilus is an excellent graphics showpiece, well worth the price. Mike Potter, the programmer, conceived it as a strategy game, and as such it is a fair one — however, with a revision in the scoring, it could easily be a very exciting arcade game.

KRAZY ANTIKS

K-BYTE
1755 Austin Road
Troy, MI 48009
$49.95 cartridge
Reviewed by Jerry White

K-BYTE has released three new games on ROM cartridges for ATARI 400/800 computers called K-STAR PATROL, K-RAZY KRITTERS, and K-RAZY ANTIKS. All three are the high quality arcade-type games you'd expect from K-BYTE. Although they are a bit expensive, it's certainly a pleasure to just pop in a cartridge and not have to wait for a program to load.

Of these three new releases, my favorite is K-RAZY ANTIKS. The object of this one-player game is to guide your white ant through a maze of tunnels while avoiding hostile enemy ants, an anteater, and torrential rain floods. You have six mazes from which to choose, and each has 99 levels of play. I seriously doubt that anyone will see level 99 in this decade.

Using a joystick, you guide your ant around the maze laying eggs along the way, and picking up enemy eggs. When an enemy ant is hot on your trail, you can drop an egg that explodes the enemy.

While all this is going on, an anteater will occasionally stick it's long tongue into the top of the maze. If you can position your white ant just ahead of the tongue, you can lure enemy ants into the danger area.

When it rains, the bottom of the maze fills with water, drowning all ants in the lower levels. Floods can be used to your advantage if you can lead the enemy ants toward the bottom as the rain begins, then retreat to the safety of higher levels.

I am not what you'd call an "arcader". I did however find this game to be addictive as well as challenging.

PUBLIC DOMAIN SOFTWARE

We at Antic are pleased to present the first of our collections of public domain software for the ATARI computers. Our first offering includes an extensive selection of programs from the libraries of the ATARI Bay Area Computer Users' Group, The ATARI Computer User Group of Dallas, the Brockport ATARI Computer Junior Users' Group, and the Eugene, Oregon ACE. On the first two disks we have compiled the best of the BASIC games programs from these libraries. Any one of these will keep you and your family absorbed for hours. One disk also includes an automatic BASIC line numbering utility and two BASIC assemblers. Other disks feature music programs, utilities, and educational programs. These disks are packed with useful information and represent an excellent value at $10.00 each, plus $1.50 per order shipping/handling. Send check or money order, indicating disk number(s) to:

Public Domain Software
ANTIC Publishing
297 Missouri Street
San Francisco, CA 94107

ANTIC UTILITY DISK #1

1. Doc: program allows you to accompany programs with separate documentation on disk
2. Microassembler: allows you to create USR routines-assembler, more
3. Assembler-Editor: BASIC, slow but versatile
4. Num: automatic line numbering utility in BASIC
5. Memtest: runs without BASIC cartridge, to test all memory
6. Pattern: graphics demo, documented
7. Color: 128 colors at once
8. Printnop: connect parallel printer from jacks 3 & 4

ANTIC GAME DISK #1

1. Chicken: a great game from ANTIC #1
2. Attack On the Death Star: from ANTIC #2
3. Hangman: the traditional word game
4. Barricade: for two players (sticks)
5. Reverse: order a string of numbers
6. Othello: for two, one or zero player — the computer plays itself.
7. Monopoly: computerized!
8. Lunar lander: select terrain (stick)
9. Escape: guide ships thru maze, skill levels (stick)
10. Zonex: discover hidden color patterns in large grid
11. Clewso: detective text adventure, graphics
12. Puff: version of Life
13. Star Destroyer: game of agility and speed (stick)
14. IQ: traditional peg-jumping game
15. Alien: target game, keyboard or sticks
16. Devil Dungeon: text adventure
17. Again: Simon-type memory game
18. Blaster: challenging Invaders-type (stick)

ANTIC GAMES DISK #2

1. Boing: trampoline game, simple but fun (stick)
2. Miner: get all the gold to the bank before the mine collapses
3. Convoy: a great WW2 strategy and action game (stick)
4. Meteor Storm: dodge the meteors and blow them up
5. Blackjack: you are in Vegas (doubling allowed)
6. Civil War: a strategic simulation
7. Artillery: firing strategy game
8. Golf: great simulation, with graphics, by David Bo
9. Super Wumpus: text adventure — find the Wumpus!
10. Reckless Driving: avoid collisions (stick)

ANTIC presents these programs in diskette form for the convenience of the ATARI community, in the belief that all of the programs offered are in the Public Domain and that no proprietary interests or rights to these programs are claimed by anyone. These diskettes are not copyprotected, nor does ANTIC claim rights to the programs themselves. The price of the diskettes is based on the cost of making them available.

TURTLE Graphics part 2

by GORDON SMITH

This is the second of two articles on implementing a Turtle Graphics system in Forth. The first one appeared in ANTIC issue #3. It discussed Turtle Graphics in general, explained why Forth is a particularly hospitable environment for a Turtle Graphics system, and gave nine screens worth of "foundation" words.

With this artcle are 17 more screens of pns-Forth source code which complete the system. I'll give an overview of the system's features; a glossary of Turtle commands, and give a few suggestions on using the system.

THE INHABITANTS AND LANGUAGE OF TURTLELAND

Four independent turtles live in Turtleland. Multiple turtles open up interesting possibilities, like having turtles chase each other. With four turtles, each can draw in a different color (there are only four colors possible at one time). If you want a different number, you can change the value of the constant #TURTLES on screen 2 before loading. One turtle at a time can be designated the "active turtle" with the SET ACTIVE command. She is the one who will respond when we type a command like "10 DRAW."

Each turtle carries a pen. The active turtle's pen can be lowered with the PENDOWN command, leaving a trail when she moves, or raised with the PENUP command. The more general SET PEN command can be used to do either.

The SET INK command fills the active turtle's pen wtih various colors of ink, depending on the Graphics Mode used. (Modes 3 through 8 can be selected with the SET MODE command.) In all modes, ink of type 0 is erasing ink. It is black, the same color as the background, except in Mode 8 when it is light blue. The command, ERASING, is the same as 0 SET INK. Both choose erasing ink. In Modes 3, 5, and 7, there is also ink of type 1 (gold), type 2 (light green), and type 3 (dark blue). In Modes 4, 6, and 8, types 2 and 3 are not available. The number of ink types is determined by the color video capabilities of the CTIA or GTIA chip. The colors are established by the Operating System when it opens the screen. You can use pns-Forth SETCOLOR word to change them.

Each turtle has a position and a heading. The heading is the number of degrees clockwise from the vertical that she is facing. The active turtle's heading can be changed directly to any value with SET HEADING, also known as TURNTO, or it can be changed incrementally by the commands RIGHT (or TURN) and LEFT.

The system keeps track of each turtle's position with X and Y coordinates. These are *not* the same as the screen column and row numbers. The SET MODE command arranges these coordinates so that the turtle's home at X=0 and Y=0 is the center of the screen, and so that there are one hundred X or Y units per pixel. This means that if a turtle is at X=1000 and Y=500 she will appear ten pixels to the right and five pixels up from the center. You can arrange the coordinates differently if you wish.

The active turtle's coordinates can be individually or jointly set with the commands SET X, SET Y, or SET POSITION (also known as GOTO). They cause the turtle to leave a track only if her pen is down. MOVETO can be used to temporarily raise the pen, or DRAWTO to lower it, before changing position. The pen is restored to its original state after the change.

The most interesting way to move the active turtle is with FORWARD, BACKWARD, DRAW, and MOVE commands. These move her a specified number of steps in whatever direction she is currently heading. FORWARD and BACKWARD draw a line only if the pen is down; DRAW always draws; MOVE never does. Each step normally moves the turtle one pixel, a distance of 100 units in XY coordinates, unless you use the SET SIZE command to alter the step size. By changing the step size you can use the same word to draw the same shape in different sizes.

A turtle's heading and her XY coordinates are always integers. The maximum range for X and Y is from –32768 to 32767. If you drive a turtle beyond this range you may see unwanted tracks as she "jumps" to the other edge of Turtleland.

Usually you can't see all of Turtleland on the screen. For example: in Mode 7 the screen displays only the part of Turtleland from X=-15900 to X=15800 and from Y=-7900 to Y=7800. You can select your own "window" into Turtleland with SET WINDOW command. Any tracks beyond the edges of the window won't be visible. Changing the window will affect the number of X or Y units per pixel. An alternate way to set the window (and the step size) is with the PERPIXEL command.

The reason that the system defaults to 100 units per pixel is to let the turtle sit "between" pixels. If we used a coordinate system as coarse as the screen pixels, then every time we moved a turtle at some angle, her new position would get "rounded" to the nearest pixel. We wouldn't be able to do a series of moves without errors accumulating. Using one hundred XY units per pixel gives us increased precision.

The SET MODE command establishes the whole screen as the "viewport". This means that the view of Turtleland visible through the window will be projected onto all of the screen. You can select any rectangular piece of the screen to be the viewport with the SET VIEWPORT command. When you experiment with this, use the FRAME or NEW commands to draw a frame around the new viewport so you can see where it is.

So far, four commands — MODE, SIZE, WINDOW, and VIEWPORT — relate to Turtleland as a whole, and seven of them — ACTIVE, PEN, INK, HEADING, X, Y, and POSITION — relate to the turtles. It is also possible for you to determine the current value of any of these parameters, by leaving out the word SET or by changing it to SHOW. For example, the command X by itself (i.e., not preceded by SET) leaves the active turtle's current X coordinate on the stack, where it can be used by any word for any purpose. So, the command SHOW X will display some message like "Turtle #1 is at X=300".

The system also has miscellaneous commands like CLEAR for clearing the screen, FRAME for drawing a frame around your picture, and HOME, START, and NEW for starting over. The command BYE leaves Turtleland and returns to pns-Forth.

Of course, all the usual Forth words are still available while you're in Turtleland, in case you need to do arithmetic, comparisons, branching, looping, or whatever. You can use the more compact loop syntax (. . .) and (. . . +) in place of the structures 0 DO . . . LOOP and 0 DO . . . +LOOP.

The important command DEFINE . . . AS . . . END allows you to add new words to the turtle's vocabulary. This makes it very easy to change any of my command names that you don't like.

As an interesting example, you might want to

```
DEFINE HILDA
   AS 1 SET ACTIVE END
DEFINE GILDA
   AS 2 SET ACTIVE END
DEFINE MATILDA
   AS 3 SET ACTIVE END
```

so that you can talk to a turtle simply by invoking her name.

USING THE SYSTEM

To start turtle-ing, just use the SET MODE command. If you want to have Turtleland displayed in Graphics Mode 7, for example, type 7 SET MODE. After this you can immediately move the turtles around with 10 DRAW, 45 TURN, etc. SET MODE initializes the system as follows:

- All four turtles are home at X=0 and Y=0, with heading 0 degrees.
- They all have their pens down.
- Their pens are filled with various ink types as described under the START command in the glossary.
- Turtle #1 is active.
- The window is such that X=0, Y=0 is in the center of the screen and there are 100 X or Y units per pixel.
- The viewport is the whole screen.

After you get acquainted with the various commands, you'll want to start extending the system by defining your own. Here is an example of a new command:

```
VALUE STEPS
VALUE INCREMENT
VALUE ANGLE

DEFINE POLYSPI AS
TO ANGLE
TO INCREMENT
0 TO STEPS
BEGIN
  STEPS INCREMENT + TO STEPS
  STEPS FORWARD
  ANGLE TURN
AGAIN
END
```

POLYSPI can make all sorts of interesting polygonal spirals. It expects to find two numbers on the stack. It stores the top one in ANGLE; this will be how many degrees the turtle will turn between each move. The one below gets stored in INCREMENT; this will be how many more steps the turtle will take each time compared to the previous time. Next STEPS is initialized to 0 and we enter a Forth BEGIN . . . AGAIN loop. The words between BEGIN and AGAIN will be executed indefinitely. (You must press a yellow console button to stop POLYSPI.) Each time through the loop, STEPS is incremented by INCREMENT, and the turtle takes the number of steps in STEPS and turns the number of degrees in ANGLE. Thus POLYSPI is just an automated sequence of FORWARDs and TURNs. For example, 2 90 POLYSPI is really the same as

```
2 FORWARD  90 TURN
4 FORWARD  90 TURN
6 FORWARD  90 TURN
```

and so on.

The three VALUE words POLYSPI uses make it easy to see what's going on. However, another definition of POLYSPI is possible which uses no variables at all:

```
DEFINE POLYSPI AS
0
BEGIN
 3 PICK +
 DUP FORWARD
 OVER TURN
AGAIN
END
```

This version keeps everything on the stack, using the Forth words PICK, DUP, and OVER for stack manipulation. You can make a variety of patterns with this one command by changing its two parameters.

Pressing a yellow console button will break out of an indefinite loop of turtle moves. In fact, every time a turtle changes position, the system checks the console buttons and returns to command level if one is depressed. This makes it easy to regain control.

As mentioned in Part I, ten of the words used in my screens are pns-Forth words which won't be available (at least not with the same meanings) in other Forth systems. Two of these, 1– and TABLE, are common Forth extensions whose high-level definitions are

```
: 1- 1 - ;
```

and

```
: TABLE  <BUILDS DOES> OVER
  + + @ ;
```

The others are highly system-specific. Four of them — SETUP S, CLOSE S, SPLIT-SCREEN, and GR. — were used in the word GRAPHICS in Part I. Their definitions are quite complex, as these words are part of pns-Forth's interface to the CIO routines in the Operating System. Their joint effect in the word GRAPHICS, however, is quite simple. Any Forth system sold for the ATARI will probably have words for opening the screen for graphics. Simply use whatever your system provides to define your own GRAPHICS, which takes one number from the stack and opens the screen in that mode, with a text window at the bottom.

The last four words specific to pns-Forth are CL#, COLOR, PLOT, and DRAWTO. These are used by LINE (in Part I), FRAME, and POSITION. The first two are simple to define; just use

```
0 VARIABLE CL#
```

and

```
: COLOR  DUP CL# ! PAD C! ;
```

CL# is a variable which is used to keep track of the color data used to plot a pixel. COLOR takes a number from the stack and stores it both in CL# and at PAD, for later use by PLOT and DRAWTO. The definitions of PLOT and DRAWTO are complicated because these words result in calls to CIO. Again, however, their functions are simple and your system probably provides similar words. Define a PLOT which takes a column and a row number from the stack, moves the screen cursor to that position, and plots a pixel there using whatever byte is at PAD as the color data. Similarly, define a DRAWTO which takes a column and a row number from the stack, and draws a line from the current position of the screen cursor to this specified position, using the byte at PAD as color data.

I believe that all the other words I've used in this system are either standard fig-Forth words or new words that I've defined.

GLOSSARY OF TURTLE COMMANDS

MODE Commands

SET MODE [mode ---]
Opens the screen in the Graphics Mode specified by mode , which should be 3–8. Sets up a default viewport, window, and step size by executing WHOLE-SCREEN SET VIEWPORT and 100 PER-PIXEL. Draws a frame around the viewport with ink of type 1. Initializes the turtles by executing START.

MODE [--- mode]
Leaves the number of the current Graphics Mode on the stack.

SHOW MODE [---]
Displays a message indicating the current Graphics Mode.

ACTIVE Commands

SET ACTIVE
[turtle# ---]
Makes the turtle whose number is turtle# the active turtle. Future commands will be directed to her.

ACTIVE [--- turtle#]
Leaves the number of the active turtle on the stack.

SHOW ACTIVE [---]
Displays a message indicating the currently active turtle.

PEN Commands

SET PEN [state ---]
Lowers the active turtle's pen if state is nonzero and raises it if state is zero.

PEN [--- state]
Leaves 1 on the stack if the active turtle's pen is down and 0 if it is up.

SHOW PEN [---]
Displays a message indicating whether the active turtle's pen is up or down.

INK Commands

SET INK [ink#]
Fills the active turtle's pen with ink of type ink# . Type 0 ink is erasing ink. Types 1, 2, and 3 are colored. Types 2 and 3 are not available in modes 4, 6, or 8.

INK [--- ink#]
Leaves on the stack the type of ink in the active turtle's pen.

SHOW INK [---]
Displays a message indicating the type of ink in the active turtle's pen.

HEADING Commands

SET HEADING
[degrees ---]
Makes the active turtle head in the direction specified by degrees . Directions are measured clockwise from the vertical.

HEADING
[--- degrees]
Leaves the active turtle's heading on the stack.

SHOW HEADING [---]
Displays a message indicating the active turtle's heading.

X Commands

SET X [x ---]
Changes the active turtle's X coordinate to x . Draws a line if her pen is down.

X [--- x]
Leaves the active turtle's X coordinate on the stack.

SHOW X [---]
Displays a message indicating the active turtle's X coordinate.

POSITION Commands

SET POSITION
[x y ---]
Changes the active turtle's coordinates to X = x and Y = y . Draws a line if her pen is down.

POSITION [--- x y]
Leaves the active turtle's X and Y coordinates on the stack.

SHOW POSITION [---]
Displays a message indicating the active turtle's X and Y coordinates.

SIZE Commands

SET SIZE
[distance steps ---]
Sets the step size so that the number of steps given by steps will cover a distance in XY coordinates given by distance .

SIZE
[--- distance steps]
Leaves the current size parameters on the steack.

SHOW SIZE [---]
Displays a message indicating the current step size.

WINDOW Commands

SET WINDOW
[xmin xmax ymin ymax ---]
Sets the window to be the region from X = xmin to X = xmax and from Y = ymin to Y = ymax .

WINDOW [--- xmin xmax ymin ymax]
Leaves the current window parameters on the stack.

SHOW WINDOW [---]
Displays a message indicating the current window.

VIEWPORT Commands

SET VIEWPORT [left right top bottom ---]
Sets the viewport to extend from screen column left to screen column right and from screen row top to screen row bottom .

WHOLE-SCREEN SET VIEWPORT [---]
Sets the viewport to extend from column 1 to the next to the last column and from row 1 to the next to the last row.

VIEWPORT [--- left right top bottom]
Leaves the current viewport parameters on the stack.

SHOW VIEWPORT [---]
Displays a message indicating the current viewport.

Y Commands

Similar to X Commands

Other Commands

CLEAR [---]
Clears the graphics screen without affecting the turtles.

FRAME [ink# ---]
Draw a frame around the viewport, using ink of type ink# .

HOME [---]
Moves the active turtle to X = 0 and Y = 0 with heading 0, without drawing a line, and then lowers her pen.

START [---]
HOMEs all the turtles first. Then fills their pens with ink. (In mode 3, 5, or 7, the Nth turtle's pen is filled with ink of type N. In mode 2, 4, or 6, turtle's 0's pen is filled with type 0 ink while the pens of turtles 1, 2, and 3 are filled with type 1 ink, the only colored ink available in these modes.) Finally, makes turtle 1 the active turtle.

NEW [---]
Clears the screen, draws a frame with type 1 ink, and initializes the turtles by executing START.

PER-PIXEL
[distance ---]
Sets the window so that the point X = 0, Y = 0 is the center of the viewport, and so that the distance in XY coordinates given by distance will be the size of one pixel. Also, sets the step size so that each step is distance units long.

FORWARD [steps ---]
Moves the active turtle forward the number of steps specified by steps . The movement is in the direction she is currently heading if steps is positive and in the opposite direction if steps is negative. The turtle's heading is unaffected. A line is drawn if her pen is down.

BACKWARD [steps ---]
Like FORWARD except in the opposite direction.

DRAW [steps ---]
Lowers the active turtle's pen so that a line will definitely be drawn as she moves forward the number of steps given by steps . Then her pen is returned to its previous state.

MOVE [steps ---]
Raises the active turtle's pen so that a line will definitely not be drawn as she moves forward the number of steps given by steps . Then her pen is returned to its previous state.

RIGHT [degrees ---]
Turns the active turtle the specifiefd number of degrees, to the right if degrees is positive and to the left if negative.

LEFT [degrees ---]
Like RIGHT except in the opposite direction.

TURN [degrees ---]
The same as RIGHT.

GOTO [x y ---]
The same as SET POSITION.

DRAWTO [x y ---]
Lowers the active turtle's pen so that a line will definitely be drawn as she moves to X = x and Y = y . Then her pen is returned to its previous state.

MOVETO [x y ---]
Raises the active turtle's pen so that a line will definitely not be drawn as she moves to X = x and Y = y . Then her pen is returned to its previous state.

TURNTO [degrees ---]
The same as SET HEADING.

PENDOWN [---]
Lowers the active turtle's pen. This is the same as 1 SET PENSTATE.

PENUP [---]
Raises the active turtle's pen. This is the same as 0 SET PENSTATE.

PENDOWN? [--- flag]
Leaves a 1 on the stack if the active turtle's pen is down and a 0 if it is up. This is the same as PEN.

PENUP? [--- flag]
Leaves a 1 on the stack if the active turtle's pen is up and a 0 if it is down. This is the opposite of PEN.

ERASING [---]
Fills the active turtle's pen with type 0 ink (the erasing type.) This is the same as 0 SET INK.

(. . .) [#loops ---]
Executes the words between the left parenthesis and the right parenthesis the number of times given by #loops .

DEFINE . . . AS . . . END
Defines the word between DEFINE and AS to be a new turtle command which will execute the words between AS and END.

BYE [---]
Leaves Turtleland and returns to pns-Forth.

```
(     Turtle Graphics II, screen 1      )
DECIMAL
: VALUES
        <BUILDS  0 DO
            0 , LOOP
      DOES> OVER + +
         TO-FLAG @ IF
  0 TO-FLAG !  !  ELSE
             @ THEN ;
VALUE PREFIX
: SET   ( --- )   2 TO PREFIX ;
: SHOW   ( --- )   4 TO PREFIX ;
: ROOT:   ( --- )
        <BUILDS SMUDGE ]
  DOES> PREFIX + @ EXECUTE
          0 TO PREFIX ;
-->
```

```
(     Turtle Graphics II, screen 2      )
4 CONSTANT #TURTLES
VALUE WHICH
  ( The number of the active turtle )
: ACTIVE!   ( n --- )   TO WHICH ;
: .WHICH   ( --- )
  ." Turtle #" WHICH . ;
: ACTIVE?   ( --- )
   .WHICH ." is active " CR ;
ROOT: ACTIVE   WHICH ACTIVE! ACTIVE? ;
-->
```

```
(     Turtle Graphics II, screen 3      )
: MODE@   ( --- n )   87 C@ ;
: MODE?   ( --- )
   ." This is graphics mode "
                   MODE@ . CR ;
TABLE MAX_COL#   ( n1 --- n2 )
  39 , 19 , 19 , 39 , 79 , 79 ,
  159 , 159 , 319 ,
TABLE MAX_ROW#   ( n1 --- n2 )
  19 , 19 ,  9 , 19 , 39 , 39 ,
  79 , 79 , 159 ,
: WHOLE-SCREEN   ( --- n1 n2 n3 n4 )
  1 MODE@ MAX_COL# 1-
  1 MODE@ MAX_ROW# 1- ;
-->
```

```
( 44 Turtle Graphics II, screen 4      )
: VIEWPORT@   ( --- n1 n2 n3 n4 )
  CLIPPING LEFT RIGHT TOP BOTTOM ;
: VIEWPORT?   ( --- )   CLIPPING
   ." The viewport is from column "
     LEFT . ." to " CR ." column "
   RIGHT . ." and from row " TOP .
         ." to row " BOTTOM . CR ;
VALUE XMIN     VALUE YMIN
VALUE XMAX     VALUE YMAX
: WINDOW@   ( --- n1 n2 n3 n4 )
  XMIN XMAX YMIN YMAX ;
: WINDOW?   ( --- )
    ." The window is from X=" XMIN .
  ." to X=" XMAX . CR ." and from Y="
       YMIN . ." to Y=" YMAX . CR ;
-->
```

```
(     Turtle Graphics II, screen 5      )
VALUE OCOL     VALUE OROW
: ORIGIN!   ( --- )      CLIPPING
    XMIN MINUS  RIGHT LEFT -
 XMAX XMIN -   */  LEFT +  TO OCOL
      YMAX MINUS  TOP BOTTOM -
  YMAX YMIN -   */  TOP +  TO OROW ;
: VIEWPORT!   ( n1 n2 n3 n4 --- )
                    CLIPPING
  MODE@ MAX_ROW# MIN  TO BOTTOM
            0 MAX  TO TOP
   MODE@ MAX_COL# MIN  TO RIGHT
          0 MAX  TO LEFT
                ORIGIN! ;
: WINDOW!   ( n1 n2 n3 n4 --- )
  TO YMAX  TO YMIN  TO XMAX  TO XMIN
                      ORIGIN! ;
-->
```

```
(     Turtle Graphics II, screen 6      )
ROOT: VIEWPORT
  VIEWPORT@ VIEWPORT! VIEWPORT? ;
ROOT: WINDOW
  WINDOW@ WINDOW! WINDOW? ;
: LEFT-   ( --- n )
  CLIPPING LEFT 1- 0 MAX ;
: TOP-   ( --- n )
  CLIPPING TOP 1- 0 MAX ;
: RIGHT+   ( --- n )   CLIPPING
  RIGHT 1+  MODE@ MAX_COL# MIN ;
: BOTTOM+   ( --- n )   CLIPPING
  BOTTOM 1+  MODE@ MAX_ROW# MIN ;
: FRAME   ( n --- )     COLOR
     LEFT-  TOP-      PLOT
     RIGHT+ TOP-      DRAWTO
     RIGHT+ BOTTOM+ DRAWTO
     LEFT-  BOTTOM+ DRAWTO
     LEFT-  TOP-    DRAWTO ;
-->
```

```
(    Turtle Graphics II, screen 7        )
#TURTLES VALUES PEN()
: PEN@    ( --- flag )    WHICH PEN() ;
: PENDOWN?    ( --- flag )    PEN@ ;
: PENUP?    ( --- flag )    PEN@ 0= ;
: PEN!    ( flag --- )
  0= 0=  WHICH  TO PEN() ;
: PENDOWN    ( --- )    1 PEN! ;
: PENUP    ( --- )    0 PEN! ;
: PEN?    ( --- )          .WHICH
  ." has her pen "  PEN@ IF
               ." down " ELSE
                 ." up " THEN
                         CR ;
ROOT: PEN    PEN@ PEN! PEN? ;
-->

(    Turtle Graphics II, screen 8        )
#TURTLES VALUES INK()
: INK@    ( --- n )    WHICH INK() ;
: INK!    ( n --- )    WHICH  TO INK() ;
: ERASING    ( --- )    0 INK! ;
: INK?    ( --- )
  .WHICH ." is using ink #" INK@ . CR ;
ROOT: INK    INK@ INK! INK? ;
-->

(    Turtle Graphics II, screen 9        )
#TURTLES VALUES HEADING()
: HEADING@    ( --- n )
  WHICH HEADING() ;
: HEADING?    ( --- )          .WHICH
  ." has heading " HEADING@ . CR ;
: HEADING!    ( n --- )
  360 MOD  WHICH  TO HEADING() ;
: TURNTO    ( n --- )    HEADING! ;
ROOT: HEADING
  HEADING@ HEADING! HEADING? ;
: TURN    ( n --- )
  HEADING@ + HEADING! ;
: RIGHT    ( n --- )    TURN ;
: LEFT    ( n --- )    MINUS TURN ;
-->

(    Turtle Graphics II, screen 10        )
#TURTLES VALUES X()
#TURTLES VALUES Y()
: X@    ( --- n )    WHICH X() ;
: Y@    ( --- n )    WHICH Y() ;
: X?    ( --- )
  .WHICH ." is at X=" X@ . CR ;
: Y?    ( --- )
  .WHICH ." is at Y=" Y@ . CR ;
: POSITION@    ( --- n1 n2 )    X@ Y@ ;
: POSITION?    ( --- )          .WHICH
  ." is at X=" X@ . ." and Y=" Y@ . CR ;
-->

(    Turtle Graphics II, screen 11        )
: X->COL    ( n1 --- n2 )    CLIPPING
  RIGHT LEFT - XMAX XMIN - */ 0COL + ;
: Y->ROW    ( n1 --- n2 )    CLIPPING
  TOP BOTTOM - YMAX YMIN - */ 0ROW + ;
: SCALE    ( n1 n2 --- n3 n4 )
  SWAP X->COL SWAP Y->ROW ;
: ?CONSOLE    ( --- flag )
  53279 C@ 7 = NOT ;
: POSITION!    ( n1 n2 --- )
                        ?CONSOLE IF
       SP! CR ." ok" QUIT THEN
                           PEN@ IF
                        INK@ COLOR
  OVER OVER SCALE  POSITION@ SCALE
                         LINE THEN
          WHICH TO Y()  WHICH TO X() ;
-->

(    Turtle Graphics II, screen 12        )
: GOTO    ( n1 n2 --- )    POSITION! ;
ROOT: POSITION
  POSITION@ POSITION! POSITION? ;
: X!    ( n --- )  Y@ POSITION! ;
: Y!    ( n --- )  X@ SWAP POSITION! ;
ROOT: X    X@ X! X? ;
ROOT: Y    Y@ Y! Y? ;
: MOVETO    ( n1 n2 --- )
  PEN@ ROT ROT PENUP POSITION! PEN! ;
: DRAWTO    ( n1 n2 --- )
  PEN@ ROT ROT PENDOWN POSITION! PEN! ;
-->

(    Turtle Graphics II, screen 13        )
VALUE SIZE_N        VALUE SIZE_D
: SIZE@    ( --- n1 n2 )
  SIZE_N SIZE_D ;
: SIZE*    ( n1 --- n2 )    SIZE@ */ ;
: SIZE!    ( n1 n2 --- )
  TO SIZE_D  TO SIZE_N ;
: SIZE?    ( --- )
                 SIZE_D DUP . 1 = IF
                    ." step is " ELSE
                  ." steps are " THEN
  ." a distance of " SIZE_N . CR ;
ROOT: SIZE    SIZE@ SIZE! SIZE? ;
-->

(    Turtle Graphics II, screen 14        )
: VECTOR    ( n --- n1 n2 )
  DUP HEADING@ SIN*  X@ +
  SWAP HEADING@ COS*  Y@ + ;
: FORWARD    ( n --- )
  SIZE* VECTOR POSITION! ;
: BACKWARD    ( n --- )  MINUS FORWARD ;
: MOVE    ( n --- )
  PEN@ SWAP  PENUP FORWARD  PEN! ;
: DRAW    ( n --- )
  PEN@ SWAP  PENDOWN FORWARD  PEN! ;
-->

(    Turtle Graphics II, screen 15        )
: PER-PIXEL    ( n --- )
                        CLIPPING >R
                  RIGHT LEFT - 2 /
  DUP MINUS R *  SWAP 1+ R *
                  BOTTOM TOP - 2 /
  DUP MINUS R *  SWAP 1+ R *
  SET WINDOW    R> 1 SET SIZE ;
( Make SURE you typed the >R and R> )
( in this correctly. )
: SCREEN-DEFAULTS    ( --- )
  WHOLE-SCREEN SET VIEWPORT
                     100 PER-PIXEL ;
TABLE GR.BYTES    ( n1 --- n2 )
  960 , 400 , 200 , 200 , 400 ,
  800 , 1600 , 3200 , 6400 ,
: CLEAR    ( --- )
  88 @ MODE@ GR.BYTES  ERASE ;
-->

(    Turtle Graphics II, screen 16        )
: HOME    ( --- )
  0 0 MOVETO  0 TURNTO  PENDOWN ;
: START    ( --- )
            #TURTLES 0 DO
      I SET ACTIVE  HOME
          MODE@ 2 MOD IF
    I ELSE  I 0= 0= THEN
            SET INK  LOOP
              1 SET ACTIVE ;
: MODE!    ( n --- )
  GRAPHICS  SCREEN-DEFAULTS
              1 FRAME  START ;
ROOT: MODE    MODE@ MODE! MODE? ;
: NEW    ( --- )  CLEAR  1 FRAME  START ;
: BYE    ( --- )  0 GRAPHICS
         0 710 C!  68 712 C! ;
-->

(    Turtle Graphics II, screen 17        )
: DEFINE   [COMPILE] : ; IMMEDIATE
: AS ; IMMEDIATE
: END   [COMPILE] ; ; IMMEDIATE
: \    ( ignores rest of line )
  IN @ C/L / 1+ C/L * IN ! ; IMMEDIATE
: (
  COMPILE 0  [COMPILE] DO ; IMMEDIATE
: )  [COMPILE] LOOP ; IMMEDIATE
: +)   [COMPILE] +LOOP ; IMMEDIATE
;S
```

A MESSAGE ON THE MEDIUM

by CARL EVANS

The ATARI 410 Program Recorder, sometimes called the cassette drive, has a mixed reputation. Many users have had exasperating problems with it that have driven some of them straight to diskettes. Still, the 410 is a popular device, especially among ATARI users with tight budgets. The 410 can be more efficient and reliable if you learn something about its quirks.

The kind and quality of tape used with the 410 are important factors contributing to successful use. You might assume that the best quality, highest priced tapes, perhaps the so-called "computer" tapes, are what you should use, but that is not necessarily so. A good brand of music tape is usually sufficient.

You should avoid really cheap tapes, or tapes with strange brand names, and also avoid the "chromium dioxide" tapes, and "digital data recording" tapes advertised for use with some computers. Short tapes are often better than long ones, and there are some cassettes out now with as little as three minutes of tape on them. Really long tapes, like the C-90 and C-120, are unnecessary and have special problems that disqualify them for computer use.

Cassette tape is made of a tough, flexible plastic, coated with a magnetic material, usually ferrous oxide. When you record a program on a tape, the recorder generates a magnetic field that leaves a magnetic trace at a certain spot on the coating. The tape is pulled past the recording heads of the recorder at a constant speed, leaving a series of magnetized traces. The strength of the traces depends on many things, but the nature of the coating, its formulation and the evenness of its distribution are important.

In general, the price and quality of a tape will reflect these factors. Very expensive tapes will have highly responsive coatings securely bonded to the plastic in very carefully controlled thicknesses. Medium priced tapes, like music tapes, sacrifice some of this quality. Cheap tapes have inferior coatings and quality control, and are really suitable only for non-critical purposes.

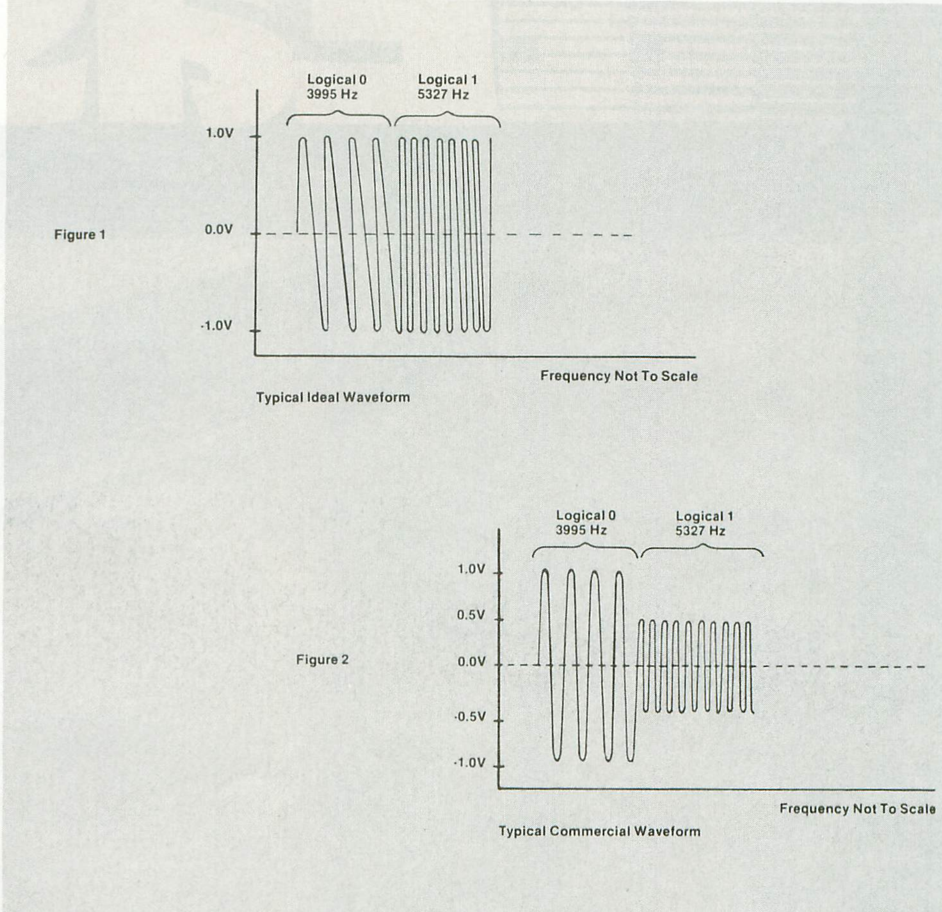
The ATARI computers record data on the tape using a method known as *frequency shift keying*, or FSK. This means that the computer actually generates sound frequencies and sends them to the recorder. The frequencies generated by the ATARI are very precise at 3995 Hz (for zero) and 5327 Hz (for one). The 410 recieves these signals from the computer and leaves the appropriate magnetic traces on tape. This means that the trace on the tape is approximately the same as generated by the computer, but not quite. Variation within components of the 410 may throw the frequency off, or the speed of the capstan motor may vary enough to change the frequency of the trace. The amount of electrical energy to the recording heads may fluctuate too, so the traces written by the 410 may stray somewhat from their intended values.

Although the ATARI computer writes sound frequencies to the tape, it does not respond to these frequencies when reading from tape. The 410 actually listens to the tape, interprets the frequencies as either low or high, and passes a voltage to the computer. Zero voltage equals zero, and approximately 5 volts equals one. The computer does not hear the frequencies, it reads voltage changes coming from the 410. This is the crux of the problem. If the 410 does not properly interpret the frequencies and pass the right voltage, the computer gets bad data.

Tape can be responsible for these errors. Look at Figure 1. In this normalized figure we find 3995 Hz and 5327 Hz as they might appear on an oscilliscope. Pretend that they represent two bits of data on tape. The amplitude (height) of each is equal, indicating the traces have equal signal strength. Tapes prepared by the ATARI Program Exchange (APEX) come close to this ideal. Tapes recorded by other vendors, or by the 410 itself most often do not. They look more like the waveforms in Figure 2, where the higher frequency has less amplitude, or signal strength. This loss of strength for higher frequencies is called attenuation and is a normal condition.

If attenuation causes the higher frequency to be recorded at a strength lower than the threshold for recognition by the 410, or the components of the 410 are not operating up to their specifications, the 410 will fail to recognize a one, and "drop" a bit. The computer is monitoring the load, and if it senses an improper sequence it will issue a loading error message. But the bad bit(s) may be in data statements about which the computer cannot judge, and so some problems won't show up till later.

Highly responsive computer tape might seem a solution to this problem, but in fact it catches us going the other way. Being highly responsive, it does not attenuate certain non-signals, like the switching current that makes a "spike" every time the recorder is started. Spikes are often interpreted by the 410 as signals, so a one is passed to the computer when none is supposed to be there. This, of course, gives a bad load. If you have this problem, you can get around it by positioning your tape on CLOADS so you start just past the point where the leader tone begins. Other intermittant electrical noises can creep onto high grade tape in a low grade recorder, and that's why medium grade tapes are better with the 410. They will tend to attenuate spurious signals to strengths below threshold.



The problem with chromium dioxide tape is that it is very abrasive to recording heads and the 410 is not built to take it. Beware of companies offering software for the ATARI recorded on chrome tape. Head wear is another cause for distortion and loss of signal.

Other things that can cause the 410 to misread a signal include "fade out" and distortion. Fade out is the loss of strength of a trace on tape due to thin spots, perhaps caused by loss of coating from repeated playing. Music buffs claim they can hear this loss after about fifty plays. That is about the point at which fade out begins to affect data too. After about fifty reads, medium grade tape may lose enough coating to lower the trace strength below threshold for higher frequencies. Running tape on Fast Forward or Reverse does not damage the coating since the heads are not engaged.

Sometimes distortion results from the physical stretching of tape. If a tape is stretched, a high frequency signal will become lower, and may not be recognized by the circuitry as a high signal. Stretching takes place most often near the beginning and end of tapes, near the hubs. Allow plenty of leader to reduce this problem. Thin tape stretches easier than thick tape, and thin tape is used with long tapes, so avoid any tape longer than 30 minutes per side.

Good practice suggests using short tapes of good quality from a manufacturer whose products give satisfaction. I usually put only one program on a tape and repeat the save a couple of times on that tape. I might also save it with other programs on a longer tape for archival purposes. Keep your tapes away from magnetic forces and environmental hazards.

This covers the main points of the tape medium as they pertain to the ATARI system. Next month we will look at the 410 machine itself. Your comments and questions are invited. Please send them to ANTIC for my attention.

Carl Evans is an electronics engineer who earned both BSEE and MSEE at Georgia Institute of Technology. He now works at Aerojet Electronics Systems, and was formerly with General Dynamics. In 1979 he bought a TRS-80 and used only tape for mass storage. When lured to the ATARI 800 in 1980 he continued to use tape with the 410 Program Recorder. He has written many programs for the ATARI, including several tape utilities, and has formed a new company, VERVAN Software, to market some of them. ANTIC readers should be pleased to learn that Carl will be coordinating this department of the magazine.

ON HAVING A GOOD TIME

continued from page 23

clear it again afterwards, getting a nice uncorrupted reading.

The timekeeping routine needs to be inserted in the interrupt service chain, and we also have to ensure that it is not erased. A number of countermeasures are necessary, including, a harmless patch to DOS.

Puting the pieces together

I have placed the module in the cassette buffer. If you don't run the cassette, this area is unused. Two points of caution about its present location: although the cassette buffer extends from 3FD hex to 4FF, "SYSTEM RESET" clears all of pages two and three so our code can't actually begin below 400; also the module slightly overflows the top end of the buffer into locations used by BASIC. Fortunately the initilization code — which only runs when the module is first loaded — can be put there, so there is no great problem.

We can partition the code as follows. The user communication area [TIMLOK, SECS...DAYS] is placed at the very beginning, so that it can be easily referenced. Then, after some local storage, comes the code to handle the RESET button, followed by the interrupt handler itself [CLOCK]. At the end — and extending into BASIC's space — is the initialization code [WINDIT]. Notice the start-address is location 2E2 (not 2E0).

Now, I'll define some of the terminology. First of all, clock "ticks" are actually VBLANK interrupts. When a VBLANK interrupt occurs, the operating system jumps to the service routine through the "immediate VBLANK vector" location VVBLKI. The service routine is normally within the operating system [SYSVBV], but we can change the contents of the vector to point to our own routine.

The vector is a two-byte address, and there would be a reasonable possibility of an interrupt happening just after one byte of the two had been changed, with disastrous results. To circumvent this hazard, the operating system provides a special routine SETVBV that should always be used to alter this. The particular vector to change is selected by the value passed in the accumulator (6 in this case). The operating system will always restore the vector to its original value on RESET, so we in turn must immediately re-fix it.

The reset code in the module is executed after the operating system's reset sequence has been initiated. This sequence occurs whenever the system has been bootstrapped (*not* simply loaded) from cassette. This CASINI vector points to the code that performs the reset sequence, and may be used for our own purposes. If the CASINI vector does happen to be already in use, the startup-time code [WINDIT] will store the current value as part of a JSR instruction [INITON] in the reset code so that it is still executed. The operating system is informed that CASINI is enabled by setting bit 1 of the flag BOOTF.

Caution! don't try to load the module more than once without a cold-start! If you do so it will find its own address in CASINI and go into an infinite loop at the next RESET.

Once the VBLANK vector is set up, our interrupt service routine [CLOCK] will go into action at each clock tick. There is little more than needs to be said about this section, except to note how the CRITIC flag is used. This flag is set non-zero by the system to inhibit deferred VBLANK processing, and we too must bypass most of the normal sequence at such times. We actually combine the CRITIC flag with our our own TIMLOK, freezing the clock if either is non-zero, so that we can avoid reading a running clock. Don't leave TIMLOK set for more than ten minutes, though. If the number of 13-second adjustments needed become greater than 60, the count-down timer [CNT60] will overflow and take several seconds to get back in step. TIMLOK *is* set initially to 255, as a signal to the user that the clock has never been run. But the user program, when setting the time of day, should always initialize the seconds-counter CSECS to zero and the count-down timer CNT60 to sixty immediately before releasing TIMLOK, thus ensuring that the clock starts in sync.

Doing it to DOS

That about covers the code itself. Now all we have to do is make the DOS patch. Under DOS 2, when you return to a cartridge program from DUP.SYS with the "B" command, the VBLANK interrupt vectors will be reset to their original system values. I have never fathomed the intention behind this. In any case, suppressing the action has absolutely no detrimental effect on normal usage.

The patch is trivial, a jump to avoid that section of code, but a little messy to install, because DUP.SYS usually goes away when not in use. It seems best to present it as a recipe; the one that follows is probably the shortest reasonable path.

1. Install the Assembler / Editor cartridge and boot up with DOS-2
2. Insert the disk you intend to patch. It should have DUP.SYS on it — and I suggest it be a scratch disk!
3. Use the editor to generate the patch:

```
10 * = $272A
20 JMP $1912
30 .END
```

4. Assemble it to a disk file with:

```
ASM,,#D:DOSPATCH.OBJ
```

5. Got to the DOS menu.
6. Give the "C" command, and in response to the file-spec query enter:

```
DOSPATCH.OBJ,DUP.SYS / A
```

At this point the patch has been tagged on to the end of the save file, making it one sector longer.

7. Use the "B" command to return to the editor, and then go back to DOS. This brings in the modified system.
8. Re-install the new system on your disk drive with the "H" command. DUP.SYS will return to its original length.

You should, of course, copy the modified system onto any disk you are going to use while the clock module is

continued next page

running. (In fact I never use anything but the patched version.) For this you can use either the "H" command to install a complete DOS, or "C" or "O" to update DUP.SYS alone.

Where we came in

If you've stuck with me this far, you probably don't need to ask "What can I use it for?". You must have desperate need for it. However, as an example of how to couple the module to BASIC, a simple digital clock is given in listing 2. Take it and go from there.

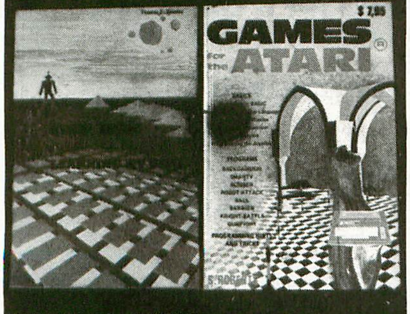
```
10 ;"ACCURATE" CLOCK MODULE
20 ;Copyright Pete Goodeve, 1982
30 ;=====================
40 ;
50 ;occupies cassette buffer
60 ;
70 ;Atari OS references:
80 SETVBV=$E45C set-vector entry
90 SYSVBV=$E45F OS VBLANK service
0100 VVBLKI=$222 immed. VBLANK vector
0110 CRITIC=$42 critical section flag
0120 CASINI=$2 "cassette" init vector
0130 BOOTF=9 boot mode flag for init
0140 ;
0150 ;
0160 *=$400 cass. buffer(1024 dec)
0170 ;
0180 TIMLOK .BYTE $FF
0190 SECS .BYTE 0
0200 MIN .BYTE 0
0210 HRS .BYTE 0
0220 DAYS .BYTE 0
0230 ;
0240 CNT60 .BYTE 60 VBLANK ticks
0250 CSECS .BYTE 0 contin. count
0260 ASECS .BYTE 13 adjustment count
0270 ;
0280 ;
0290 INITON=*+1
0300 ;Comes this way on RESET Button
0310 ;via "Cassette Init" vector!
0320 RESET
0330  JSR NUTHIN -- filled before use
0340 SETINT
0350  LDX #CLOCK/256
0360  LDY #CLOCK&$FF
0370  LDA #6 "immediate VBLANK" code
0380  JSR SETVBV set up interr. vect.
0390 NUTHIN
0400  RTS
0410 ;
0420 ;
0430 ;Immed VBLANK interrupt service
0440 ;comes through here first!
0450 ;
0460 CLOCK
0470  DEC CNT60 count 60 ticks
0480  BNE XIT before doing anything
0490  INC CSECS keep track of seconds
0500  LDX #60 (kept around for later)
0510  STX CNT60 reset count
0520  LDA CRITIC check if critical
0530  ORA TIMLOK or if locked by user
0540  BNE XIT gotta stop here
0550 ; continue on if not critical
0560 ; or locked...:
0570 ; repeats if seconds were missed
0580 CLKLP
0590  DEC ASECS 13 second count down
0600  BNE TICK
0610  LDA #13
0620  STA ASECS reset 13-sec count
0630  DEC CNT60 and skip one tick
0640 TICK
0650  INC SECS user's time
0660  CPX SECS,reached 60 yet?
0670  BNE TOK nope
0680  LDY #0
0690  STY SECS reset seconds
0700  INC MIN and bump minutes
0710  CPX MIN over the hour?
0720  BNE TOK not yet
0730  STY MIN and so on...
0740  INC HRS
0750  LDA #24
0760  CMP HRS
0770  BNE TOK
0780  STY HRS
0790  INC DAYS
0800 ;...etc. if needed
0810 TOK
0820  DEC CSECS were any missed?
0830  BNE CLKLP round again if so
0840 VVON=*+1
0850 ;continue with VBLANK chain!
0860 XIT
0870  JMP SYSVBV altered at setup
0880 ;
0890 ;
0900 ;
0910 ;*** INITIAL ENTRY HERE
0920 ;gets overwritten by BASIC
0930 WINDIT
0940  LDX CASINI+1 Cassette Init vect
0950  BEQ NOINI zero if not used
0960  LDY CASINI rest of current vect
0970 SETON
0980  STX INITON+1 set up JSR address
0990  STY INITON so stuff gets done
1000  LDA #RESET/256 plug in our own
1010  STA CASINI+1 reset sequence
1020  LDA #RESET&$FF
1030  STA CASINI
1040  LDA VVBLKI current immed VBLANK
1050  STA VVON will be done after us
1060  LDA VVBLKI+1
1070  STA VVON+1
1080  LDA BOOTF bootstrap mode flag
1090  ORA #2 must include "cassette"
1100  STA BOOTF
1110  JMP SETINT go set VBLANK vector
1120 ;
1130 NOINI
1140  LDY #NUTHIN&$FF dummy for JSR
1150  LDX #NUTHIN/256
1160  BNE SETON
1170 ;
1180 ;
1190 ;Autostart addr.
1200 *=$2E2 "init" vector
1210 .WORD WINDIT
1220 ;
1230 .END
```

```
10 GRAPHICS 2
20 REM set up mnemonics fo
   r module locations:
30 TIMLOK=1024:SECS=1025:M
   IN=1026
40 HRS=1027:CNT60=1029:CSE
   CS=1030
50 REM
60 REM if clock not set, g
   o do it:
70 IF PEEK(TIMLOK)>1 THEN
   GOSUB 210
80 POKE TIMLOK,0:REM just
   in case...
90 REM
100 REM *** main time displ
    ay loop:
110 IF PEEK(SECS)=OLDSECS T
    HEN 110
120 POKE TIMLOK,1
130 VHRS=PEEK(HRS):VMIN=PEE
    K(MIN):VSECS=PEEK(SECS)
140 OLDSECS=VSECS
150 POKE TIMLOK,0
160 POSITION 3,5
170 ? #6;"time:";VHRS;":";V
    MIN;":";VSECS;"     "
180 GOTO 110
190 REM
200 REM *** subroutine to s
    et time:
210 ? "ENTER HRS,MIN,SEC:";
220 INPUT VHRS,VMIN,VSECS
230 POKE TIMLOK,255:REM so
    it can be called from e
    lsewhere
240 POKE HRS,VHRS
250 POKE MIN,VMIN
260 POKE SECS,VSECS
270 REM reset module's inte
    rnal timers:
280 POKE CNT60,60
290 POKE CSECS,0
300 POKE TIMLOK,0
310 RETURN
```

PRODUCTS FOR ATARI* 400/800
FROM ELCOMP

BOOKS:

ATARI BASIC – Learning by using
An excellent book for the beginner. Many short programs and learning exercises. All important features of the ATARI computers are described (screen drawings, special sounds, keys, paddles, joysticks, specialized screen routines, graphics, sound applications, peeks, pokes, and special stuff). Also suggestions are made that challenge you to change and write program routines.
Order # 164 **$7.95**



GAMES for the ATARI Computer
This book describes advanced programming techniques like player-missile-graphics and use of the hardware-registers. Contains many ready to run programs in BASIC and one called GUNFIGHT in machine-Language.
Order # 162 **$7.95**

SOFTWARE IN BASIC FOR ATARI

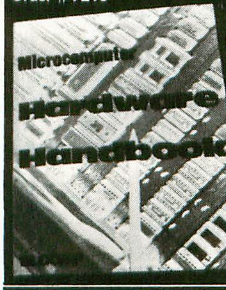
Invoice Writing for Small Business
This program makes writing invoices easy. Store your products in DATA statements with order-number, description, and price. The program later retrives the description and price matching to the entered order-number. The shipping cost and the discount may be calculated automatically depending on the quantity ordered or entered manually. The description to the program tells you how to change the program and adapt it to your own needs. Comes with a couple of invoice forms to write your first invoices on to it.
Order # 7201 cassette version **$29.95**
Order # 7200 disk version **$39.95**

Mailing List
This menu driven program allows the small business man to keep track of vendors and customers. You can search for a name or address of a certain town or for an address with a certain note. 50 addresses are put into one file.
Order # 7212 cassette version **$19.95**
Order # 7213 disk version **$24.95**

Inventory Control
This program is menu driven. It gives you the following options: read/store data, define items, entry editing, inventory maintenance (incoming-outgoing), reports. The products are stored with inventory number, manufacturer, reorder level, present level, code number, description.
Order # 7214 cassette version **$19.95**
Order # 7215 disk version **$24.95**

Programs from Book # 164
The programs from book no. 164 on cassette. (Book included)
Order # 7100 **$29.00**

Game Package
Games on cassette. (Bomber, tennis, smart, cannon fodder, etc.)
Order # 7216 **$9.95**



Microcomputer Hardware Handbook (845 pages)
Descriptions, pinouts and specifications of the most popular microprocessors and support chips.
A MUST for the hardware buff.
Order-No. 29
$14.95

Care and Feeding of the Commodore PET
Eight chapters exploring PET hardware. Includes repair and interfacing information. Programming tricks and schematics.
Order # 150 **$9.95**

ELCOMP

Payment: check, money order, VISA, MASTER-CHARGE, Euroscheck.
Orders from outside USA: add 15% shipping. CA residents add 6.5% tax
*ATARI is a registered trademark of ATARI Inc.
*VIC-20 is a registered trademark of Commodore

ELCOMP Publishing Inc.,
53 Redrock Lane
Pomona CA 91766,
Phone: (714) 623–8314

Books
+
Software
for
ATARI
VIC-20
OSI
SINCLAIR
TIMEX

SOFTWARE IN MACHINE LANGUAGE for ATARI

ATMONA-1
This is a machine language monitor that provides you with the most important commands for programming in machine-language. Disassemble, dump (hex and ASCII), change memory location, block transfer, fill memory block, save and load machine-language programs, start programs. Printer option via three different interfaces.
Order # 7022 cassette version **$19.95**
Order # 7023 disk version **$24.95**
Order # 7024 cartridge version **$59.00**

ATMONA-2
This is a tracer (debugger) that lets you explore the ATARI RAM/ROM area. You can stop at previously selected address, opcode, or operand. Also very valuable in understanding the microprocessor. At each stop, all registers of the CPU may be changed. Includes ATMONA-1.
Order # 7049 cassette version **$49.95**
Order # 7050 disk version **$54.00**

ATMAS
Macro-Assembler for ATARI-800/48k. One of the most powerful editor assemblers on the market. Versatile editor with scrolling. Up to 17k of source-Code. Very fast, translates 5k source-code in about 5 seconds. Source code can be saved on disk or cassette. (Includes ATMONA-1)
Order # 7099 disk version **$89.00**
Order # 7999 cartridge version **$129.00**

ATAS
Same as ATMAS but without macro-capability. Cassette-based.
Order # 7098 32k RAM **$49.95**
Order # 7998 48k RAM **$49.95**

ATEXT-1
This wordprocessor is an excellent buy for your money. It features screen oriented editing, scrolling, string search (even nested), left and right margin justification. Over 30 commands. Text can be saved on disk or cassette.
Order # 7210 cassette version **$29.95**
Order # 7216 disk version **$34.95**
Order # 7217 cartridge version **$69.00**

GUNFIGHT
This game (8k machine-language) needs two joysticks. Animation and sound. Two cowboys fight against each other. Comes on a bootable cassette.
Order # 7207 **$19.95**

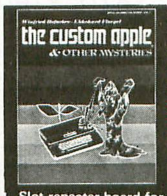
ELCOMP FORTH for the ATARI

ELCOMP FORTH is an extended Fig-Forth-version, Editor and I/O package included. Utility package includes decompiler, sector copy, Hex-dump (ASCII), ATARI Filehandling, total graphic and sound, joystickprogram and player missile.
Order # 7055 disk **$39.95**

Floating point package for ELCOMP FORTH with trigonometric functions (0 – 90°).
Order # 7230 disk **$29.95**

Learn-FORTH from ELCOMP
A subset of Fig-Forth for the beginner. On disk (32k RAM) or on cassette (16k RAM).
Order # 7053 **$19.95**

Expansion boards for the APPLE II



The Custom Apple + Other Mysteries
A complete guide to customizing the Apple Software und Hardware
Order-No. 680 **$24.95**
We also stock the boards which are used in the book "The Custom Apple . . . " (bareboards)
6522 I/O Board No. 605 **$39.00**
EPROM Burner No. 607 **$49.00**
8K EPROM/RAM Board No. 609 **$29.00**
Prototyping board for the Apple II No. 604 **$29.00**
Slot repeater board for the Apple II No. 606 **$49.00**
Order two boards and get the book free !

No. 6153 Learn FORTH for the Apple II (C) $19.95
No. 6154 Learn FORTH for the Apple II (D) $24.95
No. 6155 ELCOMP FORTH for the Apple II (D) $39.95
(Extended Fig-Forth, Editor, graphics package)
No.6156 Floating point for ELCOMP FORTH (D) $29.95

Hardware – ADD-ONS for ATARI

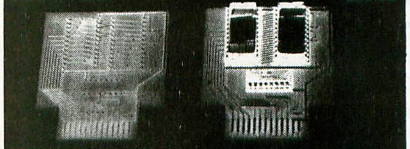
PRINTER INTERFACE
This construction article comes with printed circuit board and software. You can use the EPSON printer without the ATARI printer interface. (Works with gameports 3 and 4).
Order # 7211 **$19.95**

RS-232 Interface for your ATARI 400/800
Software with connector and construction article.
Order # 7291 **$19.95**

EPROM BURNER for ATARI 400/800
Works with gameports. No additional power supply needed. Comes compl. assembled with software (2716, 2732, 2532).
Order # 7042 **$179.00**

EPROM BURNER for ATARI 400/800 KIT
Printed circuit board incl. Software and extensive construction article.
Order #7292 **$49.00**

EPROM BOARD (CARTRIDGE)
Holds two 4k EPROMs (2532). EPROMs not included.
Order # 7043 **$29.95**



EPROM BOARD KIT
Same as above but bare board only with description.
Order # 7224 **$14.95**

ATARI, VIC-20, Sinclair, Timex and OSI

New – for your ATARI 400/800
Astrology and Biorythm for ATARI (cass. or disk).
Order # 7223 $29.95
Birth control with the ATARI (Knaus Ogino)
Order # 7222 cass. or disk $29.95

Books + Software for VIC-20 (requires 3k RAM Exp.)
4870 Wordprocessor for VIC-20, 8k RAM $19.95
4883 Mailing List for VIC-20, 16k RAM $14.95
141 Tricks for VICs - The VICstory Progr. $9.95
4880 TIC TAC VIC $9.95
4881 GAMEPACK I (3 Games) $14.95
4885 Dual Joystick Instruction $9.95
INPUT/OUTPUT Programming with your VIC
Order # 4886 $9.95
4896 Miniassembler for VIC-20 $19.95
4881 Tennis, Squash, Break $9.95
4894 Runfill for VIC $9.95

Universal Experimenter Board for the VIC-20
(Save money with this great board). This board plugs right into the expansion slot of the VIC-20. The board contains a large prototyping area for your own circuit design and expansion. The construction article shows you how to build your own 3k RAM expander and ROM-board.
Order # 4844 **$18.95**

Software for SINCLAIR ZX-81 and TIMEX 1000
2399 Machine Language Monitor $9.95
2398 Mailing List $19.95

Programming in BASIC and machine language with the ZX-81 (82) or TIMEX 1000.
Order # 140 (book) $9.95

Books for OSI
157 The First Book of Ohio $7.95
158 The Second Book of Ohio $7.95
159 The Third Book of Ohio $7.95
160 The Fourth Book of Ohio $7.95
161 The Fifth Book of Ohio $7.95

151 8K Microsoft BASIC Ref. Man. $9.95
152 Expansion Handbook for 6502 and 6802 $9.95
153 Microcomputer Appl. Notes $9.95

Complex Sound Generation
New revised applications manual for the Texas Instruments SN 76477 Complex Sound Generator.
Order # 154 **$6.95**

Small Business Programs **Order # 156**
Complete listings for the business user. Inventory, Invoice Writing, Mailing List and much more. Introduction to Business Applications. **$14.90**

PILOT continued from page 27

```
50 R:PILOT COMPOSER
60 R:ANTIC, VOL 1 # 4
70 R:
100 R:     INIT
110 *INIT
120 R:
130 C:$NOTEVALUES=
140 C:$PLAYVALUES=
150 C:$END=!
155 R: REMEMBER THE SPACE BETWEEN EACH S
ET OF CHARACTERS
160 C:$GOODNOTES=C D E F G A B O
170 C:$GOODDURATION=1 2 4 8 S O
180 C:$NOTETABLE=C. 1/ D. 3/ E. 5/ F. 6/
 G. 8/ A. 10/ B. 12/ O. 0/ 1. 1/ 2. 2/ 4
. 4/ 8. 8/ S. 16/
300 R:     FILE
310 *FILE
320 R:
330 T:ENTER DEVICE TO SAVE MUSIC ON
340 T:D=DISK, C=CASSETTE
350 A:$D
360 R:NEXT, CHECK TO SEE IF CASSETTE
370 M: C
380 CY:$FILESPEC=C:
390 JY:*FILEDONE  [IF CASS JUMP OUT
400 M: D
410 TY:ENTER FILE NAME
420 AY:$FILE    [GET FILE NAME
430 CY:$FILESPEC=$D:$FILE
440 TN:I DON'T KNOW THAT DEVICE
450 JN:*FILE
460 *FILEDONE
470 T:} [ESC-CTRL-CLEAR .. CLEARS SCREEN
500 R:       INSTRUCTIONS
510 *INSTRUCTIONS
520 R:
530 T:
540 T:NOTES ARE: C D E F G A B
550 T:       AND O FOR OFF
560 T:
570 T:DURATIONS ARE:
580 T:       1=WHOLE      2=HALF
590 T:       4=QUARTER    8=EIGHTH
600 T:       S=SIXTEENTH  O=NONE
610 T:
620 T:ENTER & TO QUIT
630 T:
700 R:      ENTER
710 *ENTER
720 R:
730 C:#A=#A+1
740 POS:1,12
750 T:ENTER 4 NOTES + DURATION FOR CHORD
 #A [SPACE, ESC-CTRL-LEFT
760 POS:17,15
770 A:$NOTES
780 M:&
790 JY:*ENDER
800 EY:
810 U:*CHECKNOTES
820 SO:20  [BEEP ON COMPLETION
830 PA:7
840 SO:0
850 WRITE(#A=10):$FILESPEC,$PLAYVALUES
860 C(#A=10):#A=0
870 J:*ENTER
900 R:       ENDER
910 *ENDER
920 R:
930 C:$PLAYVALUES=$PLAYVALUES!
940 WRITE:$FILESPEC,$PLAYVALUES
950 CLOSE:$FILESPEC
960 T:
970 T:       SAVED IN FILE $FILESPEC
980 T:
990 T:         SESSION ENDED
1000 E:
1100 R:        CHECKNOTES
1110 *CHECKNOTES
1120 R:
1130 A:=$NOTES [MOVE $N. TO ACCEPT
1140 MS:,      [MATCH ON 1ST BLANK
1150 A:=$RIGHT!/[ADD/,MOVE TO ACCEPT
1160 C:#C=O     [SETS NOTE COUNTER TO O
1170 C:$NOTEVALUES=
1180 C:#G=O
1190 *LOOP
1195 R: TWO RIGHT ARROWS AND COMMA
1200 MS:,     [SKIP 2 SPACES TYPE ESC-CTR
L-R. ARROW
1210 CN(#G=O):$PLAYVALUES=$PLAYVALUES$NO
TEVALUES
1220 POSN(#G=O):2,22
1225 R: 38 BLANKS UP ARROW NEXT LINE
1230 TN(#G=O):
          [ESC-CTRL-UP
1240 EN:
1250 MS:$RIGHT[MATCH W/O 1ST LETTER
1260 C:$SAVE=$MATCH [SAVE ALL
1270 A:=$LEFT  [$L. HAS BLANK+LETTER
1275 R: RIGHT ARROW & UNDERLINE NEXT LIN
E
1280 MS:_     [SKIP BLANK & LETTER
1290 R:$LEFT HAS THE LETTER WE NEED
1300 C:$NOTE=$LEFT
1310 U:*TRANSLATE
1320 A:=$SAVE  [PUT ALL IN BUFFER
1330 J:*LOOP
1400 R:         TRANSLATE
1410 *TRANSLATE
1420 R:
1430 C:#C=#C+1
1440 E(#C=7):
1450 A(#C<5):=$GOODNOTES
1460 A(#C=5):=$GOODDURATION
1470 M:$NOTE
1480 POSN:2,22
1490 TN:ERROR IN THIS VALUE: $NOTE
```

```
50 R:PILOT PLAYER
60 R:ANTIC, VOL 1, # 4
70 R:
80 R:
300 R:      FILE
310 *FILE
320 R:
330 T:ENTER DEVICE TO PLAY MUSIC FROM
340 T:D=DISK, C=CASSETTE
350 A:$D
360 R:NEXT, CHECK TO SEE IF CASSETTE
370 M: C
380 CY:$FILESPEC=C:
390 JY:*FILEDONE  [IF CASS JUMP OUT
400 M: D
410 TY:ENTER FILE NAME
420 AY:$FILE     [GET FILE NAME
430 CY:$FILESPEC=$D:$FILE
440 TN:I DON'T KNOW THAT DEVICE
450 JN:*FILE
460 *FILEDONE
470 T:} [ESC-CTRL-CLEAR .. CLEARS SCREEN
1000 R:      TEMPO & PLAY
1010 R:
1020 R:      TEMPO
1030 *TEMPO
1040 T:} [ESC-CTRL-CLEAR CLEARS SCREEN
1050 POS:9,5
1060 T:PLEASE ENTER A TEMPO
1070 T:
1080 T:            256 = Adagio
1090 T:            128 = Andante
1100 T:             64 = Allegro
1110 POS:17,11
1120 *RESTART
1130 A:#T
1140 J(#T=256)+(#T=128)+(#T=64):*READ
1150 T:PLEASE ENTER NUMBER AGAIN
1160 J:*RESTART
1170 R:    READ
1180 *READ
1190 T:
1200 T:      PLAYING FILE $FILESPEC
1210 READ:$FILESPEC,$PLAYVALUES
1220 R:THIS DEMOS WORD PARSING
1230 *LOOP1
1240 C:#N=0
1250 *LOOP2
1260 C:#N=#N+1
1270 A:=$PLAYVALUES
1275 R:NEXT LINE ESC-CTRL-LEFT THEN UNDE
RLINE
1280 MS:_
1290 JN:*READ
1300 C:$PLAYVALUES=$RIGHT
1310 A(#N=1):#A=$LEFT
1320 A(#N=2):#B=$LEFT
1330 A(#N=3):#C=$LEFT
1340 A(#N=4):#D=$LEFT
1350 A(#N=5):#L=$LEFT
1360 A:=$LEFT
1370 M:!
1380 EY:
1390 J(#N<5):*LOOP2
1400 SO:#A#B#C#D
1410 PA:#T/#L
1420 J:*LOOP1
1500 R:SET G FLAG FOR BAD NOTE
1510 CN:#G=1
1520 EN:
1530 A(#C=6):=$NOTE
1540 M(#C=6):!
1550 EY(#C=6):
1560 POSN(#C=6):2,22
1570 TN(#C=6):TOO MANY VALUES:$NOTE
1580 CN(#C=6):#G=1
1590 EN(#C=6):
1600 POS(#C>6):2,22
1610 T(#C>6):TOO MANY VALUES: $NOTE
1620 C(#C>6):#G=1
1630 E(#C>6):
1640 A:=$NOTETABLE
1650 MS:$NOTE.
1660 A:=$RIGHT
1670 MS:/
1680 C:$NOTEVALUES=$NOTEVALUES$LEFT
1690 E:
```

ATARI® Christmas Music!

Compare! 1. Glorious **Four** Part Harmony
2. **Accurate** Song Reproduction
3. Requires **Only Basic** Cartridge

Volume I: Silent Night, Angels From Realms of Glory, We Three Kings, Go Tell It On The Mountain, Joy To The World, O' Come, O' Come Immanuel, Good Christian Men Rejoice, Bring A Torch, Coventry Carol, It Came Upon A Midnight Clear.

Volume II: The First Noel, What Child Is This, Hark The Herald Angels Sing, There's A Song In The Air, O' Little Town Of Bethlehem, Angels We Have Heard, Away In A Manger, While Shepherds Watched, O Come All Ye Faithful, God Rest Ye Merry Gentlemen.

Volume III: Jingle Bells, All Through The Night, Please Put A Penny, We Wish You A Merry Christmas, O' Sanctissima, Wassailing Song, Deck The Halls, O Christmas Tree, Good King Wenceslas, I Saw Three Ships.

Specify: **Version 1** (8/24K Tape - 16/32K Disk)
Version 2 (32K Tape - 40K Disk)

* Prices:	Any Single Volume	Any Two Volumes	All Three Volumes
Tape -	$12.95	$21.95	$29.95
Disk -	14.95	25.95	34.95

Add $2.00 per volume ordered for lyrics.

B.I.G. Software
533 Airport Blvd. #518
Burlingame, Ca. 94010
(415) 347-1063

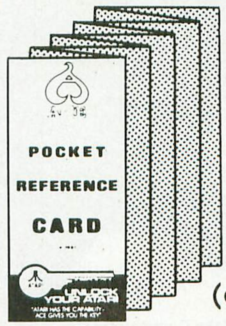
ATARI is a registered trademark of Atari, Inc.
* California residents add 6½% sales tax.
Dealer inquiries invited.

GETTING SERIOUS ABOUT PROGRAMMING YET?!?

ATARI®

UNLOCK YOUR ATARI

WITH THE WORLD'S MOST COMPLETE POCKET PROGRAMMING AID.



16 comprehensive pages

- error codes
- basic commands with abbreviations
- peek and poke locations
- internal codes
- machine language aides
- much, much more!!

Get Yours Now only **$9.95**

(dealer ad space available)

ORDER NOW-CASH, CHARGE OR C.O.D.

ACE

ADVANCED COMPUTING ENTERPRISES
5516 ROSEHILL A-2
SHAWNEE MSN., KS 66216
(913) 262-2875 • (913) 631-4180

MasterCard VISA

ATARI IS A REGISTERED TRADEMARK OF ATARI INC.

VERVAN

10072 Balsa St., Cucamonga, Ca 91730

Utility Programs for the ATARI 400/800

CASDIS - machine language program that allows you to transfer most "boot" tapes and cassette data files to disk. No special programming knowledge required. Programs that normally read cassette files during execution may be up-loaded to disk and operate normally without software modifications. Only difference is the program is on disk and loads at disk transfer speeds
Diskette only $25

CASDUP - machine language program that allows you to copy most "boot" tapes and cassette data files as easily as you normally copy a BASIC tape
Cassette only $20

FULMAP - (avail. late '82) machine language utility package for developers of BASIC programs Features: variable cross reference generator lists all program variables alphabetically with line numbers which reference them; line number cross reference generator which tells you how and where all line numbers are used; address utility which lists all indirect address references and tells you where they are used. This program resides in high memory and accessed from BASIC by entering "MAP". All outputs can be dumped to a printer
Cassette or Diskette $40

DISASM - machine language program that allows you to disassemble machine language programs. You can load a file from cassette or disk and the program will display all the file's addresses and their contents in hex, ASCII (if any), and 6502 op code mnemonics. All outputs can be dumped to a printer
Specify Cassette or Diskette $25

DISDUP - machine language program for sector level copying of disk information. User may specify single sector, range of sectors or all sectors on a disk to be copied. Copies may be made with or without a read verify. Sectors which cannot be read from or written to are displayed on the screen and optionally to a printer
Diskette only $25

Please add $2 shipping & handling per program. California residents add 6% sales tax

Back-up policy: Our disks are protected against casual copying, but we appreciate your potential need for a back-up copy. If you add $10 to your order for a program, we will send you two (2) copies of the program disk. Our normal replacement guarantee applies to both copies. This offer does not apply to dealer sales.

ATARI 400/800 is a trademark of Warner Communications, Inc. *Dealer Inquiries Welcome*

Memory Expansion Boards for the ATARI* Computer

Completely compatible with ATARI hardware and software
No modifications necessary
Fully assembled and tested
User installable - simply plug it in
Gold connector tabs and sockets
One year warranty

16K Memory board — $ 49.95
32K Memory Board — $ 99.95
48K Memory Board — $199.95

Build Your Own Memory

16K board - no components - $12.50
32K board - no components - $30.00
48K board - no components - $50.00
Add $2 Shipping and handling per board

*ATARI is a trademark of Atari Inc.

Dealer Inquiries Welcome

Tiny Tek, Inc.
P.O. Box 820249 • Dallas, TX 75382-0249
214-373-8926

ERRATA

NEC 8023A does not work with ATARI WORD PROCESSOR as indicated in Vol. 1, Issue 3.

TYPO program page 52, Vol. 1, Issue 3 mis-print line 32260. QLINF should be QLINE.

EXPAND YOUR ATARI 400®

BOOST YOUR ATARI 400® MEMORY FROM 16K OR 32K UP TO 48K

FOR ONLY

$149.95 INTRODUCTORY OFFER

FEATURES:

- Less power consumption
- Professional quality
- Allows long sophisticated programs to be run on the Atari 400®
- Easy installation
- Gold plated edge connector
- 90 day guarantee

SEND CHECK OR MONEY ORDER

Plus $2.50 Shipping and Handling
New York State Residents add 7% Tax

SAR-AN

COMPUTER PRODUCTS

12 Scamridge Curve
Buffalo, New York 14221
(716) 632-3441

Atari® and 400™ are trademarks of ATARI, Inc.

ATTENTION ATARI 400 & 800 OWNERS!

SCREEN SONICS INC. SIDEWRITER

Upgrade

your system with the "Sidewriter" a modular keyboard that gives 400 owners a full typewriter keyboard, 800 owners can now have an extension keyboard that will allow you to sit back with the "Sidewriter" on your lap and type away.

HURRY ORDER YOUR "SIDEWRITER" TODAY!

Sidewriters installed complete for $249.00 (plus shipping)
Sidewriters for the do-it-yourself owners $169.00 (conversion kits $69.00) plus shipping

DEALER INQUIRIES INVITED

FOR MORE INFORMATION
SEND $1.00 TO: SCREEN SONICS INC.

14416 S. OUTER 40 RD. • CHESTERFIELD, MO. 63017
OR CALL (314) 434-0433

Orders outside the United States must be in U.S. currency

"400" OWNERS 48K RAM upgrade installed

ONLY

$159

plus shipping

ATARI 400/800 OWNERS: THERE'S A MICROCONNECTION™ FOR YOU! —

Now you can direct-connect to the telephone with or without the 850™ interface,

there's also a serial port to drive a printer, optional autodial and autoanswer, and smart terminal software! Prices start at $199.50 For more details write or phone:

the microperipheral corporation

2643A - 151st Pl. N.E.
Redmond, WA 98052
(206) 881-7544

master charge

VISA

~~16K~~

~~32K~~

48K

RAM KIT FOR THE ATARI 400/800

USES EXISTING ATARI 8K OR 16K RAM BOARD
KIT INCLUDES ALL PARTS REQUIRED AND COMPLETE INSTRUCTIONS WITH DIAGRAMS

124.95*

SEND CHECK OR MONEY ORDER TO:

NTI™ NEOTECHNIC INDUSTRIES INC.
P.O. BOX 277
REDONDO BEACH,CA. 90277

INDICATE ATARI 400 OR 800 WITH 8K OR 16K RAM

*Calif. residents add 6% sales tax

ATARI,400,800 are Trademarks of ATARI,Inc.

Beat the clock! Outsmart your friends!

WORDRACE

The high-speed dictionary game

At last, an educational game that's really fun to play! You start each turn with a word, 6 definitions, and a counter set to 600 points. As time passes the points tick away. The sooner you pick the **correct** definition, the more points you get, but the sooner you guess wrong, the more points you **lose**.

3 levels of play on one disk:
• Beginner (ages 9-14) • Regular • Challenge
2000 words and definitions
$24.95
ATARI: 32K — disk — BASIC

Now available on cassette for the ATARI 400/800 $19.95
Features: Beginner's and Intermediate WORDRACE

YOU CAN PLAY 3 NEW GAMES WITH YOUR WORDRACE DISKETTE:

The next disk in the **WORDRACE** System. Use it along with your **WORDRACE** disk to play:

- **CLAIM TO FAME** (600 famous people in history)
- **SPORTS DERBY** (600 pieces of sports trivia)
- Plus more vocabulary words

$19.95
Requires **WORDRACE** disk

WORDRACE ACCESSORY #1

Try **ABUSE**
The insult program.

- Funny • Unpredictable • Interactive
- Guaranteed to call you something you've never been called before!

$19.95
ATARI: 40K — disk — BASIC

DON'T ASK
COMPUTER SOFTWARE INC.

2265 Westwood Blvd., Ste. B-150
Los Angeles, California 90064.
(213) 397-8811

Dealer inquiries welcome

Available at your computer store or direct from DON'T ASK
Include $2.00 shipping for each program (Calif. residents add 6% tax)

ATARI is a trademark of ATARI INC.

GOTO DIRECTORY

CALIFORNIA

THE SOFTWARE STORE

11768 West Pico
Los Angeles, CA 90064
213-473-1136

Software for ATARI computers. APX—third party—ATARI books-magazines.

HW COMPUTERS

19511 Business Center Drive
Northridge, CA 91324
213-886-9200

Full line of personal computers. Atari, TRS-80, Fortune, NEC, California Computer Systems. Complete line of software and hardware. Authorized Service Center. Other locations: Westwood, Redondo Beach and Palm Springs.

FOOTHILL COMPUTER

949 West Foothill Blvd.
Upland, CA 91786
714-985-3278

ATARI computers & software—peripherals—books—magazines.

DIMENSIONAL SOFTWARE

3954 Clairemont Mesa Blvd.
San Diego, CA 92117
714-275-4243

Software/books/peripherals/hardware

SOFTWARE EMPORIUM

4500 El Camino Real
Los Altos, CA 94022
415-941-8788

1800 S. Bascom Ave.
Campbell, CA
408-377-9311

Complete software selection for ATARI, Apple, Tandy, IBM, CP/M, books, magazines, games.

SUNSET COMPUTERS

2329 Irving
San Francisco, CA 94122
415-665-7378

ATARI hardware, software, peripherals. Repairs—Northstar, Franklin, Osborne, Kaycomb. Books, magazines.

DATA BANKS

3820 Peralta Blvd.
Fremont, CA 94536
415-790-1060

Hardwdare/software/services/peripherals and a full line of supplies for all ATARI products.

3E SOFTWARE AND SYSTEMS

22708 Mission Blvd.
Hayward, CA 94541
415-537-3637

Complete ATARI support center for hardware and software. ATARI, AXLON, OKIDATA, BIT 3, ALIEN, GROUP, BMC, PID, SYNAPSE, ON-LINE, SWIFTY, EPYX GEBELLI, DATASOFT, BRODERBUND, QS, SIRIUS, OSS, EPSON, ARTWORKX, & MORE.

P.C. COMPUTERS

10166 San Pablo Ave.
El Cerrito, CA 94530
415-527-6044

Software/books.

COMPUTERLAND

1815 C Ygnacio Valley Rd.
Walnut Creek, CA 94590
415-935-6502

Wide selection of software for ATARI. Programming seminars. IBM & Altos computers. Books, magazines.

ELECTRONIC FANTASY

2078 Vallo Fashion Park
Cupertino, CA 95014

ATARI computers & large selection of software. Repairs.

THE SOFTWARE CENTER

4720 Geary Blvd.
San Francisco, CA 94118
415-751-2231

An authorized ATARI dealer with the largest selection of ATARI software in S.F. Cartridges, tapes, discs, memory cards, joysticks, books, magazines, games.

CONNECTICUT

THE COMPUTER CENTER

Millrock Rd.
Old Saybrook, CT 06475
203-385-1587

ATARI hardware & software. Programming seminars, books, magazines, T.I., Northstar & Altos. Programmers wall chart available. Call or write for details.

EASTERN ELECTRONICS

1207 Meriden-Watby Rd.
Milldale, CT 06467

ATARI hardware & software, games, business applications, magazines and books. User group meets here.

FLORIDA

ORANGE BLOSSOM HOBBIES

1975 N.W. 36th St.
Miami, FL 33142
305-633-2522

Hardware/software/peripherals.

IOWA

COMPUTER COUNTRY

725 14th St.
Marion, IA 52302
319-377-9437

ATARI hardware, peripherals, software, games, business applications, magazines, books.

ILLINOIS

COMPLETE COMPUTING

890 E. Roosevelt Rd.
Lombard, IL 60148
312-620-0808

Best selection of ATARI software in western suburbs. 10% discount software club. Superb selection of books, magazines. Basic and advanced programming classes. Excellent technical support for ATARI. Loves bad puns.

MARYLAND

A-BIT-BETTER SOFTWARE

Post Office Box 28
Laurel, MD 20707
301-953-7256

A-Bit-Better Software offers you an extensive variety of exciting programs for your Atari 400/800. Quality is a must, and our reasonable prices give us the edge. Write for free catalog.

FREDERICK COMPUTER PRODUCTS

5726 Industry Lane
Frederick, MD 21701
301-694-8884

ATARI hardware & great selection of software from many manufacturers. ATARI & other peripherals.

MICHIGAN

BINARY CORP.

3237 Woodword Ave.
Berkeley, MI 48072
313-548-0533

ATARI hardware & software. FASTCHIP and The Terminal program. ATARI & other peripherals. Books & magazines.

NEVADA

COMPUTER CENTER

3310 S. Jones, Suite D
Las Vegas, NV 89102
702-873-5055

ATARI computers, peripherals. Large selection of business applications & game software.

NEW JERSEY

EARTHRISE COMPUTER CENTER

6 Green Village Rd.
Madison, NJ 07940
201-377-4084

ATARI, NEC, Franklin computers. Complete software selection. Computer classes.

SOFTWARE ASYLUM, INC.

626 Roosevelt Ave.
Carteret, NJ 07008.
201-969-1900

Also Old Bridge
201-536-1401

We ARE ATARI! NJ's largest retailer of Atari programs for 400/800 models; over 400 programs available from more than 60 manufacturers. Send for our latest catalog — only $2.00 (refundable as credit with purchase).

NEW YORK

LEIGH'S COMPUTER WORLD

212 E. 85th St.
New York, NY 10028
212-879-6257

ATARI hardware, software, books. Magazines, programming classes. Apple & Sinclair.

THE VIDEO CONNECTIONS

27 Merrick Ave.
Merrick, NY 11566
516-576-5050

ATARI software, books, magazines. Mention this listing for a special discount on ATARI software.

OHIO

BARNHART STORES

548 N. Main
Urbana, OH 43078
513-653-7257

Hardware/software/service/peripherals.

VIDEO DEPOT

1278 Euclid Ave.
Cleveland, OH 44115
216-696-3588

ATARI computers. Excellent selection of ATARI software. Books & magazines.

COMPUTER CORNER

5104 Mayfield Rd.
Lyndhurst, OH 44124
216-473-5010

ATARI hardware & software for games & business applications. Basic and advanced programming classes. Authorized service center for ATARI, Commodore, Eagle and Epsom.

DATA MANAGEMENT SYSTEM

2979 West Market St.

201 S. Main St.
Akron, OH 44308
216-666-3226

Factory authorized dealer/repairs. Software development marketing.

OKLAHOMA

THE COMPUTER MART

9726 East 42nd St.
Tulsa, OK 74145
918-664-8452

ATARI computers & software. Northstar & Victor computers. Books & magazines.

UTAH

THE SOFTWARE HUT

470 E. 200 South
Salt Lake City, UT 84111
801-355-0066

Hardware/software/peripherals.

WASHINGTON

PROGRAMS PLUS

16874 Southcenter Pkwy.
Tukwila, WA 98188
206-575-1375

Software, books & peripherals.

ROB ROY COMPUTER

1109 W. Yakima Ave.
Yakima, WA 98902
509-575-7704

Hardware, software, repair. ATARI & Northstar computers. Business, applications & game software. Authorized ATARI service.

SUBSCRIBE

Antic™
The ATARI® Resource

I would like to subscribe to ANTIC for
- ☐ 1 Year (6 issues) *Please enclose $15.*
- ☐ 2 Years (12 issues) *Please enclose $27.*

Make check or money order payable to:
ANTIC Publishing
297 Missouri Street
San Francisco, CA 94107

Please check where appropriate:

I own:
- ☐ 400
- ☐ 800
- ☐ Modem
- ☐ Cassette
- ☐ Disk
- ☐ Interface
- ☐ Printer
- ____ K Memory

Start with issue # ________ .

Foreign rates via air mail payable in U.S. dollars are:

	6 issues	12 issues		6 issues	12 issues
CANADA	$25	$46	SOUTH AMERICA	$33	$63
MEXICO	$27	$50	AUSTRALIA	$40	$76
EUROPE	$33	$63	NEW ZEALAND	$40	$76

Foreign rates via surface payable in U.S. dollars are $21 for six issues, $39 for twelve issues.
Single copy back issues available at $4.00.

____
Name

____
Street

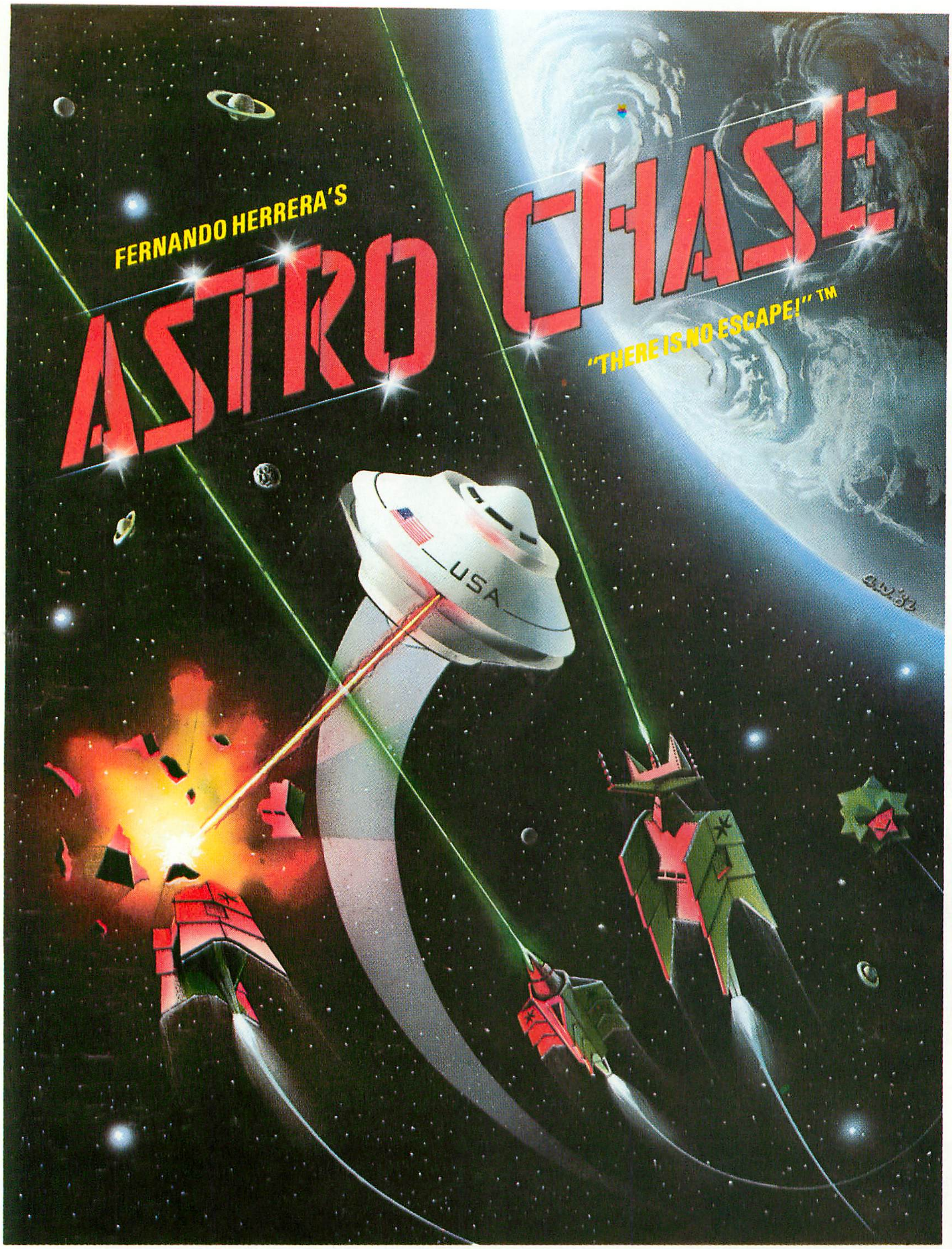
____
City, State Zip

!!COMING NEXT ISSUE!!

HOLIDAY ISSUE
BUYERS GUIDE
PRODUCT SURVEY
CHRISTMAS GREETING FOR YOUR SCREEN
HOLIDAY LIGHT SHOW
DRAGON SMOKE BY BOB ALBRECHT

ADVERTISERS

NEW



COSMIC SQUEEZE™
by Jason Pomerantz
graphics by Fernando Herrera

To Order:
Call TOLL FREE 800-223-1545
nationwide except in New York
phone 212-889-1073

MasterCard VISA

When using your MASTER CARD or VISA please have ready:
your card number & expiration date.
Send your check or money order for $29.95* plus $2.00 shipping and handling to:
First Star Software, Inc.
22 East 41st Street
New York, NY 10017

When ordering, please specify DISK or TAPE
*New york residents please add 7% sales tax.

Fernando Herrera Strikes Again!

In *ASTRO CHASE*, his first program since winning the ATARI® STAR AWARD, Fernando delivers: Fast, Super, ARCADE GRAPHICS ★ 100% MACHINE LANGUAGE ★ SMOOTH SCROLLING UNIVERSE (different for each new game) ★ AWESOME SOUND EFFECTS ★ NON-STOP ACTION ★ MULTI-DIRECTIONAL FIRING ★ INTELLIGENT ENEMY ATTACK SHIPS ★ SINGLE THRUST PROPULSION™ ★ INVISIBLE FORCE FIELDS ★ MULTIPLE SKILL LEVELS ★ Many more INNOVATIVE and UNIQUE features.

FOR THE
ATARI®
HOME COMPUTER
32K

"THERE IS NO ESCAPE!"

Ask for FIRST STAR SOFTWARE at your local dealer.
Retail and Distribution inquiries invited.

DISK AND TAPE
$29.95*
(CARTRIDGE COMING SOON)

"When being first counts...we're number one"

FIRST STAR SOFTWARE INC™

ATARI® is a registered trademark of Atari, Inc.
Single Thrust Propulsion™ is a registered trademark of First Star Software, Inc.
ASTRO CHASE, "there is no escape!"™ is a registered trademark of First Star Software, Inc.
COSMIC SQUEEZE™ is a registered trademark of First Star Software, Inc. ©, 1982 First Star Software, Inc.