NO. 33
AUGUST 1985

U.S.A. $3.00
CANADA $3.50

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

A.N.A.L.O.G. COMPUTING™

The Atari and Music

plus:

BASIC Bug Exterminator

C.COM

and Summer CES Report!

0 74470 12385 08

$SAVE BIG BUCKS$ ON ALL OSS™ PRODUCTS

OSS, in response to Atari's commitment to quality software at affordable prices, is announcing a major price reduction on its complete line of quality systems software.

BASIC XE

130XE™ Owners: Use all the memory you paid for! Get all the performance you should have gotten in the first place **$79.00**

$100 of Atari BASIC™ options free: Player/Missile Graphics, string arrays, auto line-numbering, structured programming, super-fast SORT and more!

ACTION!

Discover the benefits of structure and speed. Use ACTION!, the language ANALOG and other magazines continue to rave about! **$79.00**

Best program editor, fastest compiler, comprehensive library, fastest programs, and easy debugging all instantly at your command.

MAC/65

Edit, assemble, and debug programs the fastest and easiest way. Buy the Macro Assembler used by more leading programmers than any other! **$79.00**

Easy-to-use, familiar line-oriented editor. Powerful macros and assembler controls. Multi-breakpoint debugger with independent screen display.

THE WRITER'S TOOL

Buy it today. Write a letter tonight. The easiest word processor to learn is even easier to use! **$69.00**

With full tutorial, professional custom printing, spelling checker, form letters, and much more.

BASIC XL

Write better and faster BASIC programs with ease! Join *thousands* of pleasantly surprised Atari owners **$59.00**

Most popular advanced BASIC. For all Atari 8-bit computers. Super features: Player/Missile Graphics, renumber, DOS commands, much more.

OSS TOOLKITS
Even More for Your Money

Get a headstart on writing your own programs. Learn from the useful examples in each ToolKit. Include the numerous ToolKit routines in your own work. Give your programs to your friends, user groups, magazines, etc.

ToolKits for BASIC XL, ACTION!, and MAC/65 are now only ...... **$29.95 each**

This brief ad can only begin to tell you about the benefits and features of these and other outstanding OSS products. For a free brochure, see your dealer, or call or write OSS.

**Atari 130XE™ and Atari BASIC™ are U.S. registered trademarks of Atari Corporation.*

OSS™

Optimized Systems Software, Inc.
1221B Kentwood Avenue, San Jose, California 95129 (408) 446-3099

CIRCLE #101 ON READER SERVICE CARD

NO. 33 AUGUST 1985

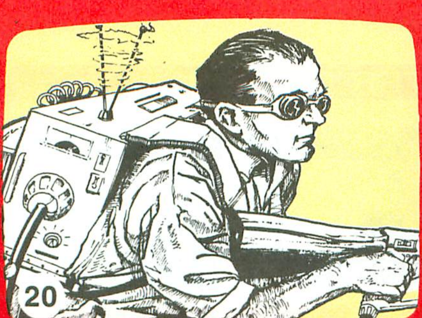
THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS ™

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

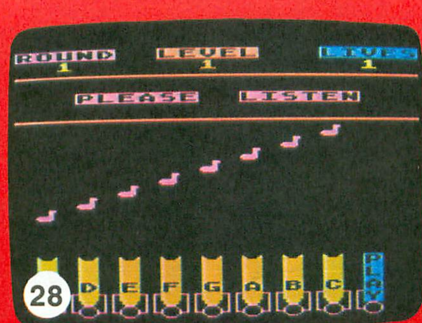
FEATURES

REVIEWS

COLUMNS



20



28

39

ANALOG COMPUTING (ISSN 0744-9917) is published monthly for $28 per year by ANALOG 400/800 Corp., 565 Main St., Cherry Valley, MA 01611. Second class postage paid at Worcester, MA and additional mailing offices.

WIN AN ATARI 520ST!

ANALOG Computing is giving away a new Atari 520ST computer! That's right, one lucky **TCS** subscriber will win one of Atari's fantastic 16-bit machines (a $599 value), just for joining the **ANALOG Computing TCS**!

Contest rules.

Everyone who joins the **ANALOG Computing TCS** before midnight Eastern Daylight Time, July 31, 1985, will be automatically entered into a special drawing that will be held on August 14, 1985.

The contest is open to all **TCS** subscribers with active accounts at the time of the drawing. **ANALOG Computing** employees are not eligible to enter.

If you're already a **TCS** subscriber, you don't have to do anything to enter. If you have a friend who isn't a **TCS** subscriber yet, tell them about the giveaway!

This drawing is our way of saying "thanks" to all the people who have made the **ANALOG Computing TCS** a terrific success. Good luck!

*For information on joining the **ANALOG Computing TCS**, see the ad elsewhere in this issue.*

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

A.N.A.L.O.G. COMPUTING™

P.O. BOX 23 WORCESTER, MASSACHUSETTS 01603 (617) 892-9230

ANALOG COMPUTING STAFF

Editors/Publishers
MICHAEL J. DES CHENES
LEE H. PAPPAS

Managing Editor
JON A. BELL

Production Editor
DIANE L. GAW

Contributing Editors
DONALD FORBES
BRADEN E. GRIFFIN, M.D.
STEVE PANAK
RUSS WETMORE

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
JIM DUNION

Contributing Artists
GARY LIPPINCOTT
LINDA RICE

Technical Division
CHARLES BACHAND
TOM HUDSON
TONY MESSINA

Advertising Manager
MICHAEL J. DES CHENES

Circulation Manager
PATRICK J. KELLEY

Accounting
ROBIN LEVITSKY

Production/Distribution
LORELL PRESS, INC.

Contributors
ANGELO GIAMBRA
RICH MOORE
CRAIG PATCHETT
MARIO PERDUE
CHUCK ROSKO
JON SNYDER
KARL E. WIEGERS

U.S. newstand distribution by
Eastern News Distributors, Inc.,
111 Eighth Ave., New York, NY 10011

WHERE TO WRITE

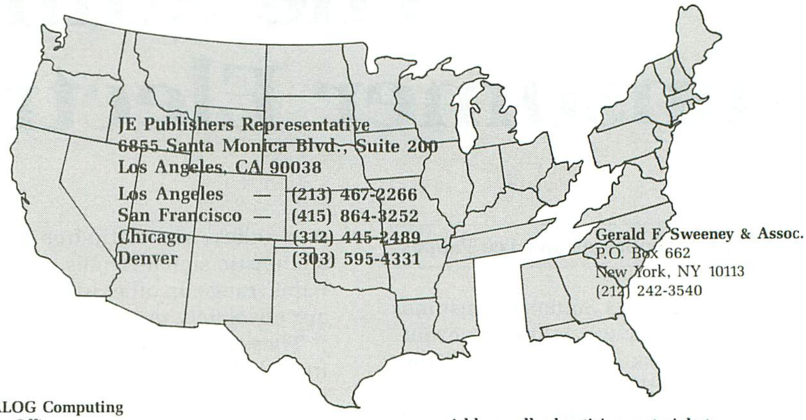
All editorial material (programs, articles, letters and press releases) should be sent to: Editor, **ANALOG Computing**, P.O. Box 23, Worcester, MA 01603.

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ANALOG Computing**, 100 Pine Street, Holmes, PA 19043, or call 1-800-345-8112 (in Pennsylvania, call 1-800-662-2444).

Correspondence concerning a regular column should be sent to our editorial address, with the name of the column included in the address.

An incorrectly addressed letter can be delayed as long as two weeks before reaching the proper destination.

ADVERTISING SALES



ANALOG Computing
Home Office
Michael DesChenes
National Advertising
(617) 892-9230

Address all advertising materials to:
Michael DesChenes—Advertising Production
ANALOG Computing
565 Main Street, Cherry Valley, MA 01611

SUBSCRIPTIONS

All subscriptions should be addressed to:

ANALOG Computing
P.O. Box 625, Holmes, PA 19043
or call our toll-free number:
1-800-345-8112
(in PA 1-800-662-2444)

Foreign subscriptions must be made payable in U.S. funds.

Subscription prices in the U.S.: $28 for 1 year; $52 for 2 years; $79 for 3 years.

Subscription prices in Canada: $36 for 1 year; $62 for 2 years; $89 for 3 years.

Foreign subscriptions: $39 for 1 year; $72 for 2 years; $99 for 3 years.

Airmail is available for foreign subscriptions at an additional $50 per year.

Please allow four to six weeks for delivery of your first issue.

PERMISSIONS

No portion of this magazine may be reproduced in any form without written permission of the publisher. Most programs are copyrighted and are not public domain. User groups should contact the publisher if they plan to place one of these programs on any type of public-accessed bulletin board or disks.

AUTHORS

When submitting articles and programs, program listings should be provided in printed *and* magnetic form, if possible. Articles should be furnished as typed or printed copy in upper and lower case with double spacing. If submissions are to be returned, please send a self-addressed, stamped envelope.

A Visit to Chicago

The Summer Consumer Electronics Show

by Arthur Leyenberger and Lee Pappas

Overall, the 1985 Summer Consumer Electronics Show was a low-key event. "What?" you ask in amazement. "How can a show with an attendance of over 100,000 people, thousands of square feet of exhibit space and hundreds of high-tech companies be low-key?" Well, to be honest, there just isn't much exciting news to report.

Several companies bowed out at the last moment. Electronic Arts, Infocom, CBS Software and other companies decided it would be more cost-effective not to have an exhibit at the show. In several cases, these companies decided, instead, to rent hotel suites at downtown Chicago hotels to court distributors, retailers and the press.

Even Atari was not going to attend the show, until, at the last minute, CES promoters made Atari an offer they couldn't refuse: a free booth. With the help of some members from the Chicagoland Atari User's Group, Atari made the best of their small conference room exhibit area. However, for the second time in a row, Atari had the most exciting product at the show. This time it was the CD-ROM.

The show-stopper.

Many of us are familiar with the new digital audio disk, otherwise known as a CD (Compact Disk). These laser "records" can contain up to 73 minutes of pop-, click- and static-free music with a fantastic signal-to-noise ratio and dynamic range. In other words, these disks are *absolutely quiet*.

These same disks can also contain an incredible amount of data. Up to 550 megabytes of data can be stored on one side of a CD. That translates into the equivalent storage of over 1500 IBM floppy disks.

Tom Rolander, Vice President of Engineering for Activenture, a California optical media company, demonstrated a prototype of the new Atari CD-ROM player. This innovative product consisted of the entire 20-volume set of *Grolier's Encyclopedia* on one CD, occupying 58 megabytes. Another 50 megabytes on the disk consisted of cross-indexing information. Every word in the text was cross indexed for retrieval purposes.

Screen display for encyclopedia on the CD-ROM.

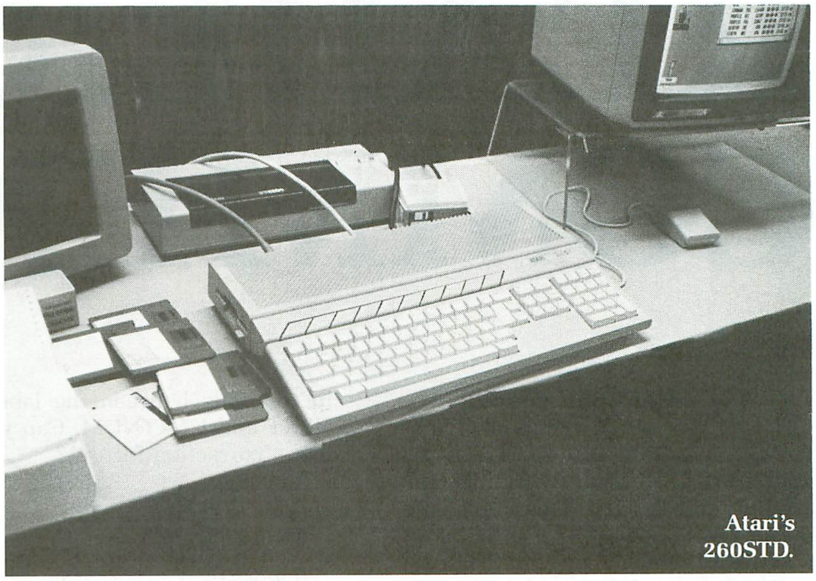
Amazingly, the text and indexing information (approximately 100K bytes) consumed only one-fourth of the disk's capacity. Rolander said that the remaining space was large enough to hold approximately 5000 high-resolution 720x 350 pixel images.

The CD-ROM was connected to the direct memory access (DMA) port of an Atari 520ST, through an interface. This DMA port allows data transfer rates of up to 1.3 megabytes per second, which yields incredibly fast access to the CD.

The CD-ROM encyclopedia lets you request all references to a particular subject or just browse through the text. Having all of the information on one CD, rather than searching through 5 cubic feet of books, is an unbelievable convenience. The speed of the search is due to the program's not looking through the actual text, but searching, instead, in the indexing portion of the information.

Regardless of what you search for, the computer screen will tell you how many entries were retrieved and allow you to look at each one. When presented, the information is displayed in black letters on a white background, with the key word highlighted in green. Once information is found, the text can be paged forward or backward, copied to a floppy disk for future use, or sent directly to a printer.

Rolander said that Activenture was developing the technology under an exclusive agreement with Atari, and the



Atari's 260STD.

decision on who would market the product had not yet been made.

The targeted price for this innovative product is $500, but Atari is shooting for about $350. In fact, Jack Tramiel told me that his target price was $150.

Consider this: an inexpensive audio CD player can be purchased for as little as $200 today, and the CD-ROM/ST interface may cost as much as $100. Therefore, an under $500 price is very realistic. The software (encyclopedia, recipe, historical stock market price, legal databases, etc.) is expected to sell initially for $100 to $150, but could come down to the $50 range if significant quantities are made.

Other Atari news.

In addition to the CD-ROM, which is expected to be available in the fall, Atari also announced a new computer. Called the 260STD, it contains 256K of RAM memory and has TOS (The Operating System), which includes GEM, in ROM.

The computer has a built-in, single-sided disk drive toward the back of the left side. The disk drive is a 500K (unformatted)/360K (formatted), 3½-inch micro drive. Unlike the 520ST model, a 260STD has a TV video output jack on the rear of the machine. The 260STD looks just like the 520ST, only slightly deeper (similar to comparing a 1200XL to an 800XL).

The 260STD is the so-called low-end, "mass market" computer that Atari has mentioned in the past. It will be sold at such places as K-Mart and Toys 'R' Us. The machine is expected to be available in the fall and will retail for $499. This price doesn't include a monitor.

The 520ST is slated to be bundled with disk drive and monochrome monitor. The retail price of the package is said to be $800, and the system will only be sold at computer specialty stores.

Sig Hartmann, Atari's President of Software, disclosed to me that the 520ST computer will never have GEM in ROM. It will always be "soft loaded." Currently the operating system called TOS and GEM, together, require about 220K of RAM.

Dave Duberman, Atari's User Group Coordinator, told me that Atari is trying to get the size down to under 200K. If this is accomplished, the 520ST will contain approximately 300K of RAM after TOS is loaded, versus the 256K of RAM on the 260STD. In terms of available memory, then, the two machines are essentially identical, especially with regard to the commercial software market.

However, in talking with Neil Harris, the publisher of Atari's *Explorer* magazine, it seems that GEM may become available in ROM for the 520ST. The 520ST, according to Harris, will contain sockets for the ROM chips, so that if (or when) they become available, they can be added. This would give the 520ST twice the memory capacity of a 260STD. Only time and Atari can tell if this will happen.

Atari's third major hardware product was a direct-connect, auto-dial, auto-answer modem that is Hayes compatible. The XM301 will sell for under $50 and plugs directly into the serial port on all Atari 800, XL and XE computers.

It takes its power from the serial port (so there's no AC cord to get in the way) and routes the dial tone through the TV speaker. The modem comes with **XE-Term**, a modem program written by Russ Wetmore that uses XMODEM protocol for up- and downloading.

The XM301 also comes with over $200 of free time on such on-line services as the Source and CompuServe. There is reportedly a similar 1200-baud modem that will sell for under $100.

New software from Atari.

Atari was again showing the **Silent Butler**, a home financial program for the XE. Available in July and costing under $50, this program lets you keep records on three checking and two savings accounts. It also allows you to categorize your expense items, for later use in filing your income tax.

Your own checks may be used on your dot-matrix or letter-quality printer, with the supplied plastic holder. **Butler** also provides a tickler file for appointments.

Atari was showing another product exclusively for the 130XE. The **VIP Professional** is a Lotus 1-2-3 clone that will sell for under $100 and should be avail-

(continued on page 10)

READER COMMENT

More RAMDISK space.

Having just purchased a new Atari 130XE, I tested the new Atari DOS 2.5 with RAMDISK. The new DOS is really quite good and *very* compatible with DOS 2.0S.

My main complaint is with the RAMDISK. Both DUP.SYS and MEM.SAV take up valuable space on this 64K disk. The 130XE and the 800XL have an additional 16K of unused memory. To free up more space on the RAMDISK, I ran Robert Luce's BASIC program **XL-DOS** (issue 24) with DOS 2.5, creating an XE(?)-DOS.

Next, I formatted drive 8 (the RAMDISK). Result: 499 free sectors on the RAMDISK, plus active DUP.SYS and MEM.SAV. The original version is still needed when using the translator disk which occupies this space.

Owners of the Atari 800 with FASTCHIP who have upgraded to XL or XE machines will be happy to learn that running **Home-made Translator** (issue 32) will copy the old operating system *complete* with the faster floating point routine. This will substantially speed up the number-crunching of these new machines.

Better yet, OMNIVIEWXL is available now for the XL machines, which gives an 80-column display plus the FASTCHIP and old OS. I hope it will also be made available for the XEs.

Yours truly,
James Evangelow
New York, NY

B-Line assembly fix.

I really appreciate the assembly listings of programs in your magazine. Typing them in or just reading them is a great way to learn both how the program works and the language at the same time.

There is an error in **B-Line** in issue 29, in the code between the labels GETNEXT and NEXTNUM. Can you publish a correction for this?

Sincerely,
Peter Wilson
Richmond, VA

If looks as if our proofreaders missed a minor printer error. The code should read:

```
STA CIX
JSR POINT      ;INIT INBUFF
TYA
PHA            ;SAVE Y
```

Setting the Atari Clock.

This short program will append a file to the **Atari Clock** (issue 31), to autorun SETCLOCK immediately after the **Clock** boots up. It saves you the task of LOADing SETCLOCK every time you boot up the **Clock**.

To create the file, make sure the autorun clock is already on your disk, then type in the following:

```
10 REM AUTORUN FOR SETCLOC
K.BAS
20 DIM N$(120):T=0
30 FOR X=1 TO 114:READ N:N
$(X)=CHR$(N):T=T+N:NEXT X
40 IF T<>9365 THEN ? "DATA
 IS INCORRECT":END
50 OPEN #1,8,0,"D:RUNCLOCK
":? #1;N$;
60 ? "DATA WRITTEN...":END
70 DATA 255,255,0,6,101,6,
160,158,162,36,142,197,2,1
42,23,208,142,240,2
80 DATA 189,28,6,145,88,13
6,202,16,247,169,13,141,74
,3,96,56,29,53,51,50
90 DATA 8,17,22,16,17,9,26
,50,53,46,2,36,26,51,37,52
,35,44,47,35,43,14,10
100 DATA 0,0,0,0,0,0,0,0,0
,104,141,240,2,168,170,145
,88,200,192,129,144,4
110 DATA 189,94,6,232,192,
140,208,241,142,74,3,202,1
42,197,6,96,0,44,47,33
120 DATA 36,41,46,39,226,2
,227,2,0,6
```

When you're done, RUN the program. When it is written properly, go to DOS and type C, then RETURN. Next, type *RUNCLOCK,AUTORUN.SYS/A* and RETURN. After a few seconds, the autorun will be ready. Boot up the disk and set the clock.

Note that the setclock program must be named SETCLOCK.

Sincerely,
Jim Little
Beaver Falls, PA

The Bushnell mystique?

Just a word or two about Nolan Bushnell. After reading his comments from the June issue of **ANALOG Computing** (31), I couldn't resist responding with a few comments of my own.

As the founding father of the Atari family, and the inspiration behind the innovative and reasonably-priced Atari 800, Mr. Bushnell has always commanded a good deal of respect from me. However, of late, I have begun to wonder if his reputation as an innovator and the Mad Muse of computing is wholly justified.

From the time he sold Atari to Warner Communications, he has loomed ominously on the horizon of the computer industry like some chained djinni, amidst rumors and pronouncements, awaiting his opportunity to loose his genius upon the world and start the real computer revolution.

It seems that there were hints everywhere in the industry magazines about the great things that were going to come from his laboratories once he was released from the constraints of his noncompetition clause that had been a part of his sale agreement with Warner. In most of his interviews, he seemed to imply that he had some Ace up his sleeve that, once played, was going to set the computer industry on its ear.

I have been faithfully waiting several years for something like that to happen. I am still waiting.

In the meantime, I have seen him involved in several enterprises, including fast-food restaurants (Pizza Time Theater), arcade games (Sente), household robots (Androbot) and now, mechanical toys. None of the above have had much impact on the computer industry.

I'm still wondering when I am going to see all those technological wonders that he keeps referring to. (Oh, by the way, most of the above-mentioned enterprises seem spectacular only by their remarkable lack of success.) Once again, in the interview with Mr. Leyenberger, I read about great things which may be in the offing (most of which cannot, unfortunately, be revealed at this time).

When are we going to be done with this mystique about our Mr. Bushnell? Fifteen years ago, when he had some very interesting things to offer the computer industry, his name seemed magical. Since then, his performance does not seem to justify the attention that we continue to shower upon him.

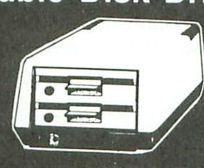
His comments about the new ST series of machines from Atari are a case in point. For example, he says: "It's a product line that makes me wonder why anyone would spend six or seven hundred dollars for a computer, when, for a few hundred dollars more, they could get a Macintosh."

My question to him is: if an Atari 520ST and a 10-megabyte hard disk drive (with all the additional power and capabilities that an ST apparently offers) can be had for $1000 or so, why in the world would anyone *want* to spend their $2500 on a Macintosh with its limitations (unacceptably small screen, monochrome only, RAM eaten up by operating system and fonts, mouse controller [ever try to *draw* with a mouse?], inefficient hard disk handling, expensive upgrades and peripherals)?

In addition to that, he then stated that the GEM system isn't as good as MacPaint! Well, why should it be? Since GEM is a user interface, and MacPaint is a drawing program, how can they even be compared? When drawing programs are designed for the ST, then they can be properly evaluated. Until then, what is he talking about? It's enough to make one wonder whether Mr. Bushnell is even aware of what the ST is.

All I can suggest is, maybe if he spent less time listening to Alexander Graham Bear, and more time watching the industry, he might have more enlightening comments to make.

Sincerely,
Bob deWitt
Provo, UT



Crashing at Vandenberg.

I have a problem with Tom Hudson's **Adventure at Vandenberg A.F.B.** I have an 800XL computer with the game stored on disk (an Atari 1050 disk drive). In typing the game in long months ago, I feel I was accurate. It checked with CHECKSUM DATA. I have visually rechecked all the lines, even had my son read back some. But it has a problem somewhere.

As the program waits for my input (What Next?) with the blinking cursor, every now and then it will just crash. The whole thing goes into Never-Never Land. It also happens as I'm entering an answer to What Next? This has happened every time I've tried to play. I can't get past the third or fourth entry, sometimes even less.

I am not a technical person, but I do feel the problem must be in the blinking cursor machine language routine. Has this problem come up before? Could you point out some places in the program which I can check again? Is there any solution?

Thank you,
Howard Gebeaux
Asheboro, NC

To make ***Adventure at Vandenberg*** *work on an XL or XE computer, delete Line 12 and make the following changes:*

```
13 Q=USR(1648,1680):POKE C
16,112:POKE 53774,112:GOTO
 75
94 CLOSE #C1:POKE 559,34:G
OTO 55
```

This will eliminate the flashing cursor, but otherwise, the game will play normally.

—*Ed.*

Send your letters to:

Reader Comment

P.O. Box 23
Worcester, MA 01603

continued from page 5

able by Christmas. It's really a spreadsheet program containing database-type operations like sorting and searching. The program uses windowing and pull-down menus, and will be functionally equivalent to a planned ST version. The **VIP Professional** is being developed by VIP Software.

Another product introduced at CES was the Atari **Planetarium**. This under-$50 program will be out by fall, with over 1200 stars and 300-plus deep sky objects. Additional features include a list of constellations, a world map pinpointing your viewing location, and the ability to put your observation point anywhere on the Earth, at nearly any point in time. Solar tracking, eclipses and plotting Halley's comet are also possible.

The **AtariWriter Plus** word processing program will be fully compatible with the current version. However, it will feature 80-column capability, count words, automatically add line numbers so files can be merged into BASIC programs, and include a spelling checker. This disk-based program will be available for under $50 by fall. A major feature touted is the ability for the user to "build" their own printer driver.

One of the most interesting new Atari software products is a GEM look-alike program that will run on all 8-bit Atari computers. Also developed by VIP Software, this is appropriately called **GEM Desktop**. The $50 price tag includes a mouse. It should be out by Christmas.

Where's the beef?

Conspicuous by their absence, several previously announced Atari software products were nowhere to be found. **Infinity**, by Matrix Software, was to be an integrated package containing a word processor, spreadsheet, relational database and telecommunications program. It was supposed to be available for both the ST and XE computers, selling for under $100.

Additional Atari titles long past due and missing from the show were: **Learning Phone**, **Song Painter**, **Shopkeeper**, **Crystal Castles** and **Mario Brothers**.

On the hardware side of the coin, the 130ST has been officially declared "cancelled," as has the 65XEP. The 130ST, with only 128K of RAM, was considered to have inadequate memory, and the 65XEP 8-bit portable was thought to have no market. There'll be little mourning for these, as the 130XE, 260STD and 520ST computers are all solid products with healthy futures.

ST software.

With the introduction of the new Atari STs just around the corner, a logical question to be asked by both experienced and novice users is "What kind of programs can I get for an ST?" **ANALOG Computing** was particularly interested to find out which companies are committed to the new Atari computers, and what kinds of software will become available for them.

The following companies have announced ST software. This is the first time this information has been made available. The companies, with descriptions of their products, are presented in alphabetical order.

Batteries Included, publisher of the well known products **B-Graph**, **Home-Pak** and **Paper Clip**, has announced a major commitment to the new ST computers. Their first GEM-based product, called **IS** (for Integrated Software), will consist of a word processor with built-in spelling checker, a combined spreadsheet and graphics package, a database manager and a stock portfolio package.

All of the programs will work together in an integrated way, according to Director of Product Development Michael Reichmann. The portfolio package will be the first release, available in the fall.

Datasoft has revealed that they have an ST development system and are planning to introduce ST software by the end of the year. Their most likely first ST titles will be **Bruce Lee** and **Goonies**.

Electronic Arts is taking a wait-and-see attitude on the new Atari STs. But Trip Hawkins, president and founder of the company, has said that they're coding their Macintosh and Amiga programs in languages such as PASCAL and C, in order to more easily port them over to the ST, should they decide to enter the market. If they did produce an ST product, it would most likely be a newer, more powerful version of **Financial Cookbook**.

Haba Systems, a West Coast software house that has previously developed and marketed software for the Macintosh, has announced that they'll be marketing a line of software for the ST computers. Included in this series of packages will be a word processor, database program, spreadsheet and communications program. They also said that they would eventually port all of their Macintosh software to the Atari ST.

ICD, makers of SpartaDOS and the US Doubler Chip, told me that they're interested in the ST, but have no current plans for specific programs. They will closely watch the market develop and see what's needed, then step in with a particular product tailored to the needs of users. Tom Harker, President, made it very clear that they don't want to abandon the Atari 8-bit line, especially the 130XE computer.

Infocom has announced that their entire line of sixteen text adventure titles will become available for the Atari ST. No firm introduction dates have yet been given.

Microprose will be concentrating on converting their new products to the ST line. These may include **Gunship**, **Acro-Jet** and **Silent Service**, a sub-warfare simulation.

Philon will be offering its line of compilers for the ST. These consist of FAST/BASIC-M (Microsoft), FAST/BASIC-C (Digital Research), C, PASCAL, FORTRAN and RPG. Prices should range from under $200 (for the BASICs) to about $400 (RPG, COBOL). Typical extensive documentation will run close to 250 pages for BASIC and 350 pages for something like COBOL. Because these are compiled, you're looking at run times fifty times faster than normal.

Sierra On-Line has announced that they'll market the popular submarine simulation game **Gato** for the ST computer family. **Gato**, originally introduced for the IBM computer, puts you in the captain's seat of a World War II submarine. Your mission: find and destroy the enemy Japanese fleet. I've spent many hours playing the PC version of the game and have recently seen the Macintosh version. The ST's **Gato** should be excellent.

Spinnaker is planning on developing ST software. Says Seth Godin, ". . .the (ST) machine is custom made for our interactive fiction games. It has the sound, graphics and memory." He also sees the ST as a strong product and is pleased with the cooperation he's received from Atari.

Next month, **ANALOG Computing** will complete our CES coverage by presenting other hardware and software announcements made at the show. □

ATARI

SUPERPRINTER PACKAGES

- SG-10 Printer and U-Print A 289
- Citoh 7500AP and U-Print A 292
- Panasonic 1091 and U-Print A 339
- Panasonic 1090 and U-Print A 269
- Legend 880 and U-Print A 287
- Powertype and U-Print A 369

Super printer packages have no extra charges added when shipped in Continental USA.

Call for prices on joysticks, printer cables, blank floppy disks, and other computer accessories.

Atari 130 XE Computer ..... Call For Current Prices
Atari 1050 Disk Drive ....

INDUS GT . 219

star micronics • inc
THE POWER BEHIND THE PRINTED WORD.
SG-10 Printer . 219

PRINTER BUFFERS

- Microfazer ...... Call
- MPP Printer Stuffer 119.95
- U-Print-16K Printer Buffer 79.95
- U-Print-32K Printer Buffer 99.95
- U-Print-64K Printer Buffer 109.95

PRINTER INTERFACES

- Aid Interfast I ... 99.95
- MPP 1150 ...... 64.95
- U-Print A 59.95
- U-Print A/16K Buffer .. 79.95
- U-Print A/32K Buffer .. 99.95
- U-Print A/64K Buffer . 109.95

PRINTERS

- Citoh 8510AP 299
- Legend 1080 239
- Citoh 7500AP 219
- Epson .... Call
- Toshiba 1351 ..... Call
- Toshiba 1340 559
- Legend 880 219
- Legend 1080 239
- Panasonic 1090 . 199
- Panasonic 1091 . 269
- Powertype 309

Buy the print shop for 27.95 with the purchase of any printer.

MODEMS

- Maxwell 300/1200 AA/DD 319
- MPP 1000E 89.95
- Volksmodem 300 .. 59
- Volksmodem 1200 189
- R-Verter Modem Adapter 39.95

U-Call Modem Adapter 44.95

Buy the Compuserve Starter Kit for 21.95 w/Any Modem Purchase

SUPER SPECIALS

- Basic XL-Cart 49.95
- Mac 65XL-Cart 62.95
- Action-Cart 62.95

BUY ANY INFOCOM TITLE AND GET THE CORRESPONDING INFOCOM INVISICLUE ABSOLUTELY *FREE*.

- Writer's Tool-Cart 62.95
- Monkey Wrench II-Cart 23.95
- Ramrod XL & Omniview XL . . 99.95

SUPER SPECIAL PRICES EFFECTIVE NOW THRU AUGUST 31, 1985

A T A R I S O F T W A R E

ACTIVISION
- Decatholon-Cart 19.95
- Hero-Cart 19.95
- Pitfall II-Cart 19.95
- River Raid-Cart 14.95
- Designer's Pencil-Cart 19.95
- Space Shuttle-Cart 19.95
- Ghostbusters-D 24.95
- Great American Cross Coontry Race-D 20.95

AVALON HILL
Call for items and prices

BRODERBUND
- Bank Street Writer-D .... [illegible]
- Loderunner-D 23.95
- Mask of the Sun-D 27.95
- Operation Whirlwind-D 27.95
- Spelunker-D 20.95
- Stelth-D 20.95
- Whistler's Brother 20.95
- Print Shop-D 34.95
- Serpent's Star-D 27.95
- Print Shop Paper 16.95

CBS SOFTWARE
Call for items and prices

DATASOFT
- Bruce Lee-D/T 27.95
- Letter Wizard + Spell-D 54.95
- Conan the Barbarian-D/T 27.95
- Mr. Do-D/T 27.95
- Alternate Reality-D 27.95

EPYX
Call for items and prices

INFOCOM
- Deadline-D 29.95
- Enchanter-D 23.95
- Infidel-D 29.95
- Planetfall-D 24.95
- Sorcerer-D 29.95
- Starcross-D 29.95
- Suspended-D 29.95
- Witness-D 29.95
- Sea Stalker-D 24.95
- Cutthroats-D 24.95
- Suspect-D 24.95
- Hitchiker-D 24.95
- Zork I-D 24.95
- Zork II or III-D 27.95
- Wisbringer-D 27.95

MICROPROSE
- Air Rescue I-D/T 23.95
- F-15 Strike Eagle-D/T 23.95
- Crusade in Europe-D 27.95
- Decision in Desert-D 27.95
- Kennedy Approach-D 23.95

MINDSCAPE
Call for items and prices

MISCELLANEOUS
- Diskey-D 34.95
- Ultra Disassembler-D 34.95
- Codewriter-D 69.95
- Star League Baseball D/T 23.95
- Star Bowl F[illegible]ball-D/T 23.95
- Master Type-D/Cart 27.95
- Flight Simulator II-D 37.95
- S.A.M.-D 41.95
- Castle Wolfenstein-D 20.95
- Compuserve Starter Kit 27.95
- Home Accountant-D 49.95
- Megafont II-D 19.95
- Monkey Wrench II-Cart 29.95
- Movie Maker-D 37.95
- Ultima III-D 41.95
- Jupiter Mission-D 34.95
- Boulder Dash-D/T 19.95
- Scraper Caper-Cart 34.95
- Miner 2049'er-Cart 34.95
- Spy Hunter-Cart/D 29.95
- Tapper-Cart/D 29.95
- Up 'N Down-Cart/D 29.95
- Pic Builder-D 27.95
- Astrochase-D 19.95
- Flip-Flop-D/T 19.95
- Sargon II-D/T 16.95
- Odesta Chess-D 49.95
- Spy vs. Spy-D 23.95
- Adventure Writer-D 27.95
- MMG Basic Compiler-D 69.95
- Summer Games-D 27.95
- Pitstop II-D 27.95
- Gateway to Apshal-Cart 27.95
- Montezuma's Revenge-D 27.95
- Dragon/Pern-D 27.95
- Adventure Master-D 34.95
- Get Rich Series-D 34.95
- MPP Modern Driver-D 19.95
- Frogger II-Cart 24.95
- Net Worth-D 49.95

MISC. (cont.)
- Stickybear-D 27.95
- Omnitrend Universe-D 69.95
- Space Beagle-D 23.95
- Adventure Writer-D 41.95
- Bea[illegible]ead-D 23.95
- Letter Perfect / Spell-D 49.95
- Harcourt / Bruce S.A.T.- 59.95
- Data-Perfect-D 49.95
- Star Wars-Cart 24.95
- S.A.G.E. Graphics Editor-D... 39.95
- Strip Poker-D 23.95
- Millionaire-D 27.95
- Scroll of Abadon-D 23.95

MISC. (cont.)
- Ultima IV-D 41.95
- Raid Over Moscow-S 27.95
- Micro-League Baseball-D 29.95
- Paper Clip-D 39.95
- Home Pak-D 34.95
- Ultima II-D 41.95
- B-Graph-D 49.95
- Trolls & Tribulations-D 19.95
- Ball Blazer-D 27.95
- Return On Fractalis-D 27.95
- Halley Project-D 29.95
- Astral Rift-D 27.95
- Back To Basic Accounting-D . . . Call

MISC. (cont.)
- Data Manager-D 19.95
- Electronic Checkbook-D 19.95
- Money Manager-D 19.95
- Qbert-Cart 24.95
- Popeye-Cart 24.95
- Starfighter Joystick 9.95
- Wico Joysticks .... Call
- Tac II Joystick 14.95

OPTIMIZED SYSTEMS
- Action-Cart 69.95
- Basic XL-Cart 54.95
- MAC / 65-Cart 69.95
- MAC / 65 Tool Kit-D 27.95
- Action Tool Kit-D 27.95
- DOS XL-D 27.95
- Writer Tool-Cart 69.95
- Basic XL Tool Kit-D 27.95

SCHOLASTIC
Call for items and prices

SCREENPLAY
Call for items and prices

SEGA
Call for items and prices

SIERRA ON-LINE
Call for items and prices

SPINNAKER
Call for items and prices

ELECTRONIC ARTS
- Archon 19.95
- Archon II 24.95
- Mule 19.95
- Realm/Impossibility 19.95
- Murder/Zinderneuf 19.95
- Music Construction 19.95
- Pinball Construction 19.95
- One on One 24.95
- Seven Cities of Gold 24.95
- Financial Cookbook 29.95

GAMESTAR
- Star League Baseball-D/T ... 20.95
- Starbowl Football-D/T 20.95
- On Track Racing-D 20.95

SSI
- Carrier Force-D 41.95
- Combat Leader-D/T 27.95
- Cosmic Balance II-D 27.95
- Cosmic Balance-D 27.95
- Broadsides-D 27.95
- War in Russia-D 55.95
- 50 Mission Crush-D 27.95
- Questron-D 34.95
- Rails West-D 27.95
- Computer Ambush-D 41.95
- Galactic Adventures-D 41.95
- Computer Baseball-D 27.95
- Reforger 88-D 41.95
- Objective Kursk-D 27.95
- Breakthrough / Ardennes/D... 41.95
- Field of Fire-D 27.95
- Imperial Galactium-D 27.95
- Oper. Market Garden-D 34.95
- Kampfgruppe-D 41.95
- Computer Quarterback-D 27.95

SYNAPSE
- Alley Cat-D/T 16.95
- Blue Max-D/T 23.95
- Dimension X-D/T 23.95
- Drelbs-D/T 23.95
- Encounter-D/T 16.95
- Fort Apocalypse-D/T 23.95
- Necromancer-D/T 23.95
- New York City-D 19.95
- Pharoah's Curse-D/T 23.95
- Quasimodo-D 19.95
- Rainbow Walker-D/T 23.95
- Relax Stress Reduction Sys.. 79.95
- Shamus Case II-D/T 23.95
- Zepplin-D/T 23.95
- Synfile-D 34.95
- Syncalc-D 34.95
- Syntrend-D 34.95
- Synchron-D 27.95
- Syncomm-D 27.95
- Synstock-D 27.95
- Blue Max 2001-D 23.95

D—DISK
T—CASSETTE
CART—CARTRIDGE

EST. 1982

ComputAbility™

P.O. Box 17882, Milwaukee, WI 53217
ORDER LINES OPEN
Mon-Fri 11 a.m. - 7 p.m. CST • Sat. 12 p.m. - 5 p.m. CST

To Order Call Toll Free

800-558-0003

For Technical Info, Order Inquiries, or for Wisc. Orders

414-351-2007

ORDERING INFORMATION: Please specify system. For fast delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 2 weeks to clear. Charges for COD are $3.00. School Purchase Orders welcome. In CONTINENTAL USA, include $3.00 shipping per software order. Include 4% shipping on all Hardware orders, minimum $4.00. Mastercard & Visa please include card # and expiration date. WI residents please add 5% sales tax. HI, AK, FPO, APO, Canadian orders — add 5% shipping, minimum $5.00. All other foreign orders, please add 15% shipping, minimum $10.00. All goods are new and include factory warranty. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call 414-351-2007 to obtain an RA# or your return will NOT be accepted for replacement or repair. Prices and availability are subject to change without notice.

ATARI is a trademark of ATARI, INC.

No surcharge for MasterCard or Visa

CIRCLE #102 ON READER SERVICE CARD

ATARI® 820 or 822 PRINTER - $39.00
With Cable and Paper *Everything you Need!*

DISK DRIVES

- **New Special Edition**
  $199.00 or $349.00 w/Happy
- Happy® Compatible
- 100% Software Compatible
- 120 Day Warranty
- Most Durable & Serviceable

DeRe ATARI $9.95

MADE FROM ATARI® 810 BOARDS
AND TANDON® MECHANISMS

410 PROGRAM RECORDERS
CASSETTE - $29.00

- Star Raiders
- Touch Typing
- States & Capitals
- Missile Command
- Time Wise
- PacMan

All for $25.00

800 Computer Boards
Complete & Tested

- Mother Board
- Power Board
- ROM Board
- CPU Board
- RAM Board

All for $39.00

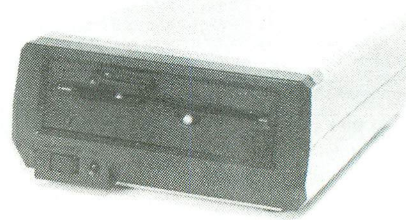
- **Numeric Keypads w/driver $19.95**
- **Disks from $1.00 each**
- **LJK® LETTER PERFECT OR DATA PERFECT $39.00**

CA. ADD APPROPRIATE SALES TAX.
ADD $5.00 MIN. SHIPPING CHARGE.

Everything for the ATARI *Systems to Parts*

San Jose Computer
1844 Almaden, Rd. Unit E
San Jose, CA 95125
(408) 723-2025

YOU CAN'T TELL A DISK DRIVE BY ITS COVER!!



WITH A **HAPPY ENHANCEMENT** INSTALLED THESE ARE
THE MOST POWERFUL DISK DRIVES FOR YOUR ATARI COMPUTER

WARP SPEED SOFTWARE DISK READING AND WRITING 500% FASTER

HAPPY BACKUP — Easy to use backup of even the most heavily protected disks
HAPPY COMPACTOR — Combines 8 disks into 1 disk with a menu
WARP SPEED DOS — Improved Atari DOS 2.0S with WARP SPEED reading & writing
SECTOR COPIER — Whole disk read, write and verify in 105 seconds
1050 ENHANCEMENT — Supports single, 1050 double, and true double density
810 ENHANCEMENT — Supports single density

SPECIAL SUGGESTED RETAIL PRICE: Get the HAPPY ENHANCEMENT 810 or 1050 version with the HAPPY BACKUP PROGRAM, plus the multi drive HAPPY BACKUP PROGRAM, plus the HAPPY COMPACTOR PROGRAM, plus the HAPPY DRIVE DOS, plus the HAPPY SECTOR COPY, all with WARP DRIVE SPEED, including our diagnostic, a $350.00 value for only $249.95, for a limited time only! Price includes shipping by air mail to U.S.A. and Canada. Foreign orders add $10.00 and send an international money order payable through a U.S.A. bank. California orders add $16.25 state sales tax. Cashiers check or money order for immediate shipment from stock. Personal checks require 2-3 weeks to clear. Cash COD available by phone order and charges will be added. No credit card orders accepted. ENHANCEMENTS for other ATARI compatible drives coming soon, call for information. Specify 1050 or 810 ENHANCEMENT, all 1050s use the same ENHANCEMENT. Please specify -H model for all 810 disk drives purchased new after February 1982, call for help in 810 ENHANCEMENT model selection. Dealers now throughout the world, call for the number of the dealer closest to you.

ATARI is a registered trademark of Atari Computer Inc.

HAPPY COMPUTERS, INC. • P. O. Box 1268 • Morgan Hill, California 95037 • (408) 779-3830

CIRCLE #106 ON READER SERVICE CARD

An Introduction to MIDI

(Musical Instrument Digital Interface)

by Craig Patchett

"Interface." *Webster's New Collegiate Dictionary* (150th edition) defines it as "the place at which independent systems meet and. . .communicate with each other." In the wonderful world of computing, this means that computers can comunicate with television monitors, disk drives, printers and even other computers. Exciting, right?

Of course not. Mostly, interfaces only allow computers to communicate with other computer-type stuff, and that gets boring quite quickly.

The people who make electronic musical instruments had an even bigger problem. Their equipment had trouble simply communicating with other electronic music-type stuff. Since few musicians use just one instrument, it was obvious that something had to be done. An interface was needed.

Although individual companies had toyed around with the idea of an interface for some time, they didn't really start getting together on it until 1981. At that point, initial proposals were worked up, and the Japanese (who'd been working on some ideas themselves) joined in.

As was to be expected, there were a lot of disagreements on exactly how the interface should function, ranging all the way down to whether data transmission should be serial (1 bit at a time) or parallel (1 byte at a time).

After a while, most of the American companies gave up, with the exception of Sequential Circuits, who had initiated the discussions. They, along with the Japanese companies, Roland, Yamaha, Korg and Kawai, shaped the interface into its final form. A working specification for the Musical Instrument Digital Interface, or MIDI, was completed around the turn of 1983. It was implemented in the first instruments, from Sequential Circuits and Roland, shortly thereafter.

So much for history. What exactly is MIDI, and what does it have to do with computers? Before we begin, it's important to realize that MIDI is only a specification for an interface. This means that it defines a set of rules and guidelines; it's not a standard circuit board or something that you buy from a designated manufacturer.

In other words, MIDI defines a language that can be used by electronic musical instruments to communicate with each other. It's up to individual manufacturers to implement it in their products.

MIDI in general.

First the basics. MIDI ended up using serial data transmission, at 31.25 Kbaud (31,250,000 bits per second). Although some critics claim that this is too slow, it's supposedly fast enough to turn up to 500 notes on and off per second.

Since MIDI can communicate with up to sixteen different instruments at once, this means that over thirty notes per instrument per second are possible. This is more than fast enough for most applications.

Communicating with so many different instruments is done through the use of sixteen MIDI "channels." Each instrument is assigned (or assigns itself) one or more channel numbers, then only tunes into or sends information out on those channels.

When a message is sent out, a channel number is attached to it, and the other instruments then check this channel number to determine whether or not they should pay attention to the message. Thus, MIDI only needs one wire to transmit its information, rather than sixteen.

The different types of messages MIDI can handle let you turn notes on and off, specify how hard and how quickly a key was hit (if you're using a keyboard with such capability), select a preset sound (program number), and manipulate various controls on the instrument (such as pitch-bend).

You can also set instruments so that they transmit note messages as they're played, but don't play the notes themselves (allowing you to play one instrument from another).

MIDI also includes commands that can be defined differently, depending on the particular instrument. These come in handy when you use MIDI with a computer, as we'll see later.

One last thing to note before we get into computer applications—MIDI will not give an instrument capabilities that weren't built into the instrument. You cannot, for example, bend the pitch on an instrument that doesn't have a pitch-bend feature, just because you send it a pitch-bend message.

This would be comparable to expecting your best friend to be able to fly just because you told him to. Silly as this may seem, many people expect MIDI to perform such miracles.

Of course, MIDI is in no way limited to musical instruments. In fact, its ability to hook up to a computer is, perhaps, its most powerful asset. Combine the processing power of a computer with the artistic quality of a musical instrument, and a new realm of possibilities appears, a few of which are listed below.

Composing.

Most of you have played around with programs like Atari's **Music Composer** or Electronic Arts' **Music Construction Set**, in which you create written music note-by-note on the screen, then have the computer play it for you.

Nice as this is, even the Atari can't do much to create music that doesn't sound like it came from a computer. But, with MIDI, you can have the computer play your music on actual instruments—orchestrated, complete with percussion (MIDI-compatible instruments include drum synthesizers, as well as keyboard and guitar synthesizers).

You can also, if you prefer, reverse the process and have the notes appear on the screen as you play them on an instrument. Edit the results until they're perfect—the computer can automatically correct your timing and transpose for you, if necessary—then the computer can print out complete sheet music for you, as well as play your finished masterpiece.

Accompaniment.

While it's relatively easy for a beginner to learn how to play a simple melody, adding chords and bass lines takes a lot of practice.

There are, however, rules that determine what goes well with what. And we all know that computers are great with rules. Therefore, MIDI gives you the capability to play a one-finger melody on a synthesizer, have a computer analyze it and generate the appropriate accompaniment.

Education.

While we're on the topic of beginners, remember that computers can also make great teachers. How about software that generates notes on a music staff on the screen and checks you as you play them on a synthesizer (sort of a musical typing tutor)?

You could have similar software to test your knowledge of pitch, timing, and so forth. There could even be computer games in which success depends on your skill with a musical keyboard.

Sound design.

The new generation digital keyboards

allow great control of the sounds they can create. Unfortunately, this control is usually in the form of numbers that have to be manipulated. We all know that humans don't fare very well with numbers.

I mentioned before that MIDI includes messages that are defined differently for different instruments. On digital synthesizers, these messages (called "system exclusive" messages) usually allow the numbers that define a sound to be transmitted over the MIDI interface.

If they're transmitted to a computer, the computer can convert them to a visual form that's easier to understand and can be manipulated with a joystick. There's also the possibility of having the computer analyze real sounds through a microphone, then convert the sounds to the numbers the synthesizer needs to imitate them.

MIDI mania.

These are just a few of the many ways in which computers can be used with MIDI. Some are already available as commercial software; some are currently in development; and some may never be seen at all. It doesn't matter.

The point is that MIDI's far more than just a specification for an interface. It represents a tremendous potential that's just waiting to be tapped. When and if it is, the ability to communicate between a computer and a musical instrument should be a great stepping stone in the advancement. . . and MIDI will have been the one to pave the way.

If you'd like more information about MIDI, the International MIDI Association (IMA) offers a membership that includes the complete MIDI specification, a newsletter to keep you up to date on MIDI developments and a phone number for technical assistance. For information, you can contact: IMA, 11857 Hartsook St., North Hollywood, CA 91607 — (818) 505-8964. □

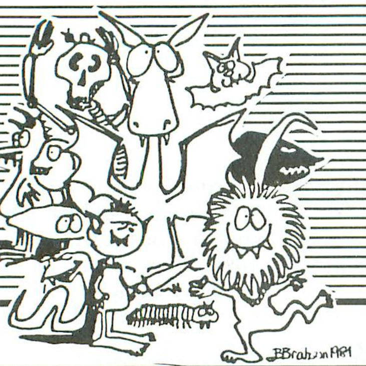
Craig Patchett is the author of several microcomputer books and a frequent contributor to ***ANALOG Computing****. A loyal Atari supporter since 1979, he's currently a software engineer for Perkin-Elmer Corp. in Norwalk, Connecticut.*

WHAT IS CHECKSUM DATA?

Most program listings in **ANALOG Computing** are followed by a table of numbers appearing as DATA statements, called "CHECKSUM DATA." These numbers are to be used in conjunction with **D:CHECK** and **C:CHECK** (which appeared in **ANALOG Computing** issue 16 and the **ANALOG Compendium**) or with **Unicheck** (from issue 24).

D:CHECK and **C:CHECK** (written by Istvan Mohos and Tom Hudson) and **Unicheck** (by Tom Hudson) are designed to find and correct typing errors when readers are entering programs from the magazine. For those readers who would like copies of these articles, you may send for back issue 16 or 24 ($4.00 each) or the **ANALOG Compendium** ($14.95 plus $2.00 shipping and handling from:

ANALOG Computing
P.O. Box 615
Holmes, PA 19045

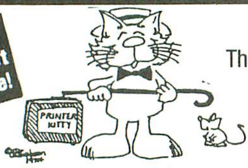


XLENT Software

Presents:

The World's Greatest Printer Utility!

NOW touch tablet compatible!



MegaFont][+

The Complete Program Lister and Graphics Dumper
by Richard Rognlie and Randy Dellinger

"... a very good program ... better than ever ..." ACE Newsletter

MORE FONTS — FASTER DUMP — FONT UPLOADER*

Allows NEC, Prowriter, Epson (w/Graftrax, RX-80, FX-80), Riteman, Gemini, Mannesmann Tally, Panasonic & other compatible dot matrix printers to dump Graphics 7+ and 8 screens in 3 sizes (4 on Epson) and print all control and inverse characters in fonts provided or use your own. New Features: Adjustable margins, FAST LISTER, Font uploader*, adjustable line spacing, Font Splicer (combine 2 fonts together).
*Prowriter 8510AP w/chip, Epson FX-80, Panasonic KX-P1092, Citizen 20 and compatibles

48K disk .... **Only $24.95**
MegaFont PC (for IBM PC, PC jr & most PC compatibles) **$39.95**

Let Your Imagination Run Free

Page Designer

by Len Dorfman and Dennis Young

"... a pretty nifty package ..." Current Notes

Page Designer lets you design a full page printout (8½ x 11) on-screen and then save it to disk. Combine text, graphics and borders. Edit on-screen in either 40 or 80 columns. Font styles can be mixed in the 40 column mode. All graphic and control characters are displayed and printed PLUS we've included specially designed border characters. Use Page Designer to add text to pictures you've created (Micropainter, Koalapad, Atari Artist). Drawing mode let's you add finishing touches (e.g., circles and spheres). Great for creating ads, reports, newsletters or for dressing up letters.

48K disk .... **Only $29.95**
For Epson, Gemini, Panasonic, Prowriter, NEC and compatible dot matrix printers.

Resolution to the MAX!
491,520 pixels per page

TypeSetter

by Dennis Young and Len Dorfman

Now your Atari can have higher resolution to the printer than the MacIntosh. The 48K version can have resolution up to 704 x 624 pixels (768 x 640 on the 130 XE version). TypeSetter accomplishes this feat by letting you scroll over multiple graphics 6 screens. You can control the style, height and width (32 sizes) of characters. Type in any direction — forward, backward, up and down. You have the flexibility to mix graphics, text and graphic characters. Sketch Pad lets you create icons and allows you to copy them or any part of the screen anywhere. Used in conjunction with Page Designer, you can add detail to standard Graphics 8 screens. This program is for ALL Atari enthusiasts who've dreamed of having the artistic capability of the MacIntosh. Light years ahead of any other Atari graphics program.

48K disk .... **Only $34.95**
Attention: 130 XE version included
For Epson, Gemini, Panasonic, Prowriter, NEC's and compatible dot matrix printers.

Jennifer Brabson's artwork .... **3 disks for $20.00**
For use with MegaFont][+, Page Designer and TypeSetter
"Chip" for Prowriter 8510A .... **adds 2K buffer** **$15.00**

The Only Serious Database on the Atari

NOW Not Copy Protected!

XLENT MegaFiler

by Jerry Kwit

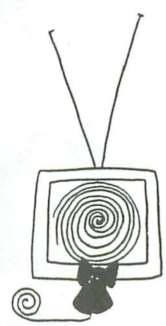
"I don't believe you can get better value for your money ..." ACE Newsletter

"Its claim to fame is the large amounts of records it can handle." ANALOG

MegaFiler is the ONLY database on the Atari that is very powerful, easy to use, and can handle over 1,500 records. Features found on expensive databases include: modifiable database structure without data loss, multiple level sorting, user defined formula for data entry, Report Generator with subtotaling of fields, subsidiary file creation and Label Generator. NOW unprotected so you can use it with your favorite DOS.

48K disk .... **Only $29.95**
Coming soon ... special version for the 130 XE.

Explore Your Inner Self



HYPNOSIS

with Brainwave Synchronization
by Gene Levine

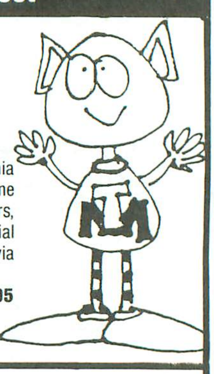
The most significant self-improvement program. Designed for anyone who is interested in hypnosis, relaxation, or self-improvement. This program replaces the synchronizing strobes that can cost hundreds of dollars. Seven screens pulsate at speeds that you can adjust with paddles or keyboard. Audio tapes may be played using the Atari cassette recorder. Create your own self-improvement tapes to be used with this program. Great for those evenings when you want to relax.

48K disk .... **Only $29.95**

The Trivia Construction Set

Trivia Mania

by Jan Iverson



Once played, most trivia games become trivial. Trivia Mania includes 1,800 questions to start, PLUS an easy to use game editor. You can create your own trivia games — computers, foreign culture, vampires — any area that is your special interest. Ideal for use by parents, teachers, students and trivia buffs. A fun and educational utility.

48K (2 double-sided disks) .... **Only $29.95**

Add $2.00 for shipping and handling. C.O.D. orders, additional fees. Virginia residents please add 4% sales tax. Send check or money order to:

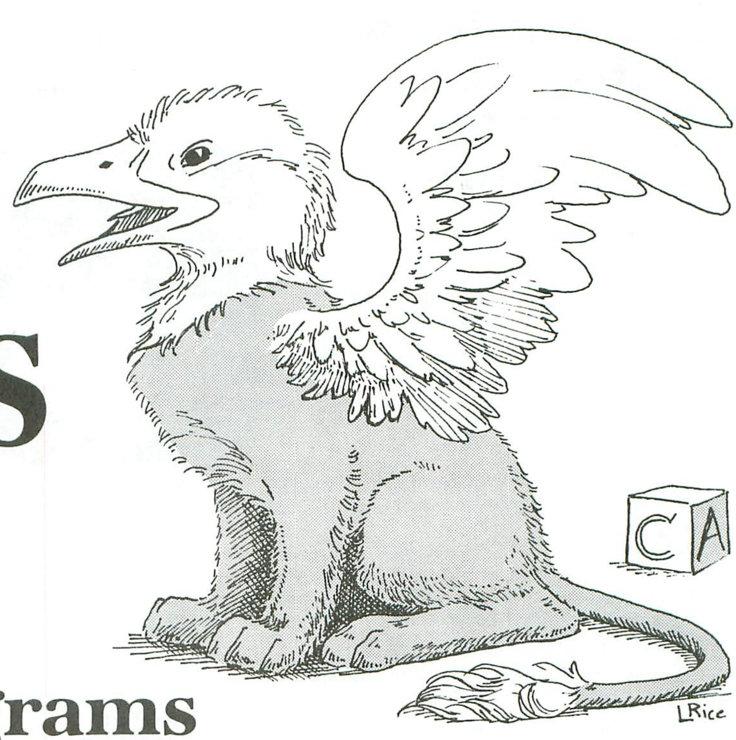
XLENT Software

XLENT Software
P.O. Box 5228, Dept. B
Springfield, Virginia 22150

Dealer Inquiries Welcome 24 Hour Order Phone **(703) 644-8881**

CIRCLE #107 ON READER SERVICE CARD

GRIFFIN'S LAIR



Educational Programs Review

by Braden E. Griffin, M.D.

At what seemed the last minute, I was asked to review a group of programs in keeping with the music theme of this issue. One of the programs is great; three are not so great; and the remaining one had documentation so confusing that I plan to give it to a higher authority. Anyway, here's this month's riff, Raff. And a-one. . .and a-two. . .

MOVIE MUSICAL MADNESS
CBS SOFTWARE
One Fawcett Place
Greenwich, CT 06836
16K Cartridge or Cassette $12.95

Who could resist the opportunity to be the writer, composer and director of a Hollywood production? With the help of the Jazz Scats, a creation of The Dovetail Group, CBS Software opens the studio doors to new vistas of cinematic excitement.

Designed for ages eight to adult, **MMM** gives you the chance to find out that there's no business like show business. At least, no business I know. Everything about it is appealing. . .Oh, you've heard that before?

MMM lets you pick one of the three Jazz Scats—Swivel Hips, Wahoo or Mr. Bassman—to star in each scene. The set is then constructed from a wide variety of props.

Animated props are in motion, but stay in one location. Included among the animated props are a villain ("Curses, foiled again!"), a madcap cop, a moon lander, a puppy and a neon sign.

Moving props are those which move across the set once the scene has begun. A starship invasion, a robot, a ghost, a witch and a train comprise the list of moving props. Only one animated or moving prop may be used in each scene.

The rest of the background is provided through the selection of stationary props. As many as fifteen of these may be used per scene.

A wide variety of scenery can be created in this way, with city skylines, fences, bricks, trees, castle parts, a piano, a fire hydrant and others. A nearly unlimited number of designs are possible using combinations of the various props.

Once the scene is set, the music score is selected. From twenty different musical interludes, just the right one for the scene is picked. A wide assortment of choices is available, from a hectic melody fit for the Keystone Kops to a sinister elegy for the walking dead.

With the music selection out of the way, the time has come to begin direction. The previously chosen Jazz Scat is moved around the scene as desired. He can even sit down or float.

When the music ends, that particular scene is over. The length of each scene determines the total length of the movie; i.e., the less activity in a scene, the greater the number of scenes that can be used.

As each scene is completed, it may be reviewed by calling up the "rushes," or it may be reshot, if desired. Once all the scenes are finished, the entire production may be viewed.

Most of the activities with this program are accomplished by using the joystick. It's very simple to manipulate

A Word from the Atari Teachers' Network

The Atari Teachers' Network is happy to announce that we are alive and well and still publishing our newsletter under the leadership of John Hanna.

The Atari Teachers' Network is a widespread non-profit organization of education-oriented computer enthusiasts dedicated to the enhancement and proper implementation of computers in schools *around the world*. The current mailing list contains over 450 members from the USA, Canada and other countries, including Singapore, New Zealand, Australia and the Fiji islands.

The Atari Teachers' Network is looking for educators and individuals interested in contributing to the dissemination and collation of educational information about computers in schools. Particular emphasis is placed on Atari computers as a basis for discussion, but the educational implications of computing are far more critical to our conscious efforts than a devotion to one particular brand of computer.

Membership in the Network is $4.00 per year. The newsletter is published quarterly at the moment, but, if demand justifies it, it will go to a bi-monthly or monthly periodical.

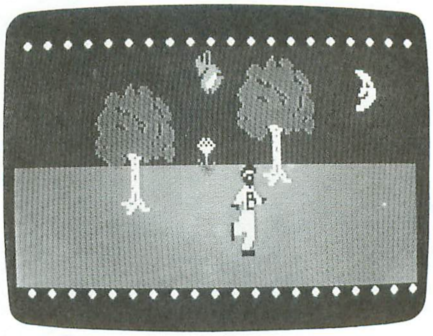
To join the Atari Teachers' Network, please send $4.00 and pertinent background information (your school, grade level, computers in use, interests, etc.) to:

Atari Teachers' Network
John E. Hanna
c/o Teaneck High School
Teaneck, NJ 07666.

If there are any European readers interested in joining, we'd be especially pleased to hear from them.

The Atari Teachers' Network is not affiliated in any way with Atari Corp. or any other commercial enterprise, or with the Teaneck, New Jersey public schools.

the various components. Children will be stimulated to create an organized continuum of scenes in the development of their movie. A lot of effort is necessary to end up with a really polished production.



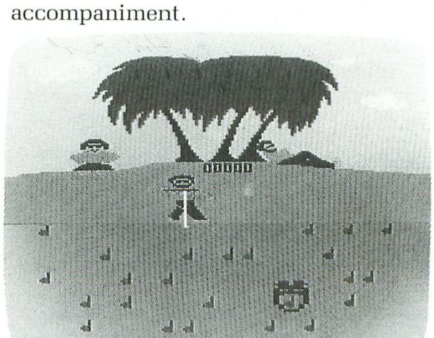
Movie Musical Madness.

The only drawback is the inability to save one's masterpiece. A VCR hooked into the system will do the trick, if available.

All in all, **Movie Musical Madness** is fun, stimulating and enjoyable.

COCO-NOTES
CBS SOFTWARE
One Fawcett Place
Greenwich, CT 06836
16K Cartridge or Cassette $12.95

This is another offering from The Dovetail Group's Jazz Scats. Using a fishing line, notes are pulled from the sea and placed in a **Coco-Note** tree. Once the tree is full, the Jazz Scats play the melody created by the combination of selected notes, with their own special accompaniment.



Coco-Notes.

Each time a new series of notes is completed, a new song is played in one of a wide variety of styles. This first game is called Catch a Coco-Note.

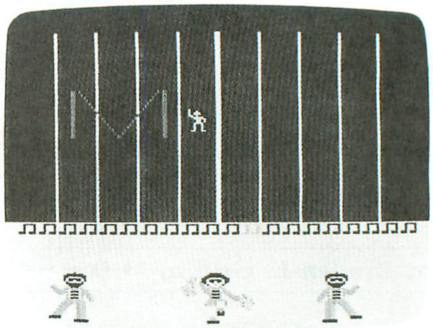
The second game option, Composin' Coco-Notes, offers the opportunity to select one's own accompaniment. A wide assortment is available, ranging in style from country to rock.

The third game pits the player against Mr. Cool Clam. Gimme a break! Notes are reeled in before they can be eaten by this menacing mollusk. A scoring system for different kinds of notes adds a competitive slant to this musical activity.

Designed for children from seven to twelve years, this game, though not terribly educational, is cute. Yeah, cute.

HALFTIME BATTLIN' BANDS
CBS SOFTWARE
One Fawcett Place
Greenwich, CT 06836
16K Cartridge or Cassette $12.95

The third and final program featuring the Jazz Scats may be played alone or against an opponent. The object is to create marching tunes and formations while trying to thwart your rival's efforts to do the same.



Halftime Battlin' Bands.

The joystick-controlled Drum Major picks up lines of bandsmen from the sides and places them in preset formations on the football field. These bandsmen play either march music or silly music. The winner is the one who completes the formation first.

A number of obstacles may be used to upset the best laid plans of one's foe. Objects obtained from the bench include a whistle that causes an opponent to drop a line of bandsmen, a football to change the musical style from march to silly music, and a hot dog to spread mustard across an unoccupied line in the formation. The line must be washed free of the mustard, using the water bucket.

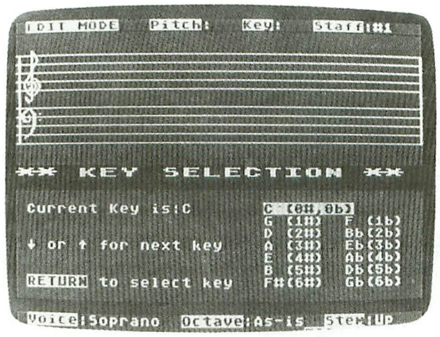
Does this sound like fun, or what? I recall advice from Thumper, the hare-brained philosopher. . . But seriously, is this game worth buying? Did John Philip Sousa dedicate all of his marches to the memory of Amelia Bedelia?

BANK STREET MUSIC WRITER
MINDSCAPE INC.
3444 Dundee Road
Northbrook, IL 60062
48K Disk $49.95

This is just a superb program. It's educational and enjoyable. But, most of all, it's easy to use. This is a tool for composing music, learning music, exploring music.

It's probably not fair to either product, but **Bank Street Music Writer** is quite similar to the **Music Construction Set** from Electronic Arts, which I reviewed in issue 25.

Notes from a variety of instruments are placed on a staff to create music. Volume, pitch, time signature, harmonies, etc. can be changed with ease to suit one's taste. Music can easily be transposed and even printed out to paper. A number of selections are included on the disk to show different examples of how the program can be used to its fullest capabilities.



Bank Street Music Writer.

There is one very basic difference in the two programs mentioned. This is in the method each uses to access the various functions. **MCS** uses various icons which are moved about the screen to write the music. The **Bank Street Music Writer** functions more like a conventional word processing program, using various keystrokes to accomplish similar tasks.

Both are excellent, so the choice may rest on the format one prefers. . ."less filling" versus "tastes great."

The manual accompanying this product is extremely well done and includes a very lucid tutorial for music aficionados from beginner on. The entire package is top notch and is what they call "user friendly." I can recommend this one without reservation. By the way, just to be fair, the creator of this gem is Glen Clancy. □

Dr. Griffin, as Chief of Newborn Medicine at a perinatal center, spends most of his time in the newborn intensive care ward. Off-hours, he's been using an Atari 800 for four years. In keeping with his gentle profession of nurturing preemies, Dr. Griffin's number one game is ***Crush, Crumble and Chomp.***

BASIC EXTERM

by Angelo Giambra

In the appendix to his *Atari BASIC Source Book*, Bill Wilkinson discusses the known bugs in Atari BASIC. The bugs are there, he tells us, because Atari was in a hurry to release BASIC and didn't give the authors sufficient time to fully debug it. He teases us by showing the assembly code which would fix some of the bugs. There's just one problem. BASIC is a ROM cartridge. We can't get in there to apply the modifications. But don't despair.

The **BASIC Bug Exterminator** will fix four of the known bugs in Atari BASIC, including the infamous lockup problem. In case you haven't experienced it yet, that's when your system suddenly locks up after you've deleted some program lines. It can be frustrating, especially when you lose several hours of work. If you have a disk drive and 48K of memory, read on.

The **Exterminator** works by copying the BASIC cartridge down to disk as an AUTORUN.SYS file. As it copies the code, it modifies the portions which cause the bugs, using Bill Wilkinson's recommended fixes, so that they work correctly.

You then load your modified version of BASIC, by removing your BASIC cartridge and booting from the disk containing the AUTORUN.SYS file.

The following bugs will be fixed:

(1) The lockup problem. Deleting programming lines will no longer cause the system to freeze.

(2) The unary minus problem. Try this in BASIC:

```
PRINT -0
```

Garbage appears on your screen because of BASIC's inability to take the unary minus of a number. With modified BASIC, this will work correctly.

(3) The GET/READ combination problem. GET statements that immediately follow READ statements can cause BASIC to change the line number of DATA statements. This will no longer happen. Note: in the October issue (23) of **ANALOG Computing** there was an article called **Another BASIC Bug**. It outlined a problem with using the VAL function and the GET statement in a program together. This problem is also corrected.

BUG INATOR

(4) The NOT operator problem. Try entering this in BASIC: *PRINT NOT NOT 1*. Crash goes your system! This nasty problem will no longer occur.

The remainder of this article will show you how to implement this new version of BASIC, and will explain some ground rules for working with it. Finally, for machine language programmers, I'll explain how the patches were applied.

Getting there.

To install your disk version of BASIC, follow these instructions *exactly*.

1. With your BASIC cartridge in the left slot of your machine, boot from any of your disks.
2. Key in *DOS* and wait for the DOS menu.
3. Select a new disk, insert it in your drive and format it, using option I.
4. Using option H, write out new DOS files to this disk.
5. Use option B to return to BASIC.
6. Key in Listing 1.
7. SAVE the program to your new disk.
8. RUN the program. If there are errors in your DATA statements, the program will inform you and stop. Correct the errors and return to Step 7.
9. Key in DOS to return to the DOS menu.
10. Using option H, rewrite the DOS files. The fixer program has made some important modifications to DOS which must now be installed.
11. Remove your BASIC cartridge and boot from the disk you have just created. Voila! The familiar *BASIC READY* prompt appears on your screen.

Now try printing *-0*. It works! So does the multiple NOT operator. Best of all, you can delete lines from programs all you want and never worry about the system locking up on you.

Using the Exterminator.

Once you get your new version of BASIC up and running, you'll have to learn a few new ground rules. For instance, after keying in DOS to perform some disk operations, using option B will no longer return you to BASIC. If you try it, the message *NO CARTRIDGE* will appear. You return to BASIC by pressing SYSTEM RESET.

MEM.SAV files will still function properly, restor-

ing your BASIC program after returning from DOS.

In fact, your new disk version of BASIC will function exactly like the cartridge version, except for one special case. Graphics modes 2 through 11 may produce strange results unless you POKE address 106 with 144 first. If a program using these modes doesn't run correctly, insert this line at the start of the program:

```
POKE 106,144
```

At the end of the program, insert this line:

POKE 106, 152.

Never POKE location 106 with any higher value. This is the RAMTOP register, and it tells the OS where free RAM ends. With cartridge BASIC, its value is usually 160. The OS multiplies this by 256 to find where free RAM ends. Since RAMTOP's new value is 152, this means that the disk version of BASIC eats up a little over 2K of extra memory, but then, you never get something for nothing.

Here's how the **Exterminator** works. A machine language procedure PEEKs into your cartridge and copies the code to an AUTORUN.SYS disk file. As it proceeds, it checks for certain addresses where the known bugs reside. Whenever it encounters one of these addresses, it substitutes code which will correct the problem.

The BASIC lockup problem is caused because the string moves involving exact multiples of 256 are not handled properly in BASIC. When you delete program lines, BASIC moves code around in memory. If it tries to move an exact multiple of 256 bytes, it crashes. This happens in a routine in BASIC called CONTRACT (since the BASIC code gets contracted). The error is located at hex address $A94E. The instructions near this address are:

```
A94C  BNE $A954
A94E  BEQ $A95B
```

What *should* be at this address is this:

```
A94C  BNE $A954
A94E  DEX
A94F  BNE $A954
A951  RTS
```

Unfortunately, the revised code assembles into two more bytes of code than the original. This problem is solved by placing a piece of revised code in memory somewhere outside of BASIC. The following code is placed in memory below the start of the BASIC code:

```
9B78  BNE $9B7E
9B7A  DEX
9B7B  BNE $9B7E
9B7D  RTS
9B7E  JMP $A954
```

Then address $A94C is modified as follows:

```
A94C  JMP $9B78
A94F  <Garbage byte>
```

Since the JMP instruction uses 3 bytes instead of the 2 used by the BNE at $A94C, a leftover garbage byte remains. If you follow the flow of logic, however, you'll see that this presents no problem, since the program flow always bypasses the garbage byte. In essence, this logic duplicates the correct sequence of events necessary to fix the problem.

Each one of the bugs in BASIC was corrected in this manner. This is why the RAMTOP register is altered from its original value of 160 to 152. Below BASIC, an area of memory is reserved that contains assembly routines which BASIC jumps into and out of. In addition, a routine is resident which handles things whenever you press the SYSTEM RESET.

SYSTEM RESET causes the OS to re-establish the top of RAM. This is at address $BFFF. The OS then takes 1K of memory below this address and makes it screen memory. The trouble is, it wipes out the BASIC also residing here. (This does not happen with the cartridge, because when the cartridge is in place, the top of RAM is at $9FFF).

An assembly routine saves the top 1K of BASIC in a different memory address at boot-up time. Then, anytime you press SYSTEM RESET, it moves this code back to high memory after the OS wipes it out.

Finally, a patch is made to the memory resident portion of DUP.SYS. Whenever you key in DOS, DUP.SYS alters DOSINI to point to the DOS initialization routine if no MEM.SAV file is present. Instead, this is pointed to $9B81, an assembly routine which reinitializes BASIC. Pressing SYSTEM RESET causes the OS to jump to DOSINI, returning us to BASIC.

Page 6 is used at boot time for some initialization procedures, but is never needed again, so you're free to use it as usual. Note that the initialization code is not cleared from this location, however. So, if you're going to use page 6, you might want to zero it out first.

Now go and put your BASIC cartridge away in a nice, safe place. You might need it for backup someday. □

Angelo Giambra is a Senior Analyst/Programmer for Marine Midland Bank in New York. With a B.A. in English Literature, he has been in the data processing field for eight years. An avid Atari hobbyist and incessant tinkerer, he enjoys writing machine language utilities and extensions to the OS and DOS.

(Listings start next page)

Listing 1.
BASIC listing.

```
10 REM BASIC BUG EXTERMINATOR
20 REM
30 ? CHR$(125):? :? "CHECKING DATA STA
TEMENTS"
40 READ A:IF A=-1 THEN PASS=PASS+1:REA
D A:IF PASS=3 THEN 60
50 CHECKSUM=CHECKSUM+A:GOTO 40
60 IF CHECKSUM<>A THEN ? CHR$(253):? "
DATA STATEMENTS INCORRECT":? "RECHECK
PROGRAM":END
70 I=1536:RESTORE 190:? CHR$(125):? :?
 "DATA STATEMENTS CORRECT--PROCEEDING"
80 READ A:IF A=-1 THEN 100
90 POKE I,A:I=I+1:GOTO 80
100 OPEN #1,8,0,"D:AUTORUN.SYS"
110 READ A:IF A=-1 THEN 130
120 PUT #1,A:GOTO 110
130 X=USR(1536)
140 READ A:IF A=-1 THEN 160
150 PUT #1,A:GOTO 140
160 CLOSE #1:? CHR$(125):? :? "NOW GO
TO DOS MENU":? "AND REWRITE DOS FILES"
170 POKE 6156,129:POKE 6160,155
180 END
190 DATA 104,169,0,133,203,169,160,133
,204,162,16,169,11,157,66,3,169,205,15
7,68,3,169,0,157
200 DATA 69,3,169,0,157,73,3,169,1,157
,72,3,160,0,177,203,133,205,32,86,228,
24,165,203
210 DATA 105,1,133,203,144,2,230,204,1
65,204,201,192,208,3,76,234,6,201,169,
208,41,165,203,201
220 DATA 76,208,217,169,76,133,205,32,
86,228,169,120,133,205,32,86,228,169,1
55,133,205,32,86,228
230 DATA 24,165,203,105,3,133,203,144,
187,230,204,76,36,6,201,172,208,30,165
,203,201,168,208,64
240 DATA 169,76,133,205,32,86,228,169,
96,133,205,32,86,228,169,155,133,205,3
2,86,228,76,96,6
250 DATA 201,188,240,3,76,36,6,165,203
,201,130,240,3,76,36,6,169,76,133,205,
32,86,228,169
260 DATA 110,133,205,32,86,228,169,155
,133,205,32,86,228,76,96,6,201,87,240,
25,201,100,240,3
270 DATA 76,36,6,169,220,133,205,32,86
,228,24,165,203,105,1,133,203,144,2,23
0,204,169,220,133
280 DATA 205,32,86,228,24,165,203,105,
1,133,203,144,2,230,204,76,36,6,96,-1
290 DATA 255,255,65,0,65,0,0,0,6,61,6,
169,129,133,12,169,155,133,13,169,0,13
3,204,169
300 DATA 188,133,205,169,0,133,206,169
,156,133,207,160,0,177,204,145,206,200
,192,0,208,247,230,205
310 DATA 165,205,201,192,240,7,230,207
,160,0,76,26,6,169,1,133,9,169,0,141,6
8,2,76,0
320 DATA 160,96,155,255,155,32,242,171
,165,212,240,4,73,128,133,212,76,177,1
72,32,81,218,169,7
330 DATA 133,192,76,134,188,208,4,202,
208,1,96,76,84,169,169,255,133,8,208,1
7,173,158,23,201
340 DATA 32,240,243,32,64,21,169,0,133
,8,76,155,155,32,64,21,32,213,155,169,
0,133,204,169
350 DATA 156,133,205,169,0,133,206,169
,188,133,207,160,0,177,204,145,206,200
,192,0,208,247,230,205
360 DATA 165,205,201,160,240,7,230,207
,160,0,76,176,155,169,129,133,12,169,1
55,133,13,76,0,160
370 DATA 83,58,169,152,133,106,162,48,
169,3,157,66,3,169,211,157,68,3,169,15
5,157,69,3,169
380 DATA 8,157,74,3,169,0,157,75,3,32,
86,228,169,12,157,66,3,32,86,228,96,22
6,2,227
390 DATA 2,213,155,0,160,255,191,-1
400 DATA 65,0,65,0,1,224,2,225,2,0,6,-
1,61046
```

•

CHECKSUM DATA.
(see page 15)

```
10 DATA 114,253,978,636,352,564,140,16
4,437,268,423,727,84,441,745,6326
160 DATA 662,936,50,469,882,779,15,674
,551,63,458,626,794,991,15,7965
310 DATA 754,938,450,116,30,866,457,61
5,795,820,5841
```

•

Listing 2.
Assembly listing.

```
; BASIC BUG FIXER
; BY A.GIAMBRA
;
IOCB1     =   $30      ;CHANNEL3
ICHID     =   $0340    ;IOCB0
ICCOM     =   ICHID+2  ;COMMAND
ICBAL     =   ICHID+4  ;BUFFER ADDRESS
ICBAH     =   ICHID+5
OPEN      =   $03      ;OPEN COMMAND
WRIT      =   $08      ;OPEN DIRECTION
CLOSE     =   $0C      ;CLOSE COMMAND
CIOV      =   $E456    ;OS I/O ROUTINE
ICAX1     =   ICHID+10 ;AUXILIARY BYTE
ICAX2     =   ICHID+11
INDEX     =   $CC      ;WORK INDEX
INDEX1    =   $CE      ;WORK INDEX
WARMSTART =   8
BOOT      =   9
COLDSTART =   $0244
DOSINIT   =   $1540
DOSINI    =   $0C
RAMTOP    =   160
          *=  65       ;TURN OFF NOISY I/O
          .BYTE 0
          *=  $0600    ;INITIALIZATION ROUTINE
STRT
          LDA #SECT&$FF ;POINT SYSTEM RESET
          STA DOSINI   ;VECTOR TO SECT
          LDA #SECT/256
          STA DOSINI+1
          LDA #0       ;POINT INDEX TO
          STA INDEX    ;LAST 1K OF BASIC
          LDA #$BC
          STA INDEX+1
          LDA #0       ;POINT INDEX1 TO $9C00
          STA INDEX1
          LDA #$9C
          STA INDEX1+1
          LDY #0       ;INITIALIZE Y REGISTER
;
;NOW WE MUST MOVE THE LAST 1K OF
;BASIC  INTO  $9C00 SINCE SYSTEM RESET
;WIPES OUT THIS PORTION OF MEMORY
;AFTER RESET IS PRESSED, WE WILL
;RESTORE THE LOST CODE FROM THE
;BUFFER AREA AS $9C00
;
LOOPAGAIN
          LDA (INDEX),Y ;MOVE 1K OF BASIC
          STA (INDEX1),Y ;INTO $9C00
          INY
          CPY #0
          BNE LOOPAGAIN
          INC INDEX+1 ;INCREMENT HIGH BYTE
          LDA INDEX+1
          CMP #$C0     ;ARE WE INTO ROM?
          BEQ DONE     ;YES!
          INC INDEX1+1 ;NOPE,KEEP GOING
          LDY #0
          JMP LOOPAGAIN
DONE
          LDA #1       ;TELL OS WE HAVE
          STA BOOT     ;SUCCESSFULLY BOOTED
          LDA #0       ;RESET COLDSTART FLAG
          STA COLDSTART
          JMP $A000    ;JUMP TO BASIC
          *=  $9B60
```

```
;
;THE RESIDENT CODE WHICH FIXES
;THE BUGS IN ATARI BASIC
;
FIXZERO
        JSR $ABF2
        LDA $D4
        BEQ NOINVERT
        EOR #$80
        STA $D4
NOINVERT
        JMP $ACB1
FIXLOCATE
        JSR $DA51
        LDA #7
        STA $C0
        JMP $BC86
FIXLOCKUP
        BNE JOUT
        DEX
        BNE JOUT
        RTS
JOUT
        JMP $A954
;
;SYSTEM RESET JUMPS TO THESE
;ROUTINES WHICH RESTORE THE
;LAST 1K OF MEMORY
;
SECT
        LDA #$FF      ;TELL BASIC TO DO
        STA WARMSTART ;A WARMSTART
        BNE SECT1
        LDA $179E     ;DOS JUMPS HERE
        CMP #32       ;IF THIS ADDRESS CONTAINS A 32
        BEQ SECT      ;A MEM.SAV FILE WAS USED
        JSR DOSINIT   ;REINIT THE DOS HANDLER
        LDA #0        ;TELL BASIC TO DO
        STA WARMSTART ;A COLDSTART
        JMP CLRR
SECT1
        JSR DOSINIT   ;REINIT THE DOS
CLRR
        JSR CLR       ;LOWER RAMTOP
        LDA #0
        STA INDEX     ;POINT INDEX TO $9C00
        LDA #$9C
        STA INDEX+1
        LDA #0        ;POINT INDEX1 TO $BC00
        STA INDEX1
        LDA #$BC
        STA INDEX1+1
        LDY #0        ;RESET Y
LOOPAAIN
        LDA (INDEX),Y ;MOVE SAVE 1K OF
        STA (INDEX1),Y ;BASIC BACK TO HIGH MEM
        INY
        CPY #0
        BNE LOOPAAIN
        INC INDEX+1
        LDA INDEX+1
        CMP #$A0      ;ARE WE FINISHED?
        BEQ DONEA     ;YES
        INC INDEX1+1
        LDY #0
        JMP LOOPAAIN
DONEA
        LDA #SECT&$FF ;POINT SYSTEM RESET TO SECT
        STA DOSINI
        LDA #SECT/256
        STA DOSINI+1
        JMP $A000     ;JUMP TO BASIC
NAME
        .BYTE "S:"    ;SCREEN HANDLER
CLR
        LDA #152      ;LOWER TOP OF RAM
        STA RAMTOP
        LDX #IOCB1    ;CHANNEL 1
        LDA #OPEN     ;OPEN COMMAND
        STA ICCOM,X
        LDA #NAME&$FF ;SCREEN HANDLER
        STA ICBAL,X
        LDA #NAME/256
        STA ICBAH,X
        LDA #WRIT     ;OPEN FOR WRITE
        STA ICAX1,X
        LDA #0
        STA ICAX2,X
        JSR CIOV      ;DO I/O TO CAUSE RAMTOP
        LDA #CLOSE    ;TO BE RESET
        STA ICCOM,X
        JSR CIOV
        RTS
        *=  $02E2
        .WORD CLR
STOP
        .END
```

●

MIDIMATE
MIDITRACK II
HYBRID ARTS
11920 West Olympic Blvd.
Los Angeles, CA 90064
(213) 826-3777
MIDIMATE — $200.00
MIDITRACK II — 48K $150.00

by Craig Patchett

If you've already read the accompanying article in this issue (page 13) on MIDI (Musical Instrument Digital Interface), then you know that MIDI provides a means of connecting a computer to one or more of a large number of keyboards and music and rhythm synthesizers. If you haven't already read the article, you may want to do so before continuing this review.

MIDI itself is no more than a specification of how the interface should be implemented—to make it functional, an actual implementation (which I'll refer to as a MIDI interface) is required. While the new Atari STs come with a MIDI interface built in, **MIDIMATE** from Hybrid Arts is the only MIDI interface currently available for the 400/600/800/1200 line.

MIDIMATE, as you might imagine, is nothing more than a small metal box with a cable that plugs into the free serial I/O port on your Atari (which means that you can't have a tape recorder and **MIDIMATE** hooked up simultaneously) and jacks for MIDI in, MIDI out, sync in and sync out.

It comes with two six-foot DIN cables that connect its MIDI jacks to the MIDI jacks on your MIDI-compatible instruments, getting all the power it needs through the serial cable. All this for a mere $200 list price, which, incidentally, doesn't include the software **MIDIMATE** requires in order to do anything. Remember high school economics? This is a clear case of low supply and high demand.

Since the power of **MIDIMATE** lies mostly in the software that supports it, let's take a good look at that software. As of the beginning of April, 1985, this is limited to **MIDITRACK II**, also from Hybrid Arts. A modem and other software is forthcoming.

MIDITRACK II is, really, a computerized version of a professional sixteen-track tape recorder (except that: (1) it has far more features; and (2) it records information about how an instrument is played, rather than the sound of the instrument itself). The storage capacity is over 3000 notes.

This "tape recorder" concept is important for those whose musical talents do not include the ability to play keyboards, since, for all the power **MIDITRACK II** offers, you still have to begin by playing notes on a music keyboard. You are not allowed, unfortunately, to enter notes from the computer keyboard, or even give a graphic depiction of what you've played. **MIDITRACK II** is, therefore, aimed more at the musician than at the average home computer user, although Hybrid Arts says that some future software releases will be aimed at the home market.

For the musician, **MIDITRACK II**'s capabilities are extensive, with well over 100 commands available. Since it would be impractical to list all the commands here, I've grouped the most important into categories and provided a summary of each category, below.

Editing commands.

Editing commands are, perhaps, the most important of all the commands; they allow you to fine tune the music that you've played onto a track. **MIDITRACK II**'s most powerful editing commands let you transpose a track (shift the pitch), quantize a track (automatically correct the timing), punch in or punch out (add or change part of a track without affecting the rest), set note duration, set velocity (set the loudness of a track for velocity-sensitive keyboards) and change relative velocity (change the loudness of each note on a track by a constant amount). You can copy, delete, add and combine tracks; perform real-time editing or step-by-step; and choose whether or not to play and/or record program (patch) number changes and pitch wheel, mod wheel and start/stop information. Finally, when recording a track, you can specify which MIDI channel number(s) you want **MIDITRACK II** to listen to.

Track commands.

Track commands allow you to select, turn on and off, and solo any of **MIDITRACK II**'s sixteen tracks. You can also name tracks (and the entire sequence), protect and unprotect tracks from being written over, and save and restore information about the track on/off settings.

Channel commands.

Channel commands let you specify which MIDI channel to use for each track. You can also select to use the channel information originally recorded onto the track.

Sync commands.

Sync (short for synchronization) commands allow you to select one of five sync modes: internal, external I (a standard clock output from tape or drum machine), external II (a TTL clock or beat input through joystick port 2), MIDI and single step. They also allow you to determine how an external clock will be used (you can divide it for compatibility, if necessary) and to lay a clock signal to an external tape recorder.

Tempo commands.

When in internal sync mode, tempo

commands allow you to set, save or recall the current tempo (which can range from 2 to 750 beats per minute, with a beat usually being held a quarter note, or 24 clocks). You can also choose to have a visual and/or audio metronome to help you keep time.

MIDI commands.

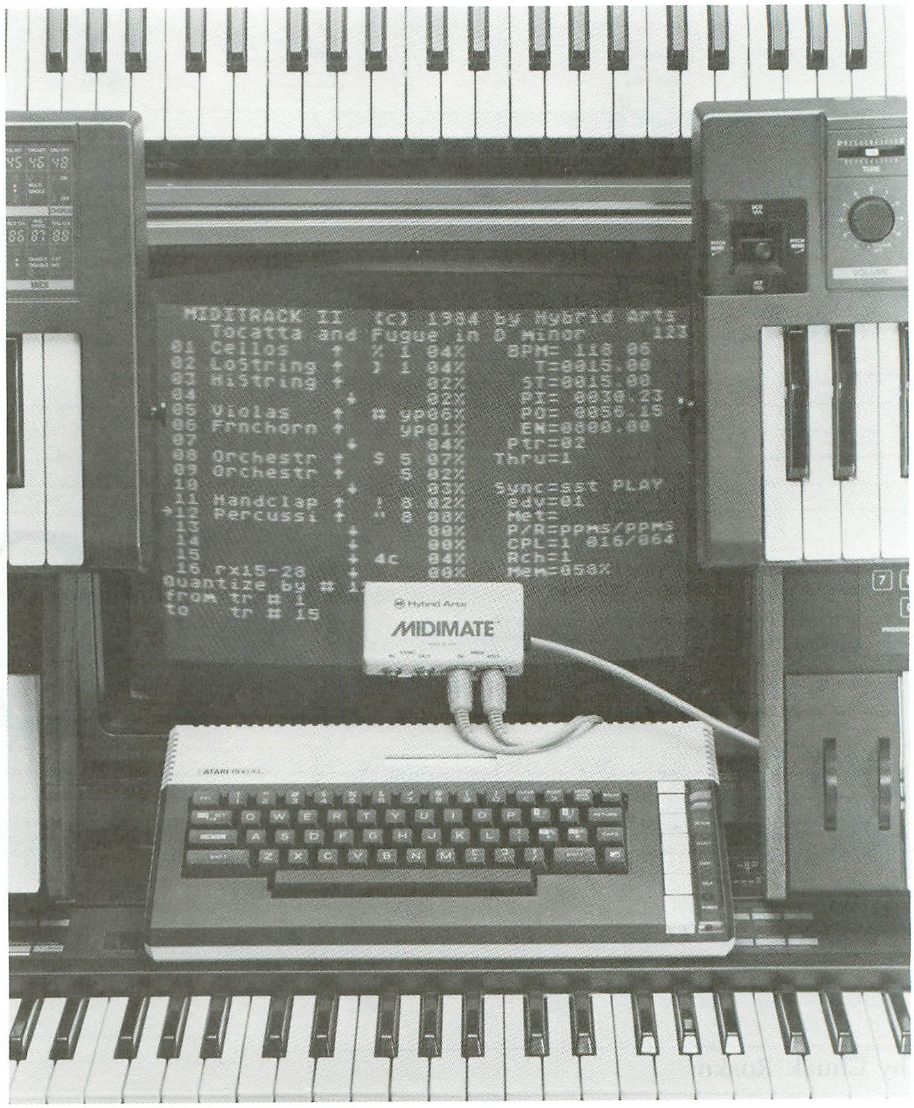
MIDI commands allow you to set the synthesizer program (patch) number, set the current MIDI mode (which determines how a MIDI instrument will listen to incoming information), and send a local on/off command (which determines whether or not information from a MIDI instrument will be sent out).

Disk commands.

Disk commands allow you to save sequences to and recall sequences from disk (three sequences can be stored on a disk), get a list of the sequences stored on a disk, format a new disk and make a backup copy of a sequence disk—or even of **MIDITRACK II** itself.

As you can see, **MIDITRACK II** offers quite a bit of power. Unfortunately, this power comes at a price—and not just monetary. First, the user interface is a perfect example of what a user interface shouldn't be. Commands seldom bear any mnemonic relationship to function, and there are no on-screen menus or help screens to overcome this problem.

The user's manual doesn't help, either. While extensive (94 pages), it is often poorly organized and confusing. At least there's a table of contents and index, and the layout format is pleasing (it was apparently done on a Macintosh). I *have* seen worse. Incidentally, Hybrid Arts' excuse for **MIDITRACK II**'s complexity was something along the lines of "musicians are used to complicated procedures." I'm glad I'm not a musician.



MIDIMATE and MIDITRACK II.

My final complaint on **MIDITRACK II** is (surprise!) the price. At $150 list, it's almost the same price as an Atari 130XE.

On the other hand, a basic four-track tape recorder (remember, **MIDITRACK II** is sixteen-track) will cost you a minimum of $700, so the **MIDIMATE/MIDITRACK II** package is definitely a better deal if you're choosing between the two.

MIDITRACK II will also yield higher quality sound than a tape recorder, since it records digital rather than analog information (although you do have to hook it up to your synthesizers, in order to play back a composition).

In conclusion, despite the drawbacks (some of which may be remedied with future releases), **MIDIMATE** and **MIDITRACK II** are a valuable asset for anyone who's serious about their music, wants to have computer control over their MIDI-compatible instruments and is willing to put up with the hassle of **MIDITRACK II**'s user interface.

Hybrid Arts informs me that the system is already being used by several well known professional musicians. For the casual home user, however, I suggest you wait. With any luck, the near future will bring a more reasonably priced MIDI interface and more generalized software, perhaps even from Hybrid Arts. □

Craig Patchett is the author of several microcomputer books and a loyal Atari supporter since 1979.

NOTE

by Chuck Rosko

Note Master is a one-player musical game designed to test your memory, increasing your ability to recognize tones and corresponding notes. The notes used comprise one octave of the C scale (C, D, E, F, G, A, B, C), starting with middle C. No sharps or flats are used, in order to simplify the game. As you move from left to right on the screen, the pitch of each note increases, just as it does on a piano keyboard.

Playing Note Master.

At the start of each round, the computer will play a random sequence of notes. You must then play back this sequence, in order, before time runs out.

Move your joystick left or right to position your blue marker under the cannon labeled with the note you want to play. To play it, press the trigger button on your joystick. If you've hit the incorrect note, or if the timer to your right has reached the bottom, you'll lose one life.

If the note was correct, it will appear in the middle portion of the screen. If you correctly play back the sequence before the time runs out, you'll move on to the next round.

There are three rounds in each of the twenty levels. As the levels increase, the sequence of notes gets longer, and your time limit increases. You start each game with five lives. If you need to hear the random note sequence played again, move under the cannon labeled *PLAY* and press the trigger button. You can only use this feature three times each game; do so wisely.

MASTER

When the title screen appears, you'll be given two choices. To immediately start playing **Note Master**, press the START key. To practice playing the notes and to get used to the sound of each, press the OPTION key. To get out of practice mode and begin playing, move your marker under the cannon labeled *PLAY* and press your trigger button. The game will then start.

Soon you'll be able to associate the pitch of each note in relation to the others, and their relative positions on a musical scale. See how far you can get . . .see if you can become a **Note Master**. □

Chuck Rosko is a Medical Technologist from Pennsylvania who's been programming for three years. He is the author of ***Number Blunder*** *and* ***Smart Shopper****, two educational programs available through T.H.E.S.I.S., and other programs which you'll be seeing in these pages.*

Program description.

Lines 10-40 — Program startup.
Lines 100-140 — Check timer, if SEC = 0, then decrease timer.
Lines 1000-2070 — Main loop, move marker, play note selected, check if note is correct, incorrect or last one in sequence.
Lines 3000-3020 — Replay the random note sequence; can be done three times per game.
Lines 3100-3220 — Wrong note routine. Check if game is over (LIVE = 0); if so, game ends.
Lines 5000-5070 — Draw main screen.
Lines 6000-6040 — Initialize all variables.
7000-7010 — Flash "round" routine.
Lines 7030-7040 — Flash "level" routine.
Line 7060 — Flashes "please listen" routine.
Lines 7200-7230 — Pick a random note sequence and store it in TUNE$; number of notes depends on the level.
Lines 7250-7300 — Play the random note sequence.
Line 7310 — Initializes timer bar.
9000-9110 — Correct note routine, increase round, increase level.
Lines 10000-10060 — Display title screen; can choose to play or GOTO practice mode.
Lines 32000-32256 — Redefine character set.

List of variables.

C(M) — The letter shot out of the cannon which corresponds to the note played.
E(M) — How far up the letter is shot.
LEVEL — The current level. There are a total of twenty levels.
LIMIT — How long you have before timer bar is decreased. As the level increases, so does the limit.
LIVE — Current number of lives; you start with five.
M — The X position of the blue marker.
N(M) — Pitch of each note.
OX — Position of note in the random note sequence.
PRAC — If PRAC = 0, you will start the game; if PRAC = 1, then GOTO practice mode.
REPLAY — Current number of times random note sequence has been played back, three per game.
ROUND — Current round, three per level.
SEC — Used to keep time. If SEC = 0, then timer bar is decreased.
TUNE$ — Holds random note sequence.
YBAR — Holds Y position of timer bar to be erased.

Listing 1.
BASIC listing.

```
1 REM *************************
2 REM *                       *
3 REM *      NOTE MASTER      *
4 REM *    BY  CHUCK ROSKO    *
5 REM *                       *
6 REM *************************
7 REM
10 DIM XD(15),N(20),E(20),C(20),TUNE$(
20)
15 POKE 559,0:GOSUB 32000:POKE 559,34:
GOSUB 10000:GOSUB 6000:GOSUB 5000
20 IF PRAC=1 THEN POSITION 3,4:? #6;"P
RACTICE  MODE":COLOR 11:PLOT M,22:GOTO
 1000
30 GOSUB 7200:GOSUB 7000:GOSUB 7030:GO
SUB 7060
40 POSITION 17,1:? #6;LIVE:COLOR 11:PL
OT M,22:GOSUB 7310:GOTO 1000
99 REM  TIMER SUBROUTINE
100 SEC=21-INT((PEEK(20)+256*PEEK(19))
/60):IF SEC>(20-LIMIT) THEN RETURN
120 COLOR 160:PLOT 19,YBAR:IF YBAR=22
THEN 140
130 YBAR=YBAR+1:POKE 19,0:POKE 20,0:RE
TURN
140 POSITION 6,23:? #6;"times up":FOR
X=1 TO 3:GOSUB 7110:NEXT X:GOTO 3140
999 REM  MAIN LOOP
1000 IF PRAC<>1 THEN GOSUB 100
1005 FOR X=1 TO 50:NEXT X:J=STICK(0):I
F STRIG(0)=0 THEN 2000
1010 ON XD(J) GOTO 1020,1040,1000
1020 COLOR 166:PLOT M,22:M=M+2:IF M>17
 THEN M=1
1022 COLOR 11:PLOT M,22:GOTO 1000
1040 COLOR 166:PLOT M,22:M=M-2:IF M<1
THEN M=17
1042 COLOR 11:PLOT M,22:GOTO 1000
2000 IF M=17 AND PRAC=0 THEN 3000
2001 IF M=17 AND PRAC=1 THEN PRAC=0:GO
SUB 5000:GOSUB 6000:GOTO 30
2005 COLOR C(M):PLOT M,17:DRAWTO M,E(M
):COLOR 160:PLOT M,17:DRAWTO M,E(M)+1
2006 IF PRAC<>1 THEN IF N(M)=ASC(TUNE$
(OX)) THEN COLOR 160:PLOT M,E(M)-1
2015 SOUND 0,N(M),10,10:POKE 540,60
2020 IF PEEK(540)>30 THEN 2020
2030 COLOR C(M):PLOT M,E(M):DRAWTO M,1
7:COLOR 160:PLOT M,E(M):DRAWTO M,17:SO
UND 0,0,0,0
2035 IF PRAC=1 THEN 1000
2040 IF N(M)<>ASC(TUNE$(OX)) THEN REST
ORE 3160:GOTO 3100
2060 COLOR C(M):PLOT OX,4:COLOR 218:PL
OT M,E(M)-1:IF OX=LEVEL THEN 9000
2070 OX=OX+1:GOTO 1000
2999 REM  REPLAY NOTES
3000 REPLAY=REPLAY+1:IF REPLAY>3 THEN
1000
3020 GOSUB 7250:GOTO 1000
3099 REM  INCORRECT NOTE
3100 POSITION 5,23:? #6;"WRONG NOTE":F
OR X=1 TO 3:GOSUB 7110:NEXT X
3140 COLOR 160:PLOT 5,23:DRAWTO 15,23:
LIVE=LIVE-1:POSITION 17,1:? #6;LIVE:FO
R X=1 TO 3
3150 POSITION 15,0:? #6;"     ":FOR C=
1 TO 15:NEXT C:POSITION 15,0:? #6;"liv
es":GOSUB 7110:NEXT X
3160 IF LIVE=0 THEN RESTORE 3220:GOTO
3180
3170 COLOR 160:PLOT 0,4:DRAWTO 19,4:GO
SUB 7060:GOSUB 7310:GOTO 1000
3180 FOR X=1 TO 15 STEP 2:READ Z:COLOR
 Z:PLOT X,17:DRAWTO X,7:SOUND 0,100,10
,14:SOUND 0,0,0,0:FOR C=1 TO 25
3190 NEXT C:NEXT X
3200 FOR X=1 TO 300:NEXT X:POSITION 0,
0:? #6;CHR$(125):GOSUB 10000:GOSUB 500
0:GOSUB 6000:GOTO 20
3220 DATA 103,97,109,101,207,214,197,2
10
4999 REM  DRAW SCREEN
5000 POKE 708,27:POKE 709,136:POKE 710
,88:POKE 711,73
5005 POSITION 1,18:? #6;"# # # # # # #
 # p   $ $ $ $ $ $ $ $ 1   C D E F G A
 B C a  ";
5010 ? #6;" %█%█%█%█%█%█%█%█y    &(&(&(
&(&(&(&(&(&"
5050 POSITION 0,0:? #6;"ROUND  level
 lives"
5060 COLOR 218:PLOT 1,14:PLOT 3,13:PLO
T 5,12:PLOT 7,11:PLOT 9,10:PLOT 11,9:P
LOT 13,8
5070 PLOT 15,7:COLOR 137:PLOT 0,6:DRAW
TO 19,6:PLOT 0,2:DRAWTO 19,2:RETURN
5999 REM  INITIALIZE
6000 FOR X=5 TO 15:XD(X)=3:NEXT X:XD(7
)=1:XD(11)=2
6010 N(1)=121:N(3)=108:N(5)=96:N(7)=91
:N(9)=81:N(11)=72:N(13)=64:N(15)=60
6020 E(1)=15:E(3)=14:E(5)=13:E(7)=12:E
(9)=11:E(11)=10:E(13)=9:E(15)=8
6030 C(1)=227:C(3)=100:C(5)=101:C(7)=1
02:C(9)=103:C(11)=97:C(13)=98:C(15)=99
6040 ROUND=1:LEVEL=1:LIVE=5:M=1:YBAR=1
3:REPLAY=0:RETURN
6999 REM  FLASH ROUND
```

```
7000 FOR X=1 TO 3:POSITION 0,0:? #6;"
     ":FOR C=1 TO 15:NEXT C:POSITION 0,
0:? #6;"ROUND":GOSUB 7100
7010 NEXT X:POSITION 2,1:? #6;ROUND:RE
TURN
7029 REM  FLASH LEVEL
7030 FOR X=1 TO 3:POSITION 7,0:? #6;"
     ":FOR C=1 TO 15:NEXT C:POSITION 7,
0:? #6;"level":GOSUB 7100
7040 NEXT X:POSITION 9,1:? #6;LEVEL:RE
TURN
7059 REM  FLASH CUE
7060 POSITION 3,4:? #6;"PLEASE  LISTEN
":FOR X=1 TO 300:NEXT X:GOSUB 7250:POS
ITION 3,4:? #6;"              "
7070 RETURN
7099 REM  SOUND SUBROUTINES
7100 FOR C=15 TO 0 STEP -0.5:SOUND 0,1
00,10,C:NEXT C:RETURN
7110 FOR C=15 TO 0 STEP -0.5:SOUND 0,2
00,10,C:NEXT C:RETURN
7120 FOR C=15 TO 0 STEP -0.5:SOUND 0,7
5,10,C:NEXT C:RETURN
7199 REM  PICK RANDOM NOTES
7200 FOR X=1 TO LEVEL:C=INT(RND(0)*8)+
1
7220 Z=121*(C=1)+108*(C=2)+96*(C=3)+91
*(C=4)+81*(C=5)+72*(C=6)+64*(C=7)+60*(
C=8)
7230 TUNE$(X)=CHR$(Z):NEXT X:RETURN
7249 REM  PLAY NOTES
7250 FOR C=1 TO LEVEL:SOUND 0,ASC(TUNE
$(C)),10,10:POKE 540,60
7270 IF PEEK(540)>30 THEN 7270
7280 SOUND 0,0,0,0:POKE 540,60
7290 IF PEEK(540)>10 THEN 7290
7300 NEXT C:RETURN
7309 REM  INITIALIZE TIMER
7310 COLOR 10:PLOT 19,13:DRAWTO 19,22:
POKE 19,0:POKE 20,0:OX=1:YBAR=13:LIMIT
=INT(LEVEL/2):RETURN
8999 REM  CORRECT SEQUENCE
9000 FOR X=1 TO 100:NEXT X:POSITION 5,
23:? #6;"very  good":FOR X=1 TO 3:GOSU
B 7120:NEXT X
9010 FOR X=1 TO 15 STEP 2:COLOR C(X):P
LOT X,E(X):DRAWTO X,17:SOUND 0,N(X),10
,10:FOR C=1 TO 20:NEXT C
9020 COLOR 160:PLOT X,17:DRAWTO X,E(X)
:NEXT X:SOUND 0,0,0,0:U=USR(ADR(RB$))
9040 COLOR 160:PLOT 5,23:DRAWTO 15,23:
PLOT 0,4:DRAWTO 19,4:ROUND=ROUND+1:IF
ROUND>3 THEN 9100
9050 GOSUB 7200:GOSUB 7000:GOSUB 7060:
GOSUB 7310:GOTO 1000
9100 ROUND=1:LEVEL=LEVEL+1:IF LEVEL>20
 THEN LEVEL=20
9110 GOSUB 7030:GOTO 9050
9999 REM  INITIAL SCREEN
10000 POKE 708,27:POKE 709,136:POKE 71
0,88:POKE 711,73
10005 FOR X=4 TO 8:COLOR 137:PLOT 0,X:
DRAWTO 19,X:NEXT X:POSITION 4,1:? #6;"
NOTE  MASTER"
10010 POSITION 3,12:? #6;"by chuck ros
ko":COLOR 122:PLOT 2,4:DRAWTO 17,8
10020 POSITION 0,18:? #6;"TO PLAY-----
--START":POSITION 0,20:? #6;"TO PRACTI
CE---select"
10030 IF PEEK(53279)=5 THEN PRAC=1:GOT
O 10060
10040 IF PEEK(53279)=6 THEN PRAC=0:GOT
O 10060
10050 POKE 711,73*B:FOR X=1 TO 50:NEXT
 X:B=-1*B+1:GOTO 10030
10060 POSITION 0,0:? #6;CHR$(125):RETU
RN
32000 REM  REDEFINE CHARACTER SET
32005 DIM ZZ$(32):RESTORE 32010:FOR I=
1 TO 32:READ A:ZZ$(I)=CHR$(A):NEXT I
32010 DATA 104,104,133,204,104,133,203
,104,133,206,104,133,205,162,4,160,0
32020 DATA 177,203,145,205,136,208,249
,230,204,230,206,202,208,240,96
32022 DIM RB$(21):RESTORE 32025:FOR I=
1 TO 21:READ A:RB$(I)=CHR$(A):NEXT I
32025 DATA 104,169,0,133,20,133,19,105
,1,232,142,22,208,142,10,212,197,19,20
8,245,96
32030 POKE 106,PEEK(106)-5:GRAPHICS 18
:START=(PEEK(106)+1)*256:POKE 752,1
32032 POSITION 5,4:? #6;"coming  up":P
OSITION 4,6:? #6;"note  master"
32040 A=USR(ADR(ZZ$),57344,START):A=US
R(ADR(ZZ$),57344,START):RESTORE 32200
32050 READ X:IF X=-1 THEN RESTORE :GRA
PHICS 17:POKE 756,PEEK(106)+1:RETURN
32060 FOR Y=0 TO 7:READ Z:POKE X+Y+STA
RT,Z:NEXT Y:GOTO 32050
32200 DATA 24,129,195,255,255,255,255,
255,255,32,255,255,255,255,255,255,255
,255
32210 DATA 40,255,255,255,255,255,126,
126,60,48,60,66,129,129,129,129,66,60
32220 DATA 56,231,36,36,36,36,36,36,36
,64,36,36,36,36,231,0,0,0,72,0,0,0,255
,255,0,0,0
32230 DATA 80,127,0,0,0,15,0,0,0,88,60
,126,255,255,255,255,126,60
32231 DATA 264,255,231,195,153,153,129
,153,255
32232 DATA 272,255,131,153,131,153,153
,131,255
32233 DATA 280,255,195,153,159,159,153
,195,255
32234 DATA 288,255,135,147,153,153,147
,135,255
32235 DATA 296,255,129,159,131,159,159
,129,255
32236 DATA 304,255,129,159,131,159,159
,159,255
32237 DATA 312,255,193,159,159,145,153
,193,255
32238 DATA 320,255,153,153,129,153,153
,153,255
32239 DATA 328,255,129,231,231,231,231
,129,255
32240 DATA 336,255,249,249,249,249,153
,195,255
32241 DATA 344,255,153,147,135,135,147
,153,255
32242 DATA 352,255,159,159,159,159,159
,129,255
32243 DATA 360,255,189,153,129,165,189
,189,255
32244 DATA 368,255,153,137,129,129,145
,153,255
32245 DATA 376,255,195,153,153,153,153
,195,255
32246 DATA 384,255,131,153,153,131,159
,159,255
32247 DATA 392,255,195,153,153,153,147
,201,255
32248 DATA 400,255,131,153,153,131,147
,153,255
32249 DATA 408,255,195,159,195,249,249
,195,255
32250 DATA 416,255,129,231,231,231,231
,231,255
32251 DATA 424,255,153,153,153,153,153
,129,255
32252 DATA 432,255,153,153,153,153,195
,231,255
32253 DATA 440,255,189,189,165,129,153
,189,255
```

```
32254 DATA 448,255,153,153,195,195,153
,153,255
32255 DATA 456,255,153,153,195,231,231
,231,255
32256 DATA 464,7,7,6,6,126,254,254,124
,-1
```

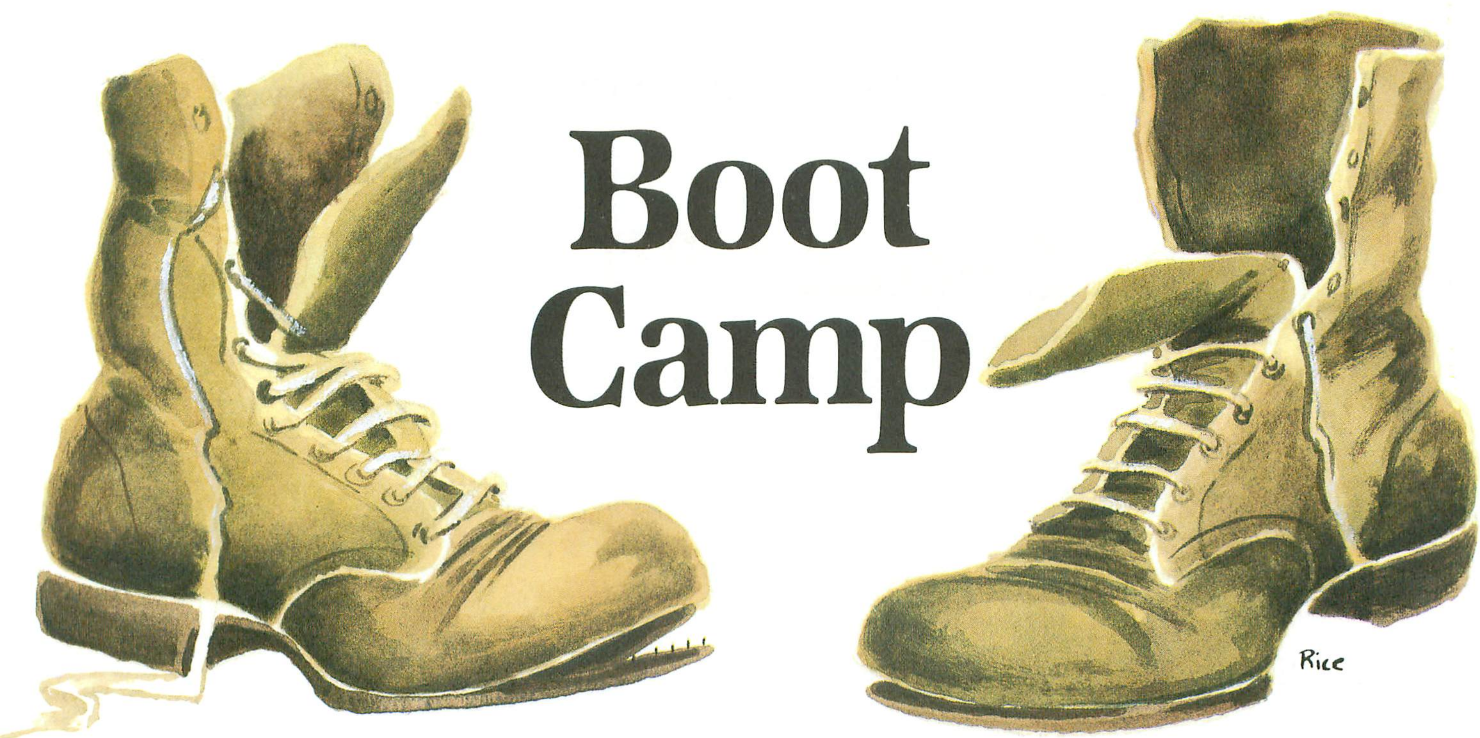
•

CHECKSUM DATA.

(see page 15)

```
1 DATA 242,784,691,897,790,252,999,459
,518,334,59,497,331,315,81,7249
130 DATA 885,151,445,906,689,686,753,1
65,739,167,921,916,667,30,76,8196
2020 DATA 504,948,965,19,949,58,813,59
4,337,694,823,825,376,343,529,8777
3180 DATA 376,498,8,500,527,156,116,74
3,761,307,421,91,232,158,403,5297
6030 DATA 346,597,548,753,111,471,94,9
3,484,143,798,57,517,521,241,5774
7199 DATA 890,9,341,15,92,800,549,656,
553,868,484,490,474,744,891,7856
9020 DATA 452,811,390,204,362,754,516,
567,802,724,520,522,437,169,793,8023
32005 DATA 98,499,222,993,668,153,16,6
54,577,970,277,887,811,681,143,7649
32232 DATA 97,158,163,190,179,166,119,
128,184,137,181,173,162,165,154,2356
32247 DATA 144,119,233,95,115,125,184,
165,138,368,1686
```

•



Boot Camp

by Tom Hudson

Yes, loyal **Boot Camp** fans, it's finally back! **Boot Camp** has been absent from the pages of **ANALOG Computing** for the last few issues because of the time required to get the **ANALOG Computing Telecommunications System** running, as well as the hours spent working with the new Atari ST computer. But enough excuses! Let's get started.

We've covered 6502 assembly language well enough that you should now be able to use some of the built-in features of your Atari computer. This issue, we'll begin working with the Central Input/Output system, or CIO.

Now the fun *really* begins.

You've probably been aching to try input and output operations on your Atari computer in machine language, but, up to this point, haven't had the information necessary to do it. Well, get ready to start learning how, because, beginning with this issue, we'll find out what CIO is and how we can put it to use in machine language programs.

Knowing how to use CIO opens doors to literally any application you want to write: games, utilities, business programs, and so on. With CIO, you can access the built-in screen editor, printers, disk drives, modems, and even create your own device handlers (such as issue 31's **V:** or my **Unicheck** U: device). The possibilities are virtually endless.

This month, we'll begin covering the CIO fundamentals, data storage and the CIO subroutine itself, and demonstrate the use of CIO with simple program segments. Future issues will cover more complex uses of CIO, including multi-file disk operations and modem usage.

In order to have a good reference for the CIO system, I suggest that you invest in the following books: *Mapping the Atari*, by Ian Chadwick from *COMPUTE!* Books, P.O. Box 5406, Greensboro, NC 27403, $14.95; and *Atari Technical Reference Notes*, from Atari Customer Relations, P.O. Box 61657, Sunnyvale, CA 94088, $29.95.

These books, especially the *Atari Technical Reference Notes*, will prove invaluable when you start working with the CIO system.

What is CIO?

When the Atari personal computers were designed, the creators of the operating system (OS) wanted a way for programs to access the various devices present in a computer system, such as disk drives, cassette drives, printers, and so on. In addition, they wanted the devices to be accessed in such a manner

WHY NOT 80 COLUMN DISPLAY ?

Until now, ATARI® owners were limited to 40 column display word processing by available software.

Now, MTS Software presents .....

Column 80

The first 80 column word processor for the ATARI® computers with NO ADDITIONAL HARDWARE REQUIRED.

COLUMN 80 features:

- 80 column display
- Easy to use
- Quick reference manual
- Ideal for home use
- NO ADDITIONAL HARDWARE REQUIRED
- Works with either TV or Monitor
- Advanced formatting capability
- Integrates with other software

Now Only $59.95

See your dealer . . .

Or call or write for more information. For direct orders enclose $59.95 plus $3.00 for shipping and specify UPS or first class mail delivery.

Order Line: 716/634-0578

MTS SOFTWARE
PO Box 623
Williamsville, NY 14221
716-634-0578

COLUMN 80 is available exclusively for the ATARI® computers (48k required).
Developed by Elmer Scharmach of MTS SOFTWARE.
ATARI is a registered trademark of ATARI Inc.

Dealer inquiries invited.

CIRCLE #112 ON READER SERVICE CARD

that the program using the device didn't necessarily have to know what the device was in order to use it.

For example, in the Atari DOS Disk Utility Program (DUP), the user may copy a file from a disk file to a printer or the screen, merely by specifying the device names *P:* or *S:*, respectively. The DUP program doesn't care which device is specified, because both the screen and the printer are accessed through CIO in the same manner. The only thing the DUP program needs to access these devices is the device name, *P:* or *S:*. This is known as "device independence."

What does device independence mean to us? It means that we can call just about any device in the Atari computer system with the same set of commands and parameters. They all conform to a set of governing rules, and to access them, all we do is provide the required information.

Some devices, such as disk drives, have additional capabilities, such as random access (the ability to write or read anywhere in a data file, and not just sequentially). In order to accommodate these additional commands, CIO treats all command numbers above a certain point specially. We'll talk more about this later, when we start working with graphics and disks.

Not only is CIO device-independent when you call it, but it provides a consistent way to report errors. This is known as "unified error handling."

Whenever a CIO device handler encounters an error, it returns a value greater than 128 in the 6502 Y register. If the end of a file is reached, CIO returns a 136 (end of file error) in the Y register. A serial bus error is indicated by the number 140.

The error-handling is called "unified" because each device handler uses the same error numbers to report common errors. That is, both the end of file on a cassette file and the end of file on a disk file are reported as an error 136. In this way, your program can be written to handle common errors automatically for any device used.

As with commands, some devices can generate errors that are unique to that type of device. For example, an error 171 (POINT invalid) can only occur on a disk drive. A cassette I/O operation would never return this code, nor would a keyboard operation. CIO sets up a standard set of error codes, but it allows the flexibility to create your own error numbers if you're writing a special device handler.

Device handlers.

Each device you want to access (screen, disk, cassette, etc.) must have a set of machine language subroutines in memory. These subroutines are known as "device handlers." When the Atari is powered on, it installs a default set of device handlers: the screen editor (E:), screen (S:), cassette (C:), printer (P:) and keyboard (K:). These were considered the essential devices at the time the OS was developed.

Additional device handlers are loaded into memory as needed. The disk handler (D:) is loaded whenever you boot a DOS disk. The RS-232 handler (R:) is loaded from the 850 interface when requested by a special booter program. Other special handlers, such as **Unicheck**'s U: and the **V:** memory storage device, may be loaded by a user program via an AUTORUN.SYS file.

The Atari OS will allow twelve device handlers to be in memory at once. With the five automatically loaded at power-up time, that leaves seven for you to work with if you ever want to add your own handlers (more, I suspect, than you'll ever need). If seven handlers aren't enough (!), you can even *replace* the default handlers with your own! We'll cover this advanced topic in the future—right now, let's find out how to access these device handlers.

The IOCBs.

From a CIO user's point of view, the most important (and most visible) feature of CIO is the Input/Output Control Block (IOCB). You've used the IOCBs in any program that did I/O with a file. For example, in the following BASIC statement:

```
OPEN #1,8,0,"P:"
```

the printer (P:) is opened by using IOCB number 1 (OPEN #1). There are eight IOCBs available for use by programs, numbered 0-7. IOCB number 0 is used by default for the screen editor (E:), which allows you to communicate with your computer via the keyboard and the screen. It's a powerful device, and one which we'll be using extensively.

All the IOCBs are independent of each other and, therefore, can all be used at the same time with different devices. This gives an incredible amount of flexibility in programming complex applications.

Whenever you open an IOCB to use a device, CIO "looks up" the specified device in the device handler table, then sets special system information in the data fields of the IOCB you specified for the I/O operation.

Each I/O operation is accomplished by placing I/O command information in the selected IOCB and then performing a JSR to the Central I/O Vector, CIOV, which is located at $E456. CIOV then takes the command information from the IOCB and attempts to perform the requested operation. Due to the device-independent nature of CIO, all I/O for all devices is done through the CIOV subroutine. This makes I/O programming on the Atari computers very easy.

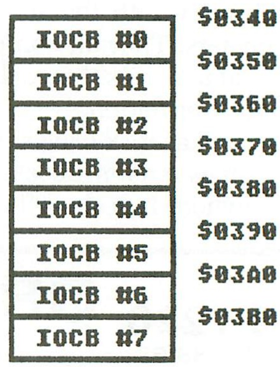
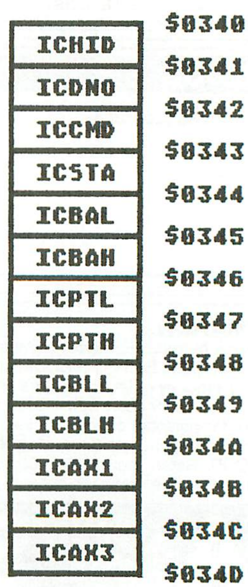


Figure 1.

The eight IOCBs are arranged in memory starting at $0340, as shown in Figure 1. Each IOCB is 16 bytes long. Each byte has a particular function, listed below. The address given for each byte is the address of that field in IOCB #0. We'll see why we use the IOCB #0 addresses in a moment.



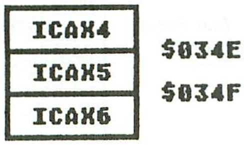


Figure 2.

For the following field descriptions, refer to Figure 2.

ICHID ($0340) is a field used by CIO indicating the I/O handler identification. It is an index into the system device table, set when the IOCB is opened. If the IOCB is not open, this byte will contain $FF. *Do not* change this byte.

ICDNO ($0341) is the device's number. When there can be more than one I/O device of a particular type, such as disk drives, the OPEN command may include a device number after the device's letter code. For example, disk drive number 2 is indicated by *D2:*, and disk drive number 4 is indicated by *D4:*. CIO determines the device number when the IOCB is opened and places the binary value of the device number in this location. For *D2:* then, ICDNO would contain *$02*. This byte is set by the system; *do not* change it.

ICCMD ($0342) is a numeric value provided by the programmer which indicates the function CIO is to perform. This value *must* be set before each call to CIO. The standard commands are: OPEN FILE ($03)—Ready file for I/O; CLOSE FILE ($0C)—Close file (I/O is finished); GET CHARACTERS ($07)—Get *n* bytes from device; PUT CHARACTERS ($0B)—Write *n* characters to device; GET RECORD ($05)—Read a line of text; PUT RECORD ($09)—Write a line of text; GET STATUS ($0D)—Determine device status; and SPECIAL (>$0D)—Device-specific command. These commands will be covered in detail in a few minutes.

ICSTA ($0343) is a 1-byte status code returned by CIO after each CIO call. The high-order (sign) bit is set to indicate error conditions (remember, any status codes greater than 127 indicate that an error has occurred). A status code of $01 indicates that the I/O operation was successfully completed.

ICBAL ($0344) and ICBAH ($0345) make up a 2-byte pointer which contains the address of the I/O data buffer. The data buffer is the place where: (1) data is to be read into or written from, or (2) the device:filename is stored for an OPEN. You set ICBAL to the low-order byte of the data buffer and ICBAH to the high-order byte. You may change this pointer at any time.

ICPTL ($0346) and ICPTH ($0347) is set up to point to the device handler's PUT CHARACTER (−1) routine when the IOCB is opened. This pointer was used by the BASIC cartridge and is not normally used.

ICBLL ($0348) and ICBLH ($0349) make up a 2-byte binary value indicating the buffer size for read and write operations. It is not required for OPEN. After the read or write, CIO will set this value to the number of bytes actually transferred into or out of the data buffer. In this way, you can test for certain errors.

ICAX1 ($034A) and ICAX2 ($034B) are auxiliary information normally used by the OPEN command. ICAX1 controls the input and output functions. The 4 bit is set for input, and the 8 bit is set for output. That is, to open a file for input, place *$04* in ICAX1 before OPENing the IOCB. For output, place *$08* in ICAX1. For both input and output, place *$0C* (12 decimal) into ICAX1. The remaining bits in ICAX1 and all of ICAX2 are device-dependent and, usually, not required. You're probably familiar with these bytes from the BASIC OPEN statement:

```
OPEN#1,8,0,"P:"
```

In this statement, the 8, corresponding to ICAX1, indicates that the file is to be opened for output. The 0, corresponding to ICAX2, indicates that ICAX2 is set to 0 for the printer. ICAX2 is most commonly used for setting the short inter-record gap mode on the cassette, where it has the value 128. These bytes must be set by the user before an OPEN command is issued.

ICAX3 ($034C), ICAX4 ($034D), ICAX5 ($034E) and ICAX6 ($034F) are all auxiliary bytes not normally used by CIO. They're primarily used by certain disk I/O operations, and we aren't concerned about them right now. Just be aware that they're available for use by the I/O device handler and may be set by the user to utilize special functions.

Now that we've defined the IOCB data fields, let's take a look at the various CIO commands and what fields we need to set in order to use them.

Command performance.

The Atari OS has a set of eight basic functions that must be supported by all device handlers. Obvious-

(continued on page 72)

SYNTRON

by Jon Snyder

You were just a renegade space traveler, but as you passed the planet **Syntron**, your ship malfunctioned, and you were forced to land. Now, as you leave your ship, you notice that you aren't alone. . .you're surrounded by menacing and deadly Syntroids. Survival is your only concern.

Typing it in.

Before typing anything, look at the listings accompanying this article.

Listing 1 is the BASIC data and data checking routine. This listing is used to create both cassette and disk versions of **Syntron**. The data statements are listed in hexadecimal (base 16), so the program will fit in 16K cassette systems.

Listing 2 is the assembly language source code for the game of **Syntron**, created with the OSS MAC/65 assembler. You *don't* have to type this listing to play the game! It is included for those readers interested in assembly language.

(continued on next page)

Follow the instructions below to make either a cassette or disk version of **Syntron**.

Cassette instructions.

1. Type Listing 1 into your computer using the BASIC cartridge and verify your typing with **Unicheck** (see page 15).

2. Type *RUN* and press RETURN. The program will begin and ask:

```
MAKE CASSETTE (0) OR DISK (1)?
```

Type *0* and press RETURN. The program will begin checking the DATA statements, printing the line number of each as it goes. It will alert you if it finds any problems. Fix any incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all of your DATA lines are correct, the computer will beep twice and prompt you to *READY CASSETTE AND PRESS RETURN.* Now, insert a blank cassette in your recorder, press the RECORD and PLAY buttons simultaneously and hit RETURN. The message *WRITING FILE* will appear, and the program will create a machine language boot tape version, printing each DATA line number as it goes. When the *READY* prompt appears, the game is recorded and ready to play. CSAVE the BASIC program onto a separate tape before continuing.

4. To play, rewind the tape created by the BASIC program to the beginning. Turn your computer OFF and remove all cartridges. Press the PLAY button on your recorder and turn ON your computer while holding down the START key. If you have a 600 or 800XL computer, you must hold the START and OPTION keys when you turn on the power. The computer will "beep" once. Hit the RETURN key, and **Syntron** will load and run automatically.

Disk instructions.

1. Type Listing 1 into your computer, using the BASIC cartridge and verify your typing with **Unicheck** (see page 15).

2. Type *RUN* and press RETURN. The program will ask:

```
MAKE CASSETTE (0) OR DISK (1)?
```

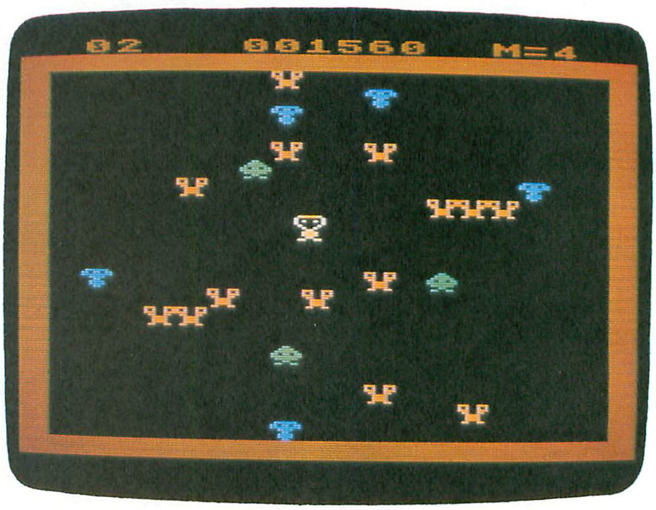
Type *1* and press RETURN. The program will begin checking the DATA lines, printing the line number of each statement as it goes. It will alert you if it finds any problems. Fix incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all the DATA lines are correct, you will be prompted to *INSERT DISK WITH DOS, PRESS RETURN.* Put a disk containing DOS 2.0S into drive #1 and press RETURN. The message *WRITING FILE* will appear, and the program will create an AUTORUN.SYS file on the disk, displaying each DATA line number as it goes. When the READY prompt appears, the game is ready to play. Be sure that the BASIC program is SAVEd before continuing.

4. To play the game, insert the disk containing the AUTORUN.SYS file into drive #1. Turn your computer OFF, remove all cartridges and turn the computer back ON. **Syntron** will load and run automatically.

How to play Syntron.

Syntron is a game for one or two players using joysticks. Select the level of difficulty (level 1 = easy; level 2 = medium, 3 = hard) by pressing the OPTION key until the desired value appears. Use the SELECT key to toggle between one- and two-player games. Press START to begin play.



Syntron.

In single-player mode, one joystick is used to control both player movement and shooting. With the trigger button released, the joystick controls the player's movement about the screen. With the trigger pressed, the joystick fires the player's gun in any of eight directions. Remember, single-player mode does not allow you to move and fire at the same time.

In two-player mode, the joystick in port 1 controls movement, while the joystick in port 2 handles firing direction.

There are three types of Syntroids. The red ones, worth 70 points, are destroyed by a single shot. Hitting a blue Syntrek turns it into a red Syntroid and yields 40 points. The green Syntris is valued at 20 points and turns into a Syntrek when hit.

Completing the tenth wave (and every ten afterwards) will give you a 10,000-point bonus.

Syntroids do not shoot, but all are deadly to the touch—and up to seventy aliens may appear in one wave. The most fun can be had with two players working as a team. One controls the direction; the other, shooting. To pause the game, press the SPACE BAR. Pressing any other key resumes the game. □

Jon Snyder is a senior in high school. He's had an Atari 800 for about 2½ years and started working in assembly language about a year ago. He finds that assembly is more satisfying to work with than BASIC —and not that hard to learn.

Listing 1.
BASIC listing.

```
10 REM *** SYNTRON ***
20 TRAP 20:? "MAKE CASSETTE (0), OR DI
SK (1)";:INPUT DSK:IF DSK>1 THEN 20
30 TRAP 40000:DATA 0,1,2,3,4,5,6,7,8,9
,0,0,0,0,0,0,0,10,11,12,13,14,15
40 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 120:? "CHECKING DATA"
50 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 220
60 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
70 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
EX(D1)*16+HEX(D2)
80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 50
90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
 THEN 50
110 GOTO 220
120 IF PEEK(195)<>6 THEN 220
130 IF PASS=0 THEN 170
140 IF  NOT DSK THEN 160
150 PUT #1,224:PUT #1,2:PUT #1,225:PUT
 #1,2:PUT #1,0:PUT #1,32:CLOSE #1:END
160 FOR X=1 TO 28:PUT #1,0:NEXT X:CLOS
E #1:END
170 IF  NOT DSK THEN 200
180 ? "INSERT DISK WITH DOS, PRESS RET
URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
,"D:AUTORUN.SYS"
190 PUT #1,255:PUT #1,255:PUT #1,0:PUT
 #1,32:PUT #1,187:PUT #1,39:GOTO 210
200 ? "READY CASSETTE AND PRESS RETURN
";:OPEN #1,8,128,"C:":RESTORE 230:FOR
X=1 TO 40:READ N:PUT #1,N:NEXT X
210 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 120:GOTO 50
220 ? "BAD DATA: LINE ";LINE:END
230 DATA 0,16,216,31,255,31,169,0,141,
47,2,169,60,141,2,211,169,0,141,231,2,
133,14,169,56,141,232,2
240 DATA 133,15,169,0,133,10,169,32,13
3,11,24,96
1000 DATA A213A9009D000ACA10FAA2809500
E8D0FB2065E4A90285A120A325203924201025
A99185808D3002A9278D3102,549
1010 DATA 85812049274C5A25A5AD0D8402F0
37AD7802C908B00AA588C9C0B010E688900CC9
0CB008A588C9399002C688AD,645
1020 DATA 78024AB009A589C930900EC68960
4AB008A589C9C4B002E68960A489A200864DBD
8C20990004C8E8E00D90F4A5,178
1030 DATA 888D00D060007E81A5814224183C
3C1866008A1869058587A9008586B4A9C89186
C8918660205F24A200A48FA5,856
1040 DATA 8218691485829002E683E88AD900
0C90EEB9010C18658285829002E683A00060A6
ADD005AD8402D029BD7802C9,985
1050 DATA 0FF0228597A202B598D017A58895
94A5979598209920A58918690695A9A9D785A3
60CA10E260A202B598F01EC9,757
1060 DATA 09B004F6949006C90CB002D69429
02D005F6A94C2B21B5986AB002D6A9CA10DB60
A58838E92D4A4A4A858AA589,29
1070 DATA 38E91F4A4A4A858B60A200868F85
A0202F2120AC20B182F040858E85A0A48FAD0A
D21048B9010CC58AB018A582,14
1080 DATA 69018584A583690020DF21D01F20
E621FE010C4C9421A58238E9018584A583E900
20DF21D00620E621DE010CE6,773
1090 DATA 8FE68FA58FC59290ACA5A0D04C68
684CB626B9000CC58BB018A58269148584A583
690020DF21D0D720E621FE00,448
1100 DATA 0C4C9421A58238E9148584A583E9
0020DF21D0BE20E621DE000C4C94218585A000
B184609182A58E9184A68F60,682
1110 DATA A9078587A202B5A98586A003B90E
2291868810F8B5949D01D0C687CA10E8600018
1800A202869BB594F03038E9,963
1120 DATA 2D4A4A4A858CB5A938E9204A4A4A
858D205F24A48CA200A58218691485829002E6
83E8E48D90F0B182F0032053,163
1130 DATA 22A69BCA10C58D1ED060A900A69B
95949598BD05D06AB0E9B182C942D011A90785
A6203A23A9459182A91485A2,809
1140 DATA C691C983D00FA90485A6A90A85A2
A9429182203A23C9C4D00FA90285A6A9839182
203A23A90A85A260AD04D0F0,273
1150 DATA FAA9A08D03D28D01D2207323C69D
10196868A208BD31239DCE0ACA10F720FD2320
73232049274C5A25A90F8DC0,991
1160 DATA 02A90D8D00D2A21420F023CAD0FA
ACC002888C01D28CC002D0ECA219A9418D00D2
A9488D01D220F023A90A8D00,298
1170 DATA D220F023CAD0E8A9FC8DC0022053
24206824203924207420A5A4859CA5910A8592
2095248D1ED0A9008D01D2AA,345
1180 DATA 20F023E8D0FA6027212D25002F36
2532AD0B0A1865A68D0B0AA204BD070AC91A90
0EE90A9D070AFE060AE002D0,305
1190 DATA 02E69DCA10E860A5A3F00CE6A3F0
098D02D2A9468D03D260A9A0D0F8A900A20295
9495989D01D0CA10F660A5A2,473
1200 DATA F0E8C6A2F00BA9118D00D2A94C8D
01D260A9A0D0F8201222205E2320072120EF21
E69FA59F2903D0EC20F023AD,873
1210 DATA FC02C921F0F9209F22A92D8D0F0A
A91D8D100AA59D09108D110A20822320352020
7420E6A7A5A72907D00620D4,64
1220 DATA 20201A24E693A59325A5D0AF2044
214C9923A003A99038E901D0FB88D0F860A282
20F02320F02320F0238E00D2,339
1230 DATA 8EC402E8D0EE8E01D2A9088D1FD0
60A900858F202F2120AC20B182C945D004A900
9182E68FE68FA58FC59290E6,160
1240 DATA 60A489A2008A990004E8C8E00DD0
F7A97885 89A97685888D1ED060A900A899140A
99000BC8D0F7A9008582A90A,678
1250 DATA 858360205F24A215D00AA901A000
9182A0139182A58218691485829002E683CA10
E8A013A90199140A99B80B88,341
```

```
1260 DATA 10F560A200AD0AD2291FC916B0F7
C90AF0F3C90BF0EFC90CF0EBC90290E79D000C
E8E8E49290DEA200AD0AD229,591
1270 DATA 0F1869029D010CE8E8E49290EFA9
00858F20AC20B182F009C692C692C6914C0525
A59CC9C8B00AE69CA5900A0A,137
1280 DATA C59C9011AD0AD2291FC59CB004A9
C4D006A983D002A9429182E68FE68FA58FC592
90C160A200A9088DF402BD00,993
1290 DATA E09D0008BD00E19D0009E8D0F1A2
27BD32259D0808CA10F760FFFFFFFFFFFFFFFF
E7A5E7663C3C3C667EDBFF3C,565
1300 DATA 183C3C18183C7EDBFF7E1866A518
4299994218A520D52520532420682420707027A5
900A0A0A18691485A8A5A10A,398
1310 DATA 0A0A0A1865A885924A8591A5A485
9C209524A5900A0A0A0A0904859E8DC4022039
24207420A20020F023E8D0FA,643
1320 DATA 4C9923A9038D1DD0A93E8D2F02A9
FC8DC002A90E8DC1028DC2028DC302A9008D07
D4A900A899000499000599000,162
1330 DATA 06990007C8D0F160205324A206A9
0F8DC402BD89269D2E0ABD97269D920ABD9026
9DCE0ACA10EBA20CBD9E269D,96
1340 DATA 530ACA10F7A203A9009D00D0CA10
FAA5A109108DA90AA5AD1869118DE50AA20ABD
AB269D800BCA10F7AD1FD0C9,300
1350 DATA 05D018E6ADA5AD290185AD186911
8DE50AA23120F023CA10FA30E1C903D01CE6A1
A5A1C904D004A90185A11869,578
1360 DATA 108DA90AA23120F023CA10FA30C1
6AB0BE8D1ED0A9008D03D28D02D28D01D2A59E
8DC402A205A9109D070ACA10,626
1370 DATA FA8D020A8D030A6073796E74726F
6E706C6179657273006C6576656C00E2F980EA
EFEE80F3EEF9E4E5F2303225,386
1380 DATA 333300333421323 4A2008E03D28E
01D2ADC402859E203924207323A20ABD28279D
690ABD33279D910ACA10F1A9,496
1390 DATA A68D01D2A59E8DC402A590C90AD0
35A900859046A5A20ABD3E279DB90ACA10F7E6
9DA90085A6EE080A203A23A2,815
1400 DATA 02BDC5021869209DC502CA10F4A5
AC18691085AC8DC80220FD238D1ED04C5D2573
796E74726F6E6F6964730064,547
1410 DATA 657374726F79656400222F2E3533
001110101010A904859DA90085908593 85AC8D
C802A93F85A5A906859EA937,795
1420 DATA 8DC502A9878DC602A9C78DC70260
A900EE030AE69085A4A590C90AD011A9328DC8
02A9108D030AEE020AA9CD85,80
1430 DATA A460707070460000A060606060606
06060606060606060606060606060606064191
27000000000000000000000000,143
1440 REM * 1980 BYTES
```

●

CHECKSUM DATA.

(see page 15)

```
10 DATA 973,351,496,811,423,729,200,60
3,555,573,694,613,29,205,214,7469
160 DATA 758,198,962,648,491,30,155,10
9,169,592,777,809,625,705,774,7802
1060 DATA 123,27,631,875,777,708,685,7
87,791,762,876,637,831,848,969,10327
1210 DATA 744,529,805,814,565,60,992,6
0,565,668,559,881,179,962,857,9240
1360 DATA 4,10,714,960,892,581,134,790
,659,4744
```

●

Listing 2.
Assembly listing.

```
;----------------------
;SYNTRON by Jon Snyder
;----------------------
;
;Page Zero variables
;
         *=  $8Ø
;
DLST     .DS 2
SCRST    .DS 2
SCRS1    .DS 2
PM       .DS 2
HPOSØ    .DS 1        ;Horiz. pos.-PMØ
VERØ     .DS 1        ;Vert. pos.
TEMPX    .DS 1        ;Holds Horiz.
YHLD     .DS 1        ;Holds Vert.
XHOLD    .DS 1
YHOLD    .DS 1
CHARHLD  .DS 1
MNUM     .DS 1        ;Counter
WAVE     .DS 1        ;Wave no.
ALN      .DS 1        ;No. of Syn.'s
WX4      .DS 1        ;(WAVE*4)+2Ø
OVMM     .DS 1        ;counter
HPLR     .DS 3        ;hpos of bullets
FHLD     .DS 1        ;holds stick(1)
MDIR     .DS 3        ;way bullet shot
CTKP     .DS 1        ;X reg. holder
ALPER    .DS 1        ;Syn's per wave
MEN      .DS 1        ;No. of men
S7Ø8     .DS 1        ;Stores loc.7Ø8
OVET     .DS 1        ;Counter
ANYLF    .DS 1        ;Flag
MNNUM    .DS 1        ;Level holder
HTSND    .DS 1        ;Stores
BSHT     .DS 1        ;sounds
SALR     .DS 1        ;Value in ALPER
WV1Ø     .DS 1        ;AND #WV1Ø
PTS      .DS 1        ;Store Syn value
OVSF     .DS 1        ;Counter
MNHLD    .DS 1
PVER     .DS 3        ;Vpos of Bullet
S712     .DS 1        ;Holds background
TEAM     .DS 1
;
;Player and Chset area
;
         *=  Ø
;
PMAREA   .DS $Ø4ØØ
PMØ      .DS $Ø1ØØ
PM1      .DS $Ø1ØØ
PM2      .DS $Ø1ØØ
PM3      .DS $Ø1ØØ
CSET     .DS $Ø2ØØ
SCRN     .DS $Ø2ØØ
MAREA    .DS $Ø1ØØ
;
;Shadow registers
;
ATRACT   =   77
SDMCTL   =   $Ø22F
SDLSTL   =   $Ø23Ø
STICKØ   =   $Ø278
STRIGØ   =   $Ø284
COLORØ   =   $Ø2C4
COLOR1   =   $Ø2C5
COLOR2   =   $Ø2C6
COLOR3   =   $Ø2C7
COLOR4   =   $Ø2C8
PCOLRØ   =   $Ø2CØ
CHBAS    =   $Ø2F4
CH       =   $Ø2FC
;
;Hardware registers
;
HPOSPØ   =   $DØØØ
PØPF     =   $DØØ4
P1PF     =   $DØØ5
GRACTL   =   $DØ1D
HITCLR   =   $DØ1E
CONSOL   =   $DØ1F
AUDF1    =   $D2ØØ
AUDC1    =   $D2Ø1
RANDOM   =   $D2ØA
PMBAS    =   $D4Ø7
SIOINV   =   $E465
;
;Character set equates
;
WALL     =   $Ø1
SYNTROID =   $42
SYNTREK  =   $83
SYNTRI   =   $C4
SYNDED   =   $45
;
;Program starts here...
;
         *=  $2ØØØ
;
GAME     LDX #19
         LDA #Ø
CLTOP    STA SCRN,X   ;Clear out top
         DEX          ;line of screen
         BPL CLTOP
         LDX #$8Ø     ;Clear top half
CLZ      STA $ØØ,X    ;of page zero
         INX
         BNE CLZ
         JSR SIOINV   ;Init sound
         LDA #2       ;Default level
         STA MNNUM
         JSR PMSETUP  ;Set up P/M's
         JSR BLPLØ    ;Put Pl.Ø in mid
         JSR CHSET    ;Set up chset
         LDA # <DLIST ;GR.1 DL
         STA DLST     ;location
         STA SDLSTL
```

```
        LDA # >DLIST
        STA SDLSTL+1
        STA DLST+1
        JSR NWGM
        JMP WSETUP   ;Set up a wave
;
;Check joystick for player movement
;
STCHK   LDA TEAM     ;# players
        ORA STRIGØ
        BEQ STRTS
        LDA STICKØ
        CMP #8       ;Right?
        BCS ST1
        LDA HPOSØ
        CMP #192
        BCS ST2
        INC HPOSØ
        BCC ST2
ST1     CMP #12      ;Left?
        BCS ST2
        LDA HPOSØ
        CMP #57
        BCC ST2
        DEC HPOSØ
ST2     LDA STICKØ   ;Up?
        LSR A
        BCS ST3
        LDA VERØ
        CMP #48
        BCC STRTS
        DEC VERØ
        RTS
ST3     LSR A        ;Down?
        BCS STRTS
        LDA VERØ
        CMP #196
        BCS STRTS
        INC VERØ
STRTS   RTS
;
;Draw your man (PlayerØ)
;
PLTPL   LDY VERØ
        LDX #Ø       ;Turns off
        STX ATRACT   ;attract mode
P1      LDA MUN,X    ;Draw PL.Ø
        STA PMØ,Y
        INY
        INX
        CPX #13
        BCC P1
        LDA HPOSØ
        STA HPOSPØ
        RTS
;
MUN     .BYTE Ø,126,129,165,129,66
        .BYTE 36,24,6Ø,6Ø,24,1Ø2,Ø
;
;The STR's erase their respective
;bullets from a former position
;STRØ-blanks out former position
;of PM to jmp it to new location.
;
STRØ    TXA
        CLC
        ADC # >PM1
        STA PM+1
        LDA #Ø
        STA PM
        LDY PVER,X
        INY
        STA (PM),Y
        INY
        STA (PM),Y
        RTS
;
;Find position of syntronoids
;using MAREA bytes
;
EPLOT   JSR SET18
        LDX #Ø       ;Draws monsters
        LDY MNUM     ;using figures
EPL1    LDA SCRST    ;from MAREA
        CLC
        ADC #2Ø
        STA SCRST
        BCC EPL2
        INC SCRST+1
EPL2    INX
        TXA
        CMP MAREA,Y ;Find vert.bytes
        BCC EPL1
        LDA MAREA+1,Y
        CLC
        ADC SCRST    ;Add horiz pos
        STA SCRST
        BCC EPL3
        INC SCRST+1
EPL3    LDY #Ø
        RTS
;
;Bullet fired? If so,make it go
;the way the stick is pushed
;
STFIR   LDX TEAM     ;# Players
        BNE S1
        LDA STRIGØ
        BNE S4
S1      LDA STICKØ,X ;Stick Ø or 1
        CMP #15      ;pressed?
        BEQ S4       ;No. skip next
        STA FHLD     ;STA strig-FHLD
        LDX #2
S2      LDA MDIR,X   ;Tells way to go
        BNE S3       ;Its gone somwhr
        LDA HPOSØ    ;No dir., so put .
        STA HPLR,X   ;Pl.x at Pl.Ø
        LDA FHLD
        STA MDIR,X   ;Put STICK here
        JSR STRØ
        LDA VERØ     ;Put vpos of
        CLC          ;pl.x
        ADC #6       ;at
        STA PVER,X   ;pl.Ø's position
        LDA #215     ;Set bullet snd
        STA BSHT
        RTS
S3      DEX          ;Check next PM
        BPL S2       ;All chked? No.
S4      RTS
;
;Move the bullet to next position
;
MVPL    LDX #2
MV1     LDA MDIR,X   ;BULLET X moved?
        BEQ XIN      ;No,so skip this
        CMP #9
        BCS LT       ;No move right
        INC HPLR,X   ;Move right
        BCC DW       ;JMP past LEFT
LT      CMP #12
        BCS DW       ;No move left
        DEC HPLR,X   ;Move left
DW      AND #$Ø2     ;Moved down?
        BNE UP       ;No dwn,try up
        INC PVER,X   ;Move down,then
        JMP XIN      ;jmp past up
UP      LDA MDIR,X
        ROR A        ;Move anyway up?
        BCS XIN      ;No up,go to XIN
        DEC PVER,X   ;Move up
XIN     DEX          ;next PM
        BPL MV1
        RTS
;
;Convert position of PMØ to
;that of a GR.1 character
;
CONV    LDA HPOSØ
        SEC
        SBC #45
        LSR A
        LSR A
         LSR A
        STA TEMPX    ;=(HPOSØ-48)/8
        LDA VERØ     ;Vert.conversion
        SEC
        SBC #31
        LSR A
        LSR A
        LSR A
        STA YHLD     ;=(VERØ-31)/8
        RTS
;
;MMOV-move Syntronoid characters
;
MMOV    LDX #Ø
        STX MNUM
        STA ANYLF
MM1     JSR CONV     ;Find Pl.Ø
        JSR EPLOT    ;find Syn. pos.
        LDA (SCRST),Y ;Syn. here?
        BEQ JMM1     ;No.
        STA CHARHLD ;Store SYN type
        STA ANYLF    ;Wave not over
        LDY MNUM
        LDA RANDOM   ;Flip coin H or
        BPL VERT     ;V course to PMØ
        LDA MAREA+1,Y
        CMP TEMPX    ;Where is PMØ?
        BCS LFT
        LDA SCRST    ;SYN moves right
        ADC #1
        STA SCRS1
        LDA SCRST+1
        ADC #Ø
        JSR SETADR
        BNE JMM1
        JSR REPLOT
        INC MAREA+1,X
        JMP JMM1
LFT     LDA SCRST    ;SYN moves LEFT
        SEC
        SBC #1
        STA SCRS1
        LDA SCRST+1
        SBC #Ø
        JSR SETADR
        BNE JMM1
        JSR REPLOT
        DEC MAREA+1,X
JMM1    INC MNUM     ;INC twice to
        INC MNUM     ;chk next SYN.
        LDA MNUM
        CMP WX4      ;All SYN's chked
        BCC MM1      ;no!
        LDA ANYLF    ;ANY SYN's LeFt?
        BNE RETN     ;Yes
        PLA          ;None left-POP
        PLA          ;stack twice
        JMP WVOV     ;Wave is over
;
VERT    LDA MAREA,Y ;Vertical
        CMP YHLD
        BCS UP1
        LDA SCRST    ;PMØ below SYN.
        ADC #2Ø
        STA SCRS1
        LDA SCRST+1
        ADC #Ø
        JSR SETADR
        BNE JMM1
        JSR REPLOT
        INC MAREA,X
        JMP JMM1
UP1     LDA SCRST    ;PMØ above SYN.
        SEC
        SBC #2Ø
        STA SCRS1
        LDA SCRST+1
        SBC #Ø
        JSR SETADR
        BNE JMM1
        JSR REPLOT
        DEC MAREA,X
        JMP JMM1
SETADR  STA SCRS1+1
        LDY #Ø
        LDA (SCRS1),Y
        RTS
;
REPLOT  STA (SCRST),Y ;Erase
        LDA CHARHLD ;Plot 1 up
        STA (SCRS1),Y
        LDX MNUM
RETN    RTS
;
;Draw bullets
;
PLBUL   LDA # >PM3
        STA PM+1
        LDX #2
PBØ     LDA PVER,X
        STA PM
        LDY #3
PB1     LDA BULTS,Y
        STA (PM),Y
        DEY
        BPL PB1
        LDA HPLR,X
        STA HPOSPØ+1,X
        DEC PM+1
        DEX
        BPL PBØ
        RTS
;
BULTS   .BYTE Ø,24,24,Ø
;
;Check-Has bullet hit a SYNTRON
;
CHK     LDX #2
CH1     STX CTKP
        LDA HPLR,X   ;If bullet off-
        BEQ INCHK    ;screen, skip
        SEC          ;Bullets to
        SBC #45      ;playfield
        LSR A
        LSR A
        LSR A
        STA XHOLD
        LDA PVER,X   ;Find vpos
        SEC          ;of bullet on
        SBC #32      ;a GR.1 screen
        LSR A
        LSR A
        LSR A
        STA YHOLD
        JSR SET18    ;Reset SCRST
        LDY XHOLD
        LDX #Ø
CH3     LDA SCRST    ;Find the
        CLC          ;right part
        ADC #2Ø      ;of GR.1 screen
        STA SCRST
        BCC CH3A
        INC SCRST+1
CH3A    INX
        CPX YHOLD
        BCC CH3
        LDA (SCRST),Y ;Anything?
        BEQ INCHK    ;Nothing!
        JSR WHIT     ;SYNTRON THERE!
INCHK   LDX CTKP     ;Next bullet
        DEX          ;Done checking?
        BPL CH1      ;No.
        STA HITCLR
        RTS
WHIT    LDA #Ø       ;Erase bullet
        LDX CTKP     ;from screen.
        STA HPLR,X
        STA MDIR,X
        LDA P1PF,X   ;LDA col. reg.
        ROR A        ;if bullet hits
        BCS INCHK    ;wall,return
        LDA (SCRST),Y
        CMP #SYNTROID
        BNE WH3
        LDA #7       ;Syntroid hit,
        STA PTS      ;worth 7Ø points
        JSR SCORE
        LDA #SYNDED ;Syntroids
        STA (SCRST),Y ;remains
        LDA #2Ø      ;Expl. noise
        STA HTSND
        DEC ALN      ;1 less SYNTRON
WH3     CMP #SYNTREK
        BNE WH4
        LDA #4       ;Hit,he's
        STA PTS      ;worth 4Ø!
        LDA #1Ø
        STA HTSND
        LDA #SYNTROID ;Turn syntrek
        STA (SCRST),Y ;into syntroid
        JSR SCORE
WH4     CMP #SYNTRI
        BNE WH5
        LDA #2       ;Syntri hit,
        STA PTS      ;worth 2Ø!
        LDA #SYNTREK ;He becomes
        STA (SCRST),Y ;syntrek.
        JSR SCORE
        LDA #1Ø
        STA HTSND
WH5     RTS
;
;Check- Is your man hit?
;
PCOL    LDA PØPF
        BEQ WH5      ;No, RETURN
        LDA #16Ø     ;You're hit!
        STA AUDC1+2 ;Some sound
        STA AUDC1
        JSR BLMIS    ;Reset player
        DEC MEN      ;1 less man
        BPL DTHC     ;Men gone? No.
        PLA
        PLA          ;Yes-game over
        LDX #8
GM1     LDA OVER,X   ;"GAME OVER"
        STA SCRN+2Ø6,X
```

```
        DEX
        BPL  GM1
        JSR  LDLAY    ;Flash-Delay
        JSR  BLMIS    ;Clear bullets
        JSR  NWGM     ;Reset game
        JMP  WSETUP
DTHC    LDA  #15      ;Player fizzles
        STA  PCOLRØ
        LDA  #13
        STA  AUDF1
BPC1    LDX  #2Ø
PC1     JSR  DLAY     ;Small delay
        DEX
        BNE  PC1
        LDY  PCOLRØ
        DEY
        STY  AUDC1
        STY  PCOLRØ
        BNE  BPC1
        LDX  #25
PC2     LDA  #65      ;Tone-65
        STA  AUDF1
        LDA  #72
        STA  AUDC1    ;SOUND Ø,X,1Ø,8
        JSR  DLAY
        LDA  #1Ø
        STA  AUDF1
        JSR  DLAY
        DEX
        BNE  PC2
        LDA  #252
        STA  PCOLRØ
        JSR  CLR      ;Clear screen
        JSR  DRWALL   ;Draw walls
        JSR  BLPLØ
        JSR  PLTPL    ;PMØ to middle
        LDA  SALR     ;Proportion
        STA  ALPER    ;of aliens
        LDA  ALN
        ASL  A
        STA  WX4
        JSR  PLTSYN   ;Plot SYN's
        STA  HITCLR   ;Clr collisions
        LDA  #Ø       ;Sound off
        STA  AUDC1
        TAX
PC4     JSR  DLAY     ;A Delay
        INX
        BNE  PC4
        RTS
;
OVER    .SBYTE "GAME OVER"
;
;Adds to and displays score
;
SCORE   LDA  SCRN+11
        CLC
        ADC  PTS
        STA  SCRN+11
        LDX  #4
SCO1    LDA  SCRN+7,X
        CMP  #26      ;internal "9"+1
        BCC  SCO2
        SBC  #1Ø
        STA  SCRN+7,X
        INC  SCRN+6,X
        CPX  #2       ;If over 1ØØØØ
        BNE  SCO2     ;multiple- get
        INC  MEN      ;a free life
SCO2    DEX
        BPL  SCO1
        RTS
;
;Sound of a fired bullet
;
BNOI    LDA  BSHT
        BEQ  BN4
        INC  BSHT
        BEQ  BN2
        STA  AUDF1+2
        LDA  #7Ø
BN3     STA  AUDC1+2
BN4     RTS
BN2     LDA  #16Ø
        BNE  BN3
;
;BLMIS resets bullets when
;you're killed, gmov, or new game
;
BLMIS   LDA  #Ø
        LDX  #2
BI1     STA  HPLR,X
        STA  MDIR,X
        STA  HPOSPØ+1,X
        DEX
        BPL  BI1
        RTS
;
;Sound of a hit monster
;
MSND    LDA  HTSND
        BEQ  BN4
        DEC  HTSND
        BEQ  MS2
        LDA  #17
        STA  AUDF1
        LDA  #76
MS3     STA  AUDC1
        RTS
MS2     LDA  #16Ø     ;Turn off So.Ø
        BNE  MS3
;
;Main loop
;
MAIN    JSR  CHK      ;Bullet to SYN.
        JSR  BNOI     ;Bullet noise
        JSR  MVPL     ;Move PMØ
        JSR  PLBUL    ;Draw bullets
        INC  OVET     ;OVET used so
        LDA  OVET     ;bullets go
        AND  #$Ø3     ;faster than PMØ
        BNE  MAIN
        JSR  DLAY
PAUSE   LDA  CH       ;Is SPACE BAR
        CMP  #33      ;hit?
        BEQ  PAUSE    ;Yes, so pause
        JSR  PCOL     ;Your man hit?
        LDA  #45      ;Internal M
        STA  SCRN+15
        LDA  #29      ;Internal =
        STA  SCRN+16
        LDA  MEN
        ORA  #16
        STA  SCRN+17
        JSR  MSND
        JSR  STCHK    ;Stick check
        JSR  PLTPL    ;Plot PMØ
        INC  OVSF     ;OVSF spaces
        LDA  OVSF     ;bullets apart
        AND  #$Ø7
        BNE  NXT
        JSR  STFIR
        JSR  ERDED
NXT     INC  OVMM     ;OVMM slows
        LDA  OVMM     ;syntronoids
        AND  WV1Ø
        BNE  MAIN
        JSR  MMOV
        JMP  MAIN     ;Re-execute loop
;
;Delay-sets general game speed
;
DLAY    LDY  #3
        LDA  #$9Ø
MN1     SEC
        SBC  #1
        BNE  MN1
        DEY
        BNE  MN1
        RTS
LDLAY   LDX  #13Ø     ;Long delay
LD1     JSR  DLAY
        JSR  DLAY
        JSR  DLAY
        STX  AUDF1
        STX  COLORØ
        INX
        BNE  LD1
        STX  AUDC1
        LDA  #8
        STA  CONSOL
        RTS
;
;Erase any SYNDED character
;
ERDED   LDA  #Ø
        STA  MNUM
ERD1    JSR  CONV
        JSR  EPLOT
        LDA  (SCRST),Y
        CMP  #SYNDED
        BNE  ERD2
        LDA  #Ø
        STA  (SCRST),Y
ERD2    INC  MNUM
        INC  MNUM
        LDA  MNUM
        CMP  WX4
        BCC  ERD1
        RTS
;
;Reset playerØ to center of screen
;
BLPLØ   LDY  VERØ
        LDX  #Ø
        TXA
BL1     STA  PMØ,Y
        INX
        INY
        CPX  #13
        BNE  BL1
        LDA  #12Ø
        STA  VERØ
        LDA  #118
        STA  HPOSØ
        STA  HITCLR   ;Clear collisons
        RTS
;
;Clear screen
;
CLR     LDA  #Ø
        TAY
CLR1    STA  SCRN+2Ø,Y
        STA  SCRN+256,Y
        INY
        BNE  CLR1
;
;Reset SCRST to top of GR.1 scrn
;
SET18   LDA  # <SCRN
        STA  SCRST
        LDA  # >SCRN
        STA  SCRST+1
        RTS
;
;Draws walls-Color in loc. 7Ø8
;
DRWALL  JSR  SET18
        LDX  #21
        BNE  DR1A
DR1     LDA  #WALL    ;Vertical walls
        LDY  #Ø
        STA  (SCRST),Y
        LDY  #19
        STA  (SCRST),Y
DR1A    LDA  SCRST
        CLC
        ADC  #2Ø
        STA  SCRST
        BCC  DR2
        INC  SCRST+1
DR2     DEX
        BPL  DR1
        LDY  #19
DR3     LDA  #WALL
        STA  SCRN+2Ø,Y
        STA  SCRN+$Ø1B8,Y
        DEY
        BPL  DR3
        RTS
;
;Plot monsters in random positions
;
PLTSYN  LDX  #Ø
PLT1    LDA  RANDOM
        AND  #$1F
        CMP  #22      ;not at 22 down
        BCS  PLT1
        CMP  #1Ø
        BEQ  PLT1
        CMP  #11
        BEQ  PLT1
        CMP  #12
        BEQ  PLT1
        CMP  #2
        BCC  PLT1
        STA  MAREA,X  ;Store location
        INX           ;at MAREA
        INX
        CPX  WX4
        BCC  PLT1
        LDX  #Ø
PLT2    LDA  RANDOM   ;Random horiz.
        AND  #$ØF
        CLC
        ADC  #2
        STA  MAREA+1,X
        INX
        INX
        CPX  WX4
        BCC  PLT2
        LDA  #Ø
        STA  MNUM
PLST    JSR  EPLOT
        LDA  (SCRST),Y ;Something
        BEQ  PLT3      ;already there?
        DEC  WX4       ;YES,-1 Syn.
        DEC  WX4
        DEC  ALN       ;WX4/2
        JMP  PLT4
PLT3    LDA  ALPER
        CMP  #2ØØ      ;At Wave 1Ø?
        BCS  SPEC      ;Yes!
        INC  ALPER
        LDA  WAVE
        ASL  A
        ASL  A
        CMP  ALPER     ;Chk proportions
        BCC  REG       ;Syntroids now!
SPEC    LDA  RANDOM
        AND  #$1F
        CMP  ALPER
        BCS  GAL2      ;> store syntrek
        LDA  #SYNTRI   ;< Store syntri
        BNE  REG1
GAL2    LDA  #SYNTREK
        BNE  REG1
REG     LDA  #SYNTROID
REG1    STA  (SCRST),Y
PLT4    INC  MNUM      ;Check next 2
        INC  MNUM      ;bytes
        LDA  MNUM
        CMP  WX4
        BCC  PLST
        RTS
;
;Redefine character set
;
CHSET   LDX  #Ø        ;Character set
        LDA  # >CSET
        STA  CHBAS
CS1     LDA  $EØØØ,X   ;Put ROM set
        STA  CSET,X    ;in RAM
        LDA  $E1ØØ,X
        STA  CSET+256,X
        INX
        BNE  CS1
        LDX  #39
CS3     LDA  ALIS,X    ;Redefine to
        STA  CSET+8,X  ;make SYN.'s
        DEX
        BPL  CS3
        RTS
;
ALIS    .BYTE 255,255,255,255
        .BYTE 255,255,255,255
        .BYTE 231,165,231,1Ø2
        .BYTE 6Ø,6Ø,6Ø,1Ø2
        .BYTE 126,219,255,6Ø
        .BYTE 24,6Ø,6Ø,24
        .BYTE 24,6Ø,126,219
        .BYTE 255,126,24,1Ø2
        .BYTE 165,24,66,153
        .BYTE 153,66,24,165
;
;This sets up a wave-JSR START
;is accessed only at a new game
;
WSETUP  JSR  START
W1      JSR  CLR
        JSR  DRWALL
        JSR  WAVSCR    ;Which wave?
        LDA  WAVE      ;Find no. of
        ASL  A         ;syn.'s that
        ASL  A         ;go on this wave
        ASL  A
        CLC
        ADC  #2Ø
        STA  MNHLD
        LDA  MNNUM
        ASL  A
        ASL  A
        ASL  A
        ASL  A
        CLC
        ADC  MNHLD
        STA  WX4       ;To read-ALN*2
        LSR  A
        STA  ALN
        LDA  SALR
        STA  ALPER
        JSR  PLTSYN    ;plot Syn's
```

```
          LDA WAVE     ;Find
          ASL A        ;color
          ASL A
          ASL A
          ASL A        ;of
          ORA #4
          STA S7Ø8     ;walls
          STA COLORØ
          JSR BLPLØ    ;Reset your man
          JSR PLTPL    ;Draw Pl.Ø
          LDX #Ø
LOOP      JSR DLAY     ;Delay
          INX
          BNE LOOP
          JMP MAIN
;
;Set up P/M graphics
;
PMSETUP   LDA #3       ;Turn on P/M's
          STA GRACTL
          LDA #62      ;DMA
          STA SDMCTL
          LDA #252     ;Pl.Ø color
          STA PCOLRØ
          LDA #14      ;Bullets colors-
          STA PCOLRØ+1 ;white!
          STA PCOLRØ+2
          STA PCOLRØ+3
          LDA # >PMAREA ;Location of
          STA PMBAS    ;P/M Table
          LDA #Ø
          TAY
CLPM      STA PMØ,Y    ;Clear P/M
          STA PM1,Y    ;areas
          STA PM2,Y
          STA PM3,Y
          INY
          BNE CLPM
          RTS
;
;Title screen-Check start key
;
START     JSR CLR
          LDX #6
          LDA #15
          STA COLORØ
SA1       LDA TITL,X   ;"SYNTRON"
          STA SCRN+46,X
          LDA LEVL,X   ;" LEVEL "
          STA SCRN+146,X
          LDA PLRL,X   ;"PLAYERS"
          STA SCRN+2Ø6,X
          DEX
          BPL SA1
          LDX #12
SA2       LDA CRED,X   ;By Jon Snyder
          STA SCRN+83,X
          DEX
          BPL SA2
          LDX #3
          LDA #Ø
PUTØ      STA HPOSPØ,X ;Set bullet
          DEX          ;locations to Ø
          BPL PUTØ
          LDA MNNUM
          ORA #16      ;Put level no.
          STA SCRN+169
          LDA TEAM
          CLC
          ADC #17
          STA SCRN+229
          LDX #1Ø
SA4       LDA PRSA,X   ;"PRESS START"
          STA SCRN+384,X
          DEX
          BPL SA4
SELECT    LDA CONSOL   ;Special keys
          CMP #5       ;SELECT pressed?
          BNE OPTION   ;No.
          INC TEAM
          LDA TEAM
          AND #1
          STA TEAM
          CLC          ;Put # players
          ADC #17
          STA SCRN+229 ;on screen
          LDX #49
SDL1      JSR DLAY
          DEX
          BPL SDL1
          BMI SELECT
OPTION    CMP #3       ;OPTION pressed?
          BNE STT2     ;No.
          INC MNNUM
          LDA MNNUM
          CMP #4
          BNE STT1
          LDA #1
          STA MNNUM
STT1      CLC          ;Put level no.
          ADC #16
          STA SCRN+169 ;on screen
          LDX #49
SDL       JSR DLAY
          DEX
          BPL SDL
          BMI SELECT
STT2      ROR A        ;Start key hit?
          BCS SELECT   ;No!
          STA HITCLR   ;Clear collis.'s
          LDA #Ø
          STA AUDC1+2  ;and sounds
          STA AUDF1+2
          STA AUDC1
          LDA S7Ø8
          STA COLORØ
          LDX #5
          LDA #16      ;Put Ø's at
STT3      STA SCRN+7,X ;score locations
          DEX
          BPL STT3
          STA SCRN+2
          STA SCRN+3
          RTS
;
TITL      .SBYTE "syntron"
PLRL      .SBYTE "players"
LEVL      .SBYTE " level "
CRED      .SBYTE +$8Ø,"by jon snyder"
PRSA      .SBYTE "PRESS START"
;
;End of a wave
;
WVOV      LDX #Ø
          STX AUDC1+2
          STX AUDC1
          LDA COLORØ
          STA S7Ø8
          JSR BLPLØ    ;Clear Pl.Ø
          JSR BLMIS    ;and bullets
          LDX #1Ø
OV1       LDA SYNT,X   ;"SYNTRONOIDS"
          STA SCRN+1Ø5,X
          LDA DSTR,X   ;"DESTROYED"
          STA SCRN+145,X
          DEX
          BPL OV1
          LDA #166
          STA AUDC1
          LDA S7Ø8
          STA COLORØ
          LDA WAVE
          CMP #1Ø      ;1Ø waves done?
          BNE KD
          LDA #Ø       ;Yes!
          STA WAVE
          LSR WV1Ø     ;Speed up action
          LDX #1Ø
RGN       LDA BONUS,X  ;Region X
          STA SCRN+185,X
          DEX
          BPL RGN
          INC MEN
          LDA #Ø
          STA PTS
          INC SCRN+8
          JSR SCORE
          LDX #2
NWCL      LDA COLOR1,X ;Change
          CLC          ;color of Syn.'s
          ADC #32
          STA COLOR1,X
          DEX
          BPL NWCL
          LDA S712
          CLC
          ADC #16
          STA S712
          STA COLOR4
KD        JSR LDLAY
          STA HITCLR
          JMP W1
;
SYNT      .SBYTE "syntronoids"
DSTR      .SBYTE " destroyed "
BONUS     .SBYTE "BONUS 1ØØØØ"
;
;Reset for a new game
;
NWGM      LDA #4       ;Reset men
          STA MEN
          LDA #Ø       ;Reset wave
          STA WAVE
          STA OVMM
          STA S712
          STA COLOR4
          LDA #$3F
          STA WV1Ø
          LDA #6
          STA S7Ø8
          LDA #55      ;Reset Syn.
          STA COLOR1   ;colors
          LDA #135
          STA COLOR2
          LDA #199
          STA COLOR3
          RTS
;
;Puts wave no. at upper left
;
WAVSCR    LDA #Ø
          INC SCRN+3
          INC WAVE
          STA SALR
          LDA WAVE
          CMP #1Ø
          BNE WV2
          LDA #5Ø      ;1Øth Wave!
          STA COLOR4
          LDA #16
          STA SCRN+3
          INC SCRN+2
          LDA #2Ø5     ;Makes all
          STA SALR     ;syntris appear!
WV2       RTS
;
DLIST     .BYTE $7Ø,$7Ø,$7Ø,$46
          .WORD SCRN
          .BYTE 6,6,6,6,6,6,6,6
          .BYTE 6,6,6,6,6,6,6,6
          .BYTE 6,6,6,6,6,6,6,$41
          .WORD DLIST

          *=  $Ø2EØ
          .WORD GAME
          .END
```

●

ATTENTION PROGRAMMERS!

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

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

Send programs to:
Editor, **ANALOG Computing**
P.O. Box 23, Worcester, MA 01603.

SALE

80 CPS PRINTER

SALE

Smith Corona

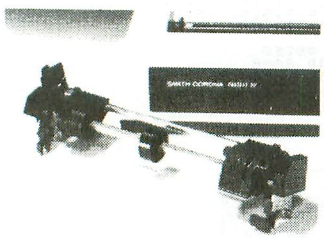
- Commodore 64 • Atari • Apple • More

• 11" Carriage • Friction/Tractor • Graphics

Super Summer Sale
Ends 9-10-85

$149

1 Year Immediate Replacement Warranty



Tractor Feed Included FREE!

SMITH CORONA

The Fastext 80 dot matrix printer from Smith-Corona. A high speed, high flexibility printer with a low suggested manufacturer's retail price of only $299.00. **Sale $149.00**. It does your graphics with ease and prints letters with speeds up to **80 CPS**. ***(Does Commodore graphics with graphics interface.)*** As for flexibility, it has bi-directional printing, friction and tracter feed . Plus six different pitches including condensed and enlarged type for impeccable characters in a variety of sizes. Also standard is a full line buffer and thrifty drop-in ribbon cassettes that yield up to a million characters. Finally, it's compatible with virtually all personal computers. Teamed up with your personal computer, it'll keep track of your transactions, fly through finances, help with homework, even plan the family menu. All this with the assurance of superb Smith-Corona quality built in, makes the Fastext-80 a lot of dot for your dollar. List $299.00. **Super Summer Sale $149.00.**

SPECIFICATIONS:

Size Weight: Height 4" Width 16.5" Depth 9" Weight 8.2 lbs.
Electrical Needs: 120V 60Hz
Internal Char. Coding: ASCII/ISO
Print Buffer Size: 132 Bytes (1 line)
No. of Char. in Char. Set: 96 ASCII/International
Graphics Capability: Standard 60, 72 Dots Per Inch Horizontal, 72 Dots Per Inch Vertical
Pitch: 10, 12, 16.7, 5, 6, 8.3
Printing Method: Impact Dot Matrix
Char. Matrix Size: (Standard) 9H x 8V, (Elongate) 10H x 8V
Printing Features: Bi-directional, Short line seeking
Printing Speed: 80 CPS

PAPER

Type: Plain
Forms Type: Fanfold, Cut Sheet
Max Paper Width: 11"
Feeding Method: Friction Feed Std.; Tractor Feed Included

RIBBON

Type: Cassette — Fabric inked ribbon
Life: 1 million characters

CHARACTER MODE

Character Font: 9 x 8 Standard, 10 x 8 Elongated, No. 8 pin to be used for underline
Character Set: 96 ASCII, 11 x 7 International Char.
Pin Graph Mode: The incoming bit pattern corresponds to the 8 pins in the print head
Resolution: Horizontal: 60, 72 dots/inch, Vertical: 72 dots/inch

PARALLEL INTERFACES

Commodore 64 — $39.00 Apple II, IIe, II + — $59.00 Atari — $59.00

Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES, EXCEPT CANADA.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!
VISA — MASTER CARD — C.O.D.
No C.O.D. to Canada, APO-FPO.

PROTECTO
We Love Our Customers
22292 N. Pepper Rd., Barrington, Illinois 60010
312/382-5244 to order
CIRCLE #115 ON READER SERVICE CARD

PANAK STRIKES!

Reviews of the latest software

by Steve Panak

Well, it seems that I missed a month, due to circumstances beyond my control. I'm none the worse for the wear though, save slightly sore from a somewhat confining canvas jacket. But, now that they've released me, I'm right back flailing with the joystick, pounding at the old keyboard, letting you know just what's going on in current entertainment software.

Little redeeming social value here, just a lot of fun. And, speaking of fun, what could be more fun than cruising a beat on the lunar surface, where the slightest miscalculation can turn you into depressurized mush?

MOON PATROL
by Williams Electronics
ATARI CORP.
P.O. Box 61657
Sunnyvale, CA 94088
16K Cartridge $19.95

Your beat is Sector Nine, an unyieldingly brutal section of the lunar surface, home of some of the roughest, toughest thugs this side of the Pecos Nebula.

Your vehicle is the latest technology can offer, a cosmic cop car crammed with such options as antigravity jump and laser cannons.

You're on **Moon Patrol**, where anything can happen to you . . . and it usually does.

This is the Atari conversion of the Williams arcade game. And while I can't agree that **Moon Patrol** was ever an arcade "smash," I must admit that it offers an exciting gaming experience.

For those unfamiliar with the arcade version, play is simple. You navigate your rover along scrolling scenery, viewed from the side. You jump over craters and blast anything that gets in your way. It's strictly "shoot first and ask questions later" —and no warning shots.

There are two courses, Beginning and Championship. Both are marked into 26 segments (A-Z) and grouped into five sections. Completing a section in record time awards bonus points; completing a course starts you over again.

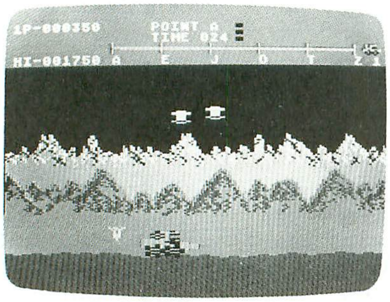
At the end of the game, when all your patrol cars have been reduced to dust, you may either start at the beginning again, or you may continue from your last completed sector. Needless to say, this makes completing the game much simpler.

Control is simple, as well. Left and right on the joystick regulate the speed, while moving the stick up jumps your car over obstacles (can you imagine the shocks on this thing, even in low gravity?) The fire button launches laser missiles both up and forward. The usual extra life, one- or two-player, and pause features round out the game.

The graphics are good, but not quite as detailed as the arcade version. The patrol car itself is disappointing; it more closely resembles a metallic anteater than a lunar rover. However, in compensation, the background does approximate a 3-D effect and is a fine backdrop for the action.

And the action is fast. What was lost in detail is made up in speed, and control is never sluggish. The UFOs dart above, taunting you into a hypnotized stupor as you explode against the rock which has blocked your path. A hovercraft lurks behind you, waiting for the chance to ram you from behind.

Strange plant creatures grab at you as you jump over the craters. Tanks roll in, their plasma cannons spewing luminous licks of death, while the coolant in your own cannon overheats, its noxious vapors singeing your nose like ammonia. Hectic, to say the least.



Moon Patrol.

For those of you familiar with the Atari cartridge manual, you don't need to be told that it's colorful, brief and concise, with a scoring chart displayed on the rear cover for easy reference.

Technically, I can criticize little in the game. On the Beginning course, after you've gone so far, you start to cycle endlessly back to Sector A. Since by then the action has become less than challenging, I consider it more of a signal that you should start the Championship course than a bug. Also, the courses don't seem to be randomly generated.

Overall, **Moon Patrol** is more than worth the money, and makes a fine addition to your game library.

TROLLS AND TRIBULATIONS
by Designer Software
CREATIVE SOFTWARE
P.O. Box 61688
Sunnyvale, CA 94086
48K Disk $24.95

In **Trolls and Tribulations**, I found myself thrust back into one of my favorite places, the dark, damp underground. Lately I've reviewed quite a number of games following this theme, and while **T&T** isn't one of the best I've seen, it may be the most difficult.

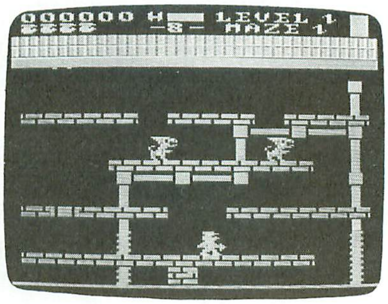
The plot is simple: disguised as a troll, you roam caverns, searching for treasures and avoiding scary skulls, boring buzzards and cretinous cretins, with only a laser cannon to protect yourself.

The caves are divided into mazes and levels. Some mazes require the killing of eight cretins before you may leave. Others need keys to unlock the doors to escape. Still others simply require that you survive until you get to the exit. If you complete five mazes, you receive an extra life.

After you shoot a cretin, it turns into an egg, which must be knocked into the water before it hatches (a la **Joust**). Although you have only eight shots, you can reload by getting to the platform and pulling the joystick down.

In addition to reloading your gun, the joystick also controls all right and left movement, while pulling back shoots, and pushing forward jumps you straight up. Pressing the fire button jumps you in the direction you're facing. At first, these controls are hard to use, but once you get used to them, they allow pinpoint control.

Unlike similar games (**Spelunker** and **Bounty Bob**), **T&T** lacked diversity and surprises. Few new things appear; all mazes are basically of the same pattern, the only differences being more and faster creatures and rearranged doors and keys.



Trolls and Tribulations.

While I can't admit to going through all the levels and mazes, I can say that all mazes on the first three levels are nearly identical. No new methods of transport (such as slides or ladders) appear, although there were trap doors—a nice surprise. Also, by level three, the creatures are nearly invincible.

To survive, a game like this must be more than merely challenging. It must entice you along with the promise of new and wonderful things ready to jump out at you from around the next corner. And, even with 200 rooms spread over 7 levels, **T&T** failed to deliver. I found little motivation to continue.

Among other problems was a misplaced pause button. The SELECT key pauses the game, and it's hard to get to quickly in an emergency. Also, while high scores are saved on the screen, this is about as worthless as a cartridge high score save, since it doesn't save them to the disk.

One plus was the container, which is plastic and opens like a cassette tape holder. It's hard and durable and stacks nicely on your shelf, which is where **T&T** may be likely to spend a lot of its time.

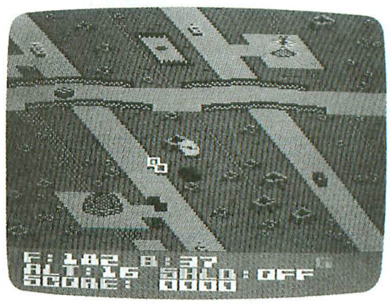
An additional note: when I contacted the manufacturer, they indicated that the programmer said **T&T** required 64K to run. However, I ran my copy successfully on my old 800 with 48K, so you can make what you will of this. They said the programmer must have reduced it down, whatever that means.

BLUE MAX 2001
by Bob Polin
SYNAPSE SOFTWARE
5221 Central Avenue
Richmond, CA 94804
48K Disk $29.95

In **Blue Max 2001**, you are the direct descendant of the **Blue Max**. As airplanes and jets collect dust in museums, you pilot your hovercraft against the evil FURXX empire. The anemic plot adds that you're saving the planet's inhabitants from the FURXX Time Masters, who will drain them of their lifespans to supplement the Masters' own.

But we're not here for plot; we're here for arcade action—and if you're looking for action, you're looking in the wrong place. **Blue Max 2001** combines old concepts with a difficult-to-control joystick, producing a mess.

The game plays much like **Zaxxon**, allowing movement in three dimensions. The joystick controls forward, back, left, right, up and down. The problems involved with placing this much control on the joystick quickly become apparent.



Blue Max 2001.

Since both forward and back movements are activated by the NW and SE compass positions on the stick, and as both contacts must be active at once to move, what happens most often is that, when you intend to move forward, you move up. This problem is worse when you wish to back up or stop the scrolling of the screen, when the danger of crashing to the ground becomes very real.

Further, the button controls both missiles (which shoot in the direction the joystick was last pointed) and bombs (which drop whenever the button is pushed while you're descending). All this becomes just a bit much to keep track of. While I'm sure it can be mastered, the game itself doesn't really seem worth the effort.

I think I would have preferred that the controls all be through the keyboard, but this would necessitate at least six keys for direction and two for missiles, again, quite a bit to keep track of.

To add to the difficulty, you can damage yourself in many ways (crashing into enemies or the ground, or being hit by a missile) and to many different extents. You could lose the ability to maneuver or to fire, or you might run out of fuel. All this is shown at the bottom of the screen, and you can obtain repairs by landing at a friendly base.

The landscape scrolls like **Zaxxon**, but lacks the color and detail, as well as the action. There are things coming at you on occasion, but, because of the difficulty of estimating whether you and the enemy are at the same altitude, I was forced to simply avoid most airborne targets. Instead, I concentrated on bombing and strafing the land-based enemy.

There are plenty of options, and you can change the way the joystick controls the ship (though you can't eliminate the difficulty), as well as a number of other game variables, such as gravity.

Because of the difficulty in getting used to the controls, I found **Blue Max 2001** to be more frustrating than fascinating. I cannot recommend it unless you really have a lot of time to invest in a game that's not very visually interesting.

THE DALLAS QUEST
by Louella Lee Caraway, Phyllis Wapner and James Garon
DATASOFT
19808 Nordhoff Place
Chatsworth, CA 91311
48K Disk $34.95

If someone were to ask me to guess which TV program has just spawned a video spin-off, "Dallas" would probably be one of my last guesses, down with "Gomer Pyle" and "My Mother the Car."

I expect "Buck Rogers" or "Star Trek" to inspire computer programs, but it's hard to imagine the treachery, deceit and hatred of "Dallas" being successfully translated to disk. Whether you can imagine it or not, it's been done—with mixed results.

In **Dallas Quest**, you play a famous detective who's been summoned to the Southfork ranch by Sue Ellen Ewing. She wishes you to retrieve an oil field map from one Chugalug Jones, an old friend of Jock's in South America.

The Dallas Quest.

Sue Ellen hopes that by obtaining the map she can become rich herself, but more important, she can rid herself of J.R. once and for all. She offers you two million dollars to get the map and return it safely to her.

However, what neither of you know is that J.R. is eavesdropping on your conversation, and he has no intention of allowing both of you to follow the little plan to its logical conclusion. And, as you'll quickly find out, a rich man is a dangerous enemy.

You start in Southfork's living room, having just completed your conversation with Sue Ellen. By typing in one- or two-word messages, you maneuver your character throughout the world of "Dallas."

The vocabulary is limited, and, often, you must work to find the synonym the program understands. Aside from this, the program executes smoothly.

One problem, however, is a clue feature. This option lets you view (by typing *CLUE*) one of the numerous clues the computer has to offer. While you get only nine per game, each time you start over you have nine more, so, by utilizing this feature in conjunction with SAVE, you'll see all the clues in no time.

It takes a great deal of will power to keep your fingers from skimming over the *C-L-U-E* keys. While this was a great aid in testing the game (it allowed me to finish in no time), it could ruin the quest's value for those who can't resist the irresistible.

The graphics range from good to excellent. Personally, I prefer all-text adventures, but all pictures here are vivid and finely detailed. In some instances, there are scrolling and moving graphics (i.e., a flying bird).

Also, the pictures can be turned on and off, allowing you to race through familiar territory. This means that you can avoid the time lag as the drive loads data from the disk, even though these loads are somewhat faster than average.

All the usual adventure game features are present. The save game feature is especially nice, allowing you to save up to eight games without a disk swap. Also, it presents you with a menu showing the games saved, along with reference names you've given them. This is a feature which even Infocom games lack, and I found it both innovative and convenient.

The only thing **Dallas Quest** lacks is the sheer hatred of the Ewings. While troublesome at times, they're mere shadows unless you've seen the show.

The puzzles are less problem-solving and more just lucky-guessing. Some examples: the owl must kill the rat, but to do so, you must give it your sunglasses; to escape the anaconda, tickle its chin (information pried from the parrot).

For the diehard "Dallas" fan, **Dallas Quest** is a must, but for the rest of us, it's simply another adventure.

ZONE RANGER
by Dan Thompson
ACTIVISION
P.O. Box 7287
Mountain View, CA 94039
16K Cartridge $24.95

I've heard a lot about Activision, most of it in connection with the 2600. Some of its games populate the bargain bins that sprout up in the local K-Marts and hobby stores. So, when I got hold of **Zone Ranger**, I was dubious and had expectations of the lowest possible quality. I was wrong. It's not a masterpiece, but it isn't all bad, either.

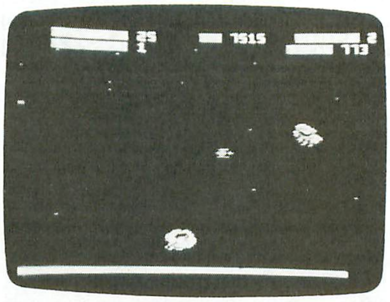
The game is a strange mix of **Asteroids** and **Omega Race**. In the primary part, you're floating in space, attempting to destroy satellites. You must also destroy and avoid floating asteroids, which break up when hit.

There are robots which repair the satellites you disable, and these must be kept at bay, as well. However, you do have help from the Skyway Patrol. You

PANAK STRIKES! *continued*

have to earn their aid by successfully negotiating the Inner Sanctum.

By passing through a Super Portal, you enter the Inner Sanctum, which is basically **Omega Race**. I always liked that game, and I liked the Inner Sanctum, with one reservation.



Zone Ranger.

There are no nasties there. All you do is go through the maze looking for and touching dots. For each dot you touch, the Patrol will eliminate one of your satellites. Your only enemy is the clock, that's ticking off your fuel supply.

Movement is fair, but you don't rotate as in **Asteroid**. Moving the joystick in any direction points you that way, a total of eight trajectories with nothing in between. Speed is also hard to regulate.

The on-screen objects are fairly detailed, but there isn't too much on the screen at once. The asteroids only split once before they disintegrate entirely. Also, while some objects are visible as they near the edge of the screen, they're invisible to your scorching laser and cannot be hit.

So, if you see **Zone Ranger** drastically discounted in a lonely bargain bin somewhere, it could be a buy. Otherwise, it's probably not a good investment.

That's a wrap for this month. I'd like to thank the Magic One Computer Shop of Barberton, Ohio for their valuable assistance in assembling this article. □

MINCE
MARK OF THE UNICORN
222 Third Street
Cambridge, MA 02142
(617) 576-2760
For 520ST $00.00

by Tom Hudson

In my analysis of the 520ST computer system, **Atari's 520ST: Our First Look**, in issue 32 of **ANALOG Computing**, I stated that when we first received the 520ST, it had no text editor included for editing programs.

I neglected to mention that, within two days of receiving the computer, a package arrived from Mark of the Unicorn, Inc., a Cambridge, Massachusetts software company. In the package: you guessed it, the 520ST's first text editor, **Mince**. With **Mince** disk in hand, I proceeded to give it a good breaking in.

What is Mince?

Mince is an extremely powerful text editor for the 520ST computer. It won't work on any of the Atari 8-bit machines, so don't get a copy unless you have (or will have) an ST.

Notice that I said **Mince** is a *text editor*. It's not a full-blown word processor, although with some work and additional software to drive a printer, it could be used as the text editor portion of a word processor. I prefer to think of it more as an editor for creating and changing the source code of programs intended for the ST. In this application, the program does a fine job.

When you purchase the **Mince** text editor, you receive a 3½-inch disk and a large user's guide.

The disk, which you should copy for a backup as soon as you receive it, contains several important files: a configuration program, **Mince** itself, a "swap" file and three tutorial text files.

The configuration program is used to tell **Mince** all about the computer system you're using. **Mince** is designed to work on a number of computer systems, and must be told how to use the features of the 520ST.

To configure **Mince** for the ST, you execute the configuration program and select items from a menu to complete the process. With the version of **Mince** we received, I had a little trouble in the configuration for the 520ST.

The ST is listed as a standard computer on the "terminal type" menu, but a "keyboard bias" figure was incorrect in the standard terminal configuration. I modified these settings to their proper values, and the configuration was complete.

Using Mince.

After you've told **Mince** that it's working on an Atari 520ST, you're ready to start editing text. If you're familiar with the use of advanced text editors, you can probably go right into the **Mince** program itself and type away.

Some users will, of course, be new to the world of electronic text editing, so Mark of the Unicorn has thoughtfully provided eight lessons on the use of **Mince** in their manual. Three of these lessons (numbers four, six and eight) are included on the **Mince** disk, in order to familiarize new users with disk I/O operations.

Lesson one, "Getting Started," shows you how to load **Mince** from the disk. Interestingly, this process is slightly different from the manual's procedure in using the ST's GEM Desktop. You simply point the mouse at the icon labeled MINCE.PRG and click the button twice. **Mince** is then brought into operation.

Lesson one continues, showing how to move the cursor around on the screen and how to enter and modify text.

Lesson number two, "Getting Around Faster," teaches some of **Mince**'s more powerful cursor movement commands: move to beginning of line, end of line, forward a word, backward a word, and handy "universal" repeat function.

With the latter, you can specify repeat values for just about any keypress with a simple CTRL-U sequence. For example, the sequence *CTRL-U 10 ** will place ten asterisks at the current cursor position; *CTRL-U 10 [DELETE]* will delete the previous ten characters. **Mince** is simply packed with commands such as this, which make text manipulation fast and easy.

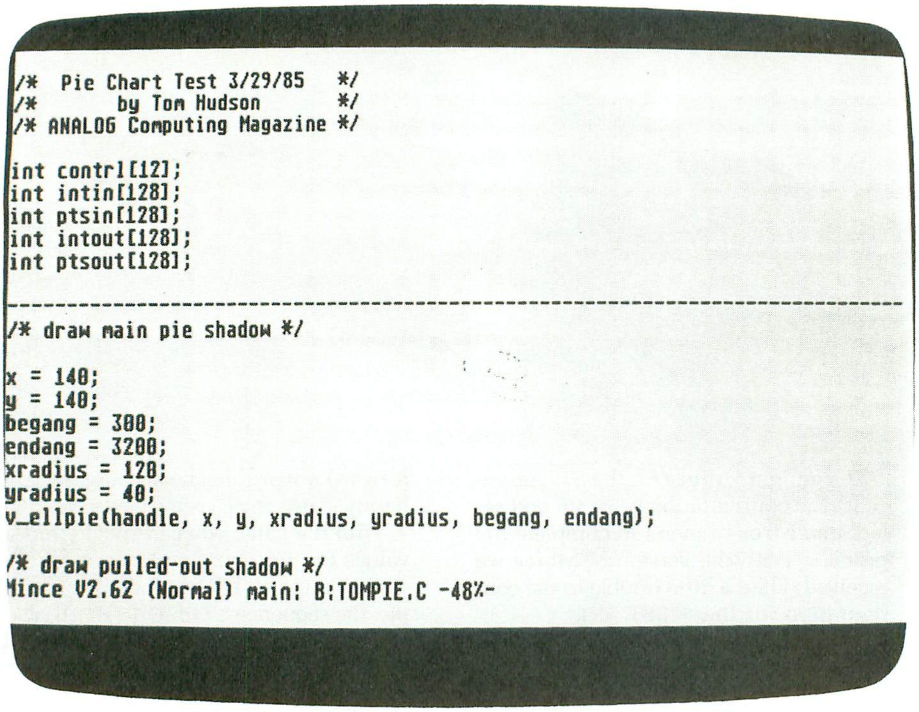
Lesson three, "Reading and Writing Files," teaches the fundamentals of working with a disk filing system, and how **Mince** uses disk files to store text. This lesson also explains the function of the "swap" file provided on the **Mince** disk.

It's used to hold older sections of text that you're not currently using. If you leave the keyboard idle for a specified time (set up when you configure **Mince**), the computer will take the opportunity to write modified pages to the swap file, so that any later page swapping can be done faster.

I have found this swapping action to be a minor nuisance when editing text. For example, pausing for a moment to think of the proper code to use in a program statement sometimes gives **Mince** the chance to do a "swap" operation.

If the swap takes a few seconds and is in progress when I'm ready to type again, I have to wait until the swap is complete before I can continue typing. As I said, it's only a minor complaint and doesn't happen when you're typing in a steady stream.

Lesson four, "Searching," which is included as a text file on the **Mince** disk, teaches about text searches in the program. **Mince** has the ability to search both forward and backward in a text file, which is a terrifically handy feature.



A Mince screen, showing the use of multiple windows.

Mince can also default to the latest search used, if you like. For example, if you search for the word *antidisestablishmentarianism*, and **Mince** finds it, you can stop, perform other editing funtions, then search for *antidisestablishmentarianism* again with only two keystrokes. This can obviously save a lot of typing headaches!

Lesson five, "Keyboard Culture," is a short section discussing conventions used in the world of computers and the subsequent **Mince** lessons, regarding special control-key sequences.

Because it has so many commands available for you, **Mince** must use two types of command key sequences: control characters and meta-commands. Got that?

Control (CTRL) characters are simply normal keyboard keys struck with the CTRL key pressed at the same time. CTRL-S, for example, starts a forward string search, and CTRL-K kills (erases) a line of text.

Meta-commands are prefixed with the ESC key. ESC-D deletes the next word in the text buffer; ESC-U makes the word under the cursor all uppercase characters, and so on.

Some commands use a combination of meta-commands and CTRL characters, such as ESC-CTRL-R, which performs a query on a search-and-replace operation. As you can see, with over 86 text manipulation commands, **Mince** must rely on several key sequences to access all the available commands. Fortunately, Mark of the Unicorn provides a welcome command summary card, which I have hanging in a convenient location above the 520ST.

Lesson six, called "Killing and Moving Text," gives a complete wrap-up of the various commands which allow you to delete or move characters, lines or whole blocks of text.

Deleting a block is easy; you simply place a text mark at the beginning of the text, move your cursor to the end of the block to delete and press CTRL-W, or "Wipe Region." The text instantly disappears.

What if you make a mistake? **Mince** allows you to re-insert the text you deleted by pressing CTRL-Y, or "Yank back deleted text." CTRL-Y can be used any number of times, to place the text in as many places in the text buffer as you like. By using the CTRL-W and CTRL-Y commands together, you can move or copy a block of text anywhere in your document with only a couple of keystrokes.

Fortunately, **Mince** will allow you to save the text from several block delete operations and place the new, larger block elsewhere in the text.

Lesson seven, "Text Processing Commands," gives information on processing words, sentences and paragraphs. **Mince** has a powerful set of commands which enable you to capitalize (change the first character to uppercase), uppercase or lowercase the entire word, starting with the current character. These commands can really be timesavers.

Other commands allow deletion of sentences (ending with periods), right-justification and indentation of paragraphs, and so on.

Finally, lesson eight covers the use of text buffers, a powerful feature which allows **Mince** to store several documents in the computer's memory at the same time. Not only can you look at any of the buffers while they're in memory, but you can use the text movement commands to move text from one buffer to another.

There's even more.

The eight lessons in the **Mince** manual are intended to get the new **Mince** user accustomed to ordinary text entry and manipulation operations. I've tried to give you a fairly complete summary of **Mince**'s commands, although there are far too many to completely summarize here.

Advanced **Mince** users, or those who have used a computer text editor before, can read more about the editor in the *Mince User's Guide*. This guide contains information on creating multiple text windows, where the screen is divided into two sections, each showing a different portion of the text document.

After using **Mince** for more than two months, I can say that I've found it an easy-to-use, well-documented program. With its eight tutorial-style lessons, it can allow even the computer novice to use it effectively, as well as the seasoned programming professional.

So far, all documents prepared with **Mince** have performed perfectly in conjunction with the compiler and assembly programs **ANALOG Computing** uses on the 520ST.

I'm not going to mince words—Mark of the Unicorn's **Mince** is an excellent first editor for the 520ST. And, if it's any indication of the software to come for Atari's 16-bit machine, we've got a lot of good programs to look forward to. □

C.COM

A machine language file utility

by Rich Moore

A little over a year ago, I bought OS/A + and was very pleased with it. . .except for the COPY. COM general purpose copy program. It was 75 sectors long —approaching twice the length of the entire Atari DUP.SYS utility package—and it had to be loaded each time that I needed to use it.

Having dabbled in 6502 machine language for a while on my 400 (with Mosaic 64K board), I felt that a much shorter program could be written to do the job. My first attempt was little more than a bare-bones copier whose only "frill" was the use of a couple of built-in OS/A + functions to read an entire command line, rather than have to resort to prompts for the source and destination filespecs.

It worked perfectly and only took up *three* sectors! With this success for encouragement, I started adding "whistles and bells" to make the program more user-friendly: error identification, copy-append to another file, copy to/from the screen editor without reopening the editor (thereby clearing the screen), *displaying* rather than executing screen editor control codes when copying a file to the editor, repeated write of a copied file *without* reloading the source file, transfer to cassette with short inter-record gaps, forced overwrite of protected files (eliminating the need to unlock them for updating), protection of files after a write, a "wait" feature to permit the changing of a filename when moving a file from one disk to another on a single-drive system, and the ability to rerun the routine without having to reload it.

These additional features lengthened **C.COM** to seven sectors, still less than one-tenth of the OSS standard copy program.

(continued on page 59)

Limitations.

C.COM is *not* intended to be a full replacement for COPY.COM but, rather, to supplement it as a quick utility for handling 99% of the files normally encountered.

The "repeat write" and "override" features make it particularly nice for updating utility files on multiple disks after making a program change, but this feature is mutually exclusive with copying very large files using multiple passes. **C.COM** will *not* copy files larger than memory. However, since very few files exceed 160 sectors in length, and a 32K system can handle 174-sector files, this limitation is rarely felt.

While **C.COM** is inherently incompatible with Atari DOS 2.0S, I followed OSS's guidelines, so that the program works "as advertised" on DOS XL, a very nice disk operating system package.

The program itself occupies 802 bytes. I assembled it between memory locations 9216 and 10017 decimal, to provide some room above the "stock" DOS XL LOMEM for additional drive buffers (more than two drives) and disk file buffers (more than three open at a time).

This assembly permits copying DOS-formatted files up to 174 sectors in length on a 32K system, 305-sector files on a 48K system and 337-sector files on a system with 52K available. If your system has a lower LOMEM (OS/A+ and/or less disk I/O buffer space), a larger copy buffer can be created by reassembling the program at a lower origin.

Program use.

C.COM is named to differentiate it from OSS's COPY.COM and make it easier to call. Syntax for using it is:

```
D1:C [sf[/options]] [df[/options]]
     sf   source filespec
     df   destination filespec
```

The brackets are to indicate that neither of the basic arguments and none of the options are required to use the routine. If the program is called with only a source filespec, the source will be read into memory and written out as the same filespec, with any designated options, after a prompt to set up the "destination."

This is a quick way to move a disk file to another disk, under the same name, on a single-drive system. If no device is specified, the default drive (normally D1:) is assumed. Naming both source and destination filespecs simply copies the source to the destination.

If *no* filespecs are named, a *COPY>* prompt will appear, waiting for the user to input the same arguments and/or options as above, execute the designated copy operation, then return to the *COPY>* prompt.

This mode is terminated by a null line (carriage RETURN only) response to the *COPY>* prompt, returning the user to the DOS XL command processor.

Options.

Options are selected by placing a / immediately following the *final* argument. More than one option may be selected at a time; order does not matter, and only the first / delimiter is required. The options are:

/R — Repeats write operation. Prompts user to set up the destination for each write subsequent to the initial one. Pressing *A* at the prompt instead of SPACE will abort the copy process and enter the COPY> mode.

/W — Waits for the user to set up the destination prior to the first write operation with appropriate prompting. Permits copying a file from one disk to another with a new destination filename on a single-drive system.

/O — Overrides PROtection of a disk file; then forces write.

/P — PROtects destination disk file after writing it out.

/A — Appends source filespec to end of destination disk file. This is a "sector" append in that the source begins on the first byte of a new sector, rather than immediately at the end of the destination disk file within its last sector. This may, in fact, waste a sector, since it's likely that neither source nor destination file entirely fills the final file sector. The combined file may be "packed" by copying it to itself:

D1:C source[/OP]

Options O, P and A are ignored for non-disk destination filespecs.

Examples.

D1:C PROGRAM.OLD PROGRAM.NEW/P — Copies D1:PROGRAM.OLD to D1:PROGRAM.NEW and protects D1:PROGRAM.NEW.

D1:C PROGRAM NEWNAME/WPR — Reads D1:PROGRAM, prompts for destination set-up, writes out D1:NEWNAME, protects D1:NEWNAME, then prompts for another disk (to write out another D1:NEWNAME and protect it).

D1:C UTILITY.COM/OPR — Reads D1:UTILITY.COM, prompts to set up destination (new disk in drive 1), forces write of D1:UTILITY.COM, protects the destination file and prompts for another disk. This is a quick and easy way to replace an old version of a program with a new version on all disks having a copy of that program.

D1:C ORIGINAL.DAT NEW.DAT/A — Appends D1:NEW.DAT to D1:ORIGINAL.DAT starting on a new sector. Pack the file thus created by:

D1:C ORIGINAL.DAT

and leaving the source disk in the drive to have the file replaced.

D1:C C: LETTER.TXT — Copies a file from cassette to D1:LETTER.TXT. Good for moving **Atari**

Writer cassette text files to disk. Tokenized BASIC programs (SAVEd to tape) may also be moved to disk this way, rather than using BASIC. Most cassette boot programs can be moved to disk files this way, but they won't load and execute from disk under DOS without significant machine language modification.

D1:C PROGRAM.OBJ C: — Copies PROGRAM.OBJ to cassette with short inter-record gaps. Files which load and execute from disk under DOS will not boot from cassette unless they are either significantly modified or are the output object code of assembly language programs specifically designed for cassette boot. The disk files of such programs must be single-load files; compound files, such as those produced by AMAC (Atari Macro Assembler), will *not* boot from cassette unless that output is first reduced to a single, contiguous load.

D1:C TEXT E: — Copies a file from D1:TEXT to the screen editor. Memory location 766 decimal is set so that screen control codes are printed, rather than executed (permitting accurate examination of files), then reset to normal after the file is completely printed on the screen.

D1:C TEXT P: — Reads D1:TEXT, then sends it to the printer.

D1:C E: NOTE.TXT — Copies text input from the screen to D1:NOTE.TXT *without* reopening the editor, so that the screen is not cleared. *Every* text line (including the last one) must end with a RETURN, and the input must be terminated with a CTRL-3 (end-of-file). Simple text data files may be set up via the editor, then added to by appending screen editor input whenever desired. For example:

```
D1:C

COPY> E: DATABASE
......NAME......     ...PHONE....
BILL                 800 555-1212
MARY                 800 554-1213
SAM                  800 553-1214
[CTRL-3]

COPY> DATABASE E:
......NAME......     ...PHONE....
BILL                 800 555-1212
MARY                 800 554-1213
SAM                  800 553-1214

COPY> E: DATABASE/A
NANCY                900 123-4567
HERB                 900 987-6543
SUE                  777 666-4321
[CTRL-3]

COPY> DATABASE
Set up destination, hit SPACE

COPY> DATABASE E:
......NAME......     ...PHONE....
BILL                 800 555-1212
MARY                 800 554-1213
SAM                  800 553-1214
NANCY                900 123-4567
HERB                 900 987-6543
SUE                  777 666-4321

COPY>

D1:
```

The above series of COPY> operations: (1) enters COPY> mode; (2) creates a data file with header label and data; (3) prints the file as a template for new data; (4) appends new data; (5) packs the appended data file; (6) prints the "updated" file; and (7) exits **C.COM** to DOS.

Error handling.

Standard CIO error codes are displayed on the screen. If an error is encountered on input, the error is displayed, and the program enters the COPY mode and awaits further action (eliminating the need to reload or rerun **C.COM**).

The user may exit to DOS via a RETURN, to check the disk directory or take other action, then return to COPY> by typing *RUN* with no arguments (an argument will be interpreted as a run address).

For output errors, the program prints the error code, then prompts for "another" destination set-up, so that it's not necessary to reload the (often lengthy) file being copied.

Creating C.COM from BASIC.

Listing 1 is a BASIC program which will create the object code file C.COM on drive 1. Line 900 may be modified to give the program another name. Type it in and save it to disk before RUNning it.

RUN the program, and it will first check to ensure that it was properly entered, then prompt you to insert the disk on which you want the object code file. It will then write out **C.COM** for you to use. Any typing entry errors will be identified by a message referencing the DATA statement where the error occurs.

Listing 2 is AMAC source code for **C.COM**. Since it's actually my first real assembly effort, it is not particularly elegant, but the object code produced is concise, effective and, certainly, useful.

As an inherently lazy programmer who likes utilities, I think I've been successful in making **C.COM** both highly versatile and friendly. □

Rich Moore, a naval Flight Officer with 1900 hours in fighters, has his M.S. in Operations Research. He is currently Data Manager for the Naval War College's computer wargaming system in Newport, Rhode Island. His principal computer interests lie in systems and utilities programming.

C.COM *continued*

Listing 1.
BASIC listing.

```
10 REM *** C.COM BASIC loader ***
30 TRAP 40000:DATA 0,1,2,3,4,5,6,7,8,9
,0,0,0,0,0,0,0,10,11,12,13,14,15
40 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 120:? "CHECKING DATA"
50 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 220
60 IF PEEK(183)+PEEK(184)*256<>LINE TH
EN ? "LINE ";LINE;" MISSING!":END
70 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X)
)-48:D2=ASC(DAT$(X+1))-48:BYTE=HEX(D1)
*16+HEX(D2)
80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 50
90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
 THEN 50
110 GOTO 220
120 IF PEEK(195)<>6 THEN 220
130 IF PASS=0 THEN 180
150 RESTORE 920:FOR X=1 TO 6:READ N:PU
T #1,N:NEXT X:CLOSE #1:END
180 ? :? "Insert disk with DOS, press
RETURN";:INPUT DAT$:RESTORE 900:READ D
AT$:OPEN #1,8,0,DAT$
190 RESTORE 910:FOR X=1 TO 6:READ N:PU
T #1,N:NEXT X:GOTO 210
210 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 120:GOTO 50
220 ? "BAD DATA: LINE ";LINE:END
900 DATA D1:C.COM
910 DATA 255,255,0,36,33,39
920 DATA 224,2,225,2,0,36
1000 DATA 18A50A69038D6125A50B69008D62
25A9008DB526205B25D049A9018DB526206F26
A9078552A9018555A208A027,968
1010 DATA 207326A9058D4203A50A18693F8D
4403A50B69008D4503A9328D4803A9008D4903
A00A910A2056E4A902855220,520
1020 DATA 6926205B25D0016048205F2668A2
008EB326C945F013C943D00AA2CFA026207326
208B26A90420C625A9079D42,805
1030 DATA 03A9229D4403A9279D450338ADE5
02E9229D4803ADE602E9279D49032056E4C088
F00CC003F008983002A0934C,499
1040 DATA 0826BD48038D2027BD49038D2127
E000F003205F26205B25F00BC943F007A9042D
B526F00AA2E9A02620732620,554
1050 DATA 8B26A2018EB326CAAD1127C945F0
1AA9082DB526F005A92420A726A208A9202DB5
26F001E88A20C625A90B9D42,264
1060 DATA 03A9229D4403A9279D4503AD2027
9D4803AD21279D4903EEFE022056E4CEFE0298
3082E000D0034C4725205F26,505
1070 DATA A9102DB526F005A92320A726A902
2DB526F0034CD924A9012DB526F0174C1924A0
0AB10A48201916A00A68D10A,482
1080 DATA D006AD1127A20060A020A200E8C8
B10AC92FD02B8A48C8A204B10AC99BF016DDB6
26F005CA10F83009BDBB260D,672
1090 DATA B5268DB5264C7E2568AAA99BE004
D002A20148689D102710C6AD1127C944F00C48
ADB52629078DB526A20268CA,383
1100 DATA 8E102760A2709D4A03A9039D4203
A9119D4403A9279D4503AD10279D4803A9009D
4903AC1127C043D007A90C8D,319
1110 DATA FC02A9809D4B032056E4A9FF8DFC
02ADB426D0C29810BF686884D4A90085D520AA
D920E6D8A200A0FFC8B1F3C9,784
1120 DATA 30F0F988CAC8E8B1F39DC72610F7
297F9DC726E8A99B9DC726A9098D4203A9C08D
4403A9268D4503A9258D4803,483
1130 DATA A2008E49032056E4205F26ADB326
D0034C19244CD924A270A90C9D42034C56E4A2
0EA027D004A20FA027A9098D,794
1140 DATA 42038E44038C4503A9FF8D4803A2
008E49034C56E4A0FF8CFC02ADFC02C921F00C
C93FD0F58CFC0268684C1924,876
1150 DATA 8CFC0260EEB426A27020CD25CEB4
2660000000141504F57522010080402457272 6F
722023202020202020FDFD9B53,839
1160 DATA 6574207570207 36F757263652C20
68697420D3D0C1C3C5FD9B536574207570 2064
657374696E6174696F6E2C20,706
1170 DATA 68697420D3D0C1C3C5FD9BC3EFF0
F9BE201C9B9B9B9B9B9B9B9B9B9B9B9B9B9B9B
9B9B00000000000000000000,283
```

●

CHECKSUM DATA.
(see page 15)

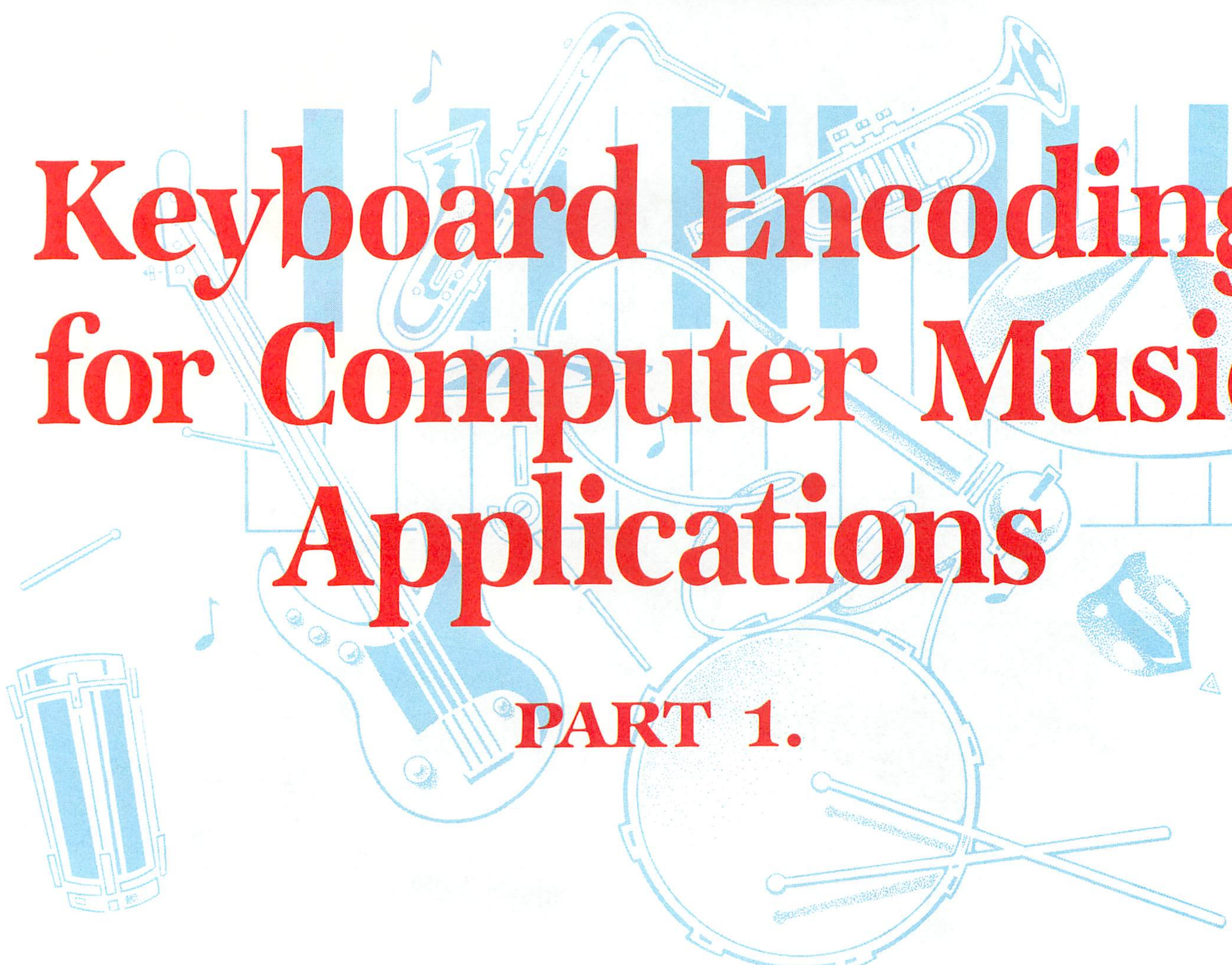
```
10 DATA 430,496,811,423,8,556,603,555,
573,694,613,30,283,642,867,7584
210 DATA 30,155,258,121,61,561,498,591
,749,586,825,617,635,898,901,7486
1100 DATA 627,174,949,701,993,475,539,
999,5457
```

●

Listing 2.
Assembly listing.

```
;
;               C.COM
;
;
;         Short COPY utility
;        for OS/A+ and DOS XL
;
;
;   Copyright 1984      H. R. Moore II
;
;          All rights reserved
;
;--------------------------------------
;
;  Syntax:
;
;          C [sf] [df[/options]]
;
;
;
;       sf => source filespec
;
;          - calling without arguments
;            puts user in "COPY>" mode
;
;          - null line ("RETURN") at
;            prompt returns user to DOS
;
;
;       df => destination filespec
;
;          - omit df to move sf on same
;            device
;
;--------------------------------------
;
;  options:
;
;          /R repeats write; to stop,
;             hit "A" to Abort and
;             go into copy mode
;
;          /W waits for destination
;             set-up. Allows changing
;             filespecs between disks
;             on one-drive systems
;
;          /O overwrite protected sf
;
;          /P protect df after write
;
;          /A appends to destination
;             (disk only)
;
;
;  options may be mixed:
;
;
;          C OLD.OBJ NEW.SYS/RWAP
;
;--------------------------------------
;
```

(continued on page 93)



Keyboard Encoding for Computer Music Applications

PART 1.

by Mario Perdue

One of the more impressive features of the Atari is its ability to produce sound. This capability inspired me to investigate the possibility of interfacing an organ-style keyboard that would enable the Atari to emulate a standard musical instrument. To do this, I needed to find a way to encode the switches of the keyboard, so that each switch would produce a unique digital code.

Note that this article and program are not intended for beginners. You should have a basic understanding of electronics before attempting the project discussed here. At the very least, you *must* know which end of the soldering iron to hold.

The simple encoder.

My first encoder (some refer to this design as the "brute force" encoder) is shown in Figure 1. This is probably the simplest type of encoder to understand. It requires very few parts, is easy to build and works well. Just what we need, right? *Wrong!*

Even with all the aforementioned attributes, this circuit won't do. Unfortunately, it's too simple. Using this design, you can play only *one note at a time.*

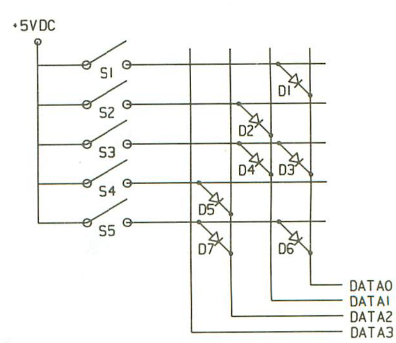


Figure 1.
"Brute force" encoder.

Let's examine this circuit and see why. Referring to Figure 1, if switch S1 is depressed, +5v forward biases diode D1 and raises DATA0 to a 1 level. Simi-

GIVE YOUR FINGERS A BREAK!

Since issue 1, **ANALOG Computing**'s cassette and disk subscriptions have eliminated the need for you to spend hours typing in programs from the magazine.

All of the programs in the magazine are on the cassette and disk versions.

1 year (12 issues) on cassette. $90.00, disk, $130.00
½ year (6 issues) cassette, $48.00, disk, $72.00

To subscribe on cassette or disk, send your check or money order to:

CASSETTE SUBSCRIPTION or **DISK SUBSCRIPTION**
P.O. BOX 615, HOLMES, PA 19043.

For faster service, call our toll-free U.S. order line:
800-345-8112 (in Pennsylvania, call 800-662-2444)

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

A.N.A.L.O.G. COMPUTING™

P.O. BOX 615, HOLMES, PENNSYLVANIA 19043 800-345-8112
(IN PENNSYLVANIA, 800-662-2444)

larly, if switch S2 is depressed, diode D2 forward biases and raises DATA1. It follows that depressing S3 will forward bias both D3 and D4 and raise both DATA0 and DATA1.

Now, what happens when you depress both S1 and S2? Diode D1 will forward bias, as will D2. This raises DATA0 and DATA1. As we've just seen, this condition is exactly the same as if you had depressed S3.

Thus, you can see that depressing multiple keys on this keyboard doesn't result in a chord. In fact, some multiple key depressions won't even result in a change. For instance, if you have S3 down, DATA0 and DATA1 will be high. If you then depress S1, without releasing S3, D1 forward biases, thereby raising DATA0. But DATA0 is already high, so there's no change.

The scanning encoder.

There are, however, other ways to encode a keyboard. One of these methods is scanning the keyboard, one note at a time. Atari's own keyboard (the one you type on) does this.

There are a number of LSI keyboard encoder chips on the market that perform this scanning function, but they're relatively expensive and don't offer enough flexibility. Because of this, I chose not to use any of them. Instead, I've connected a handful (nine, to be exact) of cheap and easy-to-obtain components to do the job.

The two key components in this circuit are the 4024 CMOS 7-stage counter and the 4051 CMOS 8-channel analog multiplier/demultiplier. The names make them sound a bit complex, but, really, they're not. Figure 2 shows a 4024.

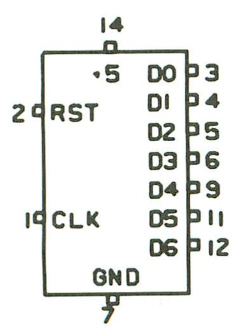


Figure 2.
4024 7-stage counter.

Pin 2 of this chip is the RESET pin. As long as this pin is held low (0v), the chip will perform normally. If this pin is raised to the high state, it will force all the outputs to zero and disable the chip until it's once again set low.

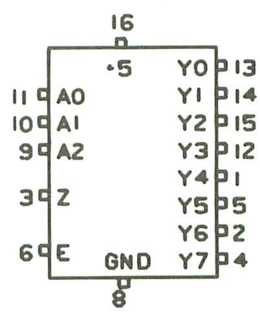
Pin 1 is the CLOCK input. This is the pin that tells the chip to count. It will count up once each time CLOCK is pulsed.

Pins 3, 4, 5, 6, 9, 11 and 12 are the output pins. The least significant bit (LSB) is pin 12 (D0). The most significant bit (MSB) is pin 3 (D6). This chip counts in binary, as shown in the following table.

		0	1	2	3	4	5	6	7	...	125	126	127
B	D6	0	0	0	0	0	0	0	0	...	1	1	1
I	D5	0	0	0	0	0	0	0	0	...	1	1	1
N	D4	0	0	0	0	0	0	0	0	...	1	1	1
A	D3	0	0	0	0	0	0	0	0	...	1	1	1
R	D2	0	0	0	0	1	1	1	1	...	1	1	1
Y	D1	0	0	1	1	0	0	1	1	...	0	1	1
	D0	0	1	0	1	0	1	0	1	...	1	0	1

Table 1.

So, using all 7 bits, this chip could count to 127, more than enough for our purposes.



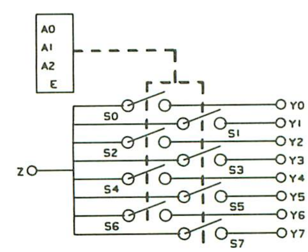


Figure 3A. **Figure 3B.**
4051 8-to-1 bipolar multiplexer/demultiplexer.

The other key component, the 4051, is the most important chip in the keyboard encoder. Inside this chip are eight bilateral CMOS switches. The mechanical equivalent of this chip is shown in Figure 3B. Notice that one side of all the switches is tied to a common line and connected to the Z pin, while the other side is tied to one of the pins Y0 to Y7.

Some of you may be wondering what good these switches are, since there are no toggles on the outside of the chip housing. That's where the address lines (A0 to A2) come in.

If you put a binary *000* on the address lines, S0 closes and Y0 is connected to Z. Referring to the binary count table, you'll see that 001 closes S1, 010 closes S2, and so on, until 111 closes S7. One other pin, the E pin, is significant. It enables the rest of the circuit. If this pin's high, all switches stay open; if it's low, the address lines can select a switch.

How it works.

Now take a look at Figure 4. There's not much to

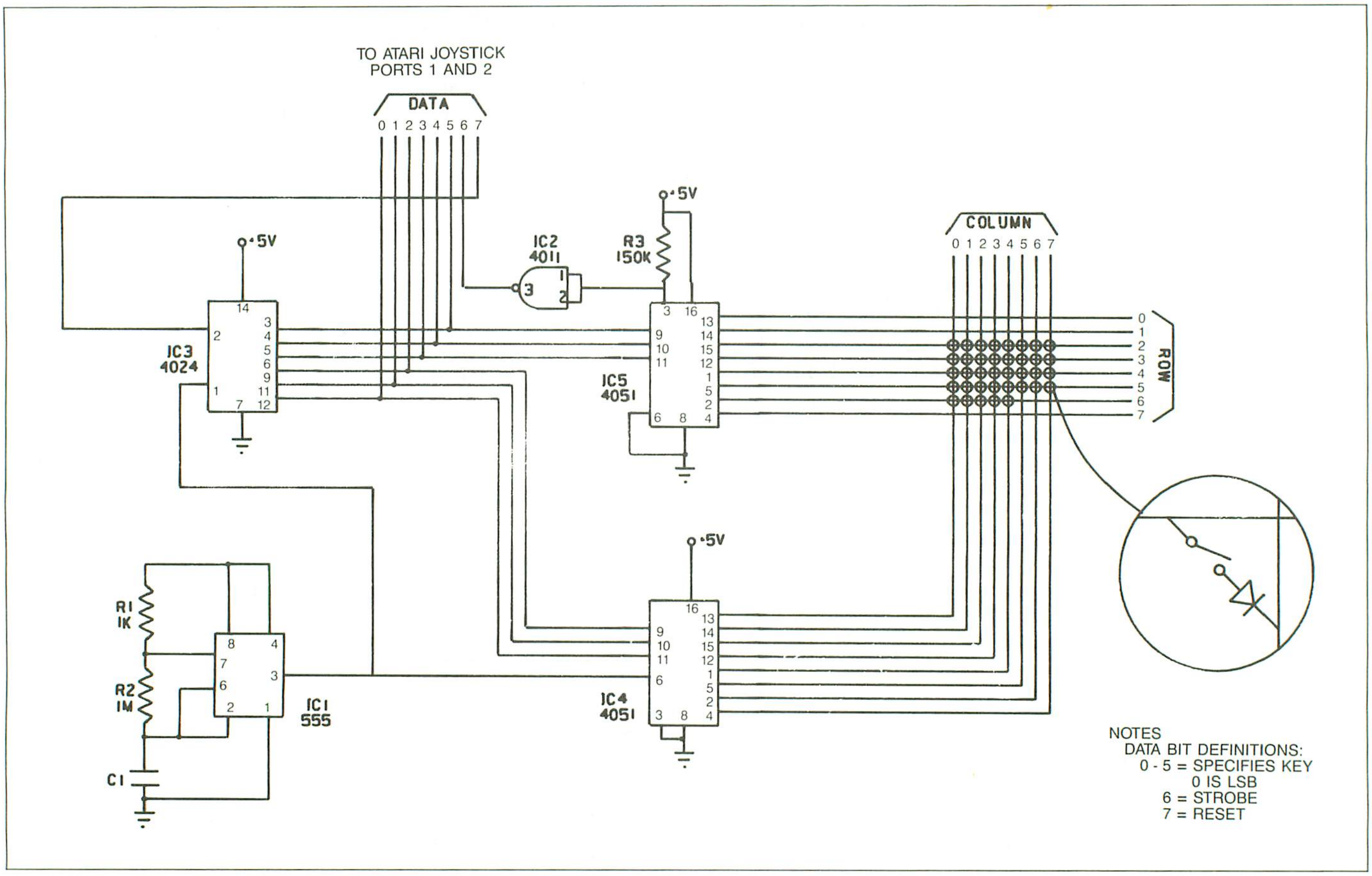


Figure 4.
"Scanning" encoder.

it, but that's our keyboard encoder. Let's examine how it works.

The first part of the circuit is made up of IC1, R1, R2 and C1. These components form an astable multivibrator. The values of R1 and R2 were selected to provide a fairly symmetrical square wave, or what's known as a "50% duty cycle."

The value of capacitor C1 was selected so that the astable clock would run at a frequency of approximately 30 KHz. The output of this clock is applied to the input of the 7-stage counter IC3 pin 1 and, for timing reasons, to IC4 pin 6.

Notice that the first 3 bits (D0-D2) are connected to the address pins of IC4, and the next 3 bits (D3-D5) are connected to the address pins of IC5. The remaining 2 bits are STROBE (D6) and RESET (D7). To start operation, the RESET line is momentarily set high. This forces IC3 to start counting at 000000 and causes both IC4 and IC5's Z pins to be connected to their Y0 pins.

If the two pins are isolated from one another, then the junction between IC4's Z pin and R3 is high, and STROBE remains low. But, if the pins are shorted together by a switch in the matrix, IC4's Z pin will go low. This low level will be inverted by IC2. The resulting high level will be passed on to the computer as STROBE and will serve to indicate that a key is down.

Next, the clock will cycle and increment the count to 000001. This doesn't affect IC4, but IC5 is now connected to its Y1 pin. As above, if these points aren't connected, nothing happens. If they are connected, STROBE goes high. Each clock cycle advances the counter. With each counter advance, a new switch in the matrix will be examined. The signal STROBE will be high for any key which is down.

Any time you look at the six data lines (D0 to D5), the number appearing there will be the number of the key being examined at that instant. If STROBE (D6) is also high, then the key being examined at that instant is down.

(continued on page 68)

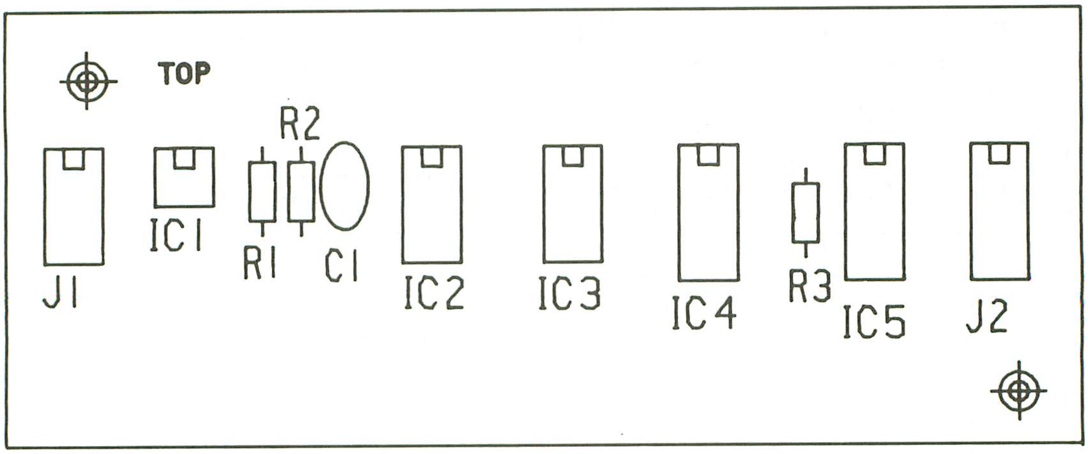


Figure 7A.

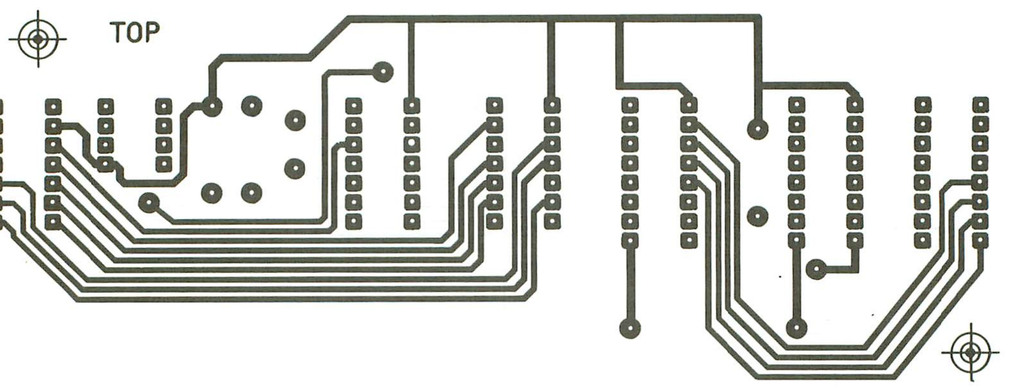


Figure 7B.

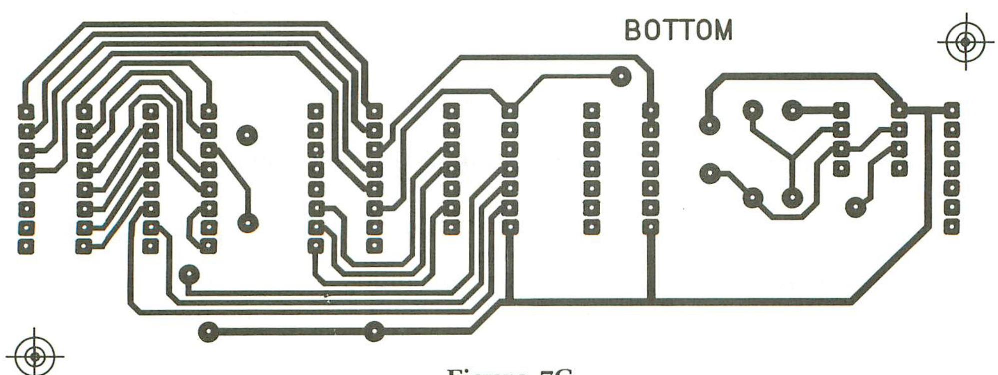


Figure 7C.

Circuit board layout.

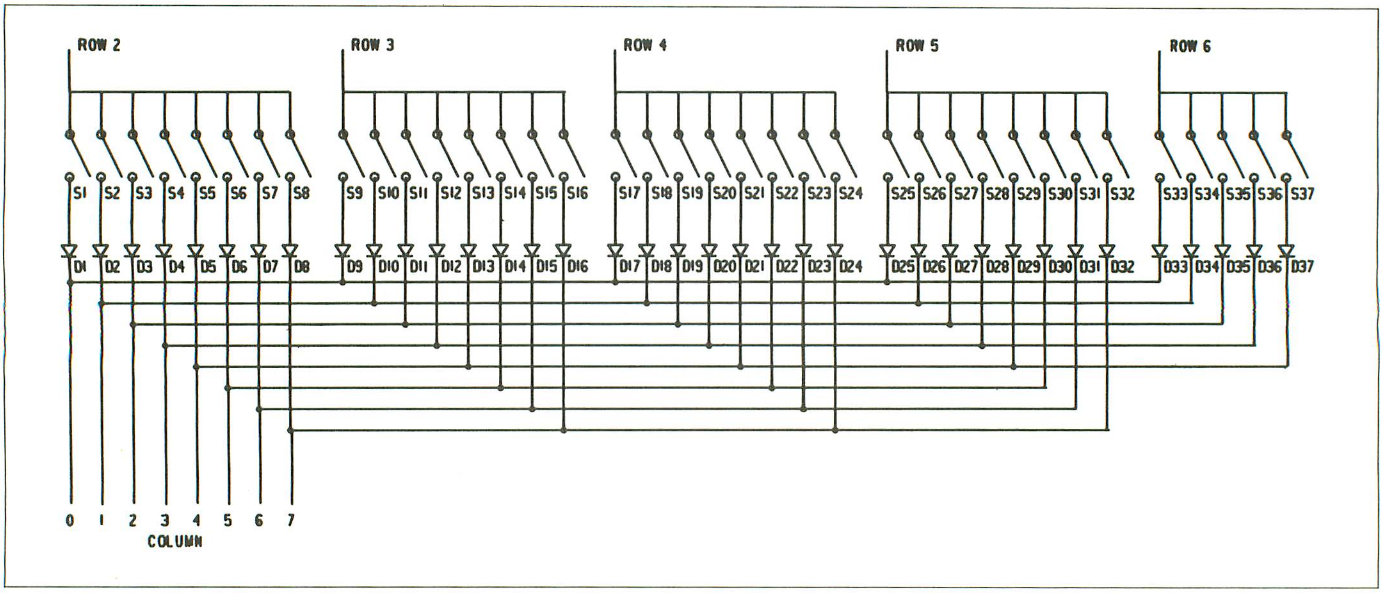


Figure 5 — Keyboard diode matrix.

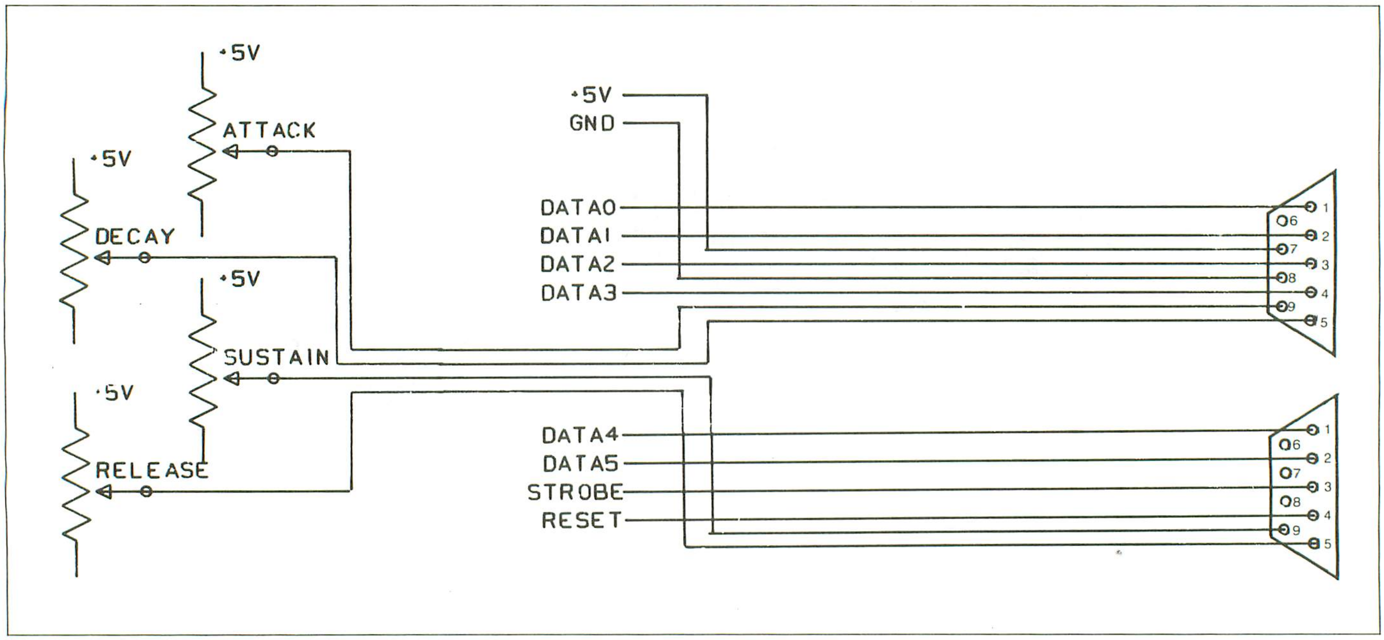


Figure 6 — Computer connection and pots.

The keyboard.

So, now that we have a keyboard encoder, what do you think we'll need next? If you guessed, "a keyboard," go to the front of the class. Figure 5 is a schematic of the keyboard. It's just a bunch of switches in series with some diodes that prevent stray current paths. These diodes are very important; you'll get pretty strange sounding chords without them.

The switches are divided into 4 groups (rows) of 8 switches and 1 group of 5 switches. Notice that the first physical group of 8 is electrically ROW 2. I have a reason for doing this that won't be covered in this article. It's important that you also do this, or the music programs won't work.

One more thing—the switches can be anything, from a series of momentary pushbuttons to a "real" keyboard. I used a Pratt and Read 3-octave unit that I purchased from PAIA Electronics (more information on this company later). This keyboard required only the addition of the diodes and has a very good feel. I'd recommend it in spite of its hefty price tag of about $100.

Connecting it to the computer.

Now that we have a keyboard connected to the encoder, we're ready to plug it into the Atari parallel I/O port. At this point, some of you are wondering if I'm playing with a full deck. . ."What parallel I/O port?" I refer, of course, to the joystick ports. Figure 6 (opposite page) shows the connection you need to make.

The +5v and GND connections supply power to the encoder. The current draw is low enough that a separate power supply is not needed. Data is transferred through the joystick connections. Four pots (labeled ATTACK, DECAY, SUSTAIN and RELEASE) are also shown on this diagram. They're attached to the paddle connections. The software, presented in the next installment of this article, will use these inputs to give you some control over the dynamics of the sound.

That about wraps up the hardware portion of this article. For those of you who want to etch a board, the layouts I used are reproduced in Figures 7A, 7B and 7C (shown, actual size, on opposite page).

If you choose to use this layout, you must be very careful when you align the two sides. If you aren't, you could end up missing a pad and not making a required connection.

Worse yet, you could end up with a connection to the wrong pad. Also, do not use sockets if you etch these boards yourself, and don't plate through the holes, because you'll need to solder both the top and the bottom of the board.

Testing the hardware.

When you get the encoder done, run Listing 1. It won't make any music, but it will tell you if your encoder is working.

When you run this program, it should print a count from 0 to 63, then *KEYBOARD ENCODER CHECKS OUT O.K.* If it prints *KEYBOARD ENCODER FAILED*, something is probably wired wrong. Check out the circuit board carefully, then try it again.

Getting parts.

Table 2 is a list of the parts necessary to build the encoder. Parts with an asterisk (*) by them are optional. Most of the parts can be obtained from any good electronics supply store.

Quan.	Description	Remarks
*1	8-Pin IC Socket	Sockets are not needed for the
*3	14-Pin IC Socket	circuit, but they do ease assembly.
*3	16-Pin IC Socket	
1	1K Resistor	¼ watt
1	1M Resistor	¼ watt
1	2.2K Resistor	¼ watt
1	22 pf Capacitor	
1	NE555 Timer IC	
1	4011 CMOS IC	
1	4024 CMOS IC	
2	4051 CMOS IC	
1	Keyboard	Or a series of switches connected as shown in Figure 5.
37	1N914 Diodes	
2	9-Pin Female "D" Connectors	
4	100K Pots	

Table 2.

If you wish to use the Pratt-Read keyboard that I used, you can order it from: PAIA Electronics, Inc., 1020 West Wilshire, Oklahoma City, OK 73116 — (405) 843-9626. The order number is AGO37; the cost is $98.50. You might also want to get their catalog. It's crammed full of music equipment.

To make it easier for those of you who haven't done much PC board building, bare PC boards are available from: ComputerWorks, 910 S. Rangeline Road, Carmel, IN 46032. The encoder alone costs $22.00; the diode only, $20.00. Both boards ordered together are $40.00. There is a shipping charge of $2.50. Allow four to six weeks for delivery.

Note that the diode board was designed for the Pratt-Read keyboard above. It would require modification to work with any other keyboard. The boards *do not* come with any of the components. But, as they're fairly common devices, they should be easy to obtain locally.

In the next installment of this article, we'll cover the software needed to make a very simple music synthesizer. Until then, have fun and good luck! □

The keyboard encoding techniques covered here aren't new. They're fairly standard circuits. Many people have written about encoders similar those covered here. So many, in fact, that I couldn't list them all, though I do wish to acknowledge two: Don Lancaster, in his book TV Typewriter Cookbook, *published by Howard W. Sams and Company, Inc.; and John S. Simonton, Jr., in his book* Friendly Stories about Computers/Synthesizers, *published by PAIA Electronics. Both of these books are excellent references.*

(Listing starts on next page)

Mario Perdue has been a Field Engineer in the computer world for about ten years. He got an Atari 800 three years ago, using it mostly for games. The 800 has been replaced with an 800XL on which his work centers around music, although he doesn't play any instruments.

Listing 1.

```
100 PORTA=54016:PACTL=54018
110 POKE PACTL,56
120 POKE PORTA,128
130 POKE PACTL,60
140 POKE PORTA,0
150 FOR X=0 TO 63
160 EFLAG=1
170 FOR Y=1 TO 1000
180 A=PEEK(PORTA)
190 IF A=X THEN EFLAG=0:Y=1000
200 NEXT Y
210 PRINT A;" ";
220 IF EFLAG=1 THEN 260
230 NEXT X
240 ? :? :? "KEYBOARD ENCODER O.K."
250 END
260 ? :? :? "KEYBOARD ENCODER FAILURE"
270 END
```

●

CHECKSUM DATA.

(see page 15)

```
100 DATA 609,350,568,340,521,157,491,6
02,316,866,762,244,873,769,448,7916
250 DATA 43,26,49,118
```

●

continued from page 37

ly, some functions cannot be used with certain devices (such as a READ from a printer); in such cases, the device handler would return an error 146, indicating that the requested function is not implemented in the handler.

In order to make CIO execute a desired I/O function, you must set up an IOCB with a numeric value indicating the function you want executed. This is the "command" byte. Each command must have various parameters set, so that CIO knows where the data is to be transferred to or from. Now, let's examine each of the eight basic commands and their use.

Opening files.

Before you use any I/O device, an IOCB pointing to that device must be OPENed by CIO. This is done with (what else?) the OPEN command.

In order to issue the OPEN command, the following fields must be set up in the desired IOCB: ICCMD—Set to $03; ICBAL and ICBAH—2-byte pointer to device/filename specification; ICAX1—Device direction and device-dependent information; and ICAX2—Device-dependent information.

Let's see how this command relates to its BASIC counterpart, the OPEN statement. In BASIC, the command to open the keyboard looks like this:

```
OPEN#1,4,0,"K:"
```

This command uses IOCB number 1 to open the keyboard. The direction value, 4, indicates that the file has been opened for input only.

In assembly language, the same open command would look like this:

```
0100   LDX #$10         ;IOCB  #1
0110   LDA #$03         ;OPEN COMMAND VALUE
0120   STA ICCMD,X      ;PUT IN IOCB #1
0130   LDA #KEYBD/256   ;HI ADDR OF "K:"
0140   STA ICBAH,X      ;PUT IN IOCB #1
0150   LDA #KEYBD&255   ;LO ADDR OF "K:"
0160   STA ICBAL,X      ;PUT IN IOCB #1
0170   LDA #$04         ;INPUT ONLY
0180   STA ICAX1,X      ;PUT IN IOCB #1
0190   LDA #$00         ;ZERO AUX BYTE 2
0200   STA ICAX2,X      ;PUT IN IOCB #1
0210   JSR CIOV         ;OPEN IT!
0220 KEYBD .BYTE "K:",$9B
```

Line 100 of this code sets the X register to $10. This is an offset which is used to point to IOCB number 1. This is *extremely* important: *ALWAYS* set the X register to the IOCB number you're using times 16 before calling CIO! The X register is used by the CIO subroutine so that it knows which IOCB it's supposed to work with. This is a simple process—if you want to use IOCB #0, set the X register to $00. If using IOCB #1, set the X register to $10. If using IOCB #7, set the X register to $70.

Now that we've set the X register to point to IOCB number 1, we need to set the rest of the CIO parameters used for the OPEN command. Lines 110-120 place the value $03 in the ICCMD location of IOCB number 1. This sets the command byte in IOCB number 1 (ICCMD,X) to $03, the numeric value of the OPEN function.

Using the X register as an offset on the ICCMD field stores the command in ICCMD + 16 (remember, the X register contains $10, or 16 decimal). Since ICCMD is defined at its location in IOCB number 0 ($0342), adding $10 to its address points to the ICCMD byte in IOCB number 1, at location $0352 (see Figure 1). By using the X register in this way, we can store commands in any IOCB by using the same X register setting that's used to tell CIO which IOCB we're using! It's a convenient feature of the CIO system.

Lines 130-160 in the code set the 2-byte pointer, ICBAL and ICBAH, to point to the address of our keyboard device string, which is stored at the label KEYBD. This is a simple text string, terminated with the byte $9B, which is the ATASCII end-of-line (EOL) character. CIO will use this string to find out which device it is to open.

Lines 170-180 set the ICAX1 byte in IOCB number 1 to $04, indicating that the keyboard is to be opened for input operations. This is the same number four that was used in the BASIC OPEN statement above.

Lines 190-200 set the ICAX2 byte in IOCB number 1 to $00. Once again, this corresponds to the 0 in the BASIC OPEN statement.

Line 210 finishes the OPEN process by calling the main CIO entry point, CIOV, located at $E456. It is imperative that you be sure the X register contains the value of the IOCB number times 16 before executing the JSR CIOV statement.

When you perform a JSR to the CIOV routine, the system sets up the requested IOCB to be used for I/O to the specified device. After the CIO operation is finished, control resumes at the statement after the JSR.

At this point, both the Y register and the ICSTA byte will contain information on the status of the CIO call. Remember, a status value of $01 indicates that the I/O operation was successful; a value greater than or equal to $80 indicates an error has occurred. You can use the 6502 CPY instruction right after the CIO call to test for specific error codes.

Now that the file is opened, you may perform other I/O operations on it, as listed below.

Close file.

The CLOSE command (numeric code $0C) is used when you're through using the I/O device. This is an especially important command when you're writing a file to disk or cassette, since it insures that all the data sent to the file is actually written to the device. Remember: *always* CLOSE your files.

The only IOCB parameter you need to set up for the CLOSE command is the command byte, ICCMD. Simply set it to $0C, load the X register with the IOCB number times 16, and JSR CIOV. The file will be closed, any data in the output buffer (if the file is opened for output) will be written, and the IOCB will be released for other uses. The following code shows how to close the file we opened in the previous OPEN example:

```
LDX #$10          ;POINT TO IOCB #1
LDA #$0C          ;CLOSE COMMAND
STA ICCMD,X       ;PUT IN IOCB #1
JSR CIOV          ;CLOSE IT!
```

Closing files is an easy operation, but its importance cannot be overestimated. Always double-check your code to be sure your files are closed properly. In addition, remember to check the status of every file CLOSE operation. Errors are possible on CLOSE commands, so be sure you test for them by checking either the Y register or the ICSTA location.

Getting characters.

Once your file is opened for input, you can read data from it in two ways. You can get characters regardless of what they are, or you can read lines of text which are terminated with the ATASCII end-of-line (EOL) character.

The first method we'll examine is GETting characters. In this mode, CIO will read the specified number of characters into the data buffer you define, no matter what the characters are. The IOCB fields used with the GET CHARACTERS command are: ICCMD—Set to $07; ICBAL and ICBAH—Pointer to input data buffer; and ICBLL and ICBLH—2-byte value indicating the data buffer length.

On a GET CHARACTERS command, the ICCMD location is set to $07, the proper numeric code for the GET operation.

The 2-byte pointer made up of ICBAL and ICBAH should be set to point to the beginning address of your data buffer. The data buffer is where CIO will place the characters read from the device. It should be set up so that it's long enough to hold all the characters you tell CIO to read. Set ICBAL to the low-order byte of the buffer's address and ICBAH to the high-order byte.

The 2-byte value made up of ICBLL and ICBLH tells CIO how many bytes you want to GET from the device. The low-order byte of the count should be placed in ICBLL, and the high-order byte should be placed in ICBLH. Be sure the byte count is not larger than the size of the buffer pointed to by ICBAL and ICBAH. If you GET more characters than your buffer can hold, CIO will clobber whatever data or program instructions follow the buffer.

The following example will GET 548 characters from the file indicated by IOCB number 6, placing them in a buffer called MYBUF. MYBUF is set up to be 1000 characters long, which is a safe size for our read of 548 characters. (IOCB number 6 is assumed to be open already.)

```
0100      LDX #$60         ;IOCB  #6
0110      LDA #$07         ;GET CHARACTERS
0120      STA ICCMD,X      ;PUT IN IOCB #6
0130      LDA #MYBUF/256   ;HI ADDR OF BUFFER
0140      STA ICBAH,X      ;PUT IN IOCB #6
0150      LDA #KEYBD&255   ;LO ADDR OF BUFFER
0160      STA ICBAL,X      ;PUT IN IOCB #6
0170      LDA #548/256     ;HI BYTE COUNT
0180      STA ICBLH,X      ;PUT IN IOCB #6
0190      LDA #548&255     ;LO BYTE COUNT
0200      STA ICBLL,X      ;PUT IN IOCB #6
0210      JSR CIOV         ;GET THE BYTES!
0220 MYBUF *= *+1000       ;DEFINE BUFFER
```

Line 100 of this code points to IOCB number 6, as explained earlier.

Lines 110-120 set the ICCMD location of IOCB number 6 to the numeric value of the GET CHARACTERS command, $07.

Lines 130-160 set ICBAL and ICBAH to point to the input buffer, MYBUF. When we call CIO, it will GET the characters from the device and place them in memory starting at the first byte of MYBUF.

Lines 170-200 set up ICBLL and ICBLH in order to tell CIO to get 548 characters from the device indicated by IOCB number 6.

Line 210 calls the CIO subroutine, which will attempt to read 548 bytes into MYBUF.

After CIO GETs the characters from the device, the Y register and ICSTA location will contain the status of the GET operation. It is important to check the status of a GET, because the end of the file may have been reached.

Also, after the GET is complete, the ICBLL and ICBLH will have been changed by CIO to tell you how many bytes were actually read into the buffer. If all 548 bytes were read, ICBLL and ICBLH will contain the value 548. If, on the other hand, the end of the data file was reached or another error occurred, ICBLL and ICBLH will indicate how many bytes were actually read into the buffer. In this way, you can properly handle end-of-file or other error conditions.

A special case of the GET CHARACTER command exists if you only want to get one character from the input device. This special option is indicated if you set the buffer length value to 0. If this is done, CIO will get one character from the input device as it becomes available, then put it in the accumulator.

Obviously, this variation on the GET CHARACTERS command does not require that the buffer address (ICBAL and ICBAH) be set before calling CIO. All you need to set is the command byte ($07) and the buffer length (0).

Putting characters.

The output equivalent of the GET CHARACTERS command is the PUT CHARACTERS command. As you might suspect, this function will write the specified number of bytes to the output device. It works just like the GET CHARACTERS command, but in reverse. You need to set the following IOCB variables to use this command: ICCMD—Set to $0B; ICBAL and ICBAH—Point to start of data buffer; and ICBLL and ICBLH—Number of bytes to write to device.

Like GET CHARACTERS, PUT CHARACTERS has a special 1-byte PUT option. By setting ICBLL and ICBLH to 0, CIO will write the byte in the accumulator to the specified device. Once again, for this special case, ICBAL and ICBAH are not used, and you don't need to set them up.

More to come.

Next issue, we continue our look at CIO and will complete the summary of CIO commands. We'll also begin writing programs which use CIO to accomplish input and output operations. Until then, it's a good idea to try and pick up the two reference manuals mentioned earlier in this article and read the sections on CIO. □

THE END USER

THIS MONTH:

Controlling a robot with your Atari computer

by Arthur Leyenberger

Measured in issues of **ANALOG Computing** (*and* **End User** columns), another month has come and gone. I do hope you've enjoyed this column the last few months—I've enjoyed being here. Those of you familiar with the movie *Tron* will remember that the Master Control Program, the MCP, would have you believe that **End Users** are a superstition or a myth. Well, as the MCP found out, we exist!

This month, we're going to take a little detour and have some fun. Like many of you, I'm interested in robots. I read about them, play with them and daydream that, perhaps someday, Asimov's *I, Robot* stories will come true. And, when I can combine robots with computers. . .well, I've been known to spend all night in my "lab."

Although we don't have an interview with Asimov's Dr. Susan Calvin, we can experiment with the robots available today. And that's exactly where we begin this month's column. Everywhere you look there are robots: industrial robots, toy robots, science fiction robots and, of course, learning robots.

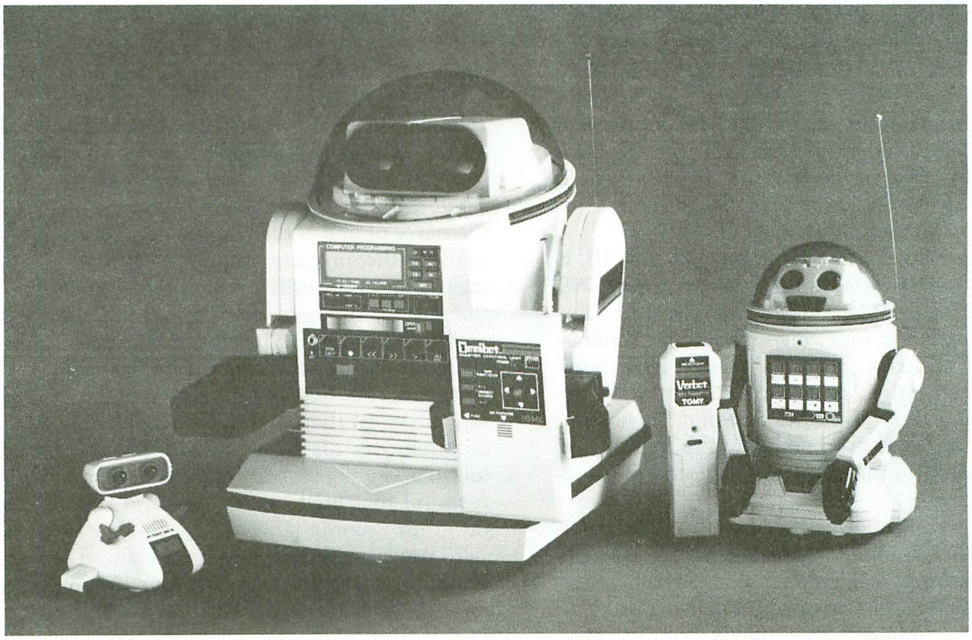
Practically speaking, most robots can't really do too much. Sure, those big industrial jobs can spray paint car doors, weld car bodies, and move raw materials. But they cost anywhere from a few thousand to hundreds of thousands of dollars. Slightly out of the reach of the typical Atari user's budget.

What about the so-called personal robots? Heathkit makes two build-'em-yourself robots. **Hero-1** was their first entry into the personal robot market, and it costs approximately $2000. It can move around on a wheeled base, can be programmed to perform certain functions and has one arm that can pick things up. **Hero Junior** is a smaller, less expensive Heathkit robot that lacks the arm and costs about $1500.

Androbot, the company founded and later sold by Nolan Bushnell, has been making a robot named **Topo** for about two years. **Topo** is slightly taller than **Hero-1**, moves around in a similar way, but lacks the capability to be fully preprogrammed. **Topo** sells for $2000 to $3000, depending upon the configuration.

In addition to industrial and personal robots, another category is educational robots. There are several companies now making robot arms which simulate the actions and programmability of industrial robots. These educational robots sell for prices from a few hundred dollars up to over $5000.

One characteristic that educational robots usually have is the ability to be interfaced to a computer and operated by a program. Typically, software used to operate these robots is available only for Apple, IBM (and clones) and Radio Shack computers. The Atari owner is left



somewhat robotless.

My favorite category is toy robots. My personal collection consists of "positionable action figures," such as **Gobots**, **Godaikens** and **Takaras**.

The **Gobot** and **Godaiken** toy robot figures typically look like something else in one state (a car, plane, tank, etc.), and by pulling a few parts out, pushing in some other parts and maybe twisting a thing or two, you turn them into a robot figure.

Takara robots from Japan are high quality, diecast metal and plastic robot figures with articulating arms and legs, and various attachable accessories.

Battery operated and wind-up robots fill out my collection. Tomy, the toy company that brought you everything from **Zoids** to **Cabbage Patch Kids** accessories, has several interesting toy robots. Their little wind-up toys called **Pocket-Bots** are cute, but don't have much educational value. However, there are two Tomy robot toys that should be of interest to the Atari user.

Omnibot.

Omnibot is a battery operated, remote control, programmable robot. It stands about a foot tall and is operated by a remote-control joystick. **Omni** has a cassette recorder in its abdomen, which can either play music or record its movements as you operate it with the joystick. Then, either directly or by means of its on-board clock, it can repeat the given program.

Unfortunately, **Omnibot** has no means for sensing its environment. Therefore, it cannot react to outside events and can only repeat a specific set of instructions that have been recorded on the cassette tape. For educational purposes, **Omnibot** can demonstrate remote control and program execution, and can wait for conditioned actions (via the clock).

Verbot.

A more interesting, less expensive toy robot that also has more educational potential is Tomy's **Verbot**. It's a battery-operated, voice-actuated robot that can move forward and backward, turn left and right, and raise and lower its arms. It can also turn on some lights in its head to simulate smiling.

A microphone is used to give **Verbot** its commands, but first these commands must be programmed. This is accomplished by holding down the various function keys on **Verbot**'s stomach as the command is spoken into the mike.

For example, to program **Verbot** to move forward, you would hold down the button labeled ∧ and say, "forward, forward, forward" into the microphone. A blinking LED indicator stops blinking when the command has been accepted.

You don't have to say "forward" as the command. You could just as well say "go forward," "move front" or even "felgercarb." The length of the spoken word must be from .75 to 1.3 seconds to work. Also, each command (for the eight functions mentioned above) must be unique.

When you've finished programming **Verbot**'s functions, it will respond to those commands spoken into the microphone by the same person, with the same inflection.

One of the problems with **Verbot** is that it often gets confused. I've been told by people at Tomy that, if you speak Japanese or Spanish, **Verbot** will work very reliably; it's not totally fluent in English.

Another problem stems from having to say the commands *exactly* as you said them when programming. In this case, the human is not very reliable. By the way, when you turn off **Verbot**, it loses the commands that were programmed, so you must go through the procedure each time it's turned on.

Verbot, like the larger **Omnibot**, really isn't that useful as an educational device, because it has limited programming capability and cannot react to the outside environment.

Because of the reliability and reprogramming problems with **Verbot**, I wondered if I could use my Atari to control it. And, if the Atari could control **Verbot**, why not add the capability for more precise control?

The program.

As they say in the movies, there *is* a happy ending. I was able to use my computer to generate eight separate, unique sounds that could be used to program **Verbot**.

Listing 1 is the "quick and dirty" program I wrote to create the sounds. It's written in Atari Microsoft BASIC (or AMSB), because I needed to control the duration of the sound. I could have used Atari BASIC or BASIC/XL, but I would have had to use FOR/NEXT loops, and their execution time can vary with their location in the program. Also, it would have been harder to measure the time for each command.

The sound command in AMSB is used in the following way: *SOUND Voice#, Pitch, Distortion, Duration*. The Atari has four voices or sound channels, so Voice# can range from 1 to 4. Pitch can range from 0 to 255, with 255 being the lowest pitch sound. Distortion can be an even number from 0 to 14, and duration is specified in 1/60ths of a second.

You'll notice that the program is neither complete nor elegant. It was only written to create sounds. I wrote using a trial and error approach, in order to get eight different sounds that were each distinct and would work with **Verbot**'s functions.

Program operation.

There are nine commands available in the program: L for left, R for right, F for forward, B for backward, U for arms up, D for arms down, S for smile, H for halt, and Q to quit the program (exit).

When the program is running, it continually scans the keyboard for input. At the entry of one of the single-letter commands, the program branches to that particular subroutine, which generates the sound for the correct duration.

After the sound is produced, the program returns to scanning the keyboard. In addition to the sound being generated, the word that's associated with the command is displayed on the screen, too. If a key is pressed that doesn't correspond to a legitimate command, the program ignores it.

To program **Verbot**, turn on the microphone first and place it near the TV or monitor speaker. Then turn on **Verbot**. LOAD and RUN the program. While holding down a function button on **Verbot**, press the corresponding command letter on the keyboard and watch **Verbot**'s red LED.

When the LED stops flashing, the function has been programmed. You may need to repeat this step a couple of times, since **Verbot** is occasionally a little hard of hearing.

When all eight of the **Verbot** functions have been programmed, you're ready to have some fun. Set the little guy down on the floor and make sure the microphone is close to the TV or monitor speaker.

Now, as you press *F* on the keyboard, your robot pal will move forward. Press *L*, and **Verbot** should turn left. If things don't seem to happen as you expect (like **Verbot** turning left if you say "smile"), go back and reprogram that particular command. Also, be sure that the batteries are good in both the microphone and **Verbot**.

So now you have a remote-controlled robot. So what? Well, I admitted earlier that my program was simple and not too useful. Here are some additional ideas that you may want to try.

Expand the program so that it will ask you not only for a command, but also for a distance. Then, when the program sends the tones to control **Verbot**, it sends a halt command after the distance has been traveled.

Of course, you'll have to figure out how fast the robot travels and how long it takes for a command to be recognized. For example, if it takes two seconds for **Verbot** to respond to one of the tone commands, and he travels one foot every five seconds, you'll have to send the halt command seven seconds after the forward command.

Another enhancement which you may want to undertake is to create a "robot editor." Here, you'd be able to enter a sequence of commands and have **Verbot** execute the entire program when you're ready.

If you tried the previous suggestion using a time interval to control **Verbot**'s movements, you know that that procedure only worked interactively. What I'm suggesting here is that you use your "robot editor" to give distance commands, as well as specific commands.

Time's up!

It looks as if I ran out of time before I could tell you about another robot, one that attaches directly to your Atari. Next month we'll finish up with robots and maybe have a few surprises.

If you're seriously interested in learning more about robotics, I recommend a very readable book called *Fundamentals of Robotics: Textbook 1*. It is published by Eshed Robotec and is part of a series of four books in a robotics training program. It may be ordered from Prep, Inc., 1007 Whitehead Road Ext., Trenton, NJ 08638 — (201) 882-2668. The cost is $19.95 plus 3% for shipping.

Once again, it's time for this **End User** to say good-bye to all you **End Users**. □

Arthur Leyenberger is a human factors psychologist and free-lance writer living in New Jersey. He has been an Atari enthusiast for four years. When not computing, he enjoys playing with robotic toys.

Companies mentioned in this column:

Androbot, Inc.
101 East Daggett Dr.
San Jose, CA 95134
(408) 262-8676

Heath Company
Benton Harbor
MI 49022
(616) 982-3200

Tomy Corporation
901 E. 233rd St.
P.O. Box 6252
Carson, CA 90749
(213) 549-2721

(BASIC listing starts opposite page)

Microsoft BASIC
listing.

```
10 REM PROGRAM TO CONTROL TOMY VERBOT
15 REM WRITTEN BY ARTHUR LEYENBERGER
20 REM ATARI MISCROSOFT BASIC - 2/23/8
5
40 PRINT "VERBOT CONTROLLER";" "
50 SOUND 3,0,10,0,120
60 GOSUB 9000
100 A$=INKEY$
110 IF A$="F" THEN GOSUB 1000
120 IF A$="B" THEN GOSUB 2000
130 IF A$="L" THEN GOSUB 3000
140 IF A$="R" THEN GOSUB 4000
150 IF A$="U" THEN GOSUB 5000
160 IF A$="D" THEN GOSUB 6000
170 IF A$="S" THEN GOSUB 7000
180 IF A$="H" THEN GOSUB 8000
190 IF A$="Q" THEN 9999
300 GOTO 100
1000 REM FORWARD
1010 PRINT "FORWARD"
1050 SOUND 3,255,10,8,90
1100 SOUND 3,0,10,0,15
1800 GOTO 9000
2000 REM BACKWARD
2010 PRINT "BACKWARD"
2050 SOUND 3,225,10,8,90
2100 SOUND 3,0,10,0,15
2800 GOTO 9000
3000 REM LEFT TURN
3010 PRINT "LEFT"
3050 SOUND 3,200,10,8,90
3100 SOUND 3,0,10,0,15
3800 GOTO 9000
4000 REM RIGHT TURN
4010 PRINT "RIGHT"
4050 SOUND 3,175,10,8,90
4100 SOUND 3,0,10,0,60
4800 GOTO 9000
5000 REM ARMS UP
5010 PRINT "ARMS UP"
5050 SOUND 3,155,10,8,30
5100 SOUND 3,102,10,8,60
5200 SOUND 3,0,10,0,15
5800 GOTO 9000
6000 REM ARMS DOWN
6010 PRINT "ARMS DOWN"
6050 SOUND 3,125,10,8,60
6100 SOUND 3,193,10,8,30
6200 SOUND 3,0,10,0,15
6800 GOTO 9000
7000 REM SMILE
7010 PRINT "SMILE"
7050 SOUND 3,100,10,8,90
7100 SOUND 3,0,10,0,180
7800 GOTO 9000
8000 REM QUIT
8010 PRINT "HALT"
8100 SOUND 3,60,10,10,90
8110 SOUND 1,47,10,10,90
8150 SOUND 3,0,10,0,15
8160 SOUND 1,0,10,0,15
9000 PRINT  "COMMAND?";
9010 RETURN
9999 END
```

•

Rescue on Fractalus and Ballblazer A REVIEW

by Arthur Leyenberger

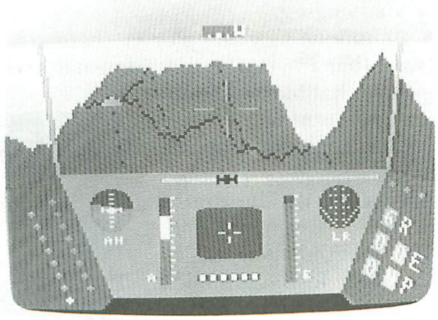
It's been a long time coming. Atari announced the deal with Lucasfilm Ltd. in May, 1984. There were to be two games coming: **Rescue on Fractalus** and **Ballblazer**. Both sounded exciting, and we looked forward to their arrival.

Then Jack Tramiel bought Atari from Warner Communications. . .we waited, and waited, and waited. Atari was unusually quiet during this time. Suddenly, at the January 1985 Consumer Electronics Show, Epyx debuted **Rescue on Fractalus** and **Ballblazer**. The two are now a reality, and I'm pleased to be able to review them.

RESCUE ON FRACTALUS
EPYX, Inc.
1043 Kiel Court
Sunnyvale, CA 94089
(408) 745-0700
48K under $30.00

At a time when video games are thought to be dead (or, at least, not healthy), **Rescue on Fractalus** is refreshingly alive, both in concept and execution.

You are the pilot of a Valkyrie fighter, the most maneuverable high-tech ship in the fleet. Your mission is tough: a low-altitude rescue of downed pilots on the uninhabitable planet Fractalus. A mother ship ferries you to within booster range of the planet, from which point you descend to the surface.



Rescue on Fractalus.

The planet itself is held by Jaggies, formidable enemies with gun emplacements and flying saucers. You must use your instruments to locate the downed space pilots, land within two units of their ships, and then open your airlock to allow each pilot to enter. Real Valkyrie pilots signal you by knocking on the hull of your ship.

Once the pilot enters your ship, his energy packs are added to your craft's energy level. When you've picked up your quota of pilots, or run out of energy, you boost up through the atmosphere back to the mother ship, to return the pilots and get your next assignment.

As you advance through the 99 levels, you must rescue increasingly more pilots. Also, enemy forces put up more resistance. Gun emplacements can withstand more of your firepower, and enemy saucers are more frequent.

Sometimes, without warning, a Jaggi warrior in disguise will attempt to sabotage your mission. If you open the airlock too quickly, he'll enter and destroy your ship. Also, if he pounds on your cockpit window, your space-faring days are over.

Of course, you can always turn on your engines and fry the Jaggi warrior. To decrease the chances of having to confront one of these nasty creatures, be sure to blast your rescued pilot's Valkyrie fighter after he's safely in your ship.

Rescue on Fractalus uses an impressive graphics technique called "fractal geometry" to portray the mountain landscape. Aside from the gun emplacements and the saucers, all of the graphics are created in real time.

The realistic 3-dimensional effect is maintained throughout the game. You can fly between mountain crevices, turn around and then fly through the same terrain, just as you would expect.

There are other realistic details that make this flight simulation game better than others that have come before. For example, the farther you land from a pilot, the longer it will take for him to reach the ship.

Also, as the pilot approaches you, he gets larger and larger. The modeled environment behaves just as the real world would. If you fly past a pilot, you can make a U-turn and pick him up—he'll be where you expect him to be.

Rescue on Fractalas is a challenging and enjoyable game that I've played for hours on end. Somehow, I never seem to tire of rescuing pilots, or lose the thrill of piloting my Valkyrie fighter. I may not retire my **Star Raiders** cartridge, but **Rescue on Fractalus** will get as much airtime. It's as close as you can get to participating in the *Star Wars* adventure, at least for the present.

BALLBLAZER
EPYX, Inc.
1043 Kiel Court
Sunnyvale, CA 94089
(408) 745-0700
48K under $30.00

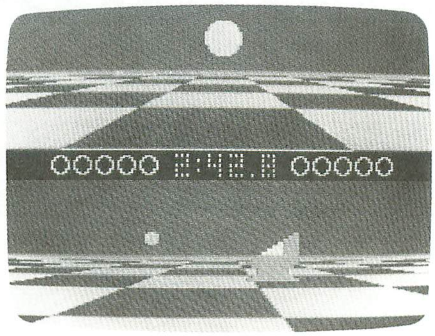
Ballblazer is the second Lucasfilm game released by Epyx. It's a mechanized, futuristic, soccer-like game that's played on a checkered grid.

Each of the two players operates a craft called a "rotofoil," trying to capture an elusive "plasmorb" (ball) and score a goal. It's not as easy, since the goalbeams move back and forth across the end zones and sides of the playing field. Also, as the game progresses, the goalbeams move closer together.

Ballblazer takes place on a scrolling grid with two 3-D perspectives on the screen. The top half of the screen is your view of the playing field, while the bottom half is that of your opponent. The object is to capture the plasmorb with your joystick-controlled rotofoil, then shoot it into the moving goals at the opposite end of the grid.

When you capture the plasmorb—or steal it from your opponent—its color changes to your color, and you will rotate to face the goal. Firing the plasmorb causes your rotofoil to recoil. It takes a while to get used to this effect, but after some play time, you can use it to your advantage in positioning your rotofoil.

Goals may be scored by blasting or pushing the plasmorb through the end posts. Scoring goals earns you 1, 2 or 3 points, depending on how far away from the goalbeams you are when you shoot. To get 3 points, though, you must shoot when the goals aren't visible—a tough shot, not to be tried by novices. A shutout is 10 points; otherwise, the highest score wins at game's end.



Ballblazer.

Game play is either hot and heavy, head-to-head action with another player, or competition against a computerized droid. Ten levels of droid players are available, ranging from an easy-to-beat, inept droid to a super droid with lightning reflexes and the killer instinct.

After spending hours in the cockpit of a rotofoil, I've only managed to defeat sixth-level droids. This game is definitely challenging.

As unique as game play is, there are two other interesting and unusual aspects of **Ballblazer**: graphics and sound.

Most computer graphics screens show a diagonal line as a jagged line, rather than a continuous one. This is typically unavoidable, due to the graphics resolution (number of pixels that can be displayed on-screen.)

A straight line has an infinite number of points, but the computer can only draw a small sample of these on the screen. **Ballblazer** overcomes the inadequacy of displaying diagonal lines by using a mathematical model that, independent of the actual graphics, provides the illusion of straight diagonal lines.

The computerized sound of **Ballblazer** is also unique. The original, continuous score adds to the excitement of the game and gives it a futuristic feeling. In addition to the rhythmic, percussive sound in the central theme, improvisational lead lines are played for a while, then fade out. This fading in and out of the assorted licks gives the composition a fresh flavor that sounds as well as the game plays.

The manual accompanying the game disk is unusual, too. It's more of a storyline than a traditional "insert the disk and press start" set of instructions.

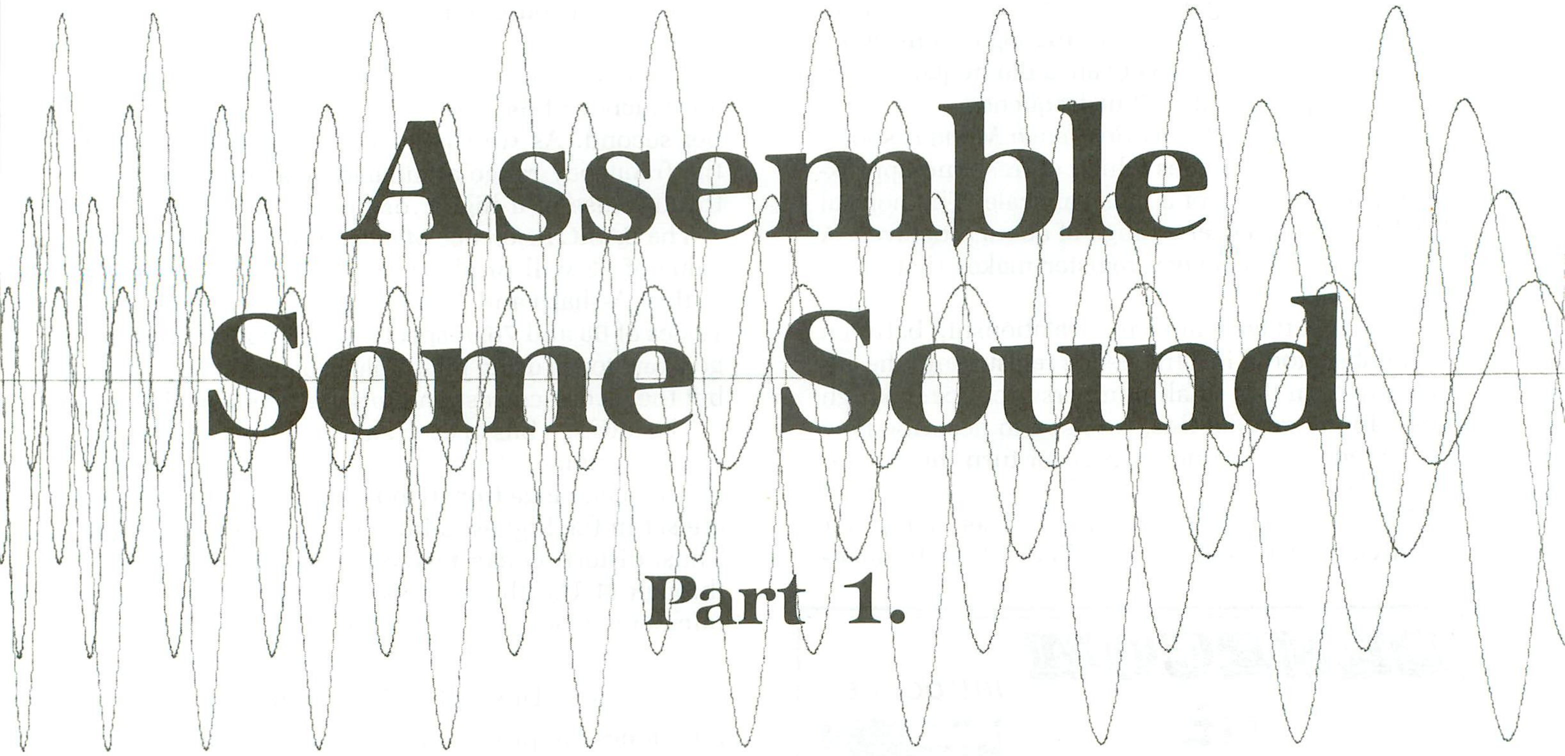
The year is 3097, and the place is a null-gravity nexus mid-space in the binary star system of Kalaxon and Kalamar. On the luminous surface of an artificial asteroid, a creature from Earth is challenging the other-world Masterblazers in this year's interstellar **Ballblazer** championship.

The manual goes on to explain the game grid, goalbeams and plasmorb, for those new to the cosmic soccer match. Renowned players are interviewed for their offensive and defensive strategies, and a galactic sportscaster does a play-by-play, complete with color from an expert player.

All in all, **Ballblazer** is an unusual, excellent game. Where else can a mere Earthling like you or me get the chance to compete with interstellar creatures for the honor of our planet—and the ultimate title of Masterblazer? □

Assemble Some Sound

Part 1.



by Karl E. Wiegers

Atari computers are respected both for their outstanding graphics capabilities and for the wide variety of sounds they're able to generate. Most discussions of sound production on the Atari deal with programs written in BASIC. It's not really obvious how to emulate the SOUND command of Atari BASIC when programming in assembly language.

This article describes the fundamentals of sound generation in assembly, using examples which you can adapt to incorporate audio effects into your own assembly language programs.

Sound theory.

The Atari computers contain a microprocessor chip called POKEY, which is responsible for generating sounds. POKEY also takes care of other input/output operations, including the keyboard, game paddles and serial devices like disk drives and the cassette recorder.

There are four separate sound channels or voices, all of which can be played simultaneously. Each voice has two memory registers in POKEY which control the sound produced. One register controls the frequency or pitch of the sound, and the other governs the volume and distortion. These parameters should be familiar from the SOUND command in Atari BASIC, which has the format:

SOUND voice,pitch,distortion,volume

The frequency registers for the four sound channels are called AUDF1-AUDF4 and are found at hexidecimal locations $D200, $D202, $D204 and $D206, respectively. Distortion/volume registers are called AUDC1-AUDC4 and are at locations $D201, $D203, $D205 and $D207. Other important POKEY registers are AUDCTL at $D208 and SKCTL at $D20F. We'll discuss their functions later.

Any pure (undistorted) tone has a characteristic number of vibrations per second (frequency), giving it a particular audible "pitch." For example, the A above middle C has a frequency of 440 cycles per second, or hertz (Hz).

However, persuading your Atari to play an A isn't just a matter of plugging the number *440* in somewhere. POKEY's sound-generating logic translates a number between 0 and 255 in an audio frequency register into a particular output frequency.

Page 58 of your *BASIC Reference Manual* shows the numeric pitch values which correspond approximately to the notes of a musical scale. The normal POKEY settings cover a range of about 3 octaves. A value of *0* in a frequency register makes that voice silent.

Notice that there's an inverse relationship between the number stored in a frequency register and the audible pitch, in that small numbers produce the high notes. To understand this, we need to describe how the contents of a frequency register turn into an audible tone.

POKEY takes the number in a frequency register and divides it into some specified "clock frequency." Roughly speaking, the result is the frequency we hear through the TV speaker. A large number in the frequency register produces a fairly small number after this division operation, yielding a low-pitched sound.

The normal clock used for sound generation has a frequency of 64 kilohertz (KHz), or 64,000 cycles per second. As we shall see later on, we can shift the frequency range attainable by a sound register, through use of a higher or lower clock frequency.

The *BASIC Reference Manual* indicates that a pitch value of 72 will produce an A. The notes adjacent to this, A-sharp and A-flat, are produced with pitch values of 68 and 76, respectively. Of course, you can generate tones using pitch values between 72 and 76, but they won't correspond to specific musical notes.

The lowest 4 bits of each audio control register govern the volume. Hence, the volume setting for each register can range from 0 (off) to 15. Distortion values are set in the highest 3 bits of each control register. Thus, distortion can take on even values of from 0 through 14. Use this equation to determine the proper number to store in a control register in order to produce a specific distortion/volume combination:

$$16 * \text{DISTORTION} + \text{VOLUME}$$

Pure tones are produced at distortion settings of 10 or 14. A typical setting for a control register would then be 168, representing a pure tone (distortion = 10) at a volume level of 8. If more than one voice is used at once, the total volume of all voices being played should not exceed 32, to avoid degradation of sound quality.

Assembly sound.

To generate sound in an assembly language program, you must perform several steps. First, initialize the POKEY chip by storing a value of *0* in AUDCTL and storing *3* in SKCTL. This is equivalent to the BASIC statement *SOUND 0,0,0,0*. Now, place an appropriate pitch value into the desired audio frequency register(s). Finally, set the corresponding audio control registers with the correct values to govern distortion and volume.

The sound will begin immediately. It will continue until explicitly turned off by placing a *0* in the frequency register, setting the volume bits to *0* in the control register or pressing SYSTEM RESET.

An important sound parameter which isn't included in the SOUND command in BASIC is duration, or how long the tone should play. The BASIC programmer solves this problem with an empty FOR/NEXT loop, using an appropriate number of loops to give the desired delay before changing or turning off the sound.

An easy way to do this in assembly language is to use the internal real-time clock which the Atari maintains at locations $12-$14 (RTCLOK, RTCLOK +1, RTCLOK+2). The value in location $14 is incremented during each vertical blank interval, 60 times per second. When it reaches 255, it is reset to *0* at the next vertical blank interval, and the value in location $13 is incremented. So, a simple way to get accurate timing in assembly is to set the RTCLOK locations to *0*, then loop until they match the contents of a timer variable which has been set to the number of sixtieths of a second ("jiffies") the sound is to be heard.

Sounding off.

Let's look at a simple example of sound in an assembly program. Listing 1 shows how to play a middle C for 1 second using an undistorted audio channel 1 (corresponding to voice 0 in BASIC).

This and most of the other examples are written in Atari Assembler Editor cartridge format. Type in the program you want to try, assemble it (adding the statement *.OPT NOLIST,OBJ* as Line 1 will save some time if you don't want to save the object code), enter the debugger and RUN the program with a G3000 instruction.

After the necessary initializations (Lines 190-220), a pitch value of 121 is stored in AUDF1, which the *BASIC Reference Manual* tells us should produce a middle C. The AUDC1 value of 168 gives us a pure tone with volume 8.

The duration is set using a 2-byte variable I called TIMER ($CB and $CC). One second equals 60 jiffies, and I need to split this duration into its high and low portions, and place them in the corresponding TIMER bytes. Obviously, the low byte is 60 and the high byte is 0, but Lines 340-370 show the general form used to handle any duration up to 65,535 jiffies (about 18.2 minutes).

The DELAY loop (Lines 470-590) zeroes the real-time clock and checks until the lowest byte (RTCLOK +2) equals the low byte of TIMER, and the middle byte (RTCLOK+1) equals the high byte of TIMER (TIMER+1). Then we return to the main program and turn off the sound by storing a *0* in AUDF1 (Lines 390-400).

Finally, terminate the program with a BRK instruction. This timing method is used in all examples in this article except the final one.

Now that you know how to generate a single note, you might like to hear the entire range of tones that can be produced using the normal sound channel settings. Listing 2 demonstrates these capabilities.

We mentioned earlier that each voice covers a range of about 3 octaves, but that you can play many tones which aren't ordinary musical notes. Listing 2 simply plugs values from 0 to 255 into AUDF1 and plays each tone for 3 jiffies.

You'll hear a sequence of tones in descending pitch, 20 tones per second, lasting a total of about 13 seconds. Notice that high frequencies appear to change in pitch more rapidly than do low frequencies. This is a consequence of the division operation that POKEY uses to convert values in a frequency register into actual tones. See the chapter on sound in *De Re Atari* for more details.

Altering the range.

We mentioned above that you can alter the frequency range covered by the sound channels. This function is controlled by the AUDCTL register. The normal setting of 0 (all bits off) yields the frequency range you heard from Listing 2, using the 64 KHz clock. However, by setting bit 1 of AUDCTL (AUDCTL = 1), a 15 KHz clock is selected.

This lower-frequency clock allows notes of much lower pitch to be played. This setting affects all four voices. Conversely, setting AUDCTL to a value of *64* (setting bit 6) makes channel 1 produce much higher frequencies; you can do the same to channel 3 by storing *32* in AUDCTL (setting bit 5). These settings select an extremely high clock rate of 1.79 MHz, the fastest clock in the Atari.

Even though a frequency register can still contain only values between 0 and 255, the different clock rates will create different actual output frequencies from the same numeric pitch value in the register. The three clock rates greatly widen the audio frequency ranges available in the Atari.

Listing 3 demonstrates the new frequency ranges. It's just like Example 2, except that each tone is only sounded for 2 jiffies, and you'll hear the normal, low and high frequency ranges in turn.

There will be a slight delay between the end of the second (low frequency) demonstration and the beginning of the third (high frequency). The computer is actually executing the third program loop, but the initial tones produced are of much higher frequency than the human ear can detect. Your dog might object to the third part of the program, since his frequency response is much higher than yours.

(continued on page 88)

CIRCLE #134 ON READER SERVICE CARD

Singing in the range.

Let's move on to something more interesting and see how to play a song in assembly language. Listing 4 does the trick, using channel 1 to play a short tune.

The basic idea is to place the pitch values you need for all the notes of your song into a data table, plug them into the frequency register one at a time, and delay briefly before continuing with the next note.

The absolute indexed addressing mode of the 6502 instruction set is a convenient way to extract values from a data table (Line 340). You'll need to refer to the *BASIC Reference Manual* again, to get the pitch values which correspond to the notes in your song. Of course, you can control the tempo of the song simply by altering the duration that each note is played: fewer jiffies = faster tempo.

There are a couple of other things to think about when playing music. First, how do you handle a song which, like most songs, contains notes played for different durations, such as quarter notes and eighth notes? Second, how do you get some separation between consecutive notes, so that the entire tune won't sound all slurred together?

Listing 4 addresses both of these questions. The data values for the notes in the tune are listed in the table at Line 760, labelled CHAN1. Notice that many notes are repeated; these will be heard as quarter notes, while notes played only once will be eighth notes. This duplication of data for some notes consumes additional memory, but it's simpler than using some algorithm to define the duration of each note.

The second problem, that of separating notes, is solved by turning off the audio channels very briefly at regular intervals, to break up the continuity of the sound.

This is clear from the pattern of durations used in the program loop which plays the notes (Lines 330-540). The first note is played for 8 jiffies (Line 360), then the next note for 5 jiffies (Line 440), then we have a 3-jiffy moment of silence (Line 490). An eighth note in this song thus lasts 8 jiffies, while a quarter note will be heard for 13 jiffies.

Notice that the sum of the second and third durations must equal the first (5 + 3 = 8) to get a regular tempo. Try changing these durations to slow the song down or speed it up. Eliminating the no-sound period will give a very *legato* or smooth sound, while lengthening it introduces a more *pizzicato* quality, as if an instrument were being plucked.

The obvious extension of this illustration is to use the other voices to play some chords. Turn now to Listing 5. This program extends that in Listing 4 by bringing into play channels 2 and 3.

New equates are included at the beginning of the program, and all three voices are set to the same volume and distortion. (Some very interesting effects can be achieved by simultaneously sounding voices with different degrees of distortion, but let's leave that for another time.)

Two new data tables are included at the end of the program, CHAN2 and CHAN3. The 0s in those tables mean that those voices will be silent at times.

Manipulating the four Atari voices this way opens up tremendous music-making potential. Think of what can be done by adding a bass line, harmony, counterpoint or percussion to a simple melody with the four instruments in your Atari orchestra.

Part 2 of **Assemble Some Sound** presents a macro which emulates the BASIC SOUND statement, for use with macro assemblers. Another example illustrates the use of a VBI routine to play repeating background music or continuous sound effects, such as racing engine sounds.

Combining two sound channels into a single voice to obtain very precise frequencies is the subject of the eighth example. Finally, volume-twiddling techniques are discussed for varying the attack, sustain and decay to make more elaborate sounds. Check back next time! □

With his B.S., M.S. and Ph.D. degrees in chemistry, Karl E. Wiegers is a Senior Research Chemist at Eastman Kodak Company. He has worked with mainframe and microcomputers for fourteen years and has written for several computer magazines, with a number of applications programs published.

Listing 1.

```
0100 ;    SOUND EXAMPLE #1
0110 ;    by Karl E. Wiegers
0120 ;
0130 ;This short program plays a
0140 ;middle C for one second using
0150 ;an undistorted channel one at
0160 ;a volume of 8.  Change value
0170 ;in line 350 for a different
0180 ;frequency.  Change the value
0190 ;in line 370 for a different
0200 ;volume or distortion.
0210 ;
0220 RTCLOK  = $12
0230 TIMER   = $CB
0240 AUDF1   = $D200
0250 AUDC1   = $D201
0260 AUDCTL  = $D208
0270 SKCTL   = $D20F
0280 ;
0290  *=$3000
0300 ;
```

```
0310  LDA #0       ;init for sound
0320  STA AUDCTL ;output by setting
0330  LDA #3       ;AUDCTL and SKCTL
0340  STA SKCTL   ;as shown here
0350  LDA #121     ;pitch for middle C
0360  STA AUDF1   ;freq control reg 1
0370  LDA #168     ;pure tone 160+vol 8
0380  STA AUDC1   ;audio control reg 1
0390 ;
0400 ;Control the length of time you
0410 ;want the sound to play by
0420 ;setting TIMER to the low byte
0430 ;of the number of jiffies
0440 ;(1/60 second) to play,
0450 ;and TIMER+1 to the high byte.
0460 ;
0470  LDA #60&255 ;really just 60 in
0480  STA TIMER   ;this example
0490  LDA #60/256 ;0 here, this is
0500  STA TIMER+1 ;the general form
0510  JSR DELAY   ;go to delay loop
0520  LDA #0       ;set freq to zero
0530  STA AUDF1   ;to turn sound off
0540  BRK          ;stop program
0550 ;
0560 ;subroutine to use internal
0570 ;real-time clock to get
0580 ;accurate timing for delay loops
0590 ;
0600 DELAY
0610  LDA #0         ;init middle and
0620  STA RTCLOK+1 ;low bytes of
0630  STA RTCLOK+2 ;clock to zero
0640 LOOP1
0650  LDA RTCLOK+1 ;loop until
0660  CMP TIMER+1  ;middle byte
0670  BNE LOOP1    ;equals value
0680 LOOP2
0690  LDA RTCLOK+2 ;loop until low
0700  CMP TIMER    ;byte equals
0710  BNE LOOP2    ;desired value
0720  RTS           ;then exit
0730  .END
```

•

Listing 2.

```
0100 ;    SOUND EXAMPLE #2
0110 ;    by Karl E. Wiegers
0120 ;
0130 ;This example plays all tones
0140 ;possible with an 8-bit register
0150 ;(0-255) on sound channel 1,
0160 ;undistorted, at a rate of
0170 ;20 tones per second.
0180 ;
0190 RTCLOK = $12
0200 TIMER  = $CB
0210 AUDF1  = $D200
0220 AUDC1  = $D201
0230 AUDCTL = $D208
0240 SKCTL  = $D20F
0250 ;
0260  *=$3000
0270 ;
0280  LDA #0
0290  STA AUDCTL
0300  LDA #3
0310  STA SKCTL
0320  LDA #168
0330  STA AUDC1
0340  LDA #3       ;3 jiffies per tone
0350  STA TIMER
0360  LDA #0
0370  STA TIMER+1
0380  LDX #0
0390 NEXTTONE STX AUDF1
0400  JSR DELAY
0410  INX
0420  BNE NEXTTONE
0430  STX AUDF1   ;0 at end to turn off
0440  BRK
0450 ;
0460 DELAY
0470  LDA #0
0480  STA RTCLOK+1
0490  STA RTCLOK+2
0500 LOOP1
0510  LDA RTCLOK+1
0520  CMP TIMER+1
0530  BNE LOOP1
0540 LOOP2
0550  LDA RTCLOK+2
0560  CMP TIMER
0570  BNE LOOP2
0580  RTS
0590  .END
```

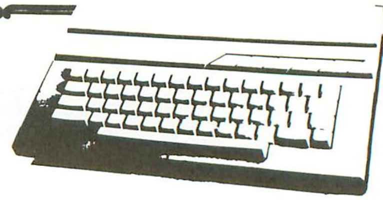
•

Listing 3.

```
0100 ;    SOUND EXAMPLE #3
0110 ;    by Karl E. Wiegers
0120 ;
0130 ;This example plays undistorted
0140 ;tones 0-255 on channel 1, but
0150 ;the value of AUDCTL is changed
0160 ;so that different frequency
0170 ;ranges are covered by those
0180 ;tone values.  The setting of
0190 ;AUDCTL affects all sound
0200 ;channels the same.
0210 ;
0220 RTCLOK = $12
0230 TIMER  = $CB
0240 AUDF1  = $D200
0250 AUDC1  = $D201
0260 AUDCTL = $D208
0270 SKCTL  = $D20F
0280 ;
0290  *=$3000
0300 ;
0310  LDA #3
0320  STA SKCTL
0330  LDA #168
0340  STA AUDC1
0350  LDA #2         ;play 30 tones
0360  STA TIMER      ;per second
0370  LDA #0
0380  STA TIMER+1
0390  LDA #0         ;normal freq range
0400  STA AUDCTL
0410  JSR PLAY
0420  LDA #1         ;low frequencies
0430  STA AUDCTL
0440  JSR PLAY
0450  LDA #64        ;high frequencies
0460  STA AUDCTL
0470  JSR PLAY
0480  BRK
0490 ;
0500 ;The routine which actually
0510 ;plays the tones is now treated
0520 ;as a subroutine
0530 ;
0540 PLAY
```

POWER WITHOUT THE PRICE AT. . .COMPUTER CREATIONS

ATARI®



FLOPPY DISKS AS LOW AS 79¢ EACH

DISKETTES (2 box minimum) 10 per box

	Generic (SKC)		Wabash		Maxell		BASF (Bulk)		BASF (Ea.) 3½"	
# Boxes	SS/DD	DS/DD	SS/DD	DS/DD	SS/SD	DS/DD	SS/DD	DS/DD	SS/DD	DS/DD
2	10.99	12.99	13.99	19.99	18.99	37.00	9.90	11.99	3.49	4.99
3 - 6	9.99	11.99	12.99	19.49	17.99	36.00	8.90	10.99	3.19	4.49
7 - 10	8.99	10.99	11.99	18.99	16.99	35.00	7.90	9.99	2.99	3.99

ATARI HARDWARE

COMPUTERS
Atari 130 XE .... Call For
Atari 520 ST .... New Prices

DISK DRIVES
Atari 1050 ....
Indus GT (Free Software) .... CALL
Astra 2001 ....
Astra "Big D" .... FOR
Atari SF 324 (3½" floppy disk 250K) .... NEW
Atari SF 354 (3½" floppy disk 500K) .... PRICES
Atari SH 317 (3½" hard disk 10 MB) ....
Happy Enhancement for Atari 810 and 1050 Drives ....

PRINTERS
Atari XTM 201 (Non-impact Dot Matrix 20 CPS) ....
Atari XTC 201 (Color, non-impact Dot Matrix 20 CPS) .... Call
Atari XDM 121 (Daisy Wheel ... For
Letter Quality 12 CPS) .... New
Atari XMM 801 (Dot Matrix, ... Prices
Impact 80 CPS) ....
Atari ST 504 (Color Dot Matrix, non-impact 50 CPS) ....

EPSON PRINTERS
LX-80 (80 column) ....
LX-80 Tractor Feed ....
Epson FX-80+ (80 col.) ......
Homewriter 10 ....

STAR MICRONICS PRINTERS
SG-10 (80 column) ....
SG-15 (136 column) .... CALL
SD-10 (80 column) .... FOR
SR-10 (80 column) .... PRICES
Powertype Daisywheel ....
Powertype Tractor Feed ....

PANASONIC PRINTERS
KX - 1090 .... Call
KX - 1091 .... For
KX-1092 .... Prices
Okimate 10 plus plug n' print

PRINTER INTERFACE CABLES
MPP-Microprint 39
MPP-1150 Parallel Printer Interface 59
U-Print A 69
A-16 Interface/Buffer 99
APE Face XLP 59
APE Face 12XLP 59
Microbits Microstuffer 109
MPP-1151 62

PRINTER RIBBONS
Gemini Printers (Black) 3
Gemini Printers (Blue/Red/Purple/Brn./Grn.) 5

Epson Printers .... Call
Panasonic Printers, Black 8
Panasonic Printers, Color 11

MONITORS
Atari XC 141 (14" Composite Color) ....
Atari XM 148 (12" Monochrome, 80 column, low resolution) .... Call For Prices
Atari SM 124 (12" Monochrome, 80 column, high resolution) ....
Atari SC 1224 (12" RGB Color)
Sanyo 12" Green Screen 79
Sanyo 12" Amber Screen 79
Monitor Cable 5
Teknika 13" Color MJ 10............ 189

MODEMS
Atari 1030 Direct 300 BAUD Connect .... CALL
Atari XM 301 Direct .... FOR
Connect 300 BAUD .... PRICES
MPP-1000E Modem 89
Signalman Mark XII Modem with R-Verter 249
Compuserve Starter Kit 24
Volksmodem 300/1200 199

UPGRADES/ ACCESSORIES/BOOKS

ACCESSORIES
Flip n' File 10 4
Flip n' File 15 7
Original Flip n' File 50 14
Flip n' File Cart Case 12
Library Cases(10 colors) 4
Disk Bank/5 (Holds 50) 12
Disk Bank (Holds 10) 5
Power Strip (6 outlet) 16
Lineguard Spike Suppressor 13
Disk Drive Cleaning Kit 13
MicroMate Paper (20#, 540 sheets.) 10
Printer Stand (wire) 16
Dust Covers .... Call for availability

UPGRADES:
Ram Rod X/L W/Omnimon 99
Omniview for 800 XL 49
U.S. Doubler 59
MPP 64 K 69

COMPLETE LINE OF ATARI SOFTWARE

SOFTWARE

NEW ATARI PROGRAMS
Codewriter 39
Filewriter 21
Reportwriter 21
Menuwriter 21
Home Integrator 20
Small Business Inventory 12
Salesman's Expenses 12
Accounts Rec./Pay. 12
Retail Invoice 12
Final Legacy 15
Adventure Writer 18
Star Voyager 12

EPYX
Dragonriders of Pern (D) 21
Summer Games 28
Pitstop II (D) 28
Ballblazer (D) 28
Rescue on Fractalus (D)......... 28
Temple of Apshai (D) 21

ACCESS
Beach Head (D) 24

ACTIVISION
Decathalon (R) 21
Pitfall II (R) 21
Space Shuttle (R) 21
Ghost Busters (D) 24

BATTERIES INCLUDED
Paperclip (D) 41
Homepak 35
B/Graph 48

BRODERBUND
Printshop (D) 31
Spelunker (D) 21
Stealth (D) 21
Whistler's Brother (D) 21

CONTINENTAL
Home Accountant (D) 50

DATASOFT
Bruce Lee 28
Dallas Quest 24
Conan 28
Letter Wizard 52

ELECTRONIC ARTS
Archon (D) 18
Pinball Construction (D) 18
M.U.L.E. (D) 18
Murder/Zinderneuf (D) 18
One on One (D) 25
Archon II (D) 25
Music Construction (D) 18
Realm/Impossibility (D) 18
Hard Hat Mack 14
Seven Cities of Gold 25
Cut and Paste 18

INFOCOM
Cut Throats (D) 24
Deadline (D) 31
Enchanter (D) 24
Hitchhiker's Guide to the Galaxy (D) 24
Infidel (D) 28
Planetfall (D) 24
Sea Stalker (D) 24
Sorcerer (D) 28
Starcross (D) 31
Suspect (D) 28
Suspended (D) 31
Witness (D) 24
Zork I (D) 24
Zork II or III (D) 28
Invisiclues Hint Books 7

MICROPROSE
F-15 Strike Eagle (D) 24
Mig Alley Ace (D) 24
Solo Flight (D) 24
Kennedy Approach 24

OSS
Action (R) 62
Action Tool Kit (D) 25
Basic XL (R) 50
DOS XL (D) 25
MAC/65 (D) 50
Mac 65 (R) 62
MAC/65 Tool Kit (D) 25
Writer's Tool Kit 62

ORIGIN
Ultima III (D) 42

SCARBOROUGH
Mastertype 28
Net Worth 55

SIERRA ON LINE
Ultima I 24
Ultima II 41

SYNAPSE
Quasimodo 17
Alley Cat 14
Syn-File + ... 35
Syn-Calc 35
Syn-Trend 28
Syn-Comm 28
Syn-Stock 28
Mindwheel (needs 2 drives) .... Call

TRONIX
S.A.M. 42
Chatterbee 28

To order call TOLL FREE
1-800-824-7506
ORDER LINE ONLY

No extra charge for MasterCard or Visa

COMPUTER CREATIONS, Inc.
P.O. BOX 493 - DAYTON, OHIO 45459
For information call (513) 435-6868 (Or to order in Ohio)

Order Lines Open 9:00 a.m. to 9:00 p.m. Mon.-Fri.; 10 a.m. to 4:00 p.m. Sat. (Eastern Standard Time). Minimum $10 per order. C.O.D. (add $3.00). Call toll free number to verify prices and availability of product. Hardware requires additional freight charges. Software and accessories add $3.00 shipping and handling in Continental United States. Actual freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico and APO. Ohio residents add 6% sales tax. For immediate delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 3 weeks to clear. School purchase orders welcome. Due to our low prices, all sales are final. NO CREDITS. All defective returns must have a return authorization number. Please call (513) 435-6868 to obtain an RA# or your return will not be accepted for replacement or repair.

```
0550  LDX #0
0560 NEXTTONE
0570  STX AUDF1
0580  JSR DELAY
0590  INX
0600  BNE NEXTTONE
0610  STX AUDF1
0620  RTS
0630 ;
0640 DELAY
0650  LDA #0
0660  STA RTCLOK+1
0670  STA RTCLOK+2
0680 LOOP1
0690  LDA RTCLOK+1
0700  CMP TIMER+1
0710  BNE LOOP1
0720 LOOP2
0730  LDA RTCLOK+2
0740  CMP TIMER
0750  BNE LOOP2
0760  RTS
0770  .END
```

•

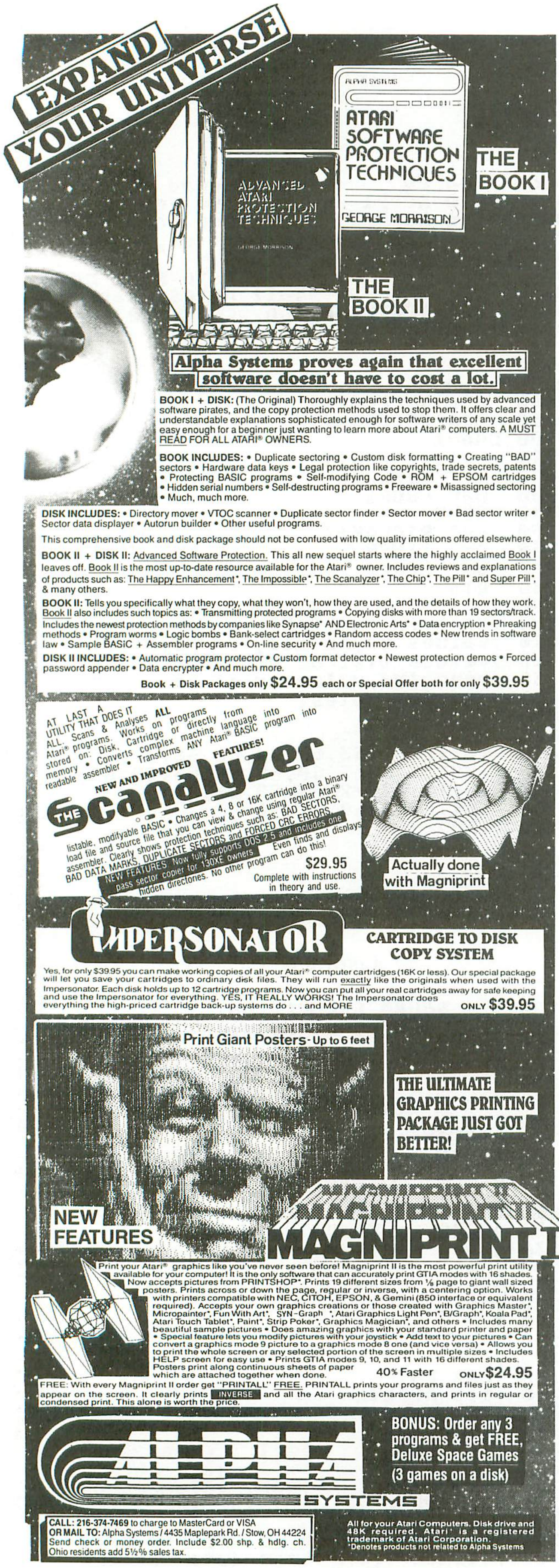
Listing 4.

```
0100 ;    SOUND EXAMPLE #4
0110 ;    by Karl E. Wiegers
0120 ;
0130 ;This example shows how to read
0140 ;frequency values from a data
0150 ;table and play a tune.  Only
0160 ;channel 1 is used.
0170 ;
0180 RTCLOK  = $12
0190 TIMER   = $CB
0200 AUDF1   = $D200
0210 AUDC1   = $D201
0220 AUDCTL  = $D208
0230 SKCTL   = $D20F
0240 ;
0250  *=$3000
0260 ;
0270 ;initializations
0280 ;
0290  LDA #0
0300  STA AUDCTL
0310  LDA #3
0320  STA SKCTL
0330  LDA #168
0340  STA AUDC1
0350 ;
0360 ;The X register is used as
0370 ;an offset pointer to read
0380 ;tone values from the data
0390 ;table at CHAN1.
0400 ;
0410  LDX #0
0420 NEXTNOTE
0430  LDA CHAN1,X    ;get next note
0440  STA AUDF1      ;put in freq reg
0450  LDA #8         ;number of jiffies
0460  STA TIMER      ;to play note goes
0470  LDA #0         ;into TIMER low
0480  STA TIMER+1    ;and high bytes
0490  JSR DELAY      ;perform time delay
0500  INX            ;point to next note
0510  LDA CHAN1,X    ;get it
0520  STA AUDF1      ;stuff it
0530  LDA #5         ;jiffies to play it
0540  STA TIMER
0550  JSR DELAY      ;play it, then turn
0560  LDA #0         ;off sound briefly
0570  STA AUDF1      ;to give separation
0580  LDA #3         ;between notes for
0590  STA TIMER      ;3 jiffies
0600  JSR DELAY
0610  INX            ;point to next note
0620  CPX #26        ;all notes played?
0630  BNE NEXTNOTE ;no, continue
0640  BRK            ;yes, quit
0650 ;
0660 DELAY
0670  LDA #0
0680  STA RTCLOK+1
0690  STA RTCLOK+2
0700 LOOP1
0710  LDA RTCLOK+1
0720  CMP TIMER+1
0730  BNE LOOP1
0740 LOOP2
0750  LDA RTCLOK+2
0760  CMP TIMER
0770  BNE LOOP2
0780  RTS
0790 ;
0800 ;Here is the table of tone
0810 ;values to be played.  Single
0820 ;values will be eighth notes,
0830 ;repeated values will be
0840 ;quarter notes.
0850 ;
0860 CHAN1
0870  .BYTE 60,53,60,64,68,68,47,47
0880  .BYTE 45,45,60,60,68,68,60,60
0890  .BYTE 72,72,60,60,81,81
0900  .BYTE 60,60,91,91
0910  .END
```

•

Listing 5.

```
0100 ;    SOUND EXAMPLE #5
0110 ;    by Karl E. Wiegers
0120 ;
0130 ;Now we will play the same tune
0140 ;as in example 4, but using three
0150 ;sound channels to play chords.
0160 ;
0170 RTCLOK  = $12
0180 TIMER   = $CB
0190 AUDF1   = $D200
0200 AUDC1   = $D201
0210 AUDF2   = $D202
0220 AUDC2   = $D203
0230 AUDF3   = $D204
0240 AUDC3   = $D205
0250 AUDCTL  = $D208
0260 SKCTL   = $D20F
0270 ;
0280  *=$3000
0290 ;
0300  LDA #0
0310  STA AUDCTL
0320  LDA #3
0330  STA SKCTL
0340  LDA #168       ;all 3 channels
0350  STA AUDC1      ;will have the
0360  STA AUDC2      ;same volume and
0370  STA AUDC3      ;distortion
0380 ;
0390 ;The X register is used to read
0400 ;values from three data tables:
0410 ;CHAN1, CHAN2, and CHAN3.
0420 ;
```



Assemble Sound *continued*

```
0430   LDX #0
0440 NEXTNOTE
0450   LDA CHAN1,X    ;get next note
0460   STA AUDF1      ;put in freq reg 1
0470   LDA CHAN2,X    ;do the same for
0480   STA AUDF2      ;registers 2
0490   LDA CHAN3,X    ;and 3
0500   STA AUDF3
0510   LDA #8         ;play for 8 jiffies
0520   STA TIMER
0530   LDA #0
0540   STA TIMER+1
0550   JSR DELAY
0560   INX            ;point to next note
0570   LDA CHAN1,X    ;get them
0580   STA AUDF1
0590   LDA CHAN2,X
0600   STA AUDF2
0610   LDA CHAN3,X
0620   STA AUDF3
0630   LDA #5         ;play for 5 jiffies
0640   STA TIMER
0650   JSR DELAY
0660   LDA #0         ;then turn
0670   STA AUDF1      ;all 3 voices off
0680   STA AUDF2
0690   STA AUDF3
0700   LDA #3         ;for 3 jiffies
0710   STA TIMER
0720   JSR DELAY
0730   INX            ;point to next note
0740   CPX #26        ;done yet?
0750   BNE NEXTNOTE   ;no, go on
0760   BRK            ;yes, quit
0770 ;
0780 DELAY
0790   LDA #0
0800   STA RTCLOK+1
0810   STA RTCLOK+2
0820 LOOP1
0830   LDA RTCLOK+1
0840   CMP TIMER+1
0850   BNE LOOP1
0860 LOOP2
0870   LDA RTCLOK+2
0880   CMP TIMER
0890   BNE LOOP2
0900   RTS
0910 ;
0920 ;Now there are three tables of
0930 ;tone values, one for each
0940 ;channel.  Zeros mean that
0950 ;channel is silent for that
0960 ;note in the tune.
0970 ;
0980 CHAN1
0990   .BYTE 60,53,60,64,68,68,47,47
1000   .BYTE 45,45,60,60,68,68,60,60
1010   .BYTE 72,72,60,60,81,81
1020   .BYTE 60,60,91,91
1030 CHAN2
1040   .BYTE 0,0,0,0,0,0,60,60,60,60,0
1050   .BYTE 0,0,0,0,0,121,121,136,136
1060   .BYTE 144,144,162,162,182,182
1070 CHAN3
1080   .BYTE 0,0,0,0,0,0,68,68,72,72
1090   .BYTE 0,0,0,0,0,0,0,0
1100   .BYTE 0,0,0,0,0,0,0,0
1110   .END
```

•

C.COM

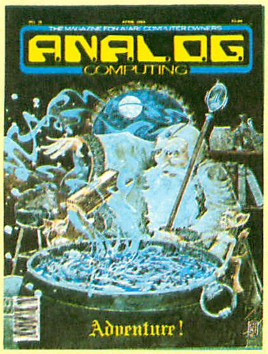
continued from page 62

```
;=======================================
;=========== E Q U A T E S ===========
;=======================================
;
;
CPALOC   = $ØA
LMARGN   = $52
COLCRS   = $55
FRØ      = $D4
INBUFF   = $F3
MEMTOP   = $2E5
CH       = $2FC
DSPFLG   = $2FE
ICCOM    = $342
ICBAL    = $344
ICBAH    = $345
ICBLL    = $348
ICBLH    = $349
ICAX1    = $34A
ICAX2    = $34B
IFP      = $D9AA
FASC     = $D8E6
CIO      = $E456
;
;
;==================================
;===== M A I N   P R O G R A M =====
;==================================
;
;
  ORG $24ØØ
;
;
START
  CLC           ;set up jump to CPGNFN
  LDA CPALOC    ;from GETFN subroutine
  ADC #3
  STA J1+1
  LDA CPALOC+1
  ADC #Ø
  STA J1+2
  LDA #Ø
  STA MODE
  JSR GETFN     ;get first filespec
  BNE HAVE1     ;and branch if called
                ;with arguments - else
                ;enter "COPY>" mode
CLOOP
  LDA #1        ;sets "COPY>" mode
  STA MODE
  JSR PCR       ;skip one line
  LDA #7        ;set margin to col. 7
  STA LMARGN
  LDA #1        ;print starting col. 1
  STA COLCRS
  LDX #LOW CMSG  ;prompt "COPY>"
  LDY #HIGH CMSG
  JSR PRINT
  LDA #5        ;screen input
  STA ICCOM
  LDA CPALOC    ;put input into CPCMDB
  CLC           ;for processing by
  ADC #63       ;GETFN subroutine
  STA ICBAL
  LDA 11
  ADC #Ø
  STA ICBAH
  LDA #5Ø
  STA ICBLL
  LDA #Ø
  STA ICBLH
  LDY #1Ø
  STA (CPALOC),Y ;reset CPBUFP
  JSR CIO        ;get command line
  LDA #2         ;reset margin
  STA LMARGN
  JSR PNUL      ;print null
  JSR GETFN     ;get first filespec
  BNE HAVE1
  RTS           ;RTS if null line
;
;
HAVE1
  PHA           ;save device name
  JSR CLOSE
  PLA           ;restore device name
  LDX #Ø        ;set IOCB #Ø
  STX RW        ;reset RW flag to Read
  CMP #'E'      ;don't reopen Editor if
  BEQ NOPN      ;filespec is "E:"
  CMP #'C'      ;prompt for source
  BNE NSW       ;if sf="C:"
W4S
  LDX #LOW SMSG ;source set-up prompt
  LDY #HIGH SMSG
  JSR PRINT
  JSR WAIT
NSW
  LDA #4        ;OPEN for read
  JSR OPEN
NOPN
  LDA #7         ;use all available
  STA ICCOM,X    ;RAM for copy
  LDA #LOW PTOP
  STA ICBAL,X
  LDA #HIGH PTOP
  STA ICBAH,X
  SEC
  LDA MEMTOP
  SBC #LOW PTOP
  STA ICBLL,X
  LDA MEMTOP+1
  SBC #HIGH PTOP
  STA ICBLH,X
  JSR CIO
  CPY #136      ;EOF status is desired
  BEQ EOF
  CPY #3        ;EOF "look-ahead"
  BEQ EOF
  TYA
  BMI JERR
  LDY #147      ;insufficient memory
                ;error for too long file
JERR
  JMP ERROR
;
;
EOF
  LDA ICBLL,X   ;save number bytes read
  STA BYTES
  LDA ICBLH,X
  STA BYTES+1
  CPX #Ø
  BEQ NC1       ;don't close if sf="E:"
  JSR CLOSE
NC1
  JSR GETFN     ;get destination file
  BEQ SWAP      ;and see if destination
                ;prompt needed
  CMP #'C'
  BEQ SWAP
  LDA #4
  AND MODE
  BEQ WRITE
SWAP             ;destination prompt
  LDX #LOW DMSG
  LDY #HIGH DMSG
  JSR PRINT
  JSR WAIT
WRITE
  LDX #1
  STX RW        ;set RW flag to Write
  DEX           ;set IOCB #Ø
  LDA FSPEC+1   ;don't reopen Editor
  CMP #'E'
  BEQ EWRT
  LDA #8        ;UNProtect if option /O
  AND MODE
  BEQ L1
  LDA #36
  JSR XIO
L1
  LDX #8        ;check for /A option
  LDA #32
  AND MODE
  BEQ Q9
  INX
Q9
  TXA
  JSR OPEN
EWRT
  LDA #11
  STA ICCOM,X
  LDA #LOW PTOP
  STA ICBAL,X
  LDA #HIGH PTOP
  STA ICBAH,X
  LDA BYTES
  STA ICBLL,X
  LDA BYTES+1
  STA ICBLH,X
  INC DSPFLG    ;set DSPFLG to show all
                ;control characters
  JSR CIO
  DEC DSPFLG    ;reset DSPFLG
  TYA
  BMI JERR
  CPX #Ø        ;don't close "E"
  BNE CL
  JMP NC2
;
;
CL
  JSR CLOSE
  LDA #16       ;PROtect if  /P
  AND MODE
  BEQ NC2
  LDA #35
  JSR XIO
NC2
  LDA #2        ;repeat write if  /R
  AND MODE
  BEQ L3
  JMP SWAP
;
;
L3
  LDA #1        ;continue if called
  AND MODE      ;without arguments
  BEQ RTN       ;else RTS
  JMP CLOOP
;
;
;=====================================
;======= S U B R O U T I N E S ======
;=====================================
;
;
GETFN
  LDY #1Ø          ;exit with fn length
  LDA (CPALOC),Y   ;in X and first
  PHA              ;character of device
                   ;name in A
J1
  JSR Ø         ;move filename into
  LDY #1Ø       ;buffer and check for
  PLA           ;change
  CMP (CPALOC),Y
  BNE CONT
  LDA FSPEC+1
  LDX #Ø
RTN
  RTS
;
;
CONT
  LDY #32
  LDX #Ø
FLOOP           ;option check
  INX
  INY
  LDA (CPALOC),Y
  CMP #'/'
  BNE NOPT
  TXA           ;save filespec length
  PHA
```

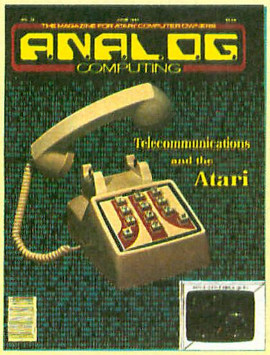
```
MLOOP               ;set mode corresponding
   INY              ;to option(s) selected
   LDX #4
   LDA (CPALOC),Y
   CMP #155
   BEQ EOFS
OLOOP
   CMP OPT,X
   BEQ L4
   DEX
   BPL OLOOP
   BMI Q1
L4
   LDA CODE,X
   ORA MODE
   STA MODE
Q1
   JMP MLOOP
;
;
EOFS
   PLA              ;restore filespec length
   TAX
   LDA #155         ;replaces "/xxxx"
   CPX #4           ;permits setting options
   BNE NOPT         ;with call   "C /xxxx"
   LDX #1
NOPT
   PHA              ;reset Z and N flags
   PLA
   STA FSPEC,X
   BPL FLOOP
   LDA FSPEC+1
   CMP #'D'         ;if not disk, then reset
   BEQ DSK          ;disk-only options /OPA
   PHA              ;save type filespec
   LDA MODE
   AND #7
   STA MODE
   LDX #2           ;set filespec length =1
   PLA              ;restore type filespec
DSK
   DEX
   STX FSPEC
RTN2
   RTS
;
;
OPEN
   LDX #$70         ;IOCB #7
   STA ICAX1,X      ;set open direction
   LDA #3
XENT
   STA ICCOM,X
   LDA #LOW [FSPEC+1]
   STA ICBAL,X
   LDA #HIGH [FSPEC+1]
   STA ICBAH,X
   LDA FSPEC        ;filename length
   STA ICBLL,X
   LDA #0
   STA ICBLH,X
   LDY FSPEC+1
   CPY #'C'
   BNE ODEV
   LDA #12          ;"press RETURN" for tape
   STA CH
   LDA #128         ;short tape gaps
ODEV
   STA ICAX2,X
   JSR CIO
   LDA #255         ;null "keystroke"
   STA CH
   LDA XF           ;ignore errors on XIO
   BNE RTN2
   TYA
   BPL RTN2
   PLA              ;pull stack for error
   PLA              ;on OPEN
ERROR               ;display error number
   STY FR0          ;and return to mode set
   LDA #0
   STA FR0+1
   JSR IFP
   JSR FASC
   LDX #0
   LDY #255
L5
   INY                ;skip leading zero's
   LDA (INBUFF),Y     ;which can show up
   CMP #'0'           ;after FASC operation
   BEQ L5
   DEY
   DEX
L6
   INY
   INX
   LDA (INBUFF),Y
   STA ENUM,X
   BPL L6
   AND #127
   STA ENUM,X
   INX
   LDA #155
   STA ENUM,X
   LDA #9             ;put record
   STA ICCOM
   LDA #LOW EMSG
   STA ICBAL
   LDA #HIGH EMSG
   STA ICBAH
   LDA #37
   STA ICBLL
   LDX #0
   STX ICBLH
   JSR CIO
   JSR CLOSE
   LDA RW           ;if WRITE error, prompt
   BNE L7           ;for another DESTINATION
   JMP CLOOP        ;else enter "COPY>" mode
;
;
L7
   JMP SWAP
;
;
CLOSE
   LDX #$70         ;IOCB #7
   LDA #12
   STA ICCOM,X
   JMP CIO          ;RTS from CIO
;
;
PNUL
   LDX #LOW NULL
   LDY #HIGH NULL
   BNE PRINT
;
;
PCR
   LDX #LOW CR
   LDY #HIGH CR
PRINT
   LDA #9
   STA ICCOM
   STX ICBAL
   STY ICBAH
   LDA #255
   STA ICBLL
   LDX #0
   STX ICBLH
   JMP CIO          ;RTS from CIO
;
;
WAIT
   LDY #255         ;null "keystroke"
   STY CH
W1                  ;wait for SPACE
   LDA CH
   CMP #33          ;internal code for SPACE
   BEQ W2
   CMP #63          ;check for "A"
   BNE W1
   STY CH
   PLA              ;reset stack, Abort and
   PLA              ;enter copy mode
   JMP CLOOP
;
;
W2
   STY CH
   RTS
;
;
XIO
   INC XF           ;set flag to ignore any
   LDX #$70         ;XIO errors
   JSR XENT         ;use common code
   DEC XF           ;reset error ignore flag
   RTS
;
;
;=====================================
;=== F L A G S  and  B U F F E R S ===
;=====================================
;
;
RW                  ;Read/Write error flag
   DB 0
;
XF                  ;XIO flag
   DB 0
;
MODE                ;mode flags
   DB 0
;
OPT                     ;table of options
   DB 'APOWR'
CODE                    ;codes for flags
   DB 32,16,8,4,2
;
EMSG
   DB 'Error #'
ENUM                    ;error # output buffer
   DB '      ',253,253,155
;
SMSG
   DB 'Set up source, hit '
   DC 'S','P','A','C','E'
   DB 253,155
;
DMSG
   DB 'Set up destination, hit '
   DC 'S','P','A','C','E'
   DB 253,155
;
CMSG
   DC 'C','o','p','y','>'
   DB ' '
NULL
   DB 28
CR
   DB 155
;
FSPEC               ;Filename buffer
   DB 155,155,155,155,155,155,155,155
   DB 155,155,155,155,155,155,155,155
;
BYTES               ;File length
   DB 0,0
;
PTOP                ;Main buffer
;
   ORG $02E0        ;set RUNAD for execution
;
   DW START
   END
```

●

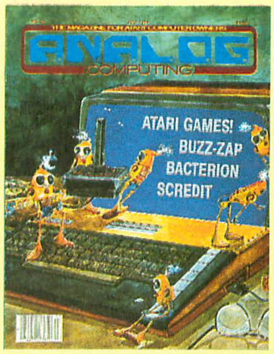
BACK ISSUES



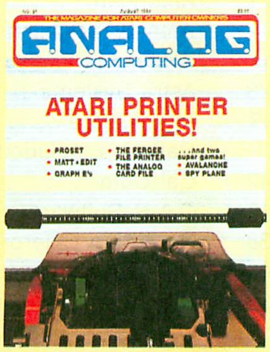
ISSUE 18 • Crash Dive • File'em • Munch'in Climb'in • H:BUG • Adventure Reviews



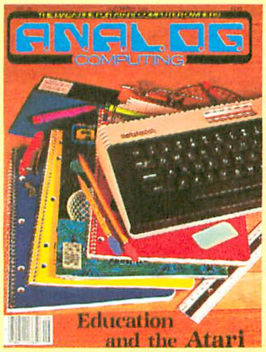
ISSUE 19 • Battle in the B-Ring • A Look at Modems • Bulletin Board Systems • Siege • Touch-Tone® Dialer



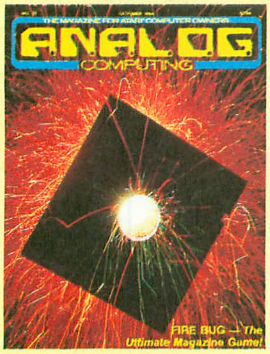
ISSUE 20 • Buzz-zap! • Scredit • AlterDOS • Bacterion! • ConTEXT



ISSUE 21 • Selecting your Perfect Printer • Matt★Edit • Graph E's • Spy Plane • Printer Survey



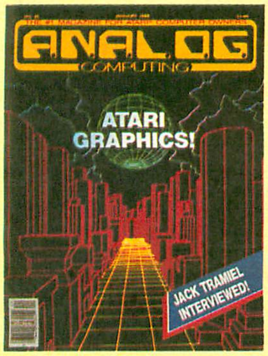
ISSUE 22 • Typing Evaluator • Math Attack • Micro-Puzzler • Air Attack • Mathman • The Reading Program



ISSUE 23 • Fire Bug • Minicomp • Dark Horse • Climber • P/M Creator/Animator

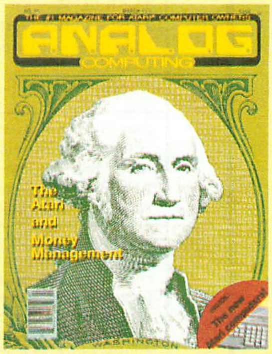
ISSUE 24 • Circuit Database • Bopotron! • XL-DOS • Cassette Compressor • Race in Space • Unicheck

ISSUE 25 • Weather Forecaster • Androton • Miner Jack • BASIC Tutorial Part 1 • Adding BASIC Function Keys

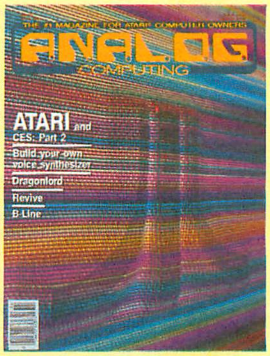


ISSUE 26 • BASIC Tutorial Part 2 • Robot Raid • Graphics Overlay • Popcorn • Magic Palette • PuLse in Action!

ISSUE 27 • English Error Messages in BASIC • Instant Renumber • MicroCheck Part 1 • Adventure at Vandenberg • Screenmaker



ISSUE 28 • MicroCheck Part 2 • TwoGun • Cascade • Monthly Mortgage Calculator • Demon Birds • MicroDOS XL



ISSUE 29 • RAMCHECK • Revive Dragonlord • XL Expansion Connector • Cheep Talk

All back issues $4.00 each

Send check or money order to:
ANALOG Computing Back Issues
P.O. Box 615, Holmes, PA 19043

MasterCard and VISA orders call:
1-800-345-8112
in PA. 1-800-662-2444

INDEX TO ADVERTISERS

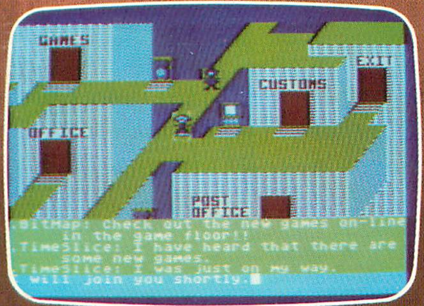
This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

THE CREATION OF A NEW BREED

The GCP Network: More Communications, More Graphics, LESS Hassle!

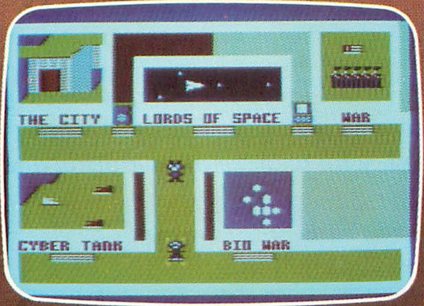
The GCP City:

The entry to worlds of adventure and challenge.



The Game Floor:

Choose your favorite and match wits with others.



Lords of Space:

Explore and conquer your way to galactic supremacy.

FEATURES: Joystick-driven; no commands to learn! Private mail, bulletin boards, program library, file transfers, automatic logons, User Group support, local phone call access from most cities, no network surcharge. SYSTEM REQUIREMENTS: 48K Atari, 1 disk drive, modem.

For more information contact:

GCP GAMES COMPUTERS PLAY, INC.

112 East Market Street, York, PA 17401 / 717-848-2660

CIRCLE #140 ON READER SERVICE CARD