ST-LOG

THE ATARI ST MONTHLY MAGAZINE

U.S.A. $3.50
CANADA $4.75

MARCH 1987 ISSUE 12

ENTERTAINMENT ISSUE

An interview with Anita Sinclair
Author of *The Pawn*

ST Nightmare Repair

Score Four

Poker Dice

MARK WILLIAMS C. AN ENLIGHTENING DEVELOPMENT FOR ATARI ST USERS.

If you've tried your hand at developing applications on the Atari ST, you know the problem. Programming tools aren't only hard to come by, they're hard to use. One might even say primitive. But now for some enlightening news: you can have all the power, portability and versatility of the C language from a leader in professional C programming tools, Mark Williams.

BRING YOUR PROGRAMMING UP TO SPEED.

The Mark Williams C compiler produces fast, dense code and supports the complete Kernighan & Ritchie industry standard C. You'll have access to GEM's AES and VDI libraries for programs using graphics, icons and the Atari mouse. And Mark Williams C lets you take advantage of the full 16 megabytes in Atari's 68000 microprocessor.

STREAMLINE DEVELOPMENT WITH POWER UTILITIES.

Mark Williams C is loaded with everything you'll need for professional development. Bring the power of the UNIX environment to your Atari ST with our Microshell Command Processor including pipes, I/O redirection and more. Edit your program with the highly acclaimed MicroEMACS full screen editor. Accelerate and simplify compiling with *make* which finds and recompiles only those modules affected by your changes. Then, when you're ready for debugging, call on our db Symbolic Debugger with single step, breakpoint and stack traceback functions. Over 40 commands, including a linker and assembler, provide a total development package for your Atari ST.

Features

C compiler
- Complete Kernighan & Ritchie C plus extensions
- Up to eight register variables
- Full access to AES and VDI libraries for programs using graphics, icons and mouse
- Complete UNIX-compatible libraries allow easy portability to and from UNIX development environment.
- Over 300 Atari-specific routines
- One-step compiling, linking with cc command
- English error messages
- Lint-like error checking

Microshell Command Processor, powerful UNIX style shell includes I/O redirection, pipes, command substitutions

MicroEMACS Full Screen Editor with commented source code included

Make Program Building Discipline

Complete symbolic debugger with single-step, breakpoints and stack traceback

Assembler, linker and archiver

Powerful Utilities Package: egrep, sort, diff, cmp, pr, tail, uniq, wc and more

Over 600 pages of documentation including 120 sample C programs

Not copy protected

MARK WILLIAMS C FOR THE ATARI ST
$179.95
60 DAY MONEY BACK GUARANTEE

DEPEND ON A NAME WITH A HISTORY OF PERFORMANCE.

Mark Williams C for the Atari ST is part of our growing line of C compilers. A line that includes the C compiler chosen by DEC, Intel, Wang and thousands of professional programmers. Now our Atari C compiler is earning its own reputation:

"Finally a great C compiler that exploits the power of the ST"–Sigmund Hartmann, President, Atari Software Group

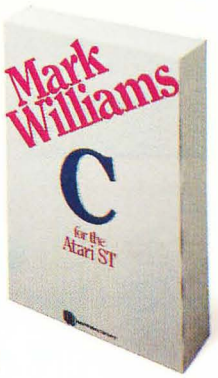
"The all-around best choice for serious software development on the ST."–Douglas Weir of *ANALOG COMPUTING*

GET WHAT YOUR ATARI ST HAS BEEN WAITING FOR.

Mark Williams C is just what your Atari ST was made for: powerful, professional programming. So now that you can have Mark Williams C for just $179.95, what are you waiting for?

Ask your Atari dealer about Mark Williams C or order today by calling **1-800-MWC-1700.***

*In Illinois call: 312-472-6659



Mark Williams Company

1430 West Wrightwood, Chicago, Illinois 60614

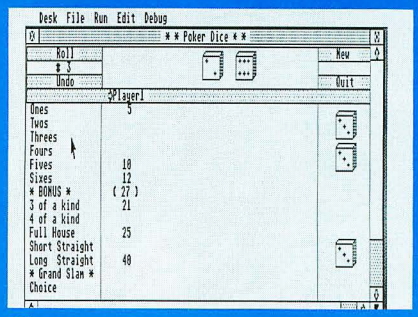
ST-LOG
THE ATARI ST MONTHLY MAGAZINE

ISSUE 12
MARCH 1987

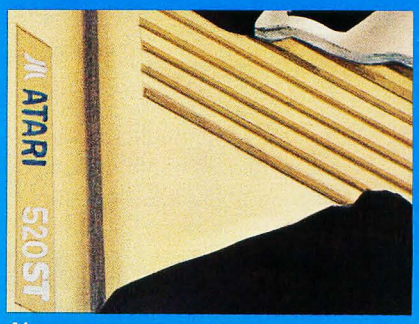
FEATURES

REVIEWS

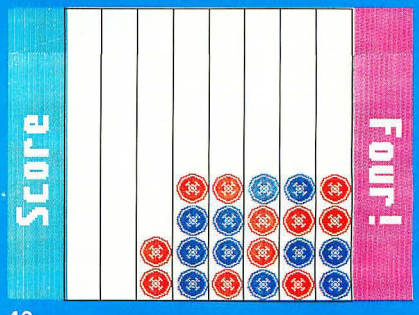
COLUMNS



17



41



49

ST-Log (ISSN 0890-9601) is published monthly for $28.00 ($36.00 in Canada, $39.00 foreign) per year by ANALOG 400/800 Corp., 565 Main Street, Cherry Valley, MA 01611. Second class postage pending at Worcester, MA and additional mailing offices. POSTMASTER: Send address changes to **ST-Log**, P.O. Box 625, Holmes, PA 19043.

Let your artistic nature bring you fame and fortune

in the

D·E·G·A·S Elite

DESIGN & ENTERTAINMENT GRAPHIC ARTS SYSTEM

CLIP-ART CONTEST

Co-sponsored and judged by

ANALOG Computing/ST-Log

FIRST PRIZE
$1000
and your choice of
3 software packages
from
Batteries Included

SECOND PRIZE
A Supra 10-meg hard disk
and your choice of
3 software packages
from **Batteries Included**

THIRD PRIZE
Double-sided drive or
monochrome monitor
and
3 Batteries Included software packages

3 RUNNERS UP
1-year ST disk subscription to ANALOG/ST-Log
and
1 Batteries Included software package

The winning illustrations will be published by Batteries Included in a Clip-Art collection disk, and will be published individually on disk versions of **ST-Log**, *as well as used for illustration in the pages of* **ANALOG Computing/ST-Log** *magazine.*

The **DEGAS Elite Clip-Art Contest** is open only to registered owners of the ***DEGAS Elite*** software program from Batteries Included.

Look for specially marked packages of ***DEGAS Elite*** for your entry card and rules. If upgrading from ***DEGAS***, ask Batteries Included for a **Clip-Art Contest** entry form.

Entries must be received by midnight, EST, May 31, 1987. Direct any inquiries about the contest to: **Batteries Included**, 30 Mural Street, Richmond Hill, Ontario, Canada L4B 1B5

Pick up Elite *and get your mouse in motion!*

MOVING?

DON'T MISS A SINGLE ISSUE

Let us know your new address right away. Attach an old mailing label in the space provided below and print your new address where indicated.

DO YOU HAVE A QUESTION ABOUT YOUR SUBSCRIPTION?

Check the appropriate boxes below:

- ☐ New subscription. Please allow 4 to 8 weeks for your first copy to be mailed.
- ☐ Renewal subscription. Please include a current address label to insure prompt and proper extension.
- ☐ 1 year — $28.00. This rate limited to the U.S. and its possessions.
- ☐ Payment enclosed. ☐ Bill me.

MAIL TO: **ST-LOG,** P.O. BOX 625, HOLMES, PA 19043

Name _____

Street Address _____

City _____ State ___ Zip ___

ATTACH LABEL HERE

(IF LABEL IS NOT HANDY, PRINT OLD ADDRESS IN THIS SPACE.)

ST-Log Staff

Editors/Publishers: Michael J. DesChenes, Lee H. Pappas. *Managing Editor:* Diane L. Gaw. *East Coast Editor:* Arthur Leyenberger. *Midwest Editor:* Matthew J.W. Ratcliff. *Contributing Editors:* Ian Chadwick, Arnie Katz, Bill Kunkel, Steve Panak, D.F. Scott, Joyce Worley. *Contributing Artists:* Mark Astrella, Isao Yonehara. *Technical Division:* Charles Bachand, Clayton Walnum, Douglas Weir. *Production:* Connie Moore, Edythe Stoddard, Jane Sullivan, Kathy Wiesner. *Advertising Manager:* Michael J. DesChenes. *Accounting/Circulation:* Robin Levitsky. *Production/Distribution:* Lorell Press, Inc. *Contributors:* Victor T. Albino, Steven Grimm, Charles F. Johnson, Mike McCarthy, Maurice Molyneaux.

U.S. newsstand distribution by Eastern News Distributors, Inc., 1130 Cleveland Rd., Sandusky, OH 44870.

ST-Log magazine (ANALOG 400/800 Corp.) is in no way affiliated with Atari. Atari is a trademark of Atari Corp.

Where to write

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

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ST-Log**, 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. We cannot reply to all letters in these pages, so if you would like an answer, please enclose a self-addressed, stamped envelope.

Advertising Sales

Permissions

Subscriptions

ST-Log, P.O. Box 625, Holmes, PA 19043; or call toll-free: 1-800-345-8112 (in Pennsylvania, 1-800-662-2444). Payable in U.S. funds only. U.S.: $28.00-1 year; $52.00-2 years; $76.00-3 years. Canada: $36.00-1 year; $68.00-2 years; $99.00-3 years. Foreign: $39.00-1 year; $72.00-2 years; $104.00-3 years. For disk subscriptions, see the cards at the back of this issue.

Authors

When submitting articles and programs, both program listings and text should be provided in printed *and* magnetic form, if possible. Typed or printed text copy is mandatory, and should be in upper- and lowercase, with double spacing. If a submission is to be returned, please send a self-addressed, stamped envelope.

Editorial

Entertainment is the theme of this month's **ST-Log**. Of course, there's more than one dimension to this idea of computer entertainment. For most of our readers it means games. For others, creating the games is where the real entertainment begins—the familiar "I write the **Pongs**" syndrome. Still others would say that if you haven't seen the staff of **ST-Log** sweating out the latest issue, you don't know what real entertainment is.

In any case, this month we have lots of the first two varieties (as for the third, you'll have to wait for *ST-Log: The Movie*). There are two games: one (**Poker Dice**) in BASIC, the other (**Score Four**) in Megamax C. Anita Sinclair, creator of the near-legendary **The Pawn**, is interviewed by our own Mike Wallace of the Microchips, Clayton Walnum. And there are plenty of reviews, most of them of games, from **King's Quest II** to **Skyfox**.

True, not everything could be included. Right up to the last minute we were considering a nifty adventure game where players wend their way through a maze of poison ivy vines. But we ran into copyright problems when the author refused to change what he insisted was an original title, "Mister Scratch."

There you have the extent of our "special" effort for this month. But that doesn't mean you can skip over **AS68 Helper** or **Status report** or **Winter CES** as if they were so much homework. No—everything this month is entertaining, as usual.

Many of you have heard of the Fog and Flesch indexes which are used to measure the complexity of English text samples. Well, we at **ST-Log** have our own index of each article's impact, based on its Amusement-to-Information Ratio—its AIR force, as we like to say. Whenever we want to test a piece, we just run it through this index and see what sort of AIR comes out. Ideally, the ratio will be about fifty-fifty. It can't always be that way, but we do our best.

Every new **ST-Log** is an entertainment issue: an ongoing variety show of fresh information that's never quite the same from month to month. Our first goal is to deliver the information quickly and accurately, but the how is just as important as the what.

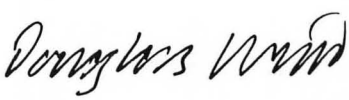
We could, say, replace our present review format with a standard checklist of desired functions and features, and have reviewers mark A, B or C, as appropriate. That would certainly put us in the machine age. But where would the fun be? No one would want to trade a Steve Panak review for a file card.

Burying the facts in fluff isn't our style either. When you turn to **Status report**, you get just that—a status report, not warmed-over ad copy. The two goals go together. The more you enjoy the magazine, the more we feel you'll get out of it. In **ST-Log**, what's good for you doesn't taste bad—it tastes good.

In short, we want our readers to turn the last page of this magazine and say, "Boy, that was fun. Not only am I well informed about all the latest Atari ST news and software, I'm also in a good mood!" Those may not be your exact words, but you get the idea.

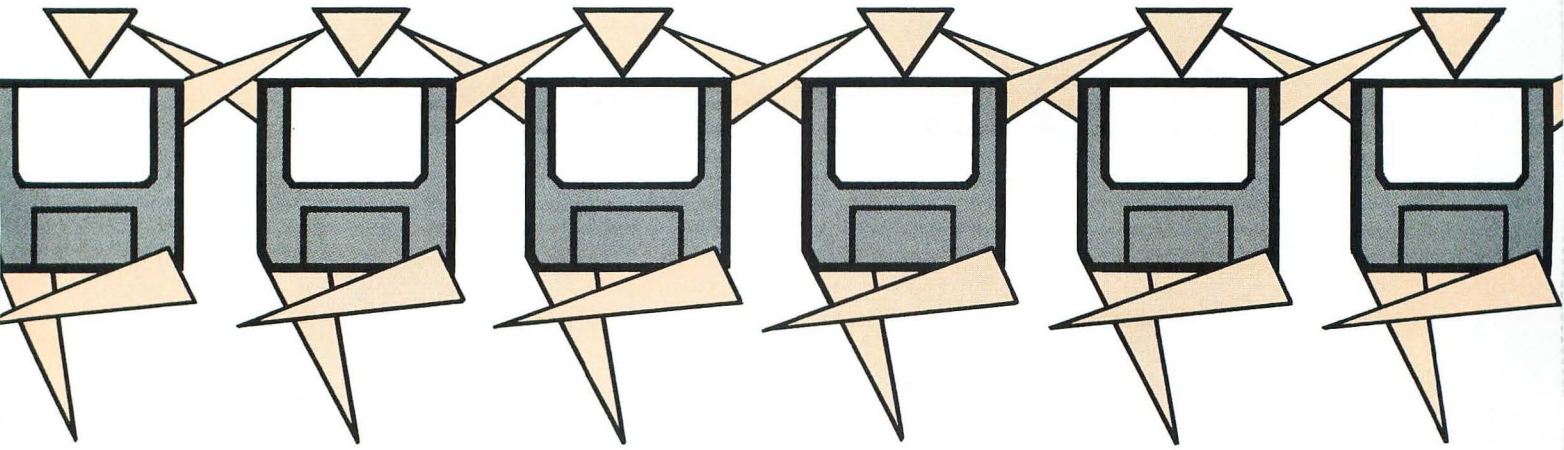
ST-Log is a variety show with a ballot box. If you don't like something, write to us. Tell us what you do like, too. Send in your reader comment cards. We do accept applause, but you have to clap very loud.

When you get the best software and the mostly timely and useful ST information we can find, all put together in the most stylish and enjoyable package we know how to make—now that's entertainment.



Douglas Weir
Technical Editor
ST-Log

The Atari ST Software Line Up



/RITE 90°
29.95 This package urns spreadsheets, ext and screen utput on their ear y allowing vertical ormatted files to be rinted sideways sing continuous or ingle sheet paper. ncredibly useful for nose over sized preadsheets or flow hart style programs. ve character sizes re available. WRITE 0° is GEM based and upports Epson FX, MM804, NEC, ROWRITER and ompatible printers.

TYPESETTER ELITE
$49.95 Is a GEM based desk top publishing package that integrates text, graphics and DEGAS like drawing tools and allows for a full lay-out page preview and a what-you-see-is-what-you-get print out. TYPESETTER ELITE can handle custom fonts and icons; and ASCII files can be loaded into TYPESETTER ELITE pages. You'll be impressed with the high resolution and clarity of your printer output when it is driven by TYPESETTER ELITE.

RUBBER STAMP ST
$39.95 A utility that lets a user manipulate pictures from DEGAS and other popular graphics programs. RUBBER STAMP ST is useful for creating icons and printing out repititious full screen pictures, graphic address labels, index, Rolodex or other card sized output. RUBBER STAMP ST can add text to pictures in multiple sizes and styles and can load in fonts from DEGAS and MEGAFONT ST.

MEGAFONT ST
$39.95 Soup-up your printer output by designing your own fonts with GEM based MEGAFONT ST. Mix font styles to indicate italics or to provide emphasis. Arrange, space, position and size text and graphics in a document with MEGAFONT ST's click through "GEM" menus. MEGAFONT ST embeds graphics, text and fonts in the same file. Compatible with DEGAS, NeoChrome and the XLEnt Line Up of ST software, MEGAFONT ST also has a built in font editor. MEGAFONT supports SMM804, NEC, PROWRITER and EPSON compatible printers.

PM INTERFACE
$29.95 Is a GEM based driver that will enhance the utility of Print Master. PM INTERFACE allows Print Master to be integrated with DEGAS and the XLEnt Line Up of ST Software. You can use PM INTERFACE to create new Print Master graphics and borders using stored portions of DEGAS and other picture files. PM INTERFACE converts Print Master graphics into icons that can be manipulated by the XLEnt ST software utility Line Up.

ST MUSIC BOX
$49.95 A MIDI utility that lends graphics support and other administrative tweaking support to your composing. Allows for changing key and time signatures, inserts, deletes and copies measures. Prints high quality sheet music with the option of lyric and graphic notation. Compatible with an ST console or a MIDI synthesizer. ST MUSIC BOX is another MIDI progression from the authors of the Abacus book, Introduction to MIDI Programing for the Atari ST.

P.O.Box 5228
Springfield, Virginia 22150
(703) 644-8881
Order Number
(703) 569-8881
Customer Service
TELEX 269728 XLENT UR

XLEnt Software

XLENT Software [U.K.] LTD
516 Alum Rock Road
Alum Rock
Birmingham B8 3HX
United Kingdom
(021) 327-6110
TELEX 265871 (MON REF G)

CIRCLE #102 ON READER SERVICE CARD

Reader comment

Alert on *Alert Boxes*.

Best of luck with **ST-Log**. Please, *please* keep up the good work. Since I'm a new ST owner (but an old 800/XL owner), I enjoy the tutorials—not only for what they show, but also for what they allow me to experiment with. I would like to focus on one: **Alert Boxes**, on page 82ST of the November 1986 **ANALOG Computing**, issue 48. Maybe you already know that the program can bomb, quite easily.

Would you like to know how to fix it? Really! What do you think about extending my subscription? No, well, how about the ST disk from issue 10? You'll think about it? Okay!

Just add *+CHR$(0)* to the end of Line 63010 (Listing 1, page 84ST). The hex zero is evidently a delimiter.

Thanks again, ladies and gentlemen, for quality publications.

Carl Jarnberg
Las Vegas, NV

Content comments.

I would like to echo the sentiments of Mr. Robert J. Fusillo, whose letter in the January 1987 **ST-Log** put my case as surely as I could state it myself.

To quote: "I do not want to be a programmer. I do not want to be a hacker. I don't care about C! I don't wish to know anything about Logo. I am interested in programs I can use, not write. . .I am not at all interested in games. . .I hope for reviews, but get yet another compiler program. . . couched in the inexplicable language of the experienced hacker."

Now, as to what I *do* want! I want product reviews that are not cold-cribbed from the manufacturers' literature. And, more than almost anything else, I want manuals that are not written by the programmers, but which are written by ordinary, literate mortals who have used the equipment or the program, as the case may be. I would also appreciate having indexes to my manuals—and I mean a complete index, which doesn't require you to know a hatful of alternate commands in order to locate the one you are looking for.

And, as one last wish, I would like to see the states pass capital punishment laws for manufacturers who advertise products "Developed especially for the ST," but who insert a disclaimer inside the shrink wrap, which says, "But only on a certain sort of ST—and then only if the manufacturer upgrades his ROM patch to fit our products!

Enough said!
Walter E. Ross
Springfield, IL

Other than by printing your letter, we can't do too much about manuals or software specs. But our reviews are just what the consumer ordered, and you'll find plenty of useful programs in these pages. Our ***Step 1*** *series, begun last issue, should help those readers who need beginning insight into and familiarization with their STs. In short,* ***ST-Log****'s staff is making an effort to publish high-quality material to meet the needs of* all *ST owners.* —*Ed.*

I own a 520ST. Bought it after reading your February 1986 issue's **High Noon** article; I liked what I read, and the computer has proved a useful addition to the household. Of course, it arrived with the usual package of language disk and **1st Word**—with which this is being written and printed on a Star NX-10 (obtained from one of your magazine's advertisers, White House Computers).

The 520ST is my second computer, the first being a Timex Sinclair 1000 with a great big 16K of added memory, no printer, just a black-and-white TV and an available audio cassette deck. It was a challenge to master programming in the T/S BASIC, and I thought that it was time to go on to bigger and better things available on the 520ST.

The ST's hardware is fairly simple and straightforward: connect this to that, etc., and turn it on. Problems of understanding the software—the owner's manual, the *ST BASIC Sourcebook* and tutorial, and the user's manual for the Star NX-10—could easily overwhelm anyone who had not just made the major investment that these purchases represent. I've found, thus far, that **1st Word**, is an adequate word processing program, although some of the articles and ads suggest that others might be more appealing, especially with a different type of printer.

Having found ST BASIC a very tiring exercise, I also bought **VIP Professional** (from Abby's—again, because of their ad) and have found it to do everything claimed, and some not mentioned. VIP's handbook is more understandable and clearer than others mentioned before. The only big problem is that it was written for the GEM version, whereas the program disk is for the text version. Very unfortunately, the VIP program leaves only about 50K of memory after loading it, thus making the claimed size of the spreadsheet a seemingly doubtful benefit on a 520 (maybe better for the 1040?)

Your decision to forge ahead with **ST-**

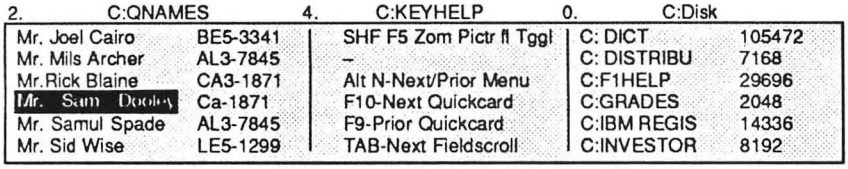
$79.95

Zoomracks II: an integrated productivity tool thats fun to use.

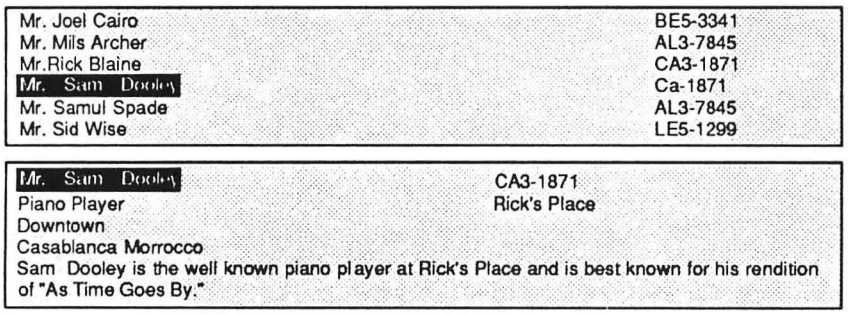
Now with math, report formatting, improved mail merge, Degas file displays, easier to learn.

Zoomracks is an integrated productivity tool based on a simple and familiar concept. You keep your information--names and addresses, invoices, correspondence, or whatever-- in cards in racks like those next to factory time clocks. It is easy to use, helps you structure your problems, and it can handle an exceptionally wide variety of applications.

Zoomracks lets you zoom back to see the first line of several cards, and several racks next to each other--compressed for display with its Smart Zoom feature.



Here you get an overview of three racks. Pressing one key zooms you in on one rack; pressing another, zooms you in on one card. To go someplace on the screen or execute a command, just click the mouse.



What the reviewers said about Zoomracks (version I)

"Zoomracks, a $79.95 wonder...from Quickview Systems"
– **Newsweek**

"The simplest thing in the world to use... very much recommended
–*Jerry Pournelle*, **Byte**

"Truly innovative, ... flexible... extremely easy to use..."
– **InfoWorld**

"practical and even fun ... you can always visualize your data as you want --- from a broad overview of the whole database to a specific detail of a single field..."
– *Art Leyenberger*, **Compute!**

"99% of the database needs in home and small business computers do not require the power and prorgramming features of DBASE III. In fact, by the time you learn DBASE III, you could have all your database needs filled by Zoomracks.

For those of us with two left thumbs...It is very hard to lose data with this well thought out system... less than $100 and worth 10 times that amount"
– *Don Terp*, **ST Business**

You can be an anarchist...and this program will bring order to your chaos...It is useful enough to become a standard equivalent to Lotus 1-2-3 in the industry.
– *Jim Bumpus*, **Atari ACE**

"*****...the more I use Zoomracks, the more I like it, and the more I can see to do with it.... will be with us for a long time, finding a solid niche in both home and office."
– *Richard Keller*, **ST Applications**

Zoomracks has established a productivity standard in design, concept, originality, and functionality."
– *Alan Glick*, **JBUG** (Boston Computer Society)

Zoomracks is totally in a class of its own... I found it so useful I considered buying another ST for my office.
– *John Leon*, **HASTE Electronik Letter**

CIRCLE #103 ON READER SERVICE CARD

60 day money back guarantee!

Project Organizer
Data Base
Calculator
Word Processor

Power without the programming

POWERFUL: Relational data base, word processor, mail merge, arithmetic, display Degas files, report formatting, organizer, macros, 250 line fields, import-export data.
FLEXIBLE: Modify templates and report formats, copy between fields and racks, view multiple racks, zoom in on a rack or card.
EASY TO USE: No field sizes to specify, few frustrating error messages, visual interface, Doesn't require programming. Few concepts to learn: everything based on cards in racks. You can undo commands
EASY TO LEARN: on-line tutorial, on-line help, 180 page manual, use Commands or menu. Works with or without mouse.
FUN: For people who hate data bases. Point and zoom in on cards, racks. Toggle commands let you try and untry things.

Uses are limited only by your imagintion: Ask some one who uses it.

OFFICE: Mailing lists, Form letters, invoicing, agendas, projects, tickler files, payables, receivables, credit records, correspondence, schedules, appointments, office procedures, check registers, employee records.
EDUCATION: annotated bibliographies, lesson planning, test banks, dissertations, organize notes for class, books, anything you do with index cards.
HOME: Address and phone lists, insurance catalogue, credit card records, shopping lists, Christmas card records, videotape/records catalogue, recipe cards, checkbook.
SALES: prospect files, tickler file, call reports, mailing lists, sales order entry.
Coming soon: Zoomracks Starter Packs with templates and forms for the office, home, school, collectors etc. ($19.95 each).

Available at your local ST retailer or mail this coupon to **Quickview Systems**, Dept. A, 146 Main St., Suite 404 Los Altos, CA 94022
California residents add 6.5% sales tax.

☐ Zoomracks II* $119.95 ($149.95 after 12/25)
☐ Zoomracks I $79.95 (Upgraded)

I understand by ordering direct I can return Zoomracks for a full refund within 60 days if not satisfied for any reason.

Name _____
Address _____
City _____
State _____ Zip _____
☐ Check enclosed ☐ Visa ☐ MC
Card # _____ Exp date: _____

*Zoomracks II lets you save macros and create multiple print formats. Registered Zoomracks owners can upgrade to Zoomracks I ($14.95) orZoomracks II ($79.95)

Zoomracks is not copy protected.

Log is commendable; now I won't have to read an article for a while before I learn that it is largely inapplicable, if not unuseable to me if it is limited to 8-bit machines.

After reading—with some care, but not total understanding—your articles on windows and **Font Tricks**, I wonder if you can give me (and others) a greater understanding of the **ST-Log** disk. It almost seems as if ST owners who merely want to use their computers might be better served by subscribing to the disk version, so that they have the programs without all the bother of typing them in. I'm not convinced, however, as I read about "unsqueezing"—this tends to put me off. I don't understand how to use programs that don't have .PRG, .TOS or .TTP extensions. Another question: I believe that there are other BASICs besides Atari ST BASIC; would these work with files with a .BAS extension? Also, what would I do about those with .C and .S extenders? Any form of C? Please, as Robert Fussilo (**Reader comment**, January 1987) points out, we're not all hackers—and we don't really want to be. Can you decode the jargon and give us the facts in plain English? Please?

Sincerely,
Richard Ostheimer
Fort Myers, FL

*Every once in a while we have so much material for an **ST-Log** disk that we have to "squeeze" the files that go on the distribution disks. When we do this, we include a program (and the instructions) for "unsqueezing" the files onto another disk.*

*Files with extensions other than .PRG, .TOS, .ACC or .TTP are not programs. Instead, they can be text files, data files used by other programs, source code files, and so on. For a fuller explanation, see this month's **Step 1: Soft Wares**.*

You are right: there are several other BASICs, and they are not necessarily compatible with each other. They should all accept source code files with a .BAS extension, but beyond that there are no guarantees. In fact, there's a strong probability that, for example, a program that runs under ST BASIC will have to be changed to work with GFA BASIC or Softworks BASIC or even LDW BASIC, to name just a few of the other versions.

The same is true for other languages. Source code written for the Megamax C compiler may not compile or run without error using the DRI (Developer's Kit) compiler, even though both compilers expect, as a default, that source code files will have an extension of .C. Files with .S extensions contain assembly language source code, and even here—believe it or not—there are incompatibilities among assemblers. —*Ed.*

I read your first separate issue of **ST-Log** with interest. However, I felt that it was missing something—there were no articles on "low-level" programming techniques. Not even a game or a graphics demo program to tinker with. The issue was informative, but didn't have anything I could really apply. Tom Hudson's slider programs are useful—if you're writing a high-level application. But what if I want to write a shoot-'em-up video game? (You remember those.) Where can I find information on scrolling, vertical blank interrupts, Line A routines, sound and the like?

Of course, there are the Abacus books; *Internals* is informative, but not the most user-friendly piece of documentation I've come across. For example, I spent four days trying to figure out their explanation of the intelligent keyboard processor. Decoding their uncommented assembly language example was no fun.

I realize that this was the first separate issue. However, I hope that in the future you'll have some articles on lower level programs, and I also would love to see some good game programs. As an owner of an Atari 400, I learned more about machine language programming from Tom Hudson's two games (**Fill 'er Up**, issue 10, and **Livewire!** from issue 12) than from any other source. I never bothered with the BASIC games—they were always slow. Now, however, with the C language implemented on the ST series, some excellent, quick games can be produced in a very short amount of time. All it takes is a few examples so that programmers can master the basic techniques. In fact, I wouldn't be surprised to see commercial quality video games published in your magazine—there just isn't a market for a $30.00 blast-'em sort of game any more—but for $3.50, I'll buy one in a magazine any day. If it looks really good, I might even buy the disk version and play it, rather than just look at the source code.

Anyhow—yes, keep up the reviews. Keep up the introductory columns. Continue the application articles. But realize there are still a lot of die-hard programmers out there who want to know how to interface C and assembly language, how to display fantastic graphics on the screen, and how to get into the BIOS and tinker. Please, please, *please* do not become the *Macworld* or the *Amiga World* of the ST community.

Sincerely,
Bruce Graves
Ithaca, NY

*We'd like to publish a steady stream of articles on low-level programming techniques. So far the main obstacle has been that non-trivial programs of any kind for the ST tend to be larger than they were on the 8-bit machines. The result: source code is often relegated to the **ST-Log** disk. We're still trying to come up with a workable solution to this space problem. One possibility: publish some programs in segments, over several months.*

*There is a compatibility problem, too: which compiler or assembler do we choose? Consider, for example, linking to external assembly language routines. Megamax isn't really set up to do this sort of thing at all—the in-line assembler is supposed to handle it. Last month's **Money** demonstrated various (Alcyon) C graphics programming techniques, and it included several assembly language subroutines called from C. This month we start a series on 68000 assembly language programming, beginning with the basics. We are looking at other possibilities as well. Suggestions are welcome.* —*Ed.*

Shiny Bubbles without a double-sided drive.

In reference to issue 10, January 1987: in the graphics article **Shiny Bubbles**, the unsqueeze instructions indicate that this cannot be unsqueezed on a single-sided disk, due to size limits.

If, however, one has an extended formatter and two drives, it *is* possible. SB.D8A can be left on drive A, while ARCX.TTP is put on drive B, on an extended formatted disk. Then type in parameters: *A:SB.D8A*. This will read the file on drive A and construct the program and related file on B—which will now fit and will run quite nicely.

It sure would be a shame to have someone miss this great demo when it is really available to them *without* a double-sided drive.

Sincerely,
John L. Owens
N. Tonawanda, NY

ST survival.

I am in the market to buy a new computer (I used to be a TI99/4A orphan). I decided to go with an Atari ST, due to all the changes going on with IBM clones (can't afford a real IBM), such as the 8088, 80286 and 80386 chip technology. While I realize that something with an 8088 chip would probably satisfy my needs, I want something a little more "state of the art." I don't want to buy something, and have it outdated and unsupported in less than five years.

No sooner had I decided on the Atari 1040ST (I have only about $1,000 to spend) than I read about a possible Atari 2080ST (*COMPUTE!* magazine), plus a few terms found in the **ST-Log** section of your December 1986 issue (49) of **ANALOG Computing**. The article by D.F. Scott talked about a 68020-based TT machine and also an EST in the idea stage. He also referred to an IBM compatibility box ("3B") with 8086 compatibility (isn't the "norm" the 8088 chip?), as well as a blitter-block memory transfer chip. I had read elsewhere that the blitter chip would take over some sprite movements, and, therefore, some programs

would run faster. I take it that this chip isn't yet or may not be included on 1040STs?

What I really need to know is where the 1040ST stands in the odds for survivability. I don't want to drop $1,000 on a 1040ST, then have a newer Atari machine (2080ST, TT, EST, etc.) come along and make the 1040 obsolete so quickly. I understand that there are risks in this computer buying game, but I am stationed in Japan (USAF) and the only way I get a pulse on the Atari market is through magazines (so far, only yours and *COMPUTE!*) I would greatly appreciate any information you can provide, so that my new purchase will be a well-researched one.

I will be in Japan for two more years, and I can't wait to get back to the U.S. with a new computer, and access some BBSs and networks such as CompuServe, Delphi, etc. I just hope anything I purchase will still be around in force.

Sincerely,
Rodney L. Billings
APO San Francisco, CA

*Since December, Atari's plans for new computer releases have become clearer (see this month's **Status report**). The new machines will have the blitter chip, more memory and other features, but underneath everything will evidently be the same 68000 processor and essentially the same TOS and GEM. Most of the radically different additions Atari is contemplating seem to involve networking and similar high-end business and professional configurations, while the blitter chip and extra memory should be installable on 1040s (and 520s, for that matter). So in the next couple of years there appears to be little danger that the 1040 will become obsolete, in the sense of software and hardware support for it ending. At the very worst, it will be too compatible to die.*

All indications are that there will be an easy way of installing blitter chips in 520s and 1040s.

The Intel 8088 and 8086 chips are the same thing, except that the 8088 performs memory and I/O operations via an 8-bit bus, and thus more slowly than the 8086, which is a full 16-bit chip. But as far as programs are concerned, the two chips are identical. —*Ed.*

Disks and that.

I am a new computer enthusiast. I have had my 1040ST for approximately six months. I've been reading many magazines, and have found **ST-Log** very valuable to me since it has become a separate publication from **ANALOG Computing**. You have helped and are continuing to help me learn more and more about the valuable resource I have with my computer. Thank you.

I have a number of questions for you. First is concerning the disk edition of your magazine. When I first ordered **ANALOG Computing**, I was under the impression that the ST supplement disk was *with* the **ANALOG** 5¼-inch disk. I am wondering if you are now shipping 3½-inch disks to your disk subscribers. If you are shipping 3½-inch disks, I would be interested in considering a subscription. However, my subscription to the magazine does not run out until sometime in 1988. What would it cost me and how could I order the disk alone, seeing that I already have a magazine subscription?

As a new Atari enthusiast I would be interested in having a listing in your magazine of user group shows. If you would be willing to put this list monthly in your magazine, those of us who are new to the Atari family would have the opportunity to attend some of these shows. I would appreciate it if you could find a place to publish the dates and locations of these shows in your magazine.

On page 4 of the February 1987 issue (**ST-Log** 11) you have a brief description of Delphi. I am wondering if there is any other literature concerning this service. I am currently in the process of purchasing a modem and would like further information if it is available. I am especially interested in cost and service availability.

Finally, I am intrigued by the map that I find on the cover of your February **ST-Log**. What program did you use to produce it? Thanks again for your magazine. I look forward to a response to these questions.

Sincerely,
William T. Scott, Jr.
Williamsport, PA

***ST-Log** disks can be obtained for $9.50 per disk, plus $1.50 for shipping and handling per order (not per disk) from: **ST-Log**, P.O. Box 23, Worcester, MA 01603.*

News about user events can be found in the Atari Fairs column.

Information about Delphi can be obtained by calling Delphi at their toll-free number 800-544-4005 (in Massachusetts, 617-491-3393).

*The map of Massachusetts seen on the cover of the February **ST-Log** was simply drawn (along with the rest of the cover) by our cartographer-at-large, Lee Pappas, using **DEGAS Elite**. He was prevented from moving Worcester to Cape Cod.*—*Ed.*

Toward Atari's future.

It was very refreshing—in your January issue of **ANALOG Computing**—to read the letter from Robert Newman regarding the appalling condition of Atari's "service network."

. . .The attitude of Atari toward its dealers is not enlightened. The "Corp." thrives on inconsistencies: Jack Tramiel would have it that dealers complain because they haven't what it takes to be successful in the market, that Atari's task is not to "support"

the dealers. Were Atari to baldly state that they don't care about anything but money, then it wouldn't matter (not in the same way), but they would have it in the same breath that the customer has priority, and that Atari is somehow prepared to "serve" the end user. How they expect to accomplish this without a standard and a service support system through Atari dealers is quite beyond my ability to imagine.

If Jack Tramiel and company wish to realize their dreams, then they will have to pay more attention to what their *customers* are telling them, *through* their dealers. "Everything for next to nothing" and "price/power ratio nonpareil" will sell some computers, but will not convince the majority—those who expect service to come *with* the machine. For all that I have little use for the PC company and phenomenon, it must be said that they originally pursued the right path. A guarantee to service ought not be something that dealers have to fight for, but, rather, something to which a dealer would have to contract *before* being allowed to purchase equipment for resale. Mr. Newman said it well: it doesn't matter what kind of goodies you offer in "the box," businesses will not generally look at it unless the after-sales structure is there. What might appear to Atari as an "inconvenience," when a person has to wait weeks and months for repairs to be effected, is to that user and the dealer attempting to serve him a matter of great frustration and annoyance. It does not *have to be* that way; that is what enrages, and that is what finally causes disillusionment, bad feelings, bad "word of mouth" and lost sales. In the long run, this amateurism on Atari's part will guarantee that they never run First.

In the helter-skelter rush to get cash flowing, Atari has made an error directly related to the problems above. It appears that anyone can "sign up" to sell Atari products, as long as they have enough capital or credit to purchase stock. A lot of these players are bandwagon riders, interested only in the main chance: the dollars. Perhaps it's from these folks that Atari Corp. has gotten the impression that *everyone* is out only for "the bucks." It seems not to have occurred to Atari that some of the resellers might have an actual *interest* in the product, and a corollary interest in the users of the product. Those who would serve Atari best are those who suffer the most. Those who care about service are those who must beat their heads against Atari's non-policy. Those who care about more than "pumping out the boxes" are those who suffer most from the price-cutting which inevitably follows when the bit-players get their hands product. The short-term vision is no doubt serving the main purpose—improving the cash flow—but, in the long run, will guarantee that Atari is seen as a reincarnation of Commodore and every other company which has not heeded the cries of dealers, to the detriment of consumers. In the long term, those dealers who were committed, concerned and sincerely interested in the Atari product will have thrown in the towel, and Atari will have to rebuild the whole structure—whenever they decide to Professionalize.

Sincerely,
George Steppo
Calgary, Alberta, Canada

The Tramiel Atari is a young corporation still, progressing and changing as time passes. Neil Harris, Director of Marketing Communications, gave us assurance at the January CES that Atari does take note of consumer opinion, and is working to upgrade service (as mentioned in this issue's ***Status report****, page 73).* —*Ed.*

ST Nightmare Repair

What to do when your ST's printer interface needs help.

by Matthew J.W. Ratcliff

This article is a tutorial on how to repair the printer interface in your ST, with a few *caveats* on how *not* to. Those of you with a lot of electronics kit building experience can learn from my mistakes if you try doing the same thing; those with less experience—or more respect for Murphy's Law—will learn the need for service centers. Either way, my pain should turn out to be your gain. And now. . .the story.

I have both an Atari 520ST and 130XE (8-bit) computer. Normally, I use an old 825 printer with the XE and my trusty Gemini 10X with the ST. I was testing the new **P:R: Connection** printer interface for the XE and doing quite a bit of cable juggling. Finally, I prepared to put **P:R: Connection** through an acid test by hooking it up to the Gemini and dumping some **Print Shop** graphics. Meanwhile, the 520ST (with the Gemini connected) was fired up with **1st Word**, ready for me to finish my product review. In too big a hurry to turn anything off, I simply disconnected the Gemini from the ST and connected it to the **P:R:** interface. It was just too much trouble to save the file and turn the ST off before making the switch.

All seemed well, until I started dumping the **Print Shop** graphic. The Gemini printed nothing but garbage. I *knew* the **P:R:** interface had to be defective. So the old reliable 850 interface was put back into service, but the printer continued to malfunction.

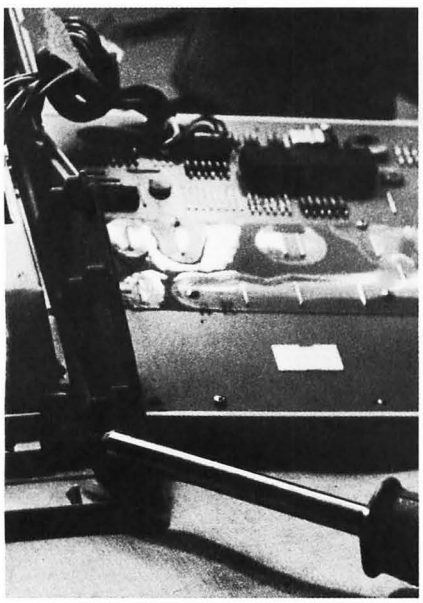
It finally dawned on me that I must have blasted some interface electronics in the Gemini printer itself. The **P:R:** was working fine on the old 825, so it had to be the Gemini. Apparently, a couple of pins had gotten shorted when I pulled the Centronics connector from the rear of the Gemini. With the power on, this had proven fatal to a 74LS374 octal latch (buffer) chip. Well . . .this wasn't the first time this had happened, so it didn't take me long to get the chip replaced and finish writing my review. The Gemini, now working fine with the XE, was reconnected to the ST (with power *off* this time!) so I could print out my article. Argh. The Gemini printed garbage again. Not only had I damaged the Gemini, the printer interface in the ST was a goner, too!

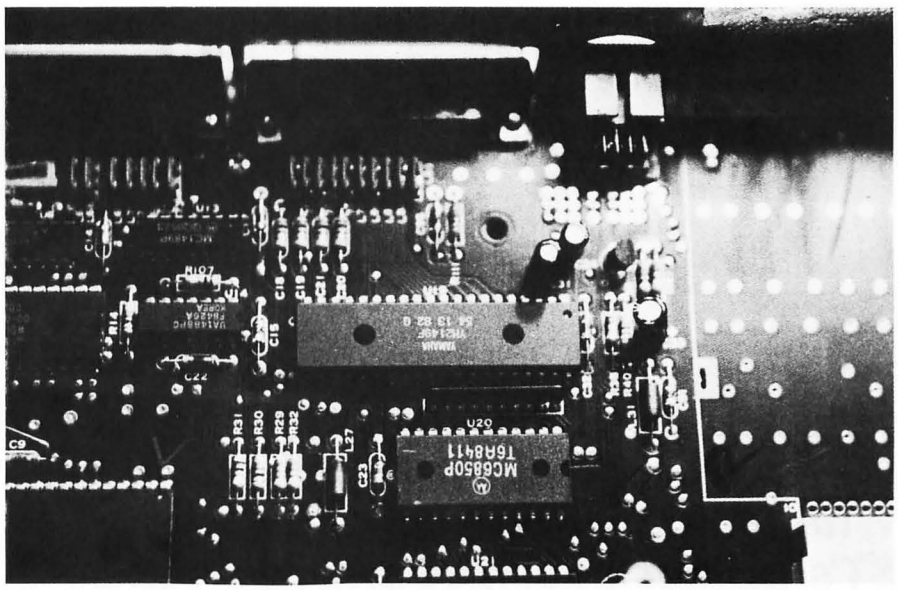
This disaster shouldn't have happened —even though I *am* an electrical engineer by profession. But I have an even better excuse. Many of us 8-bit diehards have become accustomed to Atari's well-protected architecture. During the five years I have been using 8-bit Atari computers, I have connected and disconnected devices from a hot (i.e., powered-on) serial interface bus many times, without damaging anything. All the joystick port pins are current-limit protected. The cartridge port is protected on the 400/800 systems with a power interlock door. And, from the times my two-year-old has ripped program cartridges out of a powered-up 800XL, I know that the XL/XE systems are adequately protected, as well. These superbly designed machines have lulled me into a false sense of security. It isn't that the ST is poorly designed —it just has less of the "goof proofing" hardware. Production costs have to be minimized in order to provide "Power without the Price." Common sense in using the system will provide enough protection. A good rule of thumb is to always disconnect AC power from devices at *both* ends of any cable you wish to move.

I had at first assumed that the 68901 (multifunction peripheral) chip in my ST was the casualty. But, after asking a few people on Delphi, I found that it was probably the *sound* chip that had been zapped. Not only is this chip used to produce sound on the ST, it also serves as a PIA (Peripheral Interface Adapter) with two general-purpose 8-bit I/O ports. One port is used for the eight data lines to the printer's Centronics interface. The other is used for drive selection and generating the printer strobe line. The 68901 chip only handles the returning busy line, which can be used as an interrupt. (Notice I said *I/O* ports. These registers can be reprogrammed for parallel input, too. The **Hippovision** device uses the printer interface to send digitized video *into* the ST.) The chip in my ST was a Yamaha, YM-2149. An equivalent replacement is made by General Instruments, AY-3-8910.

Thanks to the helpful folks on Delphi, I found the AY-3-8910 available from JDR Microdevices, for only $12.95. I had a huge backlog of reviews, so I ordered one chip pronto and even paid extra for shipping via UPS air. After getting the youngsters to bed late that evening, I finally got to spread the

Removing the three screws at the front of the metal RFI shield.





Here's the *dead* chip, but *no socket*. I had assumed it would be socketed. I wasn't prepared, and hastily finished the job — only to botch the works.

ST's innards all over the dining room table. I did it as follows, taking a picture of each step.

I removed the six Phillips screws from the bottom of the ST and set them aside. Then I flipped the computer over and removed the top plastic cover. Next I gently rotated the keyboard over to the right, turned it face down, and unplugged it from the circuit board. Some copper tape had to be pulled up from the top metal shield. I removed three screws from the front of the shield, and untwisted eleven retaining tabs: four at the front, two on each side and three in the rear. Finally, I lifted off the top shield to reveal the dead circuit board beneath.

I expected to be able simply to pop the old sound chip from its socket and plug in the new one. I found it near the Centronics printer connector, at location U19 on the circuit board, but there was no socket!

The 40-pin chip was soldered directly to the board. So I had to dig deeper. There were three more Phillips retaining screws near the connectors in the rear. Once they were removed, I was able to lift out the mother board and metal shield from the lower plastic case. I took the board off the shield for a close-up view of both sides. I didn't have a desoldering iron or a 40-pin socket on hand. All the Radio Shacks were closed, and common sense was telling me to call it a night and try to finish it the next day. However. . .

A phone call to a rather sleepy friend turned up a desoldering iron in a few minutes. I quickly desoldered the dead chip and set it aside. Next I cleaned the circuit board. Again, better judgement (and my sleepy friend) told me to put in a socket, in case the new chip didn't work. However, the ST was needed urgently, so I forged ahead. The new chip was soldered in. Confidently, I reassembled the ST and powered it on. Now nothing worked! The drive would spin, but it wouldn't select (no busy light). With TOS in ROM, I could get to the desktop, but I couldn't print anything. and after re-disassembling the ST, recleaning the circuit board and chip (with rubbing alcohol and cotton swabs), and reputting the mess together, it was still dead as a doornail. End of one night's work.

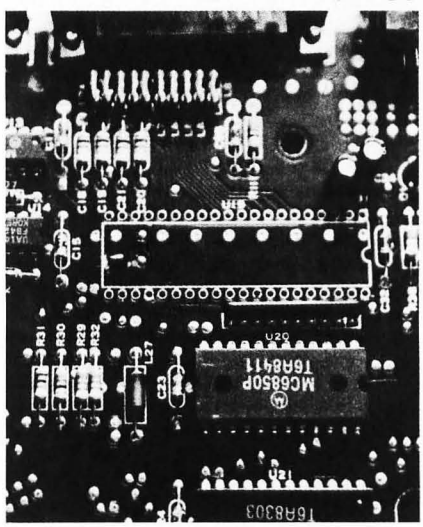
Next day I spent over $50.00 on a low wattage electronics soldering iron, a good desoldering iron and a 40-pin socket. The new (but apparently dead) chip was desoldered and removed. I cleaned the board yet again and this time noticed a small broken piece of etch in one of the solder pads. I hadn't noticed it after pulling out the original bad chip, even though it turned up on my photographs!

This is what had happened: I hadn't desoldered the first chip completely enough, so one of the pads broke when the chip was removed, and when the new chip was soldered in, the pin hadn't made contact with the broken pad.

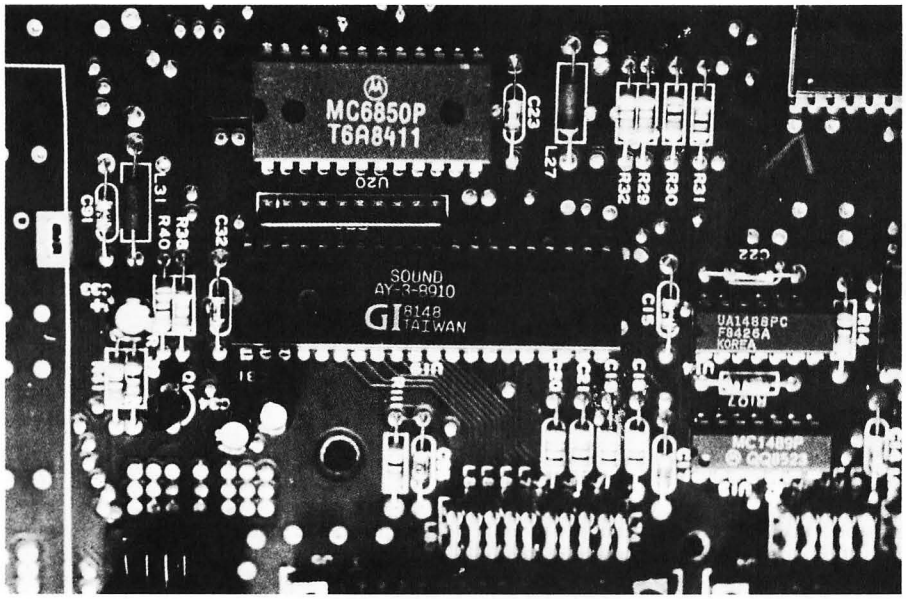
I installed the 40-pin socket and patched the broken pad with a small piece of wire, soldered to the bottom side of the board. I plugged the original defective chip into the socket and put just enough of the ST back together to test it. The drive spun to life! The new chip was tested next, and now the printer was working again.

The moral of this story is: keep your shirt on. No matter how experienced you are, it's easy to go wrong if you rush the task. Of course, "an ounce of prevention" wouldn't hurt either. If you must juggle a printer between two computers, do so with *power off*. Or, better yet, get a printer switch box. As a result of this little fiasco, I designed and built my own "Master Switch," for using the Gemini on the ST or XE.

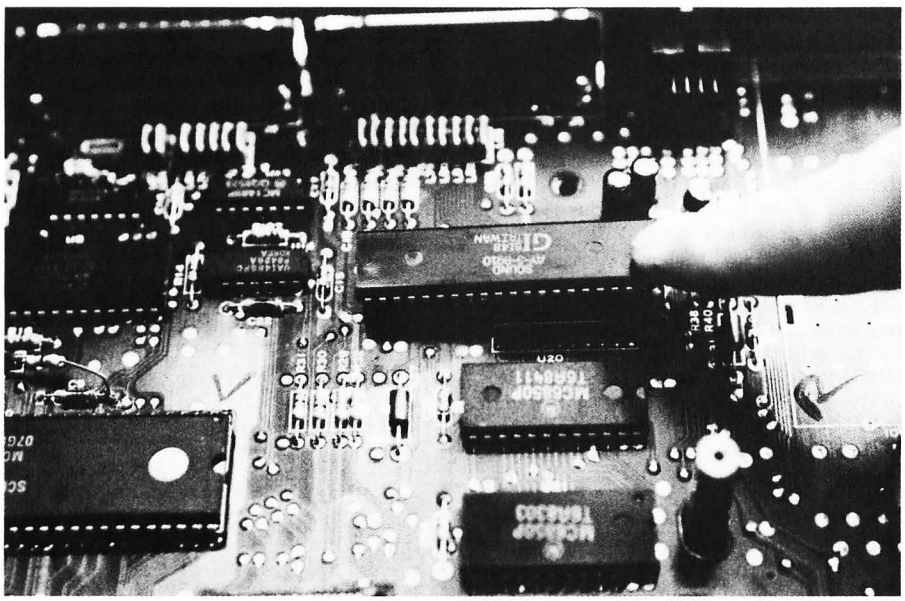
In the end, all turned out well. The computer *was* still in warranty, until I cracked the case. Remember that, if you decide to take your ST apart. If your drive's busy light won't come on, or the printer dies, or the click doesn't click anymore, it may simply



The empty socket on the ST. The damaged etch can be seen, the ninth pad from the right, in the bottom row of pads. The pad looks like a three-quarter moon. I was so busy taking pictures that I didn't inspect the board completely enough.



The gleaming new AY-3-8910 chip, soldered in place.
If you look closely, you can see the incomplete solder connection on the damaged pad.



The final step, inserting the AY-3-8910 in a socket, as I should have done in the *first* place!

be a bad chip. If your RS232 interface is behaving strangely, check the DTR and CTS output handshake lines—these are also controlled by this chip. With the proper tools and expertise, you can replace the chip yourself.

You'll save some money, *if* you do it right. Otherwise, you'll learn how I felt after I'd installed the new chip and found that now *everything* was broken. *If* it's broken, and if you're in doubt how to fix it, don't—take it to a service center. It's worth it. //

SCHEDULED ATARI FAIRS

APRIL 10 & 11, 1987
SALT LAKE CITY, UTAH
Expo Mart Salt Lake
230 W. 200 South, SLC
Contact Bob Christopolus at
(801) 486-8009 for more information

APRIL 25 & 26, 1987
BUFFALO, NEW YORK
Buffalo Convention Center
Convention Center Plaza, Downtown
For more information, contact:
Great Lakes Atari Expo,
P.O. Box 294
Buffalo, NY 14216

UPCOMING SHOWS

Additional information will be published in later issues.

MAY 1987
DALLAS, TEXAS

MAY 1987
SEATTLE, WASHINGTON

JULY 1987
CHICAGO, ILLINOIS

AUGUST 1987
DETROIT, MICHIGAN

SEPTEMBER 1987
GLENDALE, CALIFORNIA

OCTOBER 1987
WORCESTER, MASSACHUSETTS

NOT YET SCHEDULED

NEW YORK CITY, NEW YORK
ST. LOUIS, MISSOURI

Database Delphi

ST matters discussed in the Atari Users' Group SIG on Delphi.

by Matthew J.W. Ratcliff

This month, we'll look at some "Mouse-ka-mania"—problems and fixes for reading the ST mouse. We'll also take a closer look at Atari's GDOS.

Mouse-ka-mania.

11910 21-NOV 17:45 ST Programs
From: CFJ (Charles F. Johnson)
To: DLM (Daniel L. Moore)

To make matters worse with my mouse button problems, even when I use the AES graf_mkstate call exclusively, both to wait for a button press and a release, the button press is passed through to form_do somehow. I know it *does* wait for a release, but form_do still thinks the button has been pressed and insists on acting upon it. Here's the solution I devised:

Immediately after checking for the button release with graf_mkstate, call evnt_multi to wait for both a mouse button event and a timer wait of 1 millisecond. (I'm not sure if the button event is required, but stuck it in to be safe.) This seems to clear out the AES event message buffer, and form_do no longer receives extraneous button presses.

11912 21-NOV 19:23
From: DLM
To: CFJ

There is no 100-percent reliable method of mixing VDI and AES input calls, other than *not* mixing them.

Actually, it is a good idea to use evnt_multi with a timer loop to monitor the mouse. Depending on where you are in the system, and the state of wind_update calls, moving the mouse to the menu bar can lock up the system. It occurs most often when using evnt_button, but can be generated outside of that.

11962 22-NOV 14:43
From: CFJ
To: DLM

Yes, and have you noticed that the graf_rubberbox call is pretty buggy, too? If you start drawing the rubber box at a vertical coordinate of less than *20* or so, you can lock up the system very nicely. I think I'll write my own.

GDOS.

Recently, Atari Corp. announced that GDOS would be made available to copyright software developers, for release with their products, for an annual fee of $500.00. Some software developers have expressed a great deal of concern and anger over this, having expected it to be something you would get with the system, like Atari DOS 2.5 with a new 1050 disk drive on the 8-bits.

Some of the features and drawbacks of GDOS were brought out in this thread on Delphi. **DEGAS Elite** uses GDOS to load fonts, but still uses the exact same printer drivers as does **DEGAS**. **Publishing Partner** from SoftLogik was originally going to run with GDOS support, but, due to GDOS's lack of power (not enough flexibility with font controls, primarily) and quality printer drivers, it was dropped in favor of custom handlers. I have heard that *most* of the problems reported with **DEGAS Elite** can be traced directly back to GDOS, not to the program.

While GDOS itself requires very little RAM (8K), its drivers and fonts can be quite memory hungry, anywhere from 100K to 1 meg, depending on the number of fonts and the type of printer driver loaded, according to DLM. I proposed that GDOS be released in cartridge form, then sold directly to the end user. This would eliminate the need for copyright developers' paying the annual fee, and GDOS would simply become a "required peripheral" of sorts for some software. We'll pick up the thread there.

12018 23-NOV 15:14
From: DLM
To: MATRAT (Matthew J.W. Ratcliff)

That is a pretty good idea, and it might be do-able to patch the vectors to the ROM space. Putting it all in a cart would cut down memory usage to 32K (at least for the drivers I know about), which is used as scratch memory to compose printer data.

The biggest problem I can see with the carts is the variety needed, since a custom driver is needed for every printer, along with custom fonts for that printer driver. That means lots of different carts to cover all the printers. Of course, they could do what they did with the standard machine, just have an Epson and compatible GDOS cartridge only.

Conference news and Delphi tips.

I had an ST graphics conference with Tom Hudson slated for last month, but it was postponed due to the hectic holiday season. You can expect news on this, and possibly another conference with the authors of **Publishing Partner**, in the coming months.

Did you make a typographical error on the previous line? Do you hate to go back and edit it later? Did you even know that you could? Now you can edit your FORUM messages (doesn't work in EMAIL) while entering them. Just enter a */EDIT* on a line by itself, and you will temporarily be bumped into the editor of your choice (I prefer OLDIE, similar to the editor of the FoReM BBS).

Once you've made your corrections, use a */E* to exit the OLDIE editor and pick up where you left off entering the message. Use */HELP* a lot if you get lost in the edit mode. ///

Matthew Ratcliff is an electrical engineer in St. Louis, Missouri. When not using his spare time to write articles, he's president of ACE St. Louis and a remote SYSOP on Gateway City BBS, (314) 647-3290.

ST news

Disk drive cleaning kit

A new 3½-inch disk drive cleaning kit from 3M allows computer users to remove error-causing debris that accumulates on read/write heads. Regular cleaning is said to prevent accelerated wear and deterioration of micro floppies due to contamination abrasion.

Each kit contains two cleaning micro-diskettes, two half-ounce bottles of disk drive cleaning solution (enough for thirty cleanings) and an instruction booklet. The cleaning disk is designed for either single- or double-headed drives, and has been developed for just the 3½-inch floppy market. Fears of cleaning can be alleviated by the knowledge that the cleaning fabric inside this kit is less abrasive than the industry standard for actual recording media.

Retails at $19.95, from 3M, P.O. Box 33600, St. Paul, MN 55133-3600. Reader Service #140.

Innovative music software

Dr. T's new **Keyboard Controlled Sequencer ST** provides the composer with an automated 48-track "tape recorder" simulation. Using the mouse to control all major functions, the status of 36 tracks can be displayed at once on the screen.

A continuous overdub function allows patterns to be built up layer by layer, in real time, with no stopping. In addition, tracks may be muted, unmuted or soloed. Changing tempo, recording of controllers and aftertouch can be switched on or off.

Suggested retail is $195.00. Dr. T's Music Software, 66 Louise Road, Chestnut Hill, MA 02167 — (617) 244-6954. Reader Service #141.

Recent Regent release

Regent Software calls their new **Inventory Manager** a comprehensive inventory control system. With custom reporting and sales projections based on sales history, this package can handle up to 40,000 items and is ideal for a small businesses and retail operations.

Purchasing projections based on seven parameters—definable by the user—custom report generating, instant access to any item on the inventory list, overstock and obsolescence reports are some of its features. In addition, as many as 254 vendors can be maintained.

Works with both monochrome and color monitors; complete with manual and examples. Priced at $79.95. Regent Software, 7131 Owensmouth, Suite 45A, Canoga Park, CA 91303 — (818) 882-2800. Reader Service #123.

Other news

☑After some legal difficulty with Broderbund Software, Unison World, Inc has re-released its **PrintMaster** program as **PrintMaster Plus**.

Using this software package, you can create customized signs, calendars, banners, stationery and greeting cards. Various design options permit the user to select type fonts, graphics designs and borders. Used in conjunction with **Art Gallery I** or **II**, also by Unison, over 280 additional graphics are accessible. Reader Service #143.

☑**STAccounts** is an integrated accounting package, with full accounts receivable, accounts payable, inventory control, general ledger and report generator. On-line help screens provide assistance whenever you need it, and the package is designed to work within the GEM environment, using the mouse.

Batch invoicing and order entry for both sales and purchases are handled by the system, and printed copy can be altered, amended or reconfigured. This package works on a 520ST or a 1040ST, high or medium resolution.

Cost $249.95. Owners of the earlier version may upgrade at no extra cost, by contacting ISD Marketing, Inc., 2651 John Street, Unit 3, Markham, Ontario, Canada L3R 2W5 — (416) 479-1880. Reader Service #142.

Make your words perfect

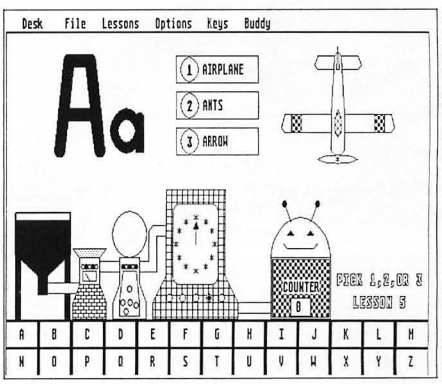
WordPerfect Corporation has introduced **WordPerfect** for the Atari ST. Supporting file compatibility with WordPerfect 4.1 for the IBM PC (and the Atari PC), a fast 115,000-word dictionary/spelling checker and built-in thesaurus, this program covers many needs. GEM is fully supported, and virtually all functions can be accessed with either the mouse or the keyboard. Desk accessories are also available inside **WordPerfect**.

Virtual memory assures that data will flow onto the disk if RAM memory is full. Also supported are math, merge and sort options. Text columns, various "undeletes," table of contents, index generation, paragraph/outline numbering and footnotes/endnotes are all at your fingertips.

With a complete manual/tutorial, **WordPerfect** sells for $395.00. WordPerfect Corporation, 288 W. Center Street, Orem, UT 84057 — (801) 225-5000. Reader Service #143.

Alpha-Bytes

A sequential and random alphabet review and quizzing program for children 2 to 6 years old, **ST Alpha-Bytes** utilizes over eighty different graphics to entertain and educate.



Both upper- and lowercase letters are displayed, along with words and pictures. Time limits can be selected, as well as a help feature.

Other options include: the ability to store and print responses, the use of mouse or keyboard and sound. There is a 20-page manual to assist adults in setting up parameters for the children.

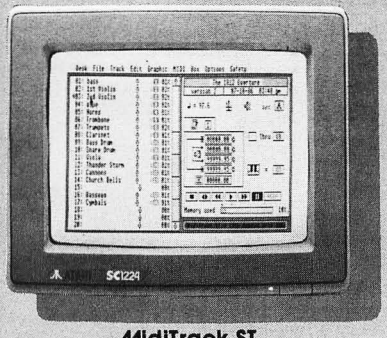
For both color and monochrome systems. Priced at $29.95, from Mission Softs, P.O. Box 3916, Seal Beach, CA 90740. Reader Service #144.

Tapeless Technology™

Inspired MIDI Software from

Hybrid Arts, Inc.™

Nobody knows MIDI on the ATARI like Hybrid Arts. We have been the undisputed experts since MIDI started, and now we've got all the bases covered.

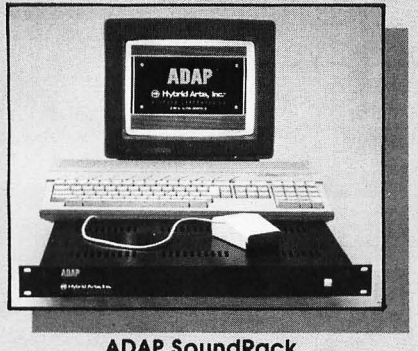


MidiTrack ST

MIDI Recorders: With several different professional quality systems to choose from, Hybrid Arts has a MIDI Recorder for every musician and any budget from hobbist to broadcast professional. Starting with **EZ-Track**, the 20 track polyphonic recorder with editing features like quantizing and instant transpose. **MidiTrack ST** has a full range of advanced editing features including an auto patch injection system and FSK, Roland and TTL sync to tape for studio use. And **MidiTrack Professional** comes complete with our **SMPTEMate** box. SMPTE specs include SMPTE read/write, drop frame, non drop frame, 24, 25, or 30 fps, jam sync, both reading and writing of user bit data. Resolution is at the bit level.

CZ-Android

Patch Librarians: A complete line of products are available for sound patch libraries. **Genpatch** is designed to work with all MIDI keyboards regardless of manufacturer. **DX-Android**, the highly acclaimed editor/librarian program for the Yamaha DX-7 and TX series. This was the first product of its kind to offer both graphic and numeric editing along with the famous android functions that create new musically useful sounds instantaneously. **CZ-Android** is the CASIO verison that works with all of the CZ keyboard series including the new CZ-1 and CZ-230S.



ADAP SoundRack

Sampling and SMPTE hardware: The revolutionary **ADAP SoundRack** (Analog to Digital Audio Processor) takes full advantage of the speed and memory of the ATARI 1040ST. The "ADAP" is a hardware/software peripheral for the ST that allows music and broadcast professionals to sample 20 seconds at 44.1 khz with 16 bit resolution. Full MIDI compatability is implemented. System is upgradeable to direct to hard disk stereo digital mastering and SMPTE interlock.

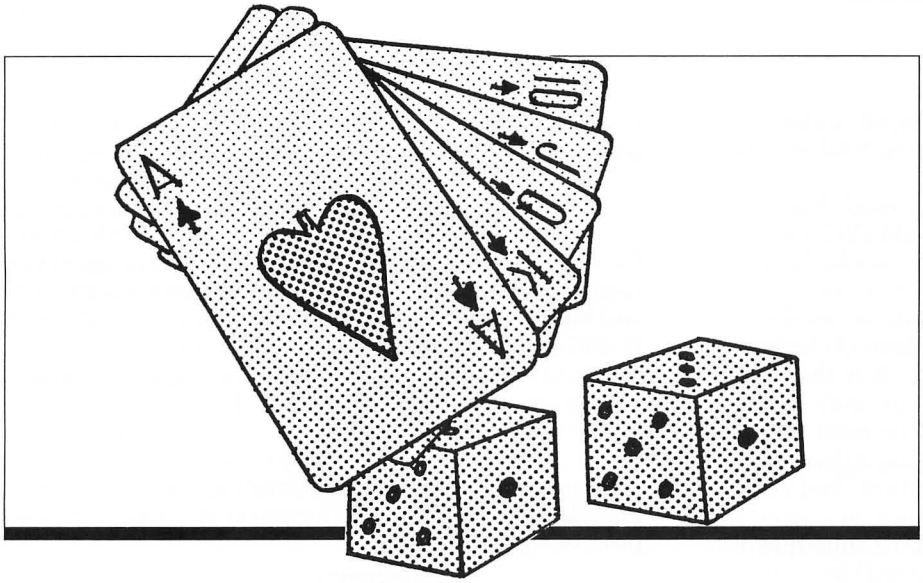
The MidiMate software series for the ATARI 8 bit computers: For the musician on a limited budget but still needing professional quality, Hybrid Arts has a complete line of studio-tested software. **MidiTrack III** is a 16 track MIDI recorder, synchronizer and remote controller for the ATARI130XE. Standard features include 10,500 note capacity, looping, copy and paste editing, programmable tempo, track combining and uncombining. **MMS** is a sophisticated monophonic single step entry sequencer. The operator can enter musical notes from either the computer keyboard or the MIDI synth keyboard. 99 tracks are available to work with. **GenPatch** is a generic patch librarian system for all MIDI keyboards. **DX-Editor** is a numeric editor and librarian for the YAMAHA DX-7 and TX. **Oasis** is our low cost Visual Editing System for the Ensoniq Mirage keyboard sampler. Features include full MASOS implementation, 8 levels of magnification, redrawing of wave forms with high resolution graphics. **MidiCom** is a MIDI telecommunications program designed for easy use.

MidiTrack III

Call or write for a free catalog and newsletter which includes a complete listing of products and pricing.

Hybrid Arts, Inc.™

11920 W. Olympic Blvd., LA, Ca. 90064
Computer BBS: (213) 826-4288 Main Office: (213) 826-3777
Telex: 4931231 (HYBRDUI)



Poker Dice

by Mike McCarthy

Poker Dice is a totally mouse-controlled computer adaptation of a popular board game, in which players roll dice and try to make poker hands out of the numbers rolled. **Poker Dice** can be played by one to four players, in high or medium resolution.

Getting started.

First you must load ST BASIC. If you load TOS from disk, you will need to disable buffered graphics in order to free up enough memory to enter **Poker Dice**. Once you have typed in the program and saved a copy to disk, you're ready to run it. Select the RUN option on the run menu.

The layout.

The first screen displayed is used to select the number of players. Just position the mouse cursor on the die with the number corresponding to the number of players desired, then click the left button once. The dice will be erased and the game screen will be displayed. Note that BASIC is very slow in responding to the mouse button; you'll have to hold the button down longer than you normally would. Do *not* try to use the scroll bars on the window border! If you do, the display will be scrambled and you'll have to restart the game.

All game functions are performed by pointing with the mouse and clicking with the left button. In the upper left corner of the screen is the ROLL command. Pointing and clicking on this area will roll the dice. The number below ROLL indicates how many times a player has already rolled the dice during that turn (each player gets three rolls per turn). Below this is the UNDO command. More about this feature later. In the upper right of the screen is the NEXT command. Clicking here will start a new game (regardless of where you are in the current one). Under NEXT is the QUIT command, which allows you to terminate a game at any time and return to BASIC.

The rest of the screen is divided into five areas. The ROLL area is at top center; here the dice are displayed when they are ROLLed. To the far right is the HOLD area. If a player does not want to roll certain dice during a turn, they can be held here. The HEADER area is where the PLAYER labels are written; an arrow points to the player whose turn it currently is. Player scores are displayed in the SCORE area, the large central portion of the screen. The OPTION area, containing a list of all the scoring options, is at the left.

The game.

The object of **Poker Dice** is to score more points than your opponent(s), by filling in as many scoring options as possible. This requires a certain amount of skill and a lot of luck.

You begin the game by clicking on the ROLL command. Five randomly rolled dice will be displayed in the ROLL area. Now you decide which of the scoring options can best be matched with this combination of dice. You have the option of rolling all five dice again to get a better result, or of holding some of the dice in the HOLD area and rolling only the remaining ones. For example, if you rolled two sixes and three other numbers, the sixes could be placed in the HOLD area and the remaining dice rolled again, to try to get more sixes to improve the score. To hold a die, simply point at it and click. It will be erased and redrawn in the HOLD area. It can be moved back to the ROLL area by the reverse process. Let's say that you get another six on the second roll. You would HOLD this six and roll the two remaining dice for the last roll of the turn. If no more sixes came up, then you would choose your scoring option at this point. You could choose sixes by pointing to the word *SIXES* in the OPTION area and

clicking. The score is automatically displayed under the column header (in high resolution a running total is also displayed at the bottom of the screen).

If, after you clicked on SIXES, you decided that you would rather score 3 OF A KIND, you would click on the UNDO command, and the SIXES score would be removed. You could then click on 3 OF A KIND, and a new score would be displayed. UNDO can only be used before the next player's first roll. Once you have clicked on your score (and relinquished the mouse), it is the next player's turn. The arrow will point to that person's PLAYER header, and the number of rolls will be reset to *0*.

The game continues until each player has filled in all options. You *must* fill in one option each turn, and each option can only be filled in once per player in a game; you cannot add to the score of an option on a subsequent turn. In some cases, this means that you will be forced to put a *0* in some option(s) if you were unlucky enough not to match any option, or only matched an option which already had a score assigned. Determining how best to use each roll is part of the skill of the game.

When the last player finishes his last turn, **Poker Dice** calculates the totals, adds bonuses where applicable, and identifies the player with the highest score as the winner. The total scores will be displayed on the bottom line of the screen. At that point, you can click on NEW to start a new game, or on QUIT to return to BASIC.

The scoring.

Poker Dice calculates all scores and totals for you, but you need to know how it works to choose your options correctly. For the top six options, the score is calculated by adding the face value of the dice which match the category name. For example, to score FOUR, you would add all your FOURs together. Any dice which are not FOURs are ignored. So three FOURs would yield a score of 12. Four FOURs would be 16. If you were scoring TWOs, then three TWOs would yield a score of 6, and so on. If the total of a player's top six options is 63 or more (63 is achieved by filling in each option with three dice of the required value) at the end of the game, then a bonus of 35 points is awarded.

The scores for 3 OF A KIND and 4 OF A KIND are tallied by adding the face value of all five dice. A FULL HOUSE (two of one kind and three of another) is always worth 25 points. A SHORT STRAIGHT (at least four consecutive dice, for example: 2-3-4-5) is always 30 points, a LONG STRAIGHT (five consecutive dice, for example: 2-3-4-5-6) is always 40 points. A GRAND SLAM (all five dice the same, for example: 6-6-6-6-6) is always 50 points. The CHOICE option is a safety valve. You can use it once when you haven't matched anything else. CHOICE is scored by adding the face value of all five dice.

The program.

The flow of the program is straightforward. The first section dimensions arrays and initializes all variables. Note that the program chooses between two complete sets of data statements, depending on the resolution of your monitor. The next section sets up the initial screen and prompts for the number of players. Once the number of players is set, the rest of the game screen is displayed.

The heart of **Poker Dice** is what I call the "Game Control Loop." Here, the program waits for you to click the mouse button and, based on the X- and Y-coordinates of the mouse, calls a subroutine to process the function you requested (the method for getting the mouse coordinates and button position is described in Appendix H of the ST BASIC manual). One of the nice features of ST BASIC is the ability to reference subroutines by name. This makes a program much easier to read and understand.

The remainder of the program consists of the subroutines to process requests from the Game Control Loop. The subroutine names and accompanying comments should provide enough explanation for you to understand the purpose of each.

Enhancements.

Of course, a programmer seldom considers a program finished. There is always one more feature to include. I would like to make use of the ST's sound capabilities to make the game a little more lively. Also, saving a running total of scores over several games would be nice, as well as saving the high score to disk, and maybe playing against the computer. . . But rather than never finish, I decided to leave the game as it is and see what enhancements others come up with. Whether you attempt some or just play the game, I hope you have fun in the process. ///

Mike McCarthy, with degrees in History and Computer Systems, has worked on IBM mainframe computers for ten years. He's currently a database systems analyst in South Florida. He bought his first Atari (an 800) in 1982 and has enjoyed experimenting with different programming languages since.

Listing 1.
ST BASIC listing.

```
100 ' ---- * * Poker Dice * * -------
200 '       by  Mike McCarthy
300 '         for ATARI 520 ST
400 ' -----------------------------
500 title$ = " * * Poker Dice * * "
600 option base 1
700 dim total(4,15)
800 dim droll(5):dim dhold(5)
900 dim holdy(6):dim rollx(6):dim opty
(15)
1000 dim box1(4):dim box2(4):dim box3(
4):dim box5(4)
1100 dim box1a(4):dim box1b(4):dim box
2a(4):dim box2b(4)
1200 dim rmsg(4):dim umsg(4):dim plhx(
4):dim plsx(4)
1300 dim nmsg(4):dim qmsg(4)
1400 dim selbox(4):dim holbox(4):dim o
ptbox(4)
1500 ' -system addresses-
1600 a#  = gb
1700 gintin  = peek(a# + 8 )
1800 gintout = peek(a# + 12)
1900 res  = peek(systab+0)
2000 ' -variable initialization-
2100 if res = 1 then restore RES1DATA
2200 if res = 2 then restore RES2DATA
2300 read scrw:read scrh:read rolly:re
ad holdx:read dw :read dh
```

```
2400 read rollmsgx:read undomsgx:read
newmsgx:read quitmsgx
2500 read startmsgx:read throwx:read t
otaly
2600 for i = 1 to 4:read plhx(i):next
i
2700 for i = 1 to 4:read plsx(i):next
i
2800 for i = 1 to 6:read rollx(i):next
 i
2900 for i = 1 to 6:read holdy(i):next
 i
3000 for i = 1 to 4:read box1(i):next
i
3100 for i = 1 to 4:read box1a(i):next
 i
3200 for i = 1 to 4:read box1b(i):next
 i
3300 for i = 1 to 4:read box2(i):next
i
3400 for i = 1 to 4:read box2a(i):next
 i
3500 for i = 1 to 4:read box2b(i):next
 i
3600 for i = 1 to 4:read box3(i):next
i
3700 for i = 1 to 4:read box4(i):next
i
3800 for i = 1 to 4:read box5(i):next
i
3900 for i = 1 to 4:read rmsg(i):next
i
4000 for i = 1 to 4:read umsg(i):next
i
4100 for i = 1 to 4:read nmsg(i):next
i
4200 for i = 1 to 4:read qmsg(i):next
i
4300 for i = 1 to 4:read selbox(i):nex
t i
4400 for i = 1 to 4:read holbox(i):nex
t i
4500 for i = 1 to 4:read optbox(i):nex
t i
4600 for i = 1 to 15:read opty(i):next
 i
4700 dh2  =  dh  / 5: dw2  = dw  / 5
4800 throw = 0
4900 round = 1
5000 up    = 0
5100 scored = 1
5200 for i = 1 to  5
5300 droll(i) = -1:dhold(i) = -1
5400 next i
5500 for i = 1  to 4:for j = 1 to 14
5600 total(i,j)=-1 :next j:total(i,7)=
0:total(i,15)=0:next i
5700 randomize 0
5800 ' -screen display setup-
5900 clearw 2:fullw 2
6000 gosub SETTITLE  ' window title
6100 color 1,2,1,1,2
6200 linef 1,1,scrw,1
6300 bx=box1(1):by=box1(2):bw=box1(3):
bh=box1(4):bfill=0:gosub box
6400 bx=box1a(1):by=box1a(2):bw=box1a(
3):bh=box1a(4):bfill=1:gosub box
6500 bx=box1b(1):by=box1b(2):bw=box1b(
3):bh=box1b(4):bfill=1:gosub box
6600 bx=box2(1):by=box2(2):bw=box2(3):
bh=box2(4):bfill=0:gosub box
6700 bx=box2a(1):by=box2a(2):bw=box2a(
3):bh=box2a(4):bfill=1:gosub box
6800 bx=box2b(1):by=box2b(2):bw=box2b(
3):bh=box2b(4):bfill=1:gosub box
6900 color 1,3,1,1,2
7000 bx=box3(1):by=box3(2):bw=box3(3):
bh=box3(4):bfill=1:gosub box
7100 bx=box4(1):by=box4(2):bw=box4(3):
bh=box4(4):bfill=0:gosub box
7200 bx=box5(1):by=box5(2):bw=box5(3):
bh=box5(4):bfill=0:gosub box
7300 gotoxy 0,4 : print " Ones "
7400 gotoxy 0,5 : print " Twos "
7500 gotoxy 0,6 : print " Threes "
7600 gotoxy 0,7 : print " Fours "
7700 gotoxy 0,8 : print " Fives "
7800 gotoxy 0,9 : print " Sixes "
7900 gotoxy 0,10: print " * BONUS * "
8000 gotoxy 0,11: print " 3 of a kind"
8100 gotoxy 0,12: print " 4 of a kind"
8200 gotoxy 0,13: print " Full House "
8300 gotoxy 0,14: print " Short Straig
ht "
8400 gotoxy 0,15: print " Long  Straig
ht "
8500 gotoxy 0,16: print " * Grand Slam
 * ";
8600 gotoxy 0,17: print " Choice ";
8700 if res = 1 then gotoxy 0,totaly:
print " TOTAL " ;
8800 gotoxy quitmsgx,2:print " Quit "
8900 gosub GETPLAYERS
9000 gotoxy plhx(1),3: print " Player1
 "
9100 if nplayers>1 then gotoxy plhx(2)
,3: print " Player2 "
9200 if nplayers>2 then gotoxy plhx(3)
,3: print " Player3 "
9300 if nplayers>3 then gotoxy plhx(4)
,3: print " Player4 "
```

```
9400 gotoxy rollmsgx,0:print " Roll "
9500 gotoxy undomsgx,2:print " Undo "
9600 gotoxy newmsgx,0 :print " New  "
9700 gotoxy throwx-res,1:print " #" 0
 ;
9800 gotoxy plhx(1),3:print chr$(3) ;
9900 ' -Game  Control Loop-
10000 while key  <> -1
10100 gosub MOUSECTL
10200 if key <> 1 then goto 12800
10300 if mx<qmsg(1) or my<qmsg(2) then
 goto 10600
10400 if mx>qmsg(3) or my>qmsg(4) then
 goto 10600
10500 goto 12900
10600 if mx<nmsg(1) or my<nmsg(2) then
 goto 10900
10700 if mx>nmsg(3) or my>nmsg(4) then
 goto 10900
10800 goto 1500
10900 if mx<rmsg(1) or my<rmsg(2) then
 goto 11300
11000 if mx>rmsg(3) or my>rmsg(4) then
 goto 11300
11100 gosub ROLLDICE
11200 goto 12800
11300 if mx<umsg(1) or my<umsg(2) then
 goto 11700
11400 if mx>umsg(3) or my>umsg(4) then
 goto 11700
11500 gosub UNDO
11600 goto 12800
11700 if mx<selbox(1) or my<selbox(2)
then goto 12100
11800 if mx>selbox(3) or my>selbox(4)
then goto 12100
11900 gosub SELECTDICE
12000 goto 12800
12100 if mx<holbox(1) or my<holbox(2)
then goto 12500
12200 if mx>holbox(3) or my>holbox(4)
then goto 12500
12300 gosub UNSELECTDICE
12400 goto 12800
12500 if mx<optbox(1) or my<optbox(2)
then goto 12800
12600 if mx>optbox(3) or my>optbox(4)
then goto 12800
12700 gosub SELECTOPT
12800 wend
12900 title$ = "OUTPUT"
13000 gosub SETTITLE
13100 end
13200 '---------------------------
13300 SETTITLE: ' modify window title
- courtesy of Russ Wetmore
13400 poke systab+24,1 :        ' turn
 off  window
13500 clearw 2: fullw 2
13600 poke gintin+0,peek(systab+8) : '
 OUTPUT window handle
13700 poke gintin+2,2 :         ' chan
ge name
13800 s# = gintin+4 :           ' DBL
address for long poke
13900 title$ = title$ + chr$(0) : ' as
sure zero terminator
14000 poke s#,varptr(title$) :  ' new
title
14100 gemsys(105) :             ' wind
_set AES call
14200 poke systab+24,0 :        ' redi
splay window
14300 return
14400 '---------------------------
14500 MOUSECTL: ' mouse x,y, button
14600 gemsys(79)
14700 mx = peek(gintout + 2)
14800 my = peek(gintout + 4)
14900 key = peek(gintout + 6)
15000 return
15100 ' -------------------------
15200 BOX:  ' draw box
15300 linef bx,by,bx+bw,by
15400 linef bx+bw,by,bx+bw,by+bh
15500 linef bx+bw,by+bh,bx,by+bh
15600 linef bx,by+bh,bx,by
15700 if bfill = 1 then  fill  bx+bw/2
,by+bh/2
15800 return
15900 '---------------------------
16000 ROLLDICE: ' roll the dice
16100 if scored = 0 and throw = 3 then
 return
16200 if scored = 0 then goto 17100
16300 throw = 0
16400 if up <> 0 then  gotoxy plhx(up)
,3  :print chr$(32);
16500 if up = nplayers then up = 0:rou
nd =  round + 1
16600 up = up + 1
16700 gotoxy plhx(up),3:print chr$(3)
16800 if round = 1 and up = 1 then got
o 17000
16900 gosub CLEARHOLD
17000 for i =  1 to 5:droll(i)=-1:dhol
d(i)=0 :next i
17100 scored = 0
17200 throw = throw  +  1
17300 gotoxy throwx,1:print throw
17400 gosub CLEARROLL
17500 for i = 1 to 5
17600 if droll(i) = 0 then goto 18000
17700 dnum = int(rnd*(6)) + 1
17800 dx = rollx(i) :dy = rolly
17900 gosub DICE:droll(i) = dnum
18000 next i
18100 return
18200 ' -------------------------
18300 SELECTDICE: ' for hold area
18400 dpos = 0
18500 if mx>rollx(1) and mx < rollx(2)
 then dpos  = 1
18600 if mx>rollx(2) and mx < rollx(3)
 then dpos = 2
18700 if mx>rollx(3) and mx < rollx(4)
 then dpos = 3
18800 if mx>rollx(4) and mx < rollx(5)
 then dpos = 4
18900 if mx>rollx(5) and mx < rollx(6)
 then dpos = 5
19000 if dpos = 0 then return
19100 if droll(dpos) = 0 then return
19200 ux = rollx(dpos):uy = rolly
19300 gosub UNDRAW
19400 dx=holdx:dy=holdy(dpos):dnum=dro
ll(dpos):gosub DICE
19500 dhold(dpos) = droll(dpos):droll(
dpos) = 0
19600 return
19700 ' -------------------------
19800 UNSELECTDICE: ' to roll area
19900 dpos = 0
20000 my = my - 20
20100 if my>holdy(1) and my < holdy(2)
 then dpos = 1
20200 if my>holdy(2) and my < holdy(3)
 then dpos = 2
20300 if my>holdy(3) and my < holdy(4)
 then dpos = 3
20400 if my>holdy(4) and my < holdy(5)
 then dpos = 4
20500 if my>holdy(5) and my < holdy(6)
 then dpos = 5
20600 if dpos = 0 then return
20700 if dhold(dpos) = 0 then return
```

```
20800 uy = holdy(dpos):ux = holdx:gosu
b UNDRAW
20900 dx = rollx(dpos):dy = rolly:dnum
 = dhold(dpos)
21000 gosub DICE
21100 droll(dpos) = dhold(dpos):  dhol
d(dpos) = 0
21200 return
21300 '-------------------------
21400 SELECTOPT: ' scoring option
21500 if scored = 1 then return
21600 opt = 0
21700 for i = 1 to 14
21800 if my>opty(i) and my<opty(i+1)th
en opt = i:goto 22100
21900 next i
22000 return
22100 if opt=7 then return
22200 if total(up,opt) =  -1 then goto
 22400
22300 sound 1,4,2,7,4:sound 1,0,0,0,0:
return
22400 gosub COLLECT
22500 if opt < 7 then gosub SCORETOP:g
oto 22700
22600 gosub SCOREBOTTOM
22700 lastopt = opt
22800 total(up,opt) = subtotal
22900 if opt > 7 then goto 23400
23000 if total(up,7) = -1 then total(u
p,7) = 0
23100 total(up,7) = total(up,7) + subt
otal
23200 s$ = "":if total(up,7) < 10 then
 s$ = " "
23300 gotoxy plsx(up)-res,10:print " (
"s$ total(up,7)")"
23400 total(up,15)  = total(up,15) + s
ubtotal
23500 scorey = opt+3
23600 scorex = plsx(up)
23700 scored = 1
23800 s$ = "":if total(up,opt) < 10 th
en s$ = " "
23900 gotoxy scorex,scorey:print s$ to
tal(up,opt) ;
24000 s$ = " ":if total(up,15)  < 100
then s$ = "  "
24100 if total(up,15)  < 10 then s$ =
"   "
24200 if res = 1 then gotoxy scorex-re
s,totaly:print s$ total(up,15) ;
24300 gotoxy plhx(up),3  :print chr$(3
2) ;
24400 if up = nplayers then gotoxy plh
x(1),3:print chr$(3) ;
24500 if up < nplayers then gotoxy plh
x(up+1),3:print chr$(3) ;
24600 gotoxy throwx,1:print " 0"  ;
24700 if round = 13 and up = nplayers
then goto ENDGAME
24800 return
24900 ' -------------------------
25000 COLLECT: ' sort for scoring
25100 for i = 1 to 5
25200 if droll(i) = 0 then droll(i) =
dhold(i)
25300 next i
25400 for i = 1 to  4:for j =  1 to 4
25500 if droll(j) > droll(j+1) then sw
ap droll(j), droll(j+1)
25600 next j:next i
25700 return
25800 ' -------------------------
25900 SCORETOP: ' score top options
26000 subtotal = 0
26100 for i = 1 to 5
26200 if droll(i) = opt then subtotal
= subtotal + opt
26300 next i
26400 return
26500 ' -------------------------
26600 SCOREBOTTOM: 'bottom options
26700 subtotal = 0
26800 on opt-7 goto 27500,27500,28600,
29700,29700,30800,31300
26900 if opt = 8 or  opt = 9 then goto
 27500
27000 if opt = 10 then goto 28600
27100 if opt = 11 or opt = 12 then got
o 29700
27200 if opt = 13 then goto 30800
27300 if opt = 14 then goto 31300
27400 return
27500 ' ... "3 and 4 of a kind"
27600 for i = 1 to 6
27700 k = 0
27800 for j =  1 to 5
27900 if droll(j) = i then k = k + 1
28000 next j
28100 if opt = 8 and k > 2 then goto 2
8400
28200 if opt = 9 and k > 3 then goto 2
8400
28300 next i:subtotal = 0:return
28400 for i = 1 to 5:subtotal = subtot
al + droll(i):next i
28500 return
28600 ' ... "Full House"
28700 d1 = droll(1):d1s=1:d2=0:d2s=0
28800 for i = 2 to 5
28900 if droll(i) = d1 then d1s = d1s
+ 1:goto 29300
29000 if d2=0 then d2 = droll(i):d2s =
 1:goto 29300
29100 if droll(i) = d2 then d2s = d2s
+ 1:goto 29300
29200 return
29300 next i
29400 if (d1s=3 and d2s=2) or (d1s=2 a
nd d2s=3) then subtotal = 25
29500 if d1s=5 or d2s=5 then subtotal
= 25
29600 return
29700 ' ... "Straights"
29800 d1s=1:d2s=0
29900 for i = 1 to 4
30000 k = droll(i+1) - droll(i)
30100 if k = 1 then d1s = d1s + 1
30200 if k = 0 then d2s = d2s + 1
30300 if k > 1 and d1s < 4 then d1s =
1
30400 next i
30500 if opt=11 then if d1s>=4 and d2s
<2 then subtotal = 30
30600 if opt=12 then if d1s=5  then su
btotal = 40
30700 return
30800 ' ... "Grand Slam"
30900 for i = 1 to 4
31000 if droll(i+1) - droll(i) <> 0 th
en return
31100 next i
31200 subtotal = 50:return
31300 ' ... "Choice"
31400 for i = 1 to 5
31500 subtotal  = subtotal + droll(i)
31600 next i
31700 return
31800 ' -------------------------
31900 DICE: ' draw dice picture
32000 if dnum = -1 then return
32100 dotsize = 2
32200 color 1,1,1,1,1
```

```
32300 linef dx,dy,dx+dw ,dy
32400 linef dx+dw,dy,dx+dw,dy+dh
32500 linef dx+dw,dy+dh,dx,dy+dh
32600 linef dx,dy+dh,dx,dy
32700 linef dx,dy,dx+dw2,dy -dh2
32800 linef dx+dw2,dy-dh2,dx+dw+dw2,dy
-dh2
32900 linef dx+dw2+dw,dy-dh2, dx+dw,dy
33000 linef dx+dw2+dw,dy-dh2,dx+dw2+dw
,dy+dh-dh2
33100 linef dx+dw2+dw,dy+dh-dh2,dx+dw,
dy+dh
33200 color 1,1,1,4,2
33300 fill dx+dw+dw2/2,dy+dh/2
33400 color 1,1,1,0,0
33500 on dnum goto 33700,33900,34200,3
4600,35100,35700
33600 return
33700 pcircle dx+dw/2,dy+dh/2,dotsize
33800 return
33900 pcircle dx+dw/4,dy+dh/4,dotsize
34000 pcircle dx+dw-dw/4,dy+dh-dh/4,do
tsize
34100 return
34200 pcircle dx+dw/4,dy+dh/4,dotsize
34300 pcircle dx+dw/2,dy+dh/2,dotsize
34400 pcircle dx+dw-dw/4,dy+dh-dh/4,do
tsize
34500 return
34600 pcircle dx+dw/4,dy+dh/4,dotsize
34700 pcircle dx+dw-dw/4,dy +dh/4,dots
ize
34800 pcircle dx+dw/4,dy+dh-dh/4,dotsi
ze
34900 pcircle dx+dw-dw/4,dy+dh-dh/4,do
tsize
35000 return
35100 pcircle dx+dw/4,dy+dh/4,dotsize
35200 pcircle dx+dw-dw/4,dy +dh/4,dots
ize
35300 pcircle dx+dw/4,dy+dh-dh/4,dotsi
ze
35400 pcircle dx+dw-dw/4,dy+dh-dh/4,do
tsize
35500 pcircle dx+dw/2,dy+dh/2,dotsize
35600 return
35700 pcircle dx+dw/4,dy+dh/4,dotsize
35800 pcircle dx+dw/2,dy +dh/4,dotsize
35900 pcircle dx+dw-dw/4,dy +dh/4,dots
ize
36000 pcircle dx+dw/4,dy+dh-dh/4,dotsi
ze
36100 pcircle dx+dw/2,dy+dh-dh/4,dotsi
ze
36200 pcircle dx+dw-dw/4,dy+dh-dh/4,do
tsize
36300 return
36400 ' ---------------------------
36500 UNDRAW:   '  erase dice
36600 color 1,0,0,0,0
36700 uy =  uy - dh2
36800 uh = dh +  dh2
36900 ux2 = ux+dw+dw2
```

```
37000 for u = 0 to uh:linef ux,uy+u,ux
2,uy+u:next u
37100 return
37200 ' --------------------------
37300 CLEARROLL:  ' clear roll area
37400 color 1,0,0,0,0
37500 ux1 = rollx(1):ux2 = rollx(5) +
dw + dw2
37600 uy1 = rolly - dh2
37700 for i = 0 to dh+dh2 :linef ux1,u
y1+i,ux2,uy1+i:next i
37800 return
37900 ' --------------------------
38000 CLEARHOLD:  ' clear hold area
38100 color 1,0,0,0,0
38200 rx1 = rollx(1):rx2 = rollx(5) +
dw + dw2
38300 ry1 = rolly - dh2
38400 ux1 = holdx
38500 uy1 = holdy(1) - dh2:uy2 = holdy
(5) + dh
38600 for i = dw+dw2 to 0 step -1
38700 linef ux1+i,uy1,ux1+i,uy2'linef
rx1,ry1+(dh+dh2-i),rx2,ry1+(dh+dh2-i)
38800 next i
38900 return
39000 ' --------------------------
39100 UNDO: ' take back a score
39200 if scored = 0 then return
39300 nextup = up + 1:if nextup > npla
yers then nextup = 1
39400 gotoxy plhx(nextup),3:print " "
;
39500 gotoxy plhx(up),3:print chr$(3)
;
39600 opt = lastopt:if total(up,opt) =
 -1 then return
39700 subtotal = total(up,opt):total(u
p,opt) = -1
39800 total(up,15) = total(up,15) -  s
ubtotal
39900 gotoxy plsx(up),3+opt: print "
 " ;
40000 if opt > 7 then goto 40500
40100 total(up,7) = total(up,7) - subt
otal
40200 gotoxy plsx(up)-res,10:print "
    "
40300 s$ = "":if total(up,7) < 10 then
 s$ = " "
40400 gotoxy plsx(up)-res,10:print " (
"s$ total(up,7)")"
40500 if res = 1 then  gotoxy plsx(up)
,totaly:print "    " ;
40600 s$ = " ":if total(up,15) < 100 t
hen s$ = "  "
40700 if total(up,15) < 10 then s$ = "
  "
40800 if res = 1 then  gotoxy plsx(up)
-res,totaly:print s$ total(up,15) ;
40900 gotoxy throwx,1:print throw ;
41000 scored = 0
41100 return
41200 ' --------------------------
41300 ENDGAME: ' bonuses,etc.
41400 if res = 2 then gotoxy 0,totaly:
print "TOTAL  " ;
41500 for i = 1 to nplayers:subtotal =
 0
41600 for j = 1 to 6
41700 subtotal = subtotal + total(i,j)
41800 next j
41900 if subtotal < 63 then goto 42200
42000 gotoxy plsx(i), 10:print "*35*"
;
42100 total(i,15) = total(i,15) + 35
42200 gotoxy plsx(i),totaly: print tot
al(i,15)  ;
42300 next i
42400 winner = 1
42500 for i = 1 to nplayers
42600 if total(i,15) > total(winner,15
) then winner = i
42700 next i
42800 key = 0
42900 gotoxy plhx(winner),3:print " Wi
nner !" ;
43000 while key <> -1 :gosub MOUSECTL:
if key <> 1 then goto 43700
43100 if mx<qmsg(1) or my<qmsg(2) then
 goto 43400
43200 if mx>qmsg(3) or my>qmsg(4) then
 goto 43400
43300 goto 43800
43400 if mx<nmsg(1) or my<nmsg(2) then
 goto 43700
43500 if mx>nmsg(3) or my>nmsg(4) then
 goto 43700
43600 goto 1500
43700 wend
43800 clearw 2:end
43900 ' --------------------------
44000 GETPLAYERS: ' nr of players
44100 for i = 1 to 4
44200 dx = rollx(i)+dw/2  :dy = rolly:
dnum = i:gosub DICE
44300 next i
44400 gotoxy startmsgx,5:print " Selec
t the number of players by "
44500 gotoxy startmsgx,6:print " point
ing to one of the dice above"
44600 gotoxy startmsgx,7:print " and c
licking the left mouse button. "
44700 key = 0
44800 while key <> -1
44900 gosub MOUSECTL
45000 if key <> 1 then goto 45900
45100 if mx<qmsg(1) or my< qmsg(2) the
n goto 45400
45200 if mx>qmsg(3) or my> qmsg(4) the
n goto 45400
45300 goto 12900
45400 if mx<selbox(1) or my< selbox(2)
 then goto 46000
45500 if mx>selbox(3) or my> selbox(4)
 then goto 46000
45600 for i = 1 to 4
45700 if mx>rollx(i) and mx<rollx(i+1)
 then nplayers = i:goto  46000
45800 next i
45900 wend
46000 gotoxy startmsgx,5:print spc(36)
46100 gotoxy startmsgx,6:print spc(36)
46200 gotoxy startmsgx,7:print spc(36)
46300 gosub CLEARROLL
46400 return
46500 RESIDATA: ' hi-res values
46600 ' scr width, scr height,dice y,d
ice x,dice width,dice height
46700 data 615,343,14,548,30,30
46800 ' rollmsgx,undomsgx,newmsgx,quit
msgx,startmsgx,throwx,totaly
46900 data 2,2,32,32,10,3,19
47000 ' player column header x's
47100 data 8,14,20,26
47200 ' player column score x's
47300 data 9,15,21,27
47400 ' roll dice x coordinates
47500 data 190,250,310,370,430,490
47600 ' hold dice y coordinates
47700 data 90,132,174,216,258,300
47800 ' box 1 x,y,width,height
47900 data 1,1,132,52
48000 ' box 1a x,y,width,height
48100 data 1,1,132,18
48200 ' box 1b x,y,width,height
```

```
48300 data 1,36,132,17
48400 ' box 2 x,y,width,height
48500 data 516,1,132,52
48600 ' box 2a x,y,width,height
48700 data 516,1,132,18
48800 ' box 2b x,y,width,height
48900 data 516,36,132,17
49000 ' box 3 x,y,width,height
49100 data 1,53,615,17
49200 ' box 4 x,y,width,height
49300 data 132,70,384,238
49400 ' box 5 x,y,width,height
49500 data 132,308,384,48
49600 ' roll message coordinates
49700 data 2,40,132,56
49800 ' undo message coordinates
49900 data 2,76,132,88
50000 ' new game message coords
50100 data 518,40,612,56
50200 ' quit game message coords
50300 data 518,76,612,88
50400 ' dice select area
50500 data 176,40,476,104
50600 ' dice hold area
50700 data 516,108,602,364
50800 ' scoring option area
50900 data 1,70,132,350
51000 ' scoring option y's
51100 data 110,126,142,160,176,194,210
,226
51200 data 244,260,278,294,312,328,346
51300 RES2DATA: ' --- data inititializ
ation values for medium-res
```



```
51400 ' scr width, scr height,dice y,d
ice x,dice width,dice height
51500 data 608,167,7,548,30,15
51600 ' rollmsgx,undomsgx,newmsgx,quit
msgx,startmsgx,throwx,totaly
51700 data 5,5,60,60,20,7,17
51800 ' player column header x's
51900 data 16,26,36,46
52000 ' player column score x's
52100 data 18,28,38,48
52200 ' roll dice x coordinates
52300 data 190,250,310,370,430,490
52400 ' hold dice y coordinates
52500 data 45,66,87,108,129,150
52600 ' box 1 x,y,width,height
52700 data 1,1,132,27
52800 ' box 1a x,y,width,height
52900 data 1,1,132,9
53000 ' box 1b x,y,width,height
53100 data 1,19,132,9
53200 ' box 2 x,y,width,height
53300 data 516,1,132,27
53400 ' box 2a x,y,width,height
53500 data 516,1,132,9
53600 ' box 2b x,y,width,height
53700 data 516,19,132,9
53800 ' box 3 x,y,width,height
53900 data 1,28,608,9
54000 ' box 4 x,y,width,height
54100 data 132,37,384,130
54200 ' box 5 x,y,width,height
54300 data 132,168,384,24
54400 ' roll message coordinates
54500 data 2,20,132,28
54600 ' undo message coordinates
54700 data 2,36,132,48
54800 ' new game message coords
54900 data 518,20,608,28
55000 ' quit game message coords
55100 data 518,36,608,48
55200 ' dice select area
55300 data 176,29,476,52
55400 ' dice hold area
55500 data 516,44,602,182
55600 ' scoring option area
55700 data 1,35,132,184
55800 ' scoring option y's
55900 data 55,68,76,86,94,104,112,120
56000 data 130,140,148,156,166,176,184
```

●

ST CHECKSUM DATA.

(see page 26)

```
 100 data  661, 761, 474, 299, 141, 12
4, 146, 554, 316, 162, 3638
 1100 data  714, 463, 357, 756, 132, 1
35, 952, 140, 20, 82, 3751
 2100 data  594, 599, 236, 361, 867, 6
95, 720, 984, 934, 555, 6545
 3100 data  833, 837, 567, 845, 849, 5
79, 585, 591, 684, 668, 7038
 4100 data  650, 662, 30, 10, 51, 923,
 67, 604, 592, 100, 3689
 5100 data  670, 91, 578, 349, 17, 387
, 2, 706, 603, 390, 3793
 6100 data  135, 391, 297, 843, 854, 3
13, 856, 867, 162, 307, 5025
 7100 data  314, 324, 54, 111, 561, 30
0, 282, 305, 385, 73, 2709
 8100 data  79, 178, 15, 845, 583, 716
, 749, 172, 758, 199, 4294
 9100 data  929, 936, 943, 108, 117, 7
41, 834, 553, 279, 416, 5856
 10100 data  737, 777, 956, 973, 748,
959, 976, 634, 958, 968, 8686
```

BRING IN THE NEW YEAR WITH ATARI FROM COMPUTABILITY

ATARI 520 SYSTEM PACKAGE*

*Including RGB or Monochrome Monitor, Mouse, Disk Drive, Basic, TOS on ROM, and RF Modulator

CALL
MONOCHROME SYSTEM

CALL
RGB/COLOR SYSTEM

ATARI 1040 ST SYSTEM PACKAGE*

*Including RGB or Monochrome Monitor, Mouse, Double-Sided Disk Drive, Basic, TOS on ROM, and Built-In Power Supply

CALL FOR CURRENT PRICE

***All ST System Packages are sold with a ComputAbility™ 90 day warranty.**

SUPRA 20 MEG 3.5 INCH HARD DRIVE
only
$679

ATARI SF 314 DISK DRIVE
Double sided/ 1 Megabyte storage
$199

PRINTERS

NX-10	209	NB-24/15	Call
NX-10/Parallel	279	Powertype	229
NP-10	189	Panasonic 1080i	199
NX-15	339	Panasonic 1091i	259
ND-15	449	Panasonic 1092i	Call
NR-15	559	Citizen 120D	Call

OKIMATE 20 COLOR PRINTER & ATARI ST PLUG N' PRICE
$199
This is a shipped price anywhere in continental USA

ST MODEM PACKAGE
NEW AVATEX 1200HC MODEM
1200 Baud-Hayes Compatible
+
ST MODEM CABLE
+
ST TALK TELECOMMUNICATIONS PACKAGE
$159

WE RECEIVE ST PRODUCTS ON A WEEKLY BASIS. IF YOU DON'T SEE IT HERE . . . CALL

SUPER SPECIALS*

Degas	24.95	BTS Spreadsheet	44.95
Paper Clip Elite	64.95	Time link	31.95
Isgur Portfolio	124.95	I.S Talk	31.95
Degas Elite	49.95	Thunder	24.95

BATTERIES INCLUDED

*ST SUPER SPECIAL PRICES EFFECTIVE NOW THRU MARCH 31, 1987

ATARI 520 ST • ATARI 1040 ST SOFTWARE

ABACUS
Textpro	34.95
Datatrieve	34.95
Forth MT	34.95
Paintpro	34.95
Text Designer	34.95
Assempro	34.95
PC Board Designer	169.95
Powerplan	34.95
Abacus Books	Call

ST GRAPHICS
Degas	27.95
Degas Elite	52.95
Easy Draw	54.95
Paintworks	49.95
Super Graphics	33.95
New Tech Color Book	16.95
Graphic Artist	149.95
Real Cadd	CALL
Aegis Animator ST	49.95

INFOCOM ST
Forever Voyaging	29.95
Bally Hoo	25.95
Cuthroats	25.95
Deadline	31.95
Enchanter	25.95
Hitchiker	25.95
Infidel	28.95
Leather Goddess	25.95
Moonmist	25.95
Planetfall	25.95
Seastalker	25.95
Sorcerer	28.95
Spellbreaker	31.95
Starcross	31.95
Suspect	28.95
Suspended	31.95
Trinity	25.95
Wishbringer	25.95
Witness	25.95
Zork I	25.95
Zork II or III	28.95

ST WORD PROCESSORS
Final Word	89.95
Paperclip Elite	64.95
Habawriter	54.95
Regent Word II	64.95
Wordwriter ST	52.95
Thunder	27.95

MICHTRON
BBS 2.0	49.95
Business Tools	34.95
Calendar	20.95
Cornerman	34.95
DOS Shell	27.95
Echo	27.95
Flipside	20.95
Kissed	34.95
M-Copy	54.95
M-Disk	27.95
Major Motion	27.95
Mi-Term	34.95
Michtron Utilities	39.95
Mudpies	16.95
Softspool	20.95
Cards	27.95
The Animator	27.95
Time Bandits	27.95
Mighty Mail	34.95
Easy Record	54.95
Intro to Logo	20.95
Personal Money Mgr.	34.95
Pinball Factor	27.95
Football Wizard	27.95
Financial Future	27.95
Eight Ball	20.95
Dot Driver	34.95
Laser Driver	34.95
Super Conductor	49.95
Hard Disk Backup	27.95
MI-Print	Call

ST LANGUAGES
Personal Pascal	49.95
Fast Basic	89.95
Fast C Compiler	99.95
Fast Fortran	199.95
Mark William's C	129.95
Metacomco Pascal	74.95
Macro Assembler	59.95
Lattice C	99.95
Fast Cobol	199.95
Metacomco BCPL	109.95
Cambridge Lisp	139.95
Modula II	54.95
Metacommco Make	49.95
Menu +	23.95
BCPL	99.95
Modula II/Developer Vers	104.95

HIPPOPOTAMUS
Call for items and prices

ST BUSINESS
VIP Professional	169.95
VIP Lite	99.95
Swiftcalc ST	52.95
Isgur Portfolio	129.95
Synsoft General Ledger	44.95
SBM Point of Sales	84.95
Sierra One Write (ea.)	69.95
Financial Cookbook	34.95
DAC Payroll	34.95
DAC Easy Accounting	49.95
Maxiplan	Call
Sylvia Porter Vol. I	52.95
Dollars and Sense	69.95
Home Accountant	44.95
BTS Spreadsheet	44.95

CASIO KEYBOARDS
CZ-101	279
CZ-230S	369

ST EDUCATIONAL
Decimal Dungeon	27.95
Fraction Action	27.95
Kinderama	27.95
Read & Rhyme	27.95
Animal Kingdom	27.95
Teddy Bear/Rainy Day	24.95
Speller Bee	33.95
Kid Talk	33.95
Math Talk	33.95
First Shapes	33.95

FUJI DISKETTES
SS/DD 3.5 in. (10 Pk) . . . 15.95
DS/DD 3.5 in. (10 Pk) . . 24.95
NOTE: Buy Fuji Diskettes at these low prices when added to any other order.

ST ACCESSORIES
Flip n' File II-Micro	19.95
Dustcovers	Call
3.5 Disk Drive Clean Kit	Call
Mouse Pad	8.95
Mouse House	6.95
RCA 6 Way Noise/Surge Prot.	34.95
Anti-Glare Screen	19.95

ST UTILITIES
Music Studio	39.95
Time Link	34.95
Middiplay	34.95
Micro-Cookbook	32.95
CZ Droid	74.95
EZ Track	49.95
Right 90	19.95
Macro-Manager	49.95
Fast	34.95
Labelmaster	27.95

DESKTOP PUBLISHING
Drawrite	99.95
Publishing Partner	Call

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
TELEX NUMBER 9102406440

ST ADVENTURES
Hacker	29.95
Hacker II	33.95
Sundog	24.95
Winnie The Pooh	19.95
Black Cauldron	27.95
Apshai Trilogy	27.95
Universe II	47.95
The Pawn	29.95
Alternate Reality	33.95
Coveted Mirror	16.95
Golden Pass	29.95
Guild of Thieves	29.95
Dungeon Master	27.95
Tass Times	33.95
Mercenary	27.95
Autoduel	34.95
Ogre	27.95
Gateway	33.95
Defender of Crown	33.95
Sinbad the Sailor	33.95
Balance of Power	33.95
King of Chicago	33.95
S.D.I.	33.95
Ultima II or III	39.95
King's Quest II or III	33.95
Portal	33.95
Space Quest	33.95
Phantasie II	27.95
Bard's Tale	Call

ST PRINT UTILITIES
Typesetter	24.95
Rubber Stamp	24.95
Printmaster	24.95
Fontwriter	27.95
Megafont ST	24.95
Art Gallery I or II	19.95
Typesetter Elite	32.95

ST TELECOMM
PC Intercom	79.95
ST Talk	17.95
I.S. Talk	39.95
Home Pak	33.95

ST DATABASES
DB Man	99.95
Regent Base	64.95
Data Manager ST	52.95
DB Master One	39.95
Zoomracks II	99.95

ST ARCADE GAMES
Bridge 4.0	20.95
Winter Games	27.95
Rogue	27.95
Super Huey	27.95
Phantasie	27.95
Mean 18	29.95
Leader Board	27.95
Brattacus	33.95
Donald Duck	20.95
Silent Service	27.95
Flight Simulator II	34.95
Champ. Wrestling	27.95
World Games	27.95
Video Vegas	24.95
Blazing Paddles	24.95
Strip Poker	27.95
Starglider	29.95
Alternate Reality	33.95
Little Comp. People	33.95
Smoothtalker	Call
Gato	33.95
Joust	20.95
Chessmaster 2000	32.95
Jet	34.95
Tenth Frame	27.95
Shanghai	29.95
3-D Helicopter	33.95
Micro League Baseball	39.95
WWF Micro Wrestling	39.95
Skyfox	32.95
Star Raiders	20.95
Super Cycle	27.95
ST Pool	24.95
Indoor Sports	Call
F-15 Strike Eagle	27.95
High Roller Simulator	Call
TwoTwo Basketball	29.95
Mastertype	27.95
Hex	24.95
Love Quest	39.95
Deep Space	29.95
Arena	24.95
ST Karate	24.95
ST Protector	23.95
Space Station	24.95
Psion Chess	39.95
Intracourse	69.95
Championship Baseball '86	29.95
GFA Championship Football	29.95
Harrier Strike	34.95

ATARI is a trademark of ATARI, INC.

No surcharge for MasterCard or Visa

ORDERING INFORMATION: Please **specify system.** For fast delivery send cashier's check or money order. Personal and company checks allow 14 business days to clear. School P.O.'s welcome. **C.O.D. charges are $3.00.** In Continental U.S.A. include $3.00 for software orders, 4% shipping for hardware. Minimum $4.00. MasterCard and Visa orders please include card #, expiration date and signature. WI residents please include 5% sales tax. HI, AK, FPO, APO, Puerto Rico and Canadian orders, please add 5% shipping, **minimum $5.00.** All other **foreign orders add 15% shipping, minimum $10.00.** All orders shipped outside the Continental U.S.A. are shipped first class insured U.S. mail. If foreign shipping charges exceed the minimum amount, you will be charged the additional amount to get your package to you quickly and safely. 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 R.A. # or your return will not be accepted. Prices and availability subject to change without notice.

CIRCLE #113 ON READER SERVICE CARD

```
 11100 data  675, 745, 976, 993, 62, 7
49, 613, 628, 868, 746, 7055
 12100 data  599, 614, 225, 750, 669,
684, 791, 171, 875, 720, 6098
 13100 data  12, 216, 589, 160, 782, 6
02, 399, 5, 253, 180, 3198
 14100 data  705, 371, 526, 221, 588,
906, 370, 376, 502, 526, 5091
 15100 data  182, 581, 567, 395, 351,
525, 318, 534, 229, 850, 4532
 16100 data  709, 184, 766, 874, 404,
841, 815, 154, 784, 1, 5532
 17100 data  844, 797, 571, 800, 247,
383, 204, 765, 64, 374, 5049
 18100 data  536, 967, 97, 576, 554, 5
61, 568, 575, 582, 655, 5671
 19100 data  642, 197, 372, 823, 848,
544, 975, 411, 584, 893, 6289
 20100 data  466, 473, 480, 487, 494,
636, 569, 21, 588, 989, 5203
 21100 data  929, 518, 988, 377, 871,
430, 273, 445, 364, 519, 5714
 22100 data  420, 229, 259, 477, 11, 2
04, 366, 940, 788, 455, 4149
 23100 data  182, 960, 28, 693, 484, 8
98, 842, 263, 1, 495, 4846
 24100 data  563, 868, 190, 458, 878,
237, 989, 533, 964, 171, 5851
 25100 data  239, 545, 370, 956, 507,
399, 535, 966, 744, 214, 5475
 26100 data  242, 926, 373, 535, 56, 6
83, 221, 537, 57, 843, 4473
 27100 data  327, 853, 848, 538, 866,
253, 107, 255, 108, 377, 4532
 28100 data  788, 794, 235, 371, 542,
204, 164, 256, 869, 881, 5104
 29100 data  868, 542, 382, 781, 72, 5
46, 103, 927, 256, 438, 4915
 30100 data  553, 555, 250, 358, 244,
912, 522, 51, 231, 743, 4419
 31100 data  358, 433, 367, 232, 611,
363, 525, 46, 835, 715, 4485
 32100 data  14, 294, 548, 418, 374, 5
40, 131, 308, 975, 144, 3746
 33100 data  347, 304, 454, 296, 795,
530, 598, 532, 608, 345, 4809
 41100 data  521, 952, 596, 820, 20, 2
42, 911, 368, 652, 606, 5688
 42100 data  951, 95, 365, 871, 296, 8
98, 369, 403, 340, 993, 5581
 43100 data  972, 989, 763, 975, 992,
647, 179, 191, 190, 115, 6013
 44100 data  237, 987, 371, 361, 717,
56, 408, 442, 763, 810, 5152
 45100 data  48, 61, 767, 669, 686, 24
5, 547, 379, 187, 83, 3672
 46100 data  85, 87, 802, 539, 937, 46
7, 154, 413, 843, 48, 4375
 47100 data  152, 9, 158, 891, 462, 87
7, 371, 727, 165, 862, 4674
 48100 data  154, 867, 257, 728, 426,
870, 438, 875, 498, 729, 5842
 49100 data  280, 733, 605, 737, 585,
257, 264, 239, 288, 842, 4830
 50100 data  473, 74, 510, 805, 583, 5
56, 718, 339, 389, 249, 4696
 51100 data  201, 932, 540, 452, 88, 3
98, 846, 40, 253, 994, 4744
 52100 data  264, 876, 447, 862, 150,
712, 146, 854, 114, 852, 5277
 53100 data  137, 713, 420, 855, 271,
860, 457, 721, 171, 718, 5323
 54100 data  584, 722, 574, 242, 244,
224, 257, 862, 485, 87, 4281
 55100 data  506, 818, 529, 569, 585,
352, 411, 269, 729, 923, 5691
 34100 data  528, 604, 597, 349, 532,
608, 106, 976, 354, 530, 5184
 35100 data  606, 104, 974, 352, 602,
536, 612, 731, 111, 974, 5602
 36100 data  973, 353, 536, 57, 241, 3
03, 122, 19, 720, 638, 3962
 37100 data  537, 58, 324, 304, 541, 8
34, 312, 544, 65, 252, 3771
 38100 data  304, 529, 828, 213, 440,
798, 396, 386, 548, 62, 4504
 39100 data  730, 892, 509, 941, 981,
874, 314, 723, 109, 773, 6846
 40100 data  179, 256, 956, 24, 308, 4
93, 561, 202, 803, 831, 4613
```

•

AS68 Helper

A SPECIAL INCLUSION

A tutorial and utility program for users of DRI's AS68 assembler.

by Charles F. Johnson

This article won't teach you 68000 assembly language, but it will show you how to set up a disk with all the files necessary to edit, assemble and debug programs using DRI's AS68 assembler, as supplied with the Atari Developer's Kit. And there's a nice treat included for anyone who uses AS68; a utility program that can *greatly* simplify the whole editing and compiling process. This utility, the **AS68 Helper**, acts as a sort of dispatcher program, letting you edit, assemble (with several different options), or debug your code—all with the click of a mouse button!

The process of editing and compiling an assembly language program on the ST is nowhere near as straightforward as on the Atari 8-bit models. If you're used to the speed and (relative) simplicity of MAC/65 by OSS (to name one excellent 8-bit assembler), the complexity of the AS68 package can be very discouraging. With MAC/65 and most 8-bit assemblers, the editor, assembler and debugger programs are all resident in memory at once. On the ST, you'll need at least ten programs (all disk resident) to develop a program. But cheer up; C programmers are even worse off in this respect.

In any case, the most important tool for anyone using the AS68 assembler is a RAMdisk, which sets aside part of the computer's memory to simulate a floppy disk drive. You can drastically speed up the assembly process by copying all the necessary files to the RAMdisk and doing your work from there. There are many commercial and public domain RAMdisk programs out for the ST; if you don't have one yet, there are several on the **ANALOG Computing/ST-Log** Atari Users' Group on Delphi. And many of the public domain versions are just as good as (in some cases, better than) the ones you pay for.

A RAMdisk is not absolutely necessary to use AS68; you can, of course, assemble just as well from a normal disk drive. But it will definitely make your life easier! Just remember to copy your source files to a real disk before turning off the computer and sending your RAMdisk to bit heaven.

I'll assume you're using the MicroEMACS text editor that comes with the Developer's Kit, but if you prefer a different editor, just substitute its name for EMACS.TTP in the discussion that follows. Please note that whichever editor you use, you *must* save source code with a filename extension of *.S*—the AS68 assembler doesn't accept others.

The files you'll need from the Developer's Kit disks are:

BATCH.TTP	The batch file processor (uses AS.BAT, see below)
EMACS.TTP	Text editor
SID.TTP	DRI's symbolic debugger
AS68.PRG	The assembler
AS68SYMB.DAT	A data file which is read by the assembler
LINK68.PRG	The linker
RELMOD.PRG	The relocator; creates a relocatable .PRG file
RM.PRG	File deleter to remove intermediate files
WAIT.PRG	Waits for a keypress after assembly

In addition to these files, you also need to create a file called AS.BAT (you can call it anything, actually, but the extension must always be *.BAT*). This file contains the commands for BATCH.TTP, which is the program you double-click on to begin an assembly; it reads the AS.BAT file and executes the list of commands contained therein. To perform a simple assembly, with no listing or symbol table, you should enter the following text into your editor:

```
as68 -l -u %1.s
link68 [u] %1.68k=%1.o
relmod %1.68k %1.prg
rm %1.68k
rm %1.o
wait
```

Save this with the name AS.BAT; you now have a complete assembler disk. Copy these ten files, plus your source code, to your RAMdisk. When you've finished editing the

source code for a program, save it and double-click on BATCH.TTP. A GEM dialog box appears to allow you to enter parameters for BATCH; enter the following text:

```
as source
```

You would replace *source* with the name of your source code file. Please note that you should *omit* the .S extension or the assembler may *delete* your source code; so be careful! Press RETURN or click on the OK button, and the assembly will begin. As BATCH calls the programs, the screen will show which one is currently active. Any errors will be printed, and, finally, you should see a message telling you to press RETURN. When you do, you'll be back at the GEM desktop, and a file with the same name as your source code but an extension of .*PRG* will now be present on the disk (or RAMdisk, as the case may be). This is, of course, your assembled program, which may be run like any other GEM program, or inspected and debugged with the SID debugger. (See the Developer's Kit documentation for more details on the operation of SID.)

And now. . .*AS68 Helper!*

AS68 Helper is a utility that can greatly speed up the assembly process. To install it, you should click *once* on the file called ASM_EDIT.PRG (included on this month's **ST-Log** disk and on our Atari Users' Group on Delphi), and then move up to the options menu heading on the desktop and select INSTALL APPLICATION. For the document type, enter *S* and click on the OK button (don't press RETURN. . .the default in this dialog box is *CANCEL*). Now, whenever you double-click on a file with an extension of .*S* (a source code file), **AS68 Helper** will load first and present you with a GEM menu bar with two titles, "Desk" and "File." *Note:* you don't have to go through the INSTALL APPLICATION procedure every time. Just do it once and save your desktop.

The menus work in typical GEM fashion: just move the mouse to one of the titles, and a box will drop down containing several choices. If you don't want to use the mouse, you can simply type the first letter of any menu selection while there are no menus pulled down (e.g. *E* for Edit, *A* for Assemble, etc.) The desk menu allows you to access any desk accessories you have installed, and also lets you view the **AS68 Helper** title box. The file menu lets you edit, assemble, or debug your program—**AS68 Helper** uses the GEMDOS EXEC function to call EMACS.TTP, BATCH.TTP, or SID.PRG. These programs, along with the others mentioned above (including ASM_EDIT.PRG) must all be present in the same directory with the source code (they're already on the RAMdisk anyway, right?) If there is no .PRG file on the disk with the same name as the source, the DEBUG option will appear in lighter type, to show that it is not currently accessible.

The "Print" option lets you send the source file you're working on to any type of printer. A dialog box allows you to choose pica, emphasized, elite, condensed, or near-letter-quality type, set your left margins, and skip over the paper's perforations. (These print options only work with Epson/Star and compatible printers.) There is also an option to expand tabs to any number of spaces (the default is 8); note that the tab space setting works for any printer, not just the Epson clan, since it expands TAB characters to strings of spaces. After starting a printout, you can abort it by pressing any key.

The file menu also lets you select a different source file to work on. Choose "New Source File" and a "File Selector" box will appear, showing all files with an extension of .S in the current directory. Since all the assembler programs must be in the same directory as your source code, you may not change directories with the File Selector. If you try, you'll see an alert box with the message *Source and assembler files must be in the directory you started with*. . .To edit several different source files during a session, just copy them all to the same directory.

When you select "Assemble," another dialog box will let you choose from several assembly options. You can specify the extension of the output file (.PRG, .ACC, .TOS, or .TTP) and choose from three preset types of assembly—a simple assembly, an assembly which produces a listing file with extension of .ASM, or an assembly which includes in the .PRG file the actual labels used in your source (this is called a "symbol table"). When a .PRG file with symbol table is examined with SID, you can refer to the same labels as the ones you used in your source code, which greatly facilitates debugging. In addition to these preset types, you can also choose "Batch File" (see below) to customize the assembly. (Batch files must be in the same directory as all the other files.) By the way, when you use **AS68 Helper**, you can eliminate WAIT.PRG from your assembler disk. If you use a preset assembly type, you can eliminate RM.PRG, too.

The key to producing these different types of assemblies is in creating the proper .BAT file for BATCH.TTP. **AS68 Helper** writes a new AS.BAT file to the disk every time you assemble, based on your choice. The source code file on the **ST-Log** disk and Atari Users' Group is called ASMEDIT.S. There you can see how these batch files should look. Bear in mind that there are many more possibilities for customizing batch files. The three choices that **AS68 Helper** gives you are not your only options!

The source code also shows how to include object trees directly into an assembly language program, instead of using a separate resource file created by the GEM Resource Construction Set, as well as a number of other interesting techniques. I would like to extend special thanks to my friend Barnaby Finch, who provided much of the information about setting up an assembler disk.

A final note on using different editors: you have two options if you want to use **AS68 Helper** with an editor other than MicroEMACS. You can simply rename your editor EMACS.TTP (this works in most cases, unless the text editor is GEM-based, like **1st Word**), or edit the source code for **AS68 Helper** to use the new name. If you choose the latter, you'll probably have to adjust some loops and some counters, as well as the obvious filename specifiers. The source is pretty well commented and should be fairly easy to follow. Have fun. . .and may all your assemblies be error-free! ///

The New British Invasion

British software makes inroads in the ST marketplace.

by Bill Kunkel, Arnie Katz and Joyce Worley

Despite its status as the most computer-crazy nation on earth—there are far more computers in Britain, per capita, than anywhere else—the British have had, until recently, only a marginal impact on the American software industry.

There are several reasons for this, all essentially economic. For one thing, until the rise of the Commodore 64, the U.S. and the U.K. had no computers in common. And after the C64 became popular in both countries, British market conditions encouraged the creation of large quantities of mostly mediocre software.

Because the U.K.'s economy made the purchase of disk drives an impractical luxury, tape cassettes evolved as the standard storage medium. British publishers avoided games with disk access in favor of tape programs which could be dumped into RAM in one (or even several) loads. Complex adventures were out; action-strategies were in. The U.S. situation, of course, was quite different.

During the early 1980s, the typical British entertainment software consumer was an adolescent male, oriented toward frequent purchases of cheap, action-oriented games. This led to the production of thousands of adaptations of popular American titles, such as **Lode Runner** (Broderbund), **Pac-Man** (Atari Corp.) and, especially, **Impossible Mission** from Epyx which has, in the years since its release in the U.K., spawned more imitators than Elvis.

Any innovations were largely superficial—improved sound and graphics, more rooms, levels and screens—all added to squeeze more "play value" into the product. But to American buyers, most of this seemed to amount to little more than a labored attempt to prolong the player's learning curve.

Two of the most popular methods of adding play value were: increasing the game speed (like, an illegally "enhanced" **Pac-Man** coin-op in which Inky, Blinky and Clyde travel only slightly slower than the speed of light) and omitting coherent documentation (which always adds a few man-hours to the fun). This produced mostly chaff, fit only for the swelling American discount bins.

The coming of the ST has abruptly changed the thinking of British publishers. Although perceived as an "economy-model" computer in the U.S., the ST costs about three times as much as a C64 with tape drive, which already rates as a top shelf entry in the United Kingdom. This has radically altered the demographics of the British software audience. Games are priced higher, because it's expected that ST users will be willing to spend more. And since the domestic market is significantly smaller than it is for less expensive computers, software publishers have begun for the first time to take the *world* market into consideration.

Much of the best ST software was created in Britain. MichTron, Firebird and Mindscape, among others, have gained major footholds in the ST universe—largely through British-produced offerings.

Some of the imported software, unfortunately, fails to bridge the gap between British and American tastes; some of it suffers from old bad habits (poor documentation, difficult play). But the best of it is leading the way in exploring the potential of 16-bit technology.

Both the old and the new can be found in **Brataccas** from Mindscape, Inc. (3444 Dundee Road, Northbrook, IL 60062). The ultimate in **Impossible Mission** imitations, **Brataccas** is burdened with the most confusing movement system in the history of electronic entertainment, as well as hopelessly inadequate instructions. In spite of these weaknesses, however, **Brataccas** (by the British design house Psygnosis) shows flashes of real brilliance. The characters met in its dual-level, horizontally-scrolling comic-book universe speak, appropriately enough, in word balloons. The graphics are stunning, though "busy"—no pixel goes unused. Walls, doors and ceilings are festooned with cameras, monitors, ads, posters, bumper stickers and anything else that the artist could think of.

Unlike U.S. designers, who are extremely conservative in terms of subject matter, the British are more sophisticated. Comic books and rock music are major influences. **Brataccas** looks like something from an issue of *2000 AD*, the popular British comic, although it plays more like *Finnegans Wake*.

Psygnosis (Freeport, Liverpool, L3 3AB, U.K., distributed exclusively in the U.S. by Computer Software Services, 495 A Busse Road, Elk Grove Village, IL 60007 and APEX Resources Inc., 129 Sherman Street, Cambridge, MA 02140) has subsequently attempted to market its own software. **Deep Space**, a beautiful and sophisticated space combat game, and **Arena**, a multi-event track-and-field competition in the **Summer Games** tradition, are its latest offerings.

Psygnosis represents most of the "old" British cliches: they continue to flog the same tired themes (no karate contest?), modernized only with the most attractive garnishes their programmers can muster.

These games are more palatable on the ST than they might otherwise be, because of the size of the ST software library. There is no surfeit of space and track games on the ST for a simple reason: there is no surfeit of *anything* on the ST (except golf games; don't ask us why).

Arena, by Ian Hetherington and Colin Rushby, is the most visually impressive track-and-field simulation we've seen yet. Its six events (100-meter race, long jump, high jump, pole vault, shot put and javelin) are each faultlessly depicted, right down to the waving flags and the cartoonish judge who sarcastically appraises your performance after each attempt (". . . In other words, give it up. . ."). If you take too long to get an athlete going once the starter's gun sounds, "he" will start razzing the coach (i.e., you). Word balloons are used extensively, but here, of course, the commentary has no effect on the game itself.

The system for moving an athlete around the screen shouldn't be all that challenging in a game like this. But it is. Speed is generated by "pounding" (as the manual puts it) two groups of keys, then hitting the SPACE BAR to jump or throw. However, even after working ourselves into a virtual paroxysm of keyboard assault, we were unable to record anything better than the most mediocre score. Of course, this could be the fault of the documentation, which manages to make even the simplest instruction hopelessly confusing.

The most successful of the British publishers in the ST market so far is clearly Firebird (74 North Central Avenue, Ramsey, NJ 07446). Firebird burst on the software scene last year with the brilliantly illustrated Electronic Scrolls text adventure, **The Pawn**, which combines a powerful, parser-driven story with graphics reminiscent of Romantic landscape paintings.

Firebird followed that success with the release of a wild, new action game under the Rainbird label. **Starglider** is a fresh, exciting shoot-'em-up that not only makes intelligent use of the ST's graphics, but actually works the mouse into the gameplay. There's even a digitized rock theme song and several snatches of actual speech.

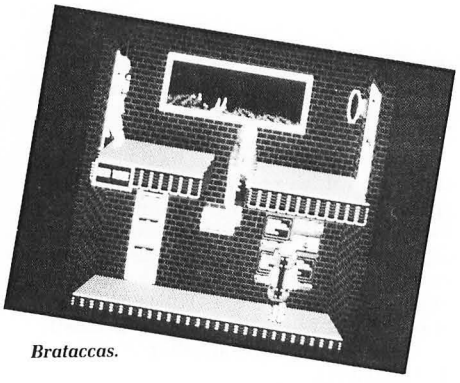
Unlike the U.S. gamers who fell out of love with vector-type line graphics after **Battlezone**, the English have mantained a strong affection for the 3-D crystaline look, especially as seen in the old Penguin Software favorite, **Stellar 7**.

Starglider is, in fact, frequently reminiscent of that earlier game, but with greatly enhanced graphics and a simulator level of realism, with operational instrument panel. **Starglider** even has adequate documentation, including a play guide, ST key commands and an accompanying novella which sets the game in context.

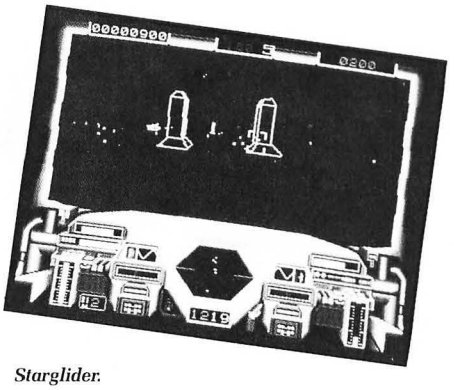
Firebird hopes to extend its dominance over the ST market with the soon-to-be-released **Guild of Thieves** by Electronic Scrolls, and **Universal Military Simulator**, which is the company's first American-designed product.

British designers are finally making a significant impact on the American market, thanks to the ST, and may well dominate the market within the next six months. In upcoming issues, we'll continue to cover new British products, while taking a closer look at the creators themselves.

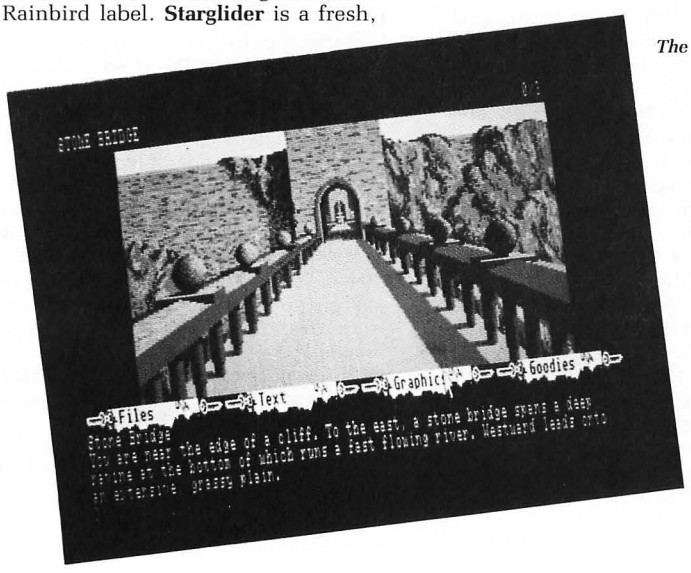
Next issue, we'll go Questing, courtesy of Sierra, and take a look at two of the newest ST offerings: **King's Quest III** and **Space Quest**. ///



Brataccas.



Starglider.



The Pawn.

Rogue

EPYX
by A.I. Design
P.O. Box 8020
Redwood City, CA 94063
(415) 366-0606
Low resolution $39.95

by Joyce Worley

A player icon moves through a multi-level dungeon, acquiring treasures and dueling with monsters, in this translation of a mainframe classic. **Rogue**'s treatment of an old theme is elevated, however, by its use of the ST's rich graphic capabilities and simplified user interface.

In all previous versions of **Rogue**, each dungeon level was represented by a full-screen overhead display. Monsters were visually portrayed by the first letter of their name: ogres were yellow *O*s; slime was a purple *S*. Treasures were simple icons. The player was indicated by the cursor arrow.

The dungeon levels are shown from the same perspective in the ST version; this makes for speedy play and is essential for establishing one's position relative to the rest of that level. But ST **Rogue** adds a

Hacker II: The Doomsday Papers

ACTIVISION
by Steve Cartwright
Drawer 7286
Mountain View, CA 94042
(415) 960-0410
Low resolution $49.95

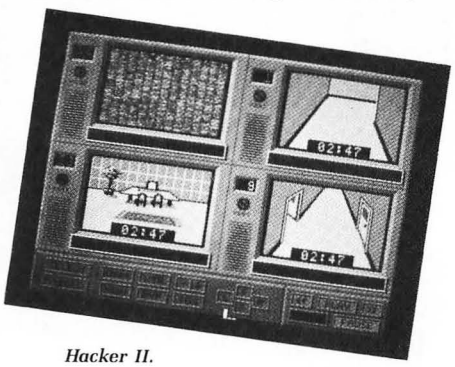
by Bill Kunkel

You've just logged onto the apocryphal Actisource Hotline, a computer bulletin board offering the usual product listings and interviews. Suddenly, transmission is interrupted by an urgent message from the Director of Special Agents in Washington, D.C. A Soviet scientist has written a "simple, yet devastating plan" to overthrow the U.S. government. These so-called Doomsday Papers are locked in a vault deep within a Siberian military installation, and the Director is counting on you, the player, to plug into a network of high-tech computer spy systems in order to snatch the secret plans.

This is the clever premise of Steve Cartwright's sequel to his own hit, **Hacker**. Like **Hacker**, **Doomsday Papers** represents a new kind of computer adventure in which the user, rather than typing a series of verb-noun commands, actually operates various devices in order to solve the game.

In **Hacker II**, you must interface with a Multi-Function Switching Matrix, or MFSM, a four-monitor console that allows access to some high-powered electronic espionage equipment. Using your trusty MFSM, you can pick up feeds from the installation's own surveillance cameras and monitors, run videotapes, and operate the three Mobile Remote Unit (MRU) robots which have been planted on-site by U.S. operatives.

Once you dope out the high-tech spy system, you can view videotapes of security patrols, for example, in order to ascertain the schedule of the guards. Multiple channels survey the entire compound. You plan a route, then switch to telemetry in order to maneuver the MRUs in search of the secret papers. But watch out; if the Russkies suspect there's a little spy-droid running around their compound, they're likely to respond by sending out an Annihilator to pound it to scrap metal.

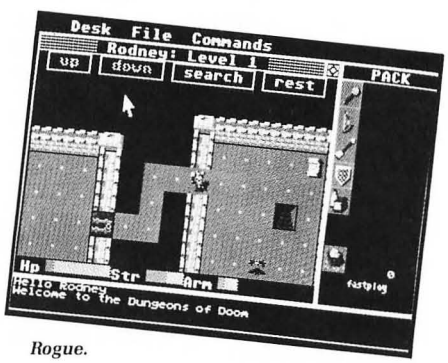


Hacker II.

Hacker II is remarkable and innovative entertainment, and make no mistake, it's extremely challenging. It assumes that you have already successfully navigated **Hacker** (which explains why the mystery spymaster is contacting you as the program begins). Familiarity with the original game is helpful, though hardly essential.

The only documentation is, basically, loading instructions (for the Actisource bulletin board, of course) and an MFSM operator's manual. The manual is written in deadpan high-fog manual-ese. (For example: "Highly advanced Adaptive Differential Pulse Coded Modulation (ADPCM) circuitry is used to provide near real-time manipulation of the MFSM subsystems.") Much as it might have spoiled

(continued on next page)



Rogue.

"zoom" mode, accessed by hitting the RETURN key. This shows the player icon and immediate environs in beautiful close-up, with monsters and treasures depicted in detailed color drawings. These two displays enrich the play value of **Rogue** tremendously, as does the ability to switch from mouse (for room-to-room movement) to keyboard control (for inspecting rooms).

Combat is simple: when two characters vie for the same space, they do battle. The results of each round are printed in a window below the visual display, while meters gauge the player's hit points, strength and armor value. Hit points are renewed by clicking on the "rest" command (though sleeping adventurers are more vulnerable to surprise attack), and an area is examined for secret doors with the "search" command. A separate window appears whenever the player needs food. (Viands as well as treasures are found in the dungeon's many rooms.) The right side of the screen contains an icon inventory, and items are accessed by clicking on the appropriate symbol. This produces a pop-up window that defines the player's options.

As the player moves deeper and deeper into the dungeon, the dangers increase—along with the value of the prizes. There is gold to be found, and weapons, and magic rings, not to mention spells and potions. But before slipping on a ring or quaffing a potion, it's a good idea to use an identify spell to determine exactly what the heck it is you've got.

Treasures, especially spells and potions, are the best thing about **Rogue**. Learning how to exploit the goodies one accumulates is the secret of successful play, but the enchanting graphics make the education an enjoyable one. When a player loses, the adventurer's name appears inscribed on an evocative tombstone, along with the rank, amount of gold amassed, the level the stalwart died on and the type of monster that snuffed him or her. There's even a Hall of Fame, recording the names of the top ten adventurers to date, with their rank, wealth, level reached and a picture of their final adversary.

Rogue is a slam-bang adventure in the trash-the-trolls-and-take-the-treasure tradition, enhanced by excellent graphics and optimum playability. ///

the mood, an English version would have been extremely helpful. Players intimidated by complicated-looking instructions will faint when they see these.

But bear with it folks; **Hacker II** unfolds like a crystal artichoke, revealing a multitude of activities through the amazing MFSM. The system includes a fully operational videotape recorder (continuous record), with a visible fast forward and rewind, as well as a pause button. You even have to adjust the vertical hold on camera transmissions!

The "player interface" is remarkably simple for such a sophisticated game. The mouse controls an on-screen hand whose pointing finger pushes buttons on the MFSM console. You select one of four monitors and determine what to display on it (tape, monitor, surveillance camera input, or telemetry guidance). The telemetry guidance system (TGS) gives an overview of one of three MRUs and a radar display covering five square miles within the installation.

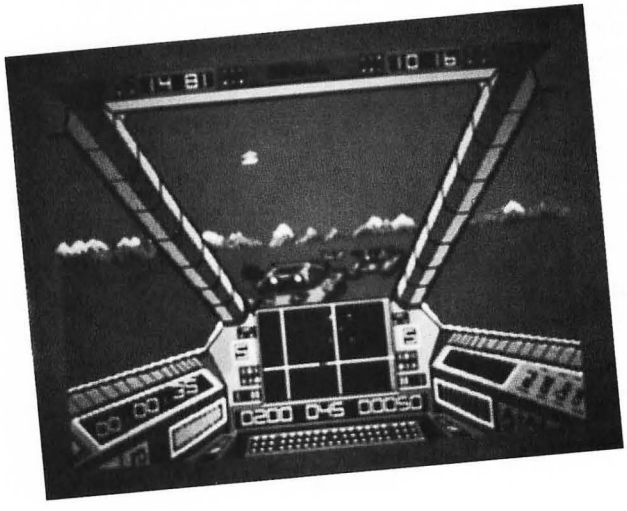
The graphics mimic perfectly the grainy black-and-white images of security monitors, right down to the vertical rolling and bands of static which appear during visible rewind and fast forward playback. Similarly, audio effects include everything from the ringing of a phone (as you attempt to log on to Actisource) and the high-speed clacking of modem-transmitted data, to the crunch of the Annihilator as it reduces one of your MRUs to postage stamp dimensions.

Hacker II is that rarest of follow-ups: a sequel that surpasses the original. Highly recommended. ///

Skyfox

by Ray Tobey
ELECTRONIC ARTS
1820 Gateway Drive
San Mateo, CA 94404
Low resolution $44.95

by Steve Panak



In *Skyfox*, piloting your Federation fighter is the easy part.

Finally, after years (okay, months) of promises and dashed expectations, we, the ever-swelling ranks of ST owners, have been offered an action game that takes full advantage of the ST's superior speed and power. I may never enter an arcade again.

Skyfox from Electronic Arts is the game, and it places you, the player, in the position of a young fighter pilot protecting his home asteroid from a nameless, faceless, ruthless enemy. After giving you the keys to a brand new Skyfox Mach4 Federation fighter, the program puts you through a series of confrontations you're unlikely to soon forget. But the main thing that sets this program apart from its predecessors is its complexity.

Your easiest task is piloting. Using the mouse or a joystick (I recommend the joystick), you control the movement through three dimensions. Smacking the SPACE BAR gives you afterburner thrust, just the ticket for getting out of a sticky situation —or for circling around on your enemy and feeding him a few pulses from your joystick-activated plasma cannon. But after you've mastered flight, you'll want to inspect the cockpit.

The visual complexity of this cabin hints at the game's numerous features. Readouts provide information on power, shield and missile levels. A radar scanner screen gives enemy positions in your quadrant. Compass heading, map coordinates and altitude displays show where you're going. Speed and time elapsed indicators tell you how fast you're getting there and how much time you have left. Ten speed settings are provided via the numeric keys.

Press *C* to bring up your computer display. A window showing the locations of all enemy forces, as well as your own bases and installations, opens up over the cabin's front viewport. This tactical map can be zoomed in on and filled with a number of reports, including the all-important score summary. Using the cursor control keys, you move a square over the area you wish to travel to; then pressing *A* engages the automatic pilot, allowing you to lean back and take a breather while the jet races you into another sticky situation.

And speaking of sticky situations, this game is full of them. Fifteen scenarios are available, and each can be played in one of five difficulty levels. You'd be well advised to start with a few of the seven training missions, in which you battle only tanks and planes, with no threat posed to your base. The base is important because it houses additional fuel, supplies and fighters, as well as all of your friends and relatives. They become endangered once you enter one of the more advanced scenarios, such as "Small Invasion" or "Full Invasion" or "Massive Onslaught."

In these advanced modes, you begin to experience tactical combat, as numerous enemy mother ships appear and begin viciously pumping out planes and tanks. You'll need to master the use of the heat-seeking and guided missiles to succeed against these vermin. Strategically, you must defend your base and the other installations, find and destroy the mother ships, and make sure your shield power stays sufficiently high to survive the hits you're going to receive. This is a lot to keep track of, but once you get into the game, you'll find it hard to stop.

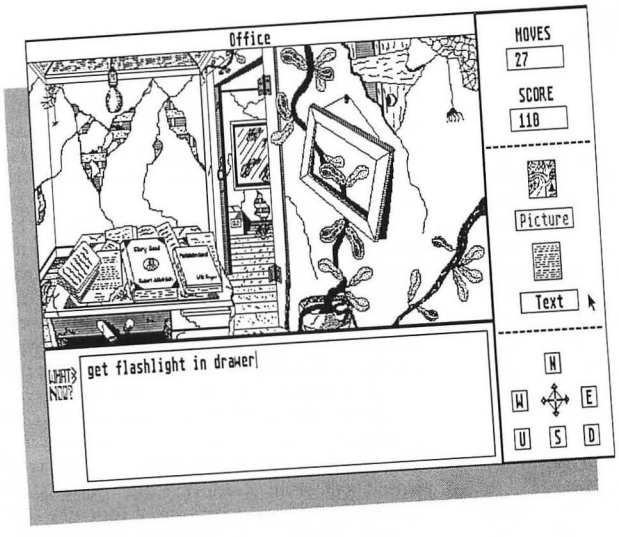
On the negative side, only one person can play at a time. We had one player control the stick, while a second manned the keyboard. This had the additional virtue of simulating a two-seater craft, with a pilot and navigator. It also made a tough game a little easier. Other negatives: high scores are not saved to disk, and you must have an ST with TOS in ROM. Positive aspects were a comprehensive albeit colorless instruction manual, and albumlike packaging which is functional and looks good on any shelf. The Apple version's graphics on the back cover do not do justice to those achieved on the mighty ST. The images are sharp and fast moving, the controls *very* responsive.

Overall, a high rating has to be given to Electronic Arts' **Skyfox**. Its series of rousing, blood-pumping battles are enough to get even the most jaded arcade junkie excited again. Even at this high price, it is a bargain. **Skyfox** is out of this world. ///

Gateway

ACTION SOFTWARE
69 Clementina St.
San Francisco, CA 94105
(415) 974-6748
High or medium resolution $39.95

by Victor T. Albino



In *Gateway*, travel to another dimension, but keep the evil from crossing back with you.

Gateway is the latest text and graphics adventure from Pryority Software, now sold and distributed by Action Software. An earlier Pryority program, **Forbidden Quest**, was one of the first adventures available for the ST series. **Quest** was a port from the Macintosh and was thus brought to market quickly. This early availability, combined with creative packaging and mouse assisted operation, helped to make it popular among software-starved ST owners. These days, new programs are appearing almost daily, and there is a wide choice of titles. Can the success of **Quest** be repeated? In my opinion, the answer is yes.

Gateway is a thoroughly delightful adventure with an exciting story line, a nice parser and easy-to-understand commands. It seems that your great-uncle Bertrand, whom you haven't seen in twenty years, has died and left you an old brownstone office building filled with musty artifacts. As you rummage through this collection of antiques, you find your uncle's journal. Its brittle, yellow pages tell an incredible story about a gateway to another dimension that lies somewhere in the old building. It's up to you to find the portal, explore the strange world that lies beyond, and prevent the evil on the other side from crossing into this world.

Like **Forbidden Quest**, **Gateway** is a conversion of a program originally written for the Macintosh. The manual was obviously intended for this other version. Still, **Gateway** exploits the ST's special features. Beautiful color or monochrome graphics can be loaded into a RAMdisk for almost immediate access. There are multiple windows and drop-down menus, a movement compass operated with the mouse—just click on the direction you want—and the capability of saving an unlimited number of games. Best of all, you don't have to pay extra for a hint book: the hints are always available on-line through a drop-down menu. You can pick both the subject you're curious about and the level of help you desire, from a subtle hint all the way to the complete answer. Or you can click on specific items shown in the pictures to seek additional information about them. You can also point and click at doorways or paths to move through or along them. I only wish there were more graphic screens included in the game.

The **Gateway** package also gives you three colorful illustrations called "Artext" prints, which resemble the covers of science fiction novels. These not only add excitement and realism to the game, but they also provide clues not given in the text description.

The normal features of a well-written text adventure are also present. The parser has an extensive vocabulary, and accepts full sentences as well as multiple commands. You can obtain detailed descriptions of your surroundings or speed up the game by selecting the "brief" mode. If you have a printer, another option allows you to make a transcript of the game as you play.

My most serious complaint concerns the incomplete and inaccurate information given about the process to follow when saving and restoring games. The manual says that you can put your saved games on a separate disk. But it doesn't mention that you must also have the program files on that same disk, or the program may lock up when it tries to resume a game. Fortunately, **Gateway** is not copy protected, so making a backup disk from which to run the program, and the saved games, is easily accomplished.

A nice feature is the ability to give a saved game any legal filename. This helps you to find the exact game you want to reload. What the documentation doesn't mention is that these names must have an extension of *.SAV*. If they don't, the names won't be automatically displayed when you select the "restore game" option.

According to the manual, there is a panic button located in the lower right-hand corner of the screen, that you can use in case of an emergency. Emergencies are defined as having your wife (or husband or parent or boss) walk into the room while you're playing the game but are supposed to be doing something else. Alternatively, you can type the word *panic*. This seemed an ingenious (and potentially useful) addition, and I was anxious to try it. Perhaps a faked spreadsheet would magically appear, only to vanish just as quickly once the danger was past. Unfortunately, there was no panic button. Apparently, this feature *wasn't* ported from the Macintosh. Undaunted, I typed the word *panic*, only to read: "Sorry, you'll have to cover for yourself."

For some reason, when a saved game is loaded, the background color of the window changes. The graphics aren't affected, but the original pleasant gray window turns into a bright blue or red that isn't as restful during long sessions at the keyboard.

All in all, though, **Gateway** offers the best of both worlds. It's a rich text adventure with exciting graphics. This combination creates a game that is, above all else, a lot of fun. ///

Victor T. Albino is a hospital administrator, self-taught computer enthusiast and vice-president of his local Atari user group. A number of his programs and articles have appeared in computer magazines, and he's received awards in educational programming and computer graphics.

ST Star Raiders

by Rod Zdybel, Jerome Domurat (based on the original by Doug Neubauer)
ATARI CORP.
1265 Borregas Avenue
Sunnyvale, CA 94086
Low resolution $29.95

by D.F. Scott

Playing the new **ST Star Raiders** reminded me of watching *Star Trek: The Motion Picture*. In fact, there's more than one parallel between the two. When the movie was announced, fans of the TV series were hoping that the grander effects, a larger screen and a bigger budget wouldn't wholly overwhelm the human conflict and intelligent dialogue that had held the series together. With **ST Star Raiders**, many of us—especially those who had been disappointed with **Star Raiders II** for the XE—hoped that splashier graphics and liberal use of RAM wouldn't detract from the intensity of the original game.

The fans who trekked to the theaters didn't want a story line that would degenerate into a shoot-'em-up, but one that would also explore matters of human resourcefulness, strategy and diplomacy. Similarly, **Raiders** devotees usually abhor shoot-'em-ups, preferring the format Doug Neubauer created in his original **Star Raiders**, which allows the player his choice of strategy and tactics.

Were expectations satisfied? With the movie, there were those who felt that, sure enough, there was too much concentration on special effects, updated gadgets and new wardrobes. As for **ST Star Raiders**, some will conclude that the new format and graphics interfere with the player's becoming totally involved in the game.

On the whole, **ST Star Raiders** is a good production; I'll explain why I think so, momentarily. First, for those of you who aren't yet **Star Raiders** fans, here is its premise, a variation of one of the very first widely distributed computer games, probably written at Dartmouth in 1968 and called, not surprisingly, **Star Trek**. . .

Yours is the only operational fighter vessel your side has in the galaxy. The galaxy map is two-dimensional and divided into "sectors." Your objective is to protect your starbases. These are good for refueling your ship and keeping it in good repair, but their hull armor seems to be made of tin foil.

You're outnumbered—at the highest level scenario, usually by about seventy to one. Your ship is alone; the enemy travels in fleets. To destroy a starbase, enemy fleets have to surround it on all sides and remain in position for several seconds. The game ends when your resources are exhausted, when you're destroyed, or when you vanquish the entire enemy armada (you win). Usually you don't win.

So much for the common elements; what are the differences between the 8- and the 16-bit **Raiders**? First of all, there's the split-screen overlay. Eight-bit players will recog-

King's Quest II: Romancing the Throne

SIERRA ON-LINE, INC.
P.O. Box 485
Coarsegold, CA 93614
(209) 683-6858
All resolutions $49.95

by Bill Kunkel

Roberta Williams' **King's Quest** was a groundbreaking attempt to move beyond the text and illustrated adventure formats that dominated the 48K era. **King's Quest** initially appeared under the IBM label as an early release for the 128K PCjr, and Williams made brilliant use of the extra memory to design a new type of joystick-driven adventure.

King's Quest combined elements of the traditional text adventure (verb-noun commands) with sound and graphics more typical of an arcade game. Granted, the adventure was a simple one, and the universe in which it was played wasn't very large, geared as it was toward younger players. But the novelty of the format and the well-written plot caught the attention of a wide range of gamers, and **King's Quest** became a major hit. Later, Sierra was able to release the program for several other computer systems.

King's Quest II: Romancing the Throne takes place several years after the end of the first game. The player, in the role of Sir Graham, is now King Graham, the just and beloved ruler of Daventry. All would be milk and honey, but for one problem: the King has no Queen!

One day, while lolling about the throne room, the King has a metaphysical encounter with his magic mirror, which shows him the image of a beautiful damsel imprisoned in a quartz tower. All of this is played out in an attractive prelude sequence, which veteran players can opt to skip over.

To reach the maiden, Graham must pass through a series of mystical doors in a nearby kingdom where things aren't nearly as conventional as they are in Daventry.

Romancing the Throne is superior to its predecessor in plot, graphics and playability. The world in which this sequel unfolds is much larger and more diverse than the setting of **King's Quest**. There's a dank and poisonous river, a crotchety dwarf's amazing underground lair, the castle of a vampiric count, and much, much more to enthrall the player.

Like its predecessor, **Romancing the Throne** is ostensibly a children's adventure, full of delightful fairy-tale images (Little Red Riding Hood has an interesting cameo, as does the wolf in granny's clothing). It's totally devoid of violence (a player who tries to kill an innocent character will be instantly banished from the game. But, once again, Roberta Williams' marvelous imagination will enchant anyone who so much as walks past the monitor while the game is in play.

King's Quest II also benefits from having been released by Sierra. Left on her own, without any creativity-stifling outside influences, Williams has fashioned a

nize the situation I call the "sandwich"—being pounded on both sides by two ships when you can see, and fire, in only one direction. The 8-bit game had a gizmo called the "tracking computer," which automatically flipped between fore and aft views each time the object being tracked changed ends, and which would go absolutely nuts in a sandwich—like a sports cameraman trying to find the ball. Many players had to *disengage* tracking in order to see what was happening.

In **ST Raiders**, the *A* key turns on the aft view in the second screen, set in a rather cluttered console. This way, the forward viewscreen remains forward at all times. The tracking computer still exists, but is rarely used for anything besides "double-zeroing" in on a target.

The big question about this game is whether split-screening was a good idea. You see, in order to fire the aft guns, you have to press the SPACE BAR. Holding the joystick and trying to battle a target up front while reaching over with a spare finger to control the aft gun is—well, inconvenient. Try dialing a phone and tapping out some morse code at the same time.

The other major change in the ST version regards galaxy representation. Eight-bit players will recall that their galaxy was rectangular with square sectors. The ST's version has rectangular sectors—ninety-one of them. A good **Raiders** player, the axiom goes, plays the map. So let's look at the map.

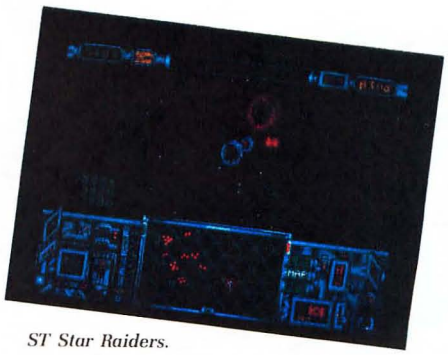
On the old one, the "Zylons" traveled in fleets, initially of one to four fighters, although you could fight only two at once. From any sector, a fleet could travel in any of eight directions; in rare instances, it could split into smaller fleets. The usual Zylon strategy was to systematically reduce starbases, converging upon and destroying them one by one, sometimes retreating from one base to attack another. Fleets moved (or didn't) every fifty seconds according to your ship's chronometer.

ST Raiders fleets are composed initially of two to six fighters, and all six can be engaged at once within a sector. On the map, each fleet can move in any of twelve directions, and a starbase can be surrounded on three sides instead of four. The tactical algorithms of this game seem to dispose fleets in such a way that equal numbers loiter in the proximity of each starbase, and several bases can be attacked at once. But fleets do not split in this version, so a one- or two-ship fleet is tactically as powerful as a fleet of six.

In the old **Raiders**, I found it effective to take out the large fleets first, thus isolating the smaller ones within cordons of starbases before finishing them off. In **ST Raiders**, the best strategy seems exactly the opposite: eliminate the small fleets first and keep the large fleets spread out, motionless, two around each starbase.

There is no fifty-second "motion clock" in this game; in fact, if the ship's chronometer seems Greek to you, that's because it is. No offense to the Greeks, but the clock is meaningless. Anyhow, a base is destroyed and turned into two more fighters after being surrounded for almost any random amount of time—sometimes as little as fifteen seconds.

As for individual fighters: there are seven types of target, each with its own movement strategy. Some are dangerous enough to be excluded from easier play levels. The manual gives them rather silly names, so I took the liberty of giving them sillier ones. First, there's the "fuzzball," which hangs around like a half-deflated basketball. No pattern to its movement; it's just an asteroid with a gun. Next, the "boxer," a telephone booth with boxing gloves. It tends to get in your blind spot (above you) and stay there. Then there's the "bicycle guy," which looks like a Kawasaki. It comes close, fires a few rounds and maintains range. "King Tut" looks like an Egyptian space shrine, dodges fire extremely well, keeps its distance, knows how to follow you on parallel course. "The Minus Sign" (the manual calls it "Subtractor," I admit) flies between other ships, using them as shields. Finally, "Orion" (after the old **Starfleet: Orion**) is O-shaped, dances around like Linda Fratianne, fires often, hides occasionally in the blind spot.



ST Star Raiders.

Fighters all suffer from the same weaknesses, despite their varying strengths. The first and most obvious is that it is far easier to hit a far-away target than a close one. Just "double-zero" it on the tracking computer, fire and wait for the debris. As for their second major weakness, I'll quote Spock from *Star Trek II*: their patterns "suggest two-dimensional thinking." As you watch the long-range scan on the second display (yellow ships are above you, orange below, green on your equatorial firing level), all the targets tend to remain near the same level. Thus it's preferable to evade ships vertically rather than horizontally.

That's only one of the reasons why this game seems so two-dimensional. Two-dimensionality does not necessarily make for a bad game: chess comes across pretty well. But **Raiders** is supposed to be a *space* game, not hypergalactic backgammon. Maybe I'm precocious, but . . . Star Commander, Class 1, on my third mission in **Star Raiders**?

Still, the old **Raiders** intensity is there. Some of the 8-bit's fun trademarks are gone—the double-barreled torpedo chutes, for example, have become a straight-firing, needle-nose gun pair—but there's still a game to be found in this package. ///

King's Quest II.

whimsical wonderland from her fairy-tale database, full of humor and unforgettable images.

It is unfortunate that the ST version is in no way enhanced over editions published for computers with less memory and graphics resolution, but there's no faulting this game's quality. Even the crustiest computer curmudgeon will be romanced by **King's Quest II**. ///

Make the connection!

ANALOG Computing on Delphi puts you on-line with the world.

Delphi, an on-line, full-service information network, offers news and sports from the Associated Press, weather reports, movie reviews, shopping services, travel information, and more.

ANALOG Computing, the #1 magazine for Atari owners, brings you the Atari Users' Group on Delphi. We offer a message forum and an extensive database for up- or downloading—all from as little as 10 cents per minute from most U.S. cities, with no additional telephone charges and no extra charge for 1200 or 2400 bps. We'll use the group's conference feature for electronic meetings with well known Atarians and, of course, **ANALOG** staff. Bring on your toughest questions!

Special rates

Subscribers to **ANALOG Computing** or **ST-Log** may join without charge, and will receive a free lifetime Delphi membership, a Delphi Command Card and $5.00 of line-time credit applicable to their account. If you purchase the *Delphi Handbook*—the highly detailed manual on using the whole Delphi system—for $29.95, you will get an additional $20.00 of line-time credit. And you can subscribe to either **ANALOG Computing** or **ST-Log** directly, while on-line, to be eligible for these bonuses.

How to connect

First, select a data communications network: Telenet or Tymnet (in the U.S.), or DataPac (in Canada). In the Boston area, dial Delphi direct (617-576-0862). To determine your local Telenet number, dial 800-TELENET or 703-689-5700 (in Alaska, 907-264-7391). To obtain a Tymnet number, call 800-336-0149. If you have difficulty, call Delphi at 800-544-4005 (in Massachusetts, 617-491-3393). Current subscribers to **ANALOG Computing** or **ST-Log** should type *JOINATARI* when asked for user name. When asked for a password, type *ANALOG*. Those who wish to subscribe to either magazine on-line should, instead, type *SUBSCRIBE* at the password prompt. Once on Delphi, you'll find our group on the "Groups & Clubs" menu. To get there, just type *GR.ATARI* from the main menu prompt.

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

P.O. BOX 23, WORCESTER, MA 01603

Cards — Version 2.0

MICHTRON
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700
All resolutions $39.95

by Matthew J.W. Ratcliff

Cards from MichTron is a new ST program that can play five different card games. Tease your brain with an unusual version of Solitaire, or try Klondike (the traditional "seven-stack" solitaire), or Poker Squares. You can hone your cribbage skills, or up to four people can challenge the house dealer in Blackjack. These games all come in a single program file. The program has been advertised as Solitare (*sic*—this is MichTron's spelling; we'll do it the right way in this review), but the final version has been renamed **Cards**.

The graphics are sharp. The cards are colorful and easy to read, and are always arranged so that you can see all the cards in a stack. The dealing effects are fast, and the playability is quite good.

The documentation is a bit too brief for my taste. It is assumed that you have some familiarity with every game in **Cards**. You are referred to the *Official Rules of Card Games* (56th edition, published by U.S. Playing Card Company) for game rules, but I feel that these should have been included in the manual.

Once **Cards** is loaded, you can pull down a menu of game selections or adjust some playing features from an options menu. For those who prefer function keys, F1 through F5 will start one of the games, while F6 through F10 activate various playing options.

Solitaire is the most unusual game in the **Cards** package. You won't find it in the book mentioned above. It was taught to the program's author, John Weaver, by his mother, and he tells me he couldn't find references to it in rule books anywhere.

The game begins as all fifty-two cards are dealt, face up, to the board in groups of eight. The object is to move all the cards to suit stacks, much as in Klondike.

Each top card can be played to the suit stack in ascending order (ace through king), or to any other stack that has the next higher card of the same suit. Just point and click, and the card is played automatically, if possible. When no other plays can be made, all the cards left on the board can be reshuffled and dealt again. You can do this twice. The game is then over if no more plays can be made.

Klondike is the traditional seven-stack solitaire that most people are familiar with. The deck costs you $52.00 (you start with −$52.00), and each card played onto one of the four suit stacks pays $5.00.

The computer shuffles and deals. You can drag your cards to the appropriate stacks in descending order from king to deuce, alternating suit colors. If no play is available, you can click on the deck to play a single card. If an ace or a card that can be played to a suit stack pops up, you need only point and click to send it to its destination automatically.

Klondike is the game I have played most often in **Cards**. I find it highly addictive, but frustrating. My biggest complaint is that *you can't cheat!* And it's so difficult to win this form of solitaire that cheating is almost a necessity.

For example, it would be nice if you could use the UNDO key to revoke the last play and attempt an alternate route. (Not "according to Hoyle," but a commonly accepted "play option" in *my* book.)

Another example: since you play a single card at a time from the deck, you can make only one pass through it. But it's very common to play solitaire in "three-card" style, where you count three cards and must play the top one first to get to the ones below. You are allowed to play through the deck as many times as you wish, provided at least one card was played in the previous pass.

Also, it would be nice if you could peek at the hidden cards and perhaps play any face card (instead of kings only) on an empty slot. I would like to see high scores saved, with points deducted for each of the "cheat features" used.

According to the manual, if you reach a "winning position," where all cards in the stack and building stacks are revealed, and the draw stack is empty, the remaining cards are played to the suit stacks automatically. I haven't won, so I don't know if this in fact happens. I got close once, but the program crashed, whether because of a bug or a hardware glitch, I don't know. The problem has not recurred in the many games I've played since.

Klondike is played "according to Hoyle," with no cheating allowed. Since the computer does all the shuffling and dealing for you, the game is played very quickly (and you lose much sooner).

Poker Squares is another form of solitaire, where you play cards from the deck onto a 5-by-5 grid. The object is to build the best combination of poker hands, five horizontally and five vertically. Points are awarded on a scale directly proportional to the hand's value in a real poker game. High score (but no player name) can be saved to disk. This is the only game in **Cards** where a high score may be saved.

In Cribbage, one person plays against the computer. The game is controlled via mouse or with the number keys. The program plays a pretty fair game, but it would be nice if it allowed a second player of the human persuasion.

Up to four people can play Blackjack against the (computer) dealer. Each player enters his name, starting dollar amount and bet per hand. You should start with a relatively low value ($200 to $500). If your winnings come to a total of more than $999, the display drops a digit: when you hit $1000, it looks as though your money has fallen back to $100. This is a minor limitation in the display routine, but the program does keep track of your winnings properly at all times.

This version of Blackjack is complete, and adheres to "Vegas house rules." The dealer gets one card down, one up. Each player gets two cards up. A blackjack pays one and a half times the bet.

After the deal, a flashing message appears under the active player's cards. You can take a hit (extra card) by pressing the number 1 key. You press 2 to stand pat or 3 to "double down" (double your bet while taking only one card). If your first two cards are a pair, you also have the option of pressing 4 to split into two separate hands.

Drawing five cards with a total of 21 or less beats all other hands, except a blackjack. When everyone has bet, the dealer reveals his card and plays. He must take a hit on a total of 16 or less, and stand on 17 or more. A push occurs when you tie the dealer, in which case no money is lost.

It may seem odd that Blackjack uses number keys for control, rather than letter keys or the mouse. But this works well for two players, where one can use the nu meric keypad and the other the main keyboard. With more than two players, the best system is to have one player make all the entries.

To sum up, all the games are well done. Graphics are superb, and the user interface and mouse controls are complete. My complaints are few. Klondike needs "cheat factors." I think all the games should allow you to save the high scores (and players' names). Blackjack could then start with a fixed stake amount, making a high score meaningful. But even without these enhancements, all the **Cards** games are excellent and will provide many hours of entertainment. //

Super Cycle

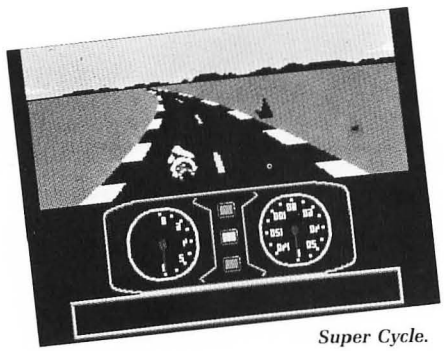
EPYX
P.O. Box 8020
600 Galveston Drive
Redwood City, CA 94063
(415) 366-0606
Low resolution $39.95

by Bill Kunkel

Super Cycle is a motorcycle racing game that might have run a truer course on the Atari 2600 than on the 520 or 1040ST. Certainly the graphics show off little of the ST's capabilities. You're steering a generic-looking chopper, seen in the foreground of a rather bland playfield, to the left and right as other cyclists try to pass you.

Your cycle has three gears, indicated on-screen by three vertical lights. You shift with the joystick button.

Available options are mostly trivial. You can choose your bike's color, as well as the style and color of your racing leathers. Now, you proceed to the track for some of the dullest competition this side of Tag-Team Tiddlywinks. You get the standard **Pole Position** view, but without the sign-posts or changing horizon topography most similar games have.



Super Cycle.

It isn't even possible to nudge another racer off the road; your bike simply bumps off any others it encounters as if it were meeting an ambulatory stone wall.

Racing games are few and far between on the ST, but even the most rabid road warrior would be well advised to pass on **Super Cycle**. You don't want to encourage this sort of thing. //

Championship Wrestling

EPYX
P.O. Box 8020
600 Galveston Drive
Redwood City, CA 94063
(415) 366-0606
Low resolution $39.95

by Bill Kunkel

The difficulty of simulating professional wrestling with its show biz trappings and gymnastic ultra-violence is vividly demonstrated by Epyx' **Championship Wrestling**. The two previous computer wrestling contests (Data East's coin-op, **Tag Team Wrestling** and Mindscape's **Bop 'n' Wrestle** which this game resembles) dealt unsuccessfully with the joystick's inability to serve as a sophisticated command device. Once you position yourself, how do you use the joystick to call for a punch, kick, suplex, hiproll or any of the several dozen holds that the pro grapplers routinely execute?

Tag Team Wrestling opted to ignore the problem. The player used the joystick exclusively to position his matman; the computer selected and displayed the appropriate holds. **Bop 'n' Wrestle** attempted to give the player more control and, in the process, produced a game that cannot be played by mere mortals.

Championship Wrestling offers twelve moves, not counting the pin. This would be a manageable number, if there were any reasonable way for the player to execute all of them. Some moves are possible only when the opponents are in close (for example, punch, kick), while others (flying dropkick) work only from a distance. Still others must be performed as part of a sequence. There are so many different joystick positions that simply learning them all will require more hours than mastery of the *Kama Sutra*, with considerably less entertaining results.

The wrestlers are profiled at great length in the documentation, and descriptions of their "specialty" holds are given. Having custom holds is nice, but in "real" wrestling, such maneuvers are only used as either "submission" or "finishing" holds. Here, they supplant the headlock, which makes no sense at all.

Most of the competitors are generic types, such as Zeke Weasel, the mountain man; or the Howling Manslayer, a weird permutation of the classic Indian grappler, who actually wrestles with his headdress on, or K.C. Colossus, the obligatory Hulk Hogan swipe. Each of these characters appears in a nicely animated close-up window while his "theme song" plays in the background. Inane quotations underneath ("I'll rip yer ears off and feed 'em to ya!") are a lame attempt at capturing the ambiance of the wrestling "interview."

The scoring system has only a slight bearing on the match result. It's possible to be ahead by over 200 points, with your opponent out flat on his back, only to be unexpectedly pinned when he suddenly executes one move successfully.

A "strength meter" lets you monitor your physical condition; a useful feature, since you can collapse from exhaustion while in the midst of thrashing your adversary, another thing that never happens on TV.

Championship Wrestling often betrays an ignorance of how pro wrestling works. There's a vanity board listing "World Records," à la **Summer/Winter/World Games**, but the ratings are based on point totals (which pro wrestling doesn't use), rather than win-loss record.

In addition, the game is very fast, with 3-second victories no rarity (especially by the computer, which doesn't need the joystick and never forgets holds). Even when you win, most of the "strategy" involves making a beeline for your opponent and whaling away on the action button until someone scores a pinfall.

There are some nice touches, however. The vertical suplex looks fantastic, as does the airplane spin. It's even possible to come off the top rope (though you'll come to the end of yours trying to figure out how). At times, things get very close to the feel of

Shanghai

ACTIVISION
P.O. Box 7287
Mountain View, CA 94039
(800) 227-9759
In CA (415) 940-6044, 940-6045
Low resolution $44.95

by Bill Kunkel

Perfectly executed, truly original computer games are so rare that it's difficult to keep from going overboard on the superlatives when one does come along. **Shanghai**, Brodie Lockard's classic strategy contest (now adapted for the ST by Grossman, Covitz, Anderson and Snyder) is one of that rare breed.

Inspired by the ancient Chinese pastime of mah-jongg, **Shanghai** requires you to clear a board of 144 strategically layered tiles, by matching pairs. Playing is easy; you click the mouse on desired tiles and they disappear. As simple as it sounds, **Shanghai** is the most compelling computer game since **Pac-Man**. Once hooked, you become a virtual Shanghaiholic, sneaking off for "one more game" between bites of dinner and station breaks.

Shanghai.

The tiles are reshuffled and restacked before each game, so it's always a fresh contest. Not every round can be brought to a successful conclusion. However, when you do succeed in eliminating all the tiles, a dragon appears and breathes some congratulatory flame.

Visually, **Shanghai** is a knockout: delicately rendered tiles, "embossed" with a colorful symbol against a white background. Of the 144 tiles, 108 are suits, 12 dragon, 16 wind, 4 season and 4 are flower tiles, each appropriately Oriental. The layered arrangement is known as the "Dragon" (hence the incendiary lizard who appears at the dénouement), with tiles stacked in piles from one to five high. In order for you to be able to remove a tile from the formation, it must be "free" (able to slide from the board either to the left or right). The board itself is seen from above, and the height of a given stack is determined by the color and thickness of the border around the topmost tile.

Shanghai, essentially a one-player game, also offers Team Effort (any number of players take turns), Tournament (the top five scores are recorded) and Challenge (like Team Effort, but with a time limit). Games can be saved to disk at any time and a Help menu lets you back up any number of moves, restart, display possible moves and (a forfeit option) peek at what lies beneath any given tile.

Shanghai may not be every gamer's cup of tea, but, to my mind, it's the most elegant entertainment to come along for the thoughtful player, since reversi was computerized. Can the first national **Shanghai** Tournament be far away? //

pro wrestling. But the ring is much too small, and so are the wrestlers. In clinches, it's impossible to tell who's doing what to whom, especially when opponents look alike. When trying to bounce off the ropes, you often "stick" suddenly, and are at your opponent's mercy.

But the basic flaw of **Championship Wrestling** is its overly complicated play system. Trying to remember the joystick moves for twelve different holds at the same time that you're trying to get from place to place in the ring is just too much. Maybe this will only add to the challenge for fanatics; others may prefer to stick with the TV version. //

Step 1
SOFT WARES

by Maurice Molyneaux

Last time, I talked about the various types of system hardware and their capabilities. This month, we'll look at a variety of subjects, most of them related to disks and programs. As always, my goal is to clarify or point out details inadequately treated or ignored by Atari's ST manuals. But first . . .

The Three Laws of Computer Errors.

Law 1. — Ninety percent of all computer errors are operator related.

Law 2. — Eight percent of the remaining errors are software related.

Law 3. — The remaining two percent are the @%$&# computer's fault!*

The point I'm trying to make is: when something goes wrong with your ST, first ask yourself what *you* might have done wrong before pointing an accusing finger at the program or the computer.

The most common user boo-boos include overwriting existing files, deleting files you didn't mean to, formatting disks with important data on them, not regularly saving your work (which is just like *asking* for a power surge to erase RAM), not making backup disks, and, the greatest sin of all, not reading or consulting the manual. So, if you lose something because of one of the above, blame yourself!

The case of the missing TOS.

One of the most flagrant mistakes in Atari's 520ST manual (not the 1040ST's) is right up front on page 10, which shows an ST screen displaying an image of a disk drive and floppy disk. Step 3 in the instructions on that page tells you to "hold the TOS System Disk" and insert it into the drive when this screen appears (right after you turn the computer on). But of course, you never see this screen when you turn your ST on—and where's that System Disk, anyway? The answer is simple: there *is* no System Disk, bacause the STs don't need it anymore. The first STs had to load their operating system (GEM) from disk because Atari was still making changes in GEM itself. Later, Atari put TOS (another part of the operating system or OS) and GEM into 192K of ROM (six chips), in every ST manufactured after 1985. Most early ST owners bought the chips and had them installed, and keep the System Disk only for special occasions (see the next section.) Unfortunately, as far as I know, Atari never bothered to update the manual and eliminate this now useless step.

TOS it in RAM.

Even with TOS and GEM in ST ROM chips, it is still possible to use the original disk-based, RAM-consuming TOS and GEM system that was packaged with the 1985 520STs. This old disk-based TOS is far more buggy (i.e., has more errors) than the current ROM version, but it has a few advantages. First and foremost, some programs written when TOS was disk based won't work with TOS in

When you want to talk Atari

XL/XE HARDWARE

INTERFACES

ICD
P:R Connection.....$59.99
Printer Connection.....49.99
Supra
Microprint.....39.99
1150.....39.99
1151 (1200 XL).....39.99
Xetec
Graphix Interface.....39.99
Atari
850 Interface.....109.00

COMPUTERS

Atari 130XE **$129.00**

65XE.....97.99
800 XL.....63.99
800 XL Package (1050 Drive, 1025 Printer, AtariWriter, Joystick & Diskettes)....349.00

MEMORY

Axlon
32K Board.....24.99
48K Board.....49.99
128K RAM Disk.....119.00

MODEMS

Atari
SX212 300/1200 (ST).....94.99
835.....24.99
XMM301.....44.99
Anchor
VM520 300/1200 Direct Connect (ST)134.00
Volksmodem.....49.99
Volksmodem 1200.....129.00
Supra
1200 AT or ST.....139.00

ST HARDWARE

1040 Monochrome System **$699**

Includes: 1040ST, 1 mb RAM with 3½" drive built-in, 192K ROM with TOS, Basic, Logo, ST language, power supply and monochrome monitor.

1040 RGB/Color System.....859.00
520 Monochrome System (Includes: 520ST computer with modulator, disk drive, mouse, logo, Basic and monochrome monitor)499.00
520 RGB/Color System.....749.00
SM124 Monochrome Monitor.....129.00
SM1224 Color Monitor.....309.00
1040 Computer (no monitor).....569.00
Call for prices on the new Atari line of computers: 520ST FM/the increased RAM 1040 systems/Atari PC compatible/Atari Laser Printer/XDM Daisywheel.

DRIVES

Atari
1050 Disk Drive (XL/XE).....$139.00
SHD204 20 Meg Hard Drive (ST)...599.00
AA314 DS/DD Disk (ST).....199.00

Atari AA354 SS/DD Disk (ST) **$99.99**

Indus
GT Disk Drive (XL/XE).....179.00
Supra
20 Meg Hard Drive (XL/XE).....679.00
20 Meg Hard Drive (ST).....569.00
30 Meg Hard Drive (ST).....Call

PRINTERS

Atari
1020 XL/XE.....$29.99
1025 XL/XE.....124.00
1027 XL/XE.....119.00
XMM801 XL/XE.....199.00

Atari XMM804 ST **$189**

Citizen
MSP-10 (80 col.).....299.00
MSP-20 (80 col.).....349.00
Premier 35 cps Daisywheel.....499.00
C.Itoh
Prowriter 1550P (132 col.).....269.00
Prowriter 8510SP.....Call
Epson
LX86, FX85.....Call
FX86E, FX286E (240 cps).....Call
EX800, EX1000, RX100.....Call
LQ800, LQ1000.....Call
Juki
5510 Color Dot Matrix.....Call
6100 10 cps Daisywheel.....Call
6300 40 cps Daisywheel.....Call
NEC
ELF360 20 cps.....389.00
3000 Series Spinwriter 35 cps.....779.00
Pinwriter 560-660-760.....Call
Okidata
Okimate 20.....129.00
ML-182, ML-192, ML-193.....Call
ML-292, ML-293.....Call
Panasonic
KX-1080I 120 cps 80 col.....219.00
KX-1592 180 cps 132 col.....439.00
Star Micronics
NX-10 120 cps 80 col.....219.00
NX-15 120 cps 132 col.....379.00
Toshiba
P321 216 cps 24 pin.....479.00
P351 288 cps 24 pin.....1049.00

COMPUTER MAIL ORDER

CIRCLE #117 ON READER SERVICE CARD

... you want to talk to us.

XL/XE SOFTWARE

Access
Leaderboard Golf.....$26.99
Tournament Disk.....14.99
Accolade
Fight Night.....19.99
Hardball.....19.99
Atari
Various Titles from.....$3.49
Batteries Included
Paperclip w/Spellpack.....39.99
B-Graph.....37.99
Broderbund
Printshop.....28.99
Graphics Library 1-2-3 (ea.).....16.99
Cygnus
Starfleet I.....34.99
Datasoft
221 Baker St.....21.99
Electronic Arts
Touchdown Football.....13.99
Chessmaster 2000.....29.99
Epyx
World Karate Championship.....24.99
Infocom
Zork III.....29.99
Microprose
Top Gunner.....19.99
F-15 Strike Eagle.....24.99
Origin Systems
Ultima 4.....39.99
Roklyn
Gorf/Wizard of Wor/AT Deluxe/
Anti-Sub/Journey to Planet (ea.).....3.49
Strategic Simulations
Gemstone Warrior.....12.95
Sublogic
Flight Simulator II.....37.99
X-Lent
Typesetter.....24.99

AtariWriter Plus **$34.99**

DISKETTES

Maxell
MXD1 SS/DD 5¼".....$9.99
MXD2 DS/DD 5¼".....10.99
MXD3 SS/DD 3½".....12.99
MXF2 DS/DD 3½".....21.99

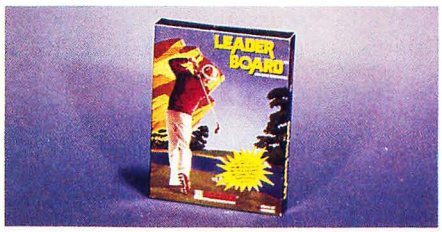
Sony
SCMD1 SS/DD 5¼".....8.99
SCMD2 DS/DD 5¼".....10.99
SCMFD1 SS/DD 3½".....13.99
SCMFD2 DS/DD 3½".....20.99
Flip 'N File Disk Holders
FF100 5¼" (100).....19.99
FF15D 5¼" (15).....6.99
FF737 3½" (25).....9.99
Curtis Protectors & Accessories
Emerald.....39.99
Safety Strip.....19.99
Universal Printer Stand.....14.99
ICD (XL/XE)
Sparta DOS Contruction Set.....28.99
US Doubler/Sparta DOS.....49.99
Happy Archiver (1050).....36.99
Rambo XL.....29.99
Multi I/O Board 256K.....179.00
Multi I/O Board 1 Meg.....299.00

ST SOFTWARE

Activision
Music Studio.....44.99
Antic
Cad 3-D.....37.99
Atari
Star Raiders.....19.99
Batteries Included
Degas Elite.....49.99
Paperclip Elite.....79.99
Cygnus
Starfleet I.....39.99

ST SOFTWARE

Ditek
STA Accounts.....174.99
Electronic Arts
Chessmaster 2000.....34.99
Epyx
Winter Games.....29.99
First Byte
Math Talk (Educ.).....39.99
Haba
Habawriter.....34.99
Hippopotamus
Hippo Backgammon.....29.99
Infocom
Zork Trilogy.....49.99
Michtron
Major Motion.....26.99
Microprose
Silent Service.....27.99
F-15 Strike Eagle.....29.99
OSS
Personal Pascal.....49.99



Access Leaderboard Golf **$26.99**

Paradox
Wanderer (3-D).....29.99
Polarware
Crimson Crown.....26.99
Psygnosis
Arena.....26.99
Sierra On Line
Winnie The Pooh (Educ.).....19.99
Sublogic
Flight Simulator II.....37.99
Timeworks
Wordwriter ST.....59.99
VIP
Professional (GEM).....144.99
X-Lent
Typsetter.....27.99

In the U.S.A. and in Canada

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

Outside the U.S.A. call 717-327-9575 Telex 5106017898

Educational, Governmental and Corporate Organizations call toll-free 1-800-221-4283

CMO. 477 East Third Street, Dept. B704, Williamsport, PA 17701

ALL MAJOR CREDIT CARDS ACCEPTED.

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

CIRCLE #117 ON READER SERVICE CARD

ROM. Furthermore, when TOS is loaded in RAM (it'll consume over 200K), you can actually modify it (impossible to do with the version in ROM, because ROM—as explained last time—is *Read-Only* Memory, and cannot be altered). This is desirable when running a program like **Format Plus** (mentioned later), which alters the OS in order to get a desired result.

If you boot your ST with a disk that contains the TOS.IMG file, the system will load it into RAM, and give control to it and not its own ROM-based TOS. Loading TOS off a disk (with desk accessories) takes about 30 to 35 seconds. The screen will go white during that time. By the way, it's best to turn off your ST before booting TOS from a disk, and turn it off again before rebooting it normally, because sometimes things don't go right with a mere RESET. Don't use this old TOS unless you have to, because, as stated above, it contains numerous bugs and occupies a fair portion of your computer's RAM, limiting the amount you can use. If you are interested in obtaining a disk-based TOS, contact your local ST users group, or ask your local ST dealer for information.

Track that sector!

While discussing hardware last time out, I mentioned Atari's two models of disk drives (SF354 single-sided and SF314 double-sided) and their respective storage capacities per disk (360K and 720K) in standard format. There *are* other formats you can use, but they usually require special software to create. One program in the public domain called FMTCOPY.PRG (by "Ron," who unfortunately didn't give his last name in the program) gives SF354 users three disk formats, and SF314 users six formats. They are as follows (the totals listed—in K bytes—indicate usable disk space, not total capacity):

1 . . . . . Single sided with 40 tracks (174.5K)
2 . . . . . Double sided with 40 tracks (352K)
3 . . . . . Single sided with 80 tracks (349K)
4 . . . . . Double sided with 80 tracks (709K)
5 . . . . . Single sided with 82 tracks & 10 sectors per track (399K)
6 . . . . . Double sided with 82 tracks & 10 sectors per track (809K)

Formats 1 and 2 yield the same capacity as single- and double-sided IBM 5¼-inch disks, and formats 3 and 4 the same capacity and format as IBM 3½-inch disks (these are also the ST default). Formats 5 and 6 are special: be forewarned that TOS and most programs will not sector copy these formats (in other words, the GEM desktop disk copy routines and most TOS copy programs won't work.) FMTCOPY.PRG features a disk utility that will allow you to copy disks in these formats.

Normally, you'll use the ST's own disk formatting option. But the ST's GEMDOS routines for disk formatting actually *inhibit* the disk drives from reading and writing at full speed. For those of you who want more speed, the September 1986 **ST-Log** section in issue 46 of **ANALOG Computing** presented a BASIC program (**Format Plus**) by Brian Duggan, which allows you to format disks that read/write faster than normal. A stand-alone version of this program (FRMTPLUS.PRG) is also in the public domain. In order to use either version, you'll have to obtain (if you haven't already gotten it) the *old* mid-1985 disk-based version of TOS (TOS.IMG). If you have a late-1985 disk-based TOS, it won't work (the program checks and will tell you if you've got the right TOS). You boot the old TOS, run the **Format Plus** program (you'll have to load BASIC first if using the original **ANALOG** version), format some disks, then turn your computer off and back on again without the System Disk, allowing the ROM-based operating system to take over once again.

Interestingly, the default ST disk format conforms to the IBM 3½-inch disk standard, and (according to an Atari representative I spoke to) a 3½-inch disk formatted on an IBM can be read by an ST. Of course, a double-sided IBM format disk cannot be read by an Atari single-sided drive.

Warning: IBM programs won't work on your ST without an IBM emulator! If you have one (they have been announced), consult your manual before trying to run an IBM program.

What's in a (file) name?

Atari's manuals omit this subject entirely. Filenames can help you in more than one way to find what you're looking for. The conventions for filenames are as follows:

All files can have a primary name of up to eight characters, a separator in the form of a period, and an extension of up to three characters. In most cases, you can only use letters and numbers for a filename and its extension; other symbols are not allowed. You cannot put a space in a filename, but you can use the underline character. The following are examples of illegal filenames:

```
STWRIT;.PRG
1ST WORD.PRG
SHOW_PICTURES.PRG
PICTURE.NEOC
```

You'll notice that the illegal filenames contain symbols like the semicolon and/or blank spaces, or they exceed eight letters in the primary filename or three in the extension, etc. GEM is usually smart enough not to allow you to enter illegal filenames when using the various built-in desktop options (e.g., "Show Info," "New Folder," etc.) However, it *is* possible to enter an illegal filename from a program which doesn't use the GEM item selectors and dialog boxes. Don't assume that it's safe to do something just because a particular program might allow it. A disk's file structure and directory can be scrambled and all of your data lost if you do something patently illegal. By the way, *never* use a question mark (*?*) or an asterisk (***) in a filename when saving. They are "wildcard" symbols (explained later in this article) and have other uses.

Choosing a filename is something you should consider carefully; the name and extension should clearly express what the file is or what it does. A **NEO-Chrome** picture file of a Klingon battlecruiser could be named KLINGON.NEO. The filename *KLINGON* describes the subject, while the extension tells you that it is a **NEO-Chrome** file. Another example: I currently use **ST Writer** for all of my article writing, so these files have the extension *STW*, and I know they are to be used with **ST Writer**. Some standards in filename extensions are as follows:

.APP	Program file (GEM)
.BAS	BASIC program (must run BASIC to use)
.C	C language source code file
.DAT	Data file
.DOC	Document/text file (sometimes for **1st Word**)
.GEM	GEM data file
.LOG	Logo program (must run Logo to use)
.NEO	**NEO-Chrome** picture file
.RSC	Resource file (for a GEM program)
.PRG	Program file (GEM)
.SYS	System file (used by the ST itself)
.TOS	Program file (TOS)
.TTP	Program file (TOS takes parameters)
.TXT	Text file

Folders.

"Folder" is just (in GEM) another word for directory. When you double-click the mouse on a disk icon in the desktop, the window displayed by GEM shows the contents of that disk's "main" directory. However, this directory can contain other directories (i.e., folders), as well as files, and those folders can in turn contain other folders. . . Each time you double-click on a folder icon or name, GEM changes the window to show the contents of the folder you just selected. On the other hand, when you click on the close box of a window, GEM returns you "upward" to the folder that contains the folder you were just "in," and displays its contents. If you close the window when it is displaying the contents of the main directory or folder, the window disappears.

The main reason for this complicated structure is to make things easier for you, the user. It allows you to organize files into related groups. Instead of having to thread your way through a huge listing of dozens of files contained on one disk, many of which probably have nothing to do with others, you can create folders to hold groups of files that go together. It's as though you had many little disks, each of which contains only a certain kind of file. One obvious example of this method in action can be found on the **1st Word** distribution disks, where the "Printer" folder contains all the files used for adapting **1st Word** to various printers.

There seems to be no simple way to rename a folder, other than creating a new folder with the name you want, copying the old folder's contents into the new one, and finally deleting the old folder. Note that when you delete a folder you also delete everything in it.

AUTO-mation.

There is a special folder that is not mentioned by Atari's manuals, and has been poorly and badly documented on bulletin boards and in newsletters. . . namely, the AUTO folder.

I'm assuming you use one particular disk to boot your

system every time (such disks are called, appropriately enough, "boot disks"). If you create a folder on your boot disk and name it AUTO, when you boot the ST any programs you placed in the AUTO folder will be automatically run *before* the system loads desk accessories and goes to the GEM desktop. So, if you had a program for setting up a RAMdisk that you wanted to run "automatically" when you turn on or reset your ST, you would copy that program into the AUTO folder and boot the system with the disk on which that folder resides.

An AUTO folder can be very useful, but it does have some limitations. The most serious is that any programs which use the GEM interface (windows, dialog boxes, menu bars, etc.) will not work if copied into an AUTO folder. This is because the system runs the AUTO programs *before* activating GEM itself. If a program tries to use GEM before GEM is run by your ST, the program will crash, and you will have to reboot (with a different disk this time!) Programs like **1st Word**, ST BASIC, Logo, etc., will not work from an AUTO folder. Another limitation is that, on a color system, all AUTO programs will be executed in low resolution—which can play havoc with programs designed to run in medium or high resolution.

Running programs which have to access data files from the AUTO folder can be a little confusing. If, for example, you have a boot disk on which a slideshow program is placed in AUTO, the picture files it must read *cannot* be placed in AUTO with it, because, if they are, the program won't be able to find them! This is because your ST doesn't treat AUTO as the currently active directory. It treats the main directory as active, so any such data files must be *there*. In such cases, put the program to "autorun" in the AUTO folder, and all its data files in the main directory.

What's a program?

A lot of novices have a tough time telling programs from other files, and sometimes don't even know what the difference is. As is so often the case, Atari's manuals don't help much, so I'll attempt to explain.

All files, program or data, are saved to a disk in the same manner. However, program files and data files are two entirely different beasties. A program consists of data that, when read by your computer, will instruct it to do various things. For example, running NEO.PRG will allow you to draw neat low-resolution pictures on your color monitor, but trying to run PICTURE.NEO will just not work. Why? The PICTURE.NEO file contains the data that **NEOChrome** uses to put your picture on the screen. Used by NEO.PRG, it's wonderful; without it, it's only a jumble of senseless numbers.

To sum up: a program file (including .APP, .PRG, .TOS and .TTP files) contains that sort of data which will directly instruct the computer to carry out various tasks, while data files (including .DAT, .NEO, .RSC, etc.,) contain information which the programs use in order to do the things you want them to. The only way your ST has of knowing whether a file contains a valid program is to examine the filename extension, so don't rename non-program files with an extension like .PRG; trying to "run" such a file will surely crash the system.

The asterisk's wild.

There are two symbols that are illegal to use when saving or loading a particular file, but are of great use when looking for specific file types in a directory. These symbols, the asterisk (*) and the question mark (?), are known as "wildcards," because, just as in card games, they can represent *anything*. The reason for having two wildcards is that each represents a different number of characters in a filename.

The ? wildcard represents one variable character, and, if you were to search a directory for all files matching the following...

```
DATFILE?.DAT
```

...the files DATFILE1.DAT, DATFILE9.DAT, and DATFILEQ.DAT would all be displayed. In other words, you're telling the system to display any filename that exactly matches the listed characters, with the exception of the ? in the eighth place (in this example) in the primary filename, where any character (or none) will do. **DEGAS Elite** in low resolution looks for any files with the extension *P?1*, which means PICTURE.PC1 or PICTURE.PI1 will both be listed.

The * wildcard means any number of characters (from *1* to *8* for a filename and/or *0* to *3* for an extension). If you used the following...

```
*.NEO
```

...as the parameter for searching a particular directory, any file with the extension *NEO* (such as: BEE.NEO, TRAIN.NEO, etc.) would be listed. Or, if you used...

```
*.*
```

...in a search, the system would list every single file in the selected directory, because the wildcard indicates any filename and any extension of any length.

You can use both types of wildcards in the same search. For example, in low resolution **DEGAS Elite** will search for all picture files with the extension *P?1*. The wildcard displayed in the item selector box looks like this:

```
A:\*.P?1
```

The *A:* designates the drive being searched, but the **.P?1* tells the system to look for any filename, with an extension that has as its first character *P*, as its third character *1*, and as its second, any character.

The new math.

If you're new to computing, you may wonder why there are no multiplication or division keys on the keyboard. Well, they *are* there; it's just that the common computer symbols for these functions are not the ones you learned in school. Add and subtract are represented as expected by + and –, but multiply and divide are represented by the symbols * and /. If you have a program or desk accessory which acts like a calculator, you may be able to use the numeric keypad and its math function symbol keys in place of clicking on the on-screen buttons. In such a case, the ENTER key often acts as an "equals" command. So, "six times nine divided by four equals..." would be entered by pressing *6*, *, *9*, /, *4* and then ENTER. The same math symbols apply in programming with BASIC

and Logo, but the syntax with which they are used varies depending on the language.

Putting the *boom* in "bomb."

There are a variety of terms used to describe a system failure, all of them picturesque. Your ST or a particular program you are running may "hang, hang up, lock up, crash, bomb," or even go "down in flames." I like to classify such failures under two general headings: "Lockups" and "Bombs."

A Lockup is where everything just stops working; the system becomes paralyzed either partially or fully. In such cases nothing may work, or perhaps you can still move the mouse pointer, but most of the things you try to do result in no action at all.

A Bomb is self explanatory, because, generally speaking, you will *see* bombs on the screen. The new user may not have experienced this (yet), but when something goes terribly wrong in a program or on the GEM desktop, the system will display a row of small cherry bombs (resplendent with lit fuses) on the screen. The number of bombs corresponds to a type of error known to the ST's "brain," the 68000 microprocessor chip. Two bombs represent a "Bus Error" (not Greyhound), three an "Address Error," four an "Illegal Instruction," etc.

Of Bomb crashes, there are also two types, from the user's point of view: what are sometimes called "soft" and "hard" crashes. A soft crash has occurred when you are running a program, something goes wrong, bombs appear, immediately disappear, and you find yourself back in the GEM desktop. A hard crash has occurred when the bombs appear and remain on the screen until you either reset or turn off the computer. Sometimes a soft crash will precede a hard crash. In such a case, the currently running program will soft crash back to the GEM desktop, where the system will then follow suit and hard crash. *Boom!*

At times, resetting your computer after a crash won't solve the problem, because certain memory locations will remain unchanged during the reset, and your system may crash all over again. Thus, it's advisable to turn off your ST for a few seconds following a crash, then restart it.

Getting on the "Parallel Bus."

Well, time to go again. In this and the last article we've covered most of the hardware and software points Atari's manuals missed, so next time it's down to the business of using the Graphics Environment Manager. Yes, we'll be cracking GEM in the next installment, so hang onto your mice—we're going to enter the world of windows, dialog boxes, installed applications, and a whole lot more. *Plus* a full-page quick reference sheet on common ST and GEM keystrokes for beginners. Until then... //

Allergic to all things Commodore and never bitten by Apples, Maurice Molyneaux purchased an Atari 800XL for animation work, but upgraded to an ST as soon as they became available. Currently slaving to complete the fifteenth draft of a science-fiction novel, he also masochistically churns out free-lance articles, artwork and animation on his ST, and hopes to dig out of a mountain of pending projects by the year 2000.

COMPUTER PALACE WE KNOW ATARI ST!

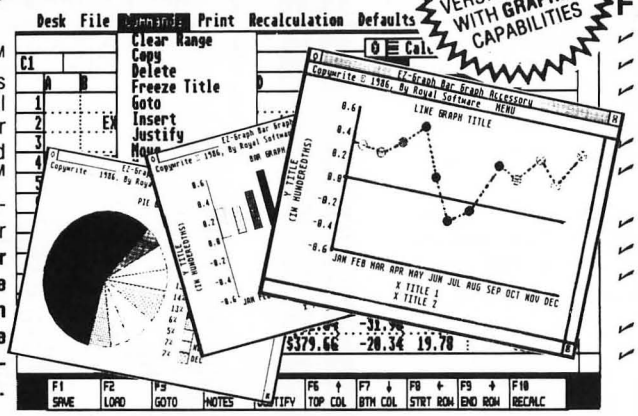
EZ Calc™

PUT EZ CALC™ TO WORK FOR YOU AND BENEFIT FROM THE RESULTS

NEW! Enhanced VERSION 1.2 NOW WITH **GRAPHING** CAPABILITIES

ONLY $69.95

EZ CALC™ is a fully implemented **GEM**™ based spreadsheet for home and business use. This is by far the most powerful spreadsheet available for the price. Better yet, all commands are mouse controlled for speed and ease of use. EZ CALC™ also uses **less memory** than other spreadsheets for the ST, leaving more room for your data and formulas. **If you've never used a spreadsheet before, you'll be amazed how easy** EZ CALC™ **is to learn and use. The experienced user will love the speed of a mouse controlled spreadsheet.** ★ **DEMO AVAILABLE FOR $5.**



FEATURES

- 300 columns by 999 rows
- Extensive use of **GEM**™ windows
- All commands are under mouse control
- Built in 10 keypad calculator
- On-line help windows (No commands to memorize)
- Built in sort routine
- Developed exclusively for the Atari ST
- 10 macros controlled by the function keys
- Split-screen capabilities
- Note Pad

GEM is a Trademark of DIGITAL RESEARCH, INC.
EZ CALC is a Trademark of ROYAL SOFTWARE.

GREAT BATTLES 1789-1865

NEW!!

ONLY $34.95

FOUR GAMES IN ONE

- **Napoleonic Battles and Civil War**

Features:

- 1 to 4 hours per game
- Realistic 16-color graphics
- 1 to 4 players and/or the computer can play any of the sides
- Optional limited intelligence of enemy positions
- See all movement, plots and unmoved units at the click of a mouse button.
- Multiple weapon types
- Infantry, artillery, cavalry, horse artill- ery, gunboats, and leaders
- Optional rules for hidden units, variable reinforcements

Battles Include:

- ☐ **AUSTERLITZ 1805**
- ☐ **WATERLOO 1815**
- ☐ **GETTYSBURG 1863**
- ☐ **SHILOH 1862**

Relive some of the greatest battles in history, such as Waterloo, Austerlitz, Shiloh and Gettysburg. You make leadership decisions just as General Grant and Emperor Napoleon did. Great sounds of battle and colorfully detailed maps aid in leading you to the final victory.

PAYROLL MASTER

NEW!!

- State tax information for multiple states.
- Prints Employee Checks - Address Labels - W-2 Forms - Employee List - End of Quarter/Year Reports - PLUS MUCH MORE!
- Ram resident for lightning fast access.

Only $79.95

Payroll Master is a **GEM** based payroll program that will give the average user the power to perform any payroll task large or small. All the information is available at a click of a button, so it is fast and easy to operate. No need to buy extra modules because the program has everything you need, for completing tasks from printing checks to detailed reports.

INVENTORY MASTER

INVENTORY MASTER is an easy to use **Gem** based inventory program that will do everything from keeping track of inventory to generating detailed printed reports. With **GEM** pull down menus you can execute fast and easy searches, sorts and printouts. It is so easy you can do it almost without looking at the manual. Additionally the program is written to be easily integrated with **Invoice Master**, giving you a powerful automatic inventory control system.

EZ DATA NEW

Only $49.95

- ☐ Fully **GEM** based
- ☐ Very powerful and versatile report capability
- ☐ **Three Windows:** - Data window - Selection window - Model window
- ☐ **Free form fields** - No need to define lengths or types
- ☐ **Extremely fast search and sort** - all records in memory
- ☐ **Built-in functions:** square root, logarithm, exponent, absolute value
- ☐ **Arithmetic operators:** +,-,*,/,%, (remainder) ± (exponent),=
- ☐ **Relational operators:** ×,†,1,(equal), !(unequal)

A powerful, easy-to-use **GEM** interface filing/data base system with many math and formula functions not found on any database program. Complex, formula intensive, programs are easy to design with EZ DATA from calculating statistics to a detailed billing/report generator program. The best part is that you design the program that fits your personal home or business needs the best. EZ DATA allows any one who has had very little or no computer programming at all to make a fast filing system for the ST. You can create easy home programs such as: name and addresses book, a checking account balancer, a home or business inventory filing system. Use EZ DATA'S versatile report capability to preform a multiable of different printing tasks. By inputing printer codes directly into the the form to be outputed you can use all the fonts, form feeds, tabbing and any thing else your printer is capable of doing. The sorting and searching capabilites are extremely fast because the whole file is loaded into memory. No need to take a coffee break while the computer searches for data because EZ DATA will find it within a few seconds. After all, the reason for a database system is to speed filing up, not to slow it down. We think this is the easiest, fastest, best documented, and best priced filing system on the market today

EZ THESAURUS $39.95 COMING SOON

This real-time or stand-alone program is a window based Thesaurus. Helping the user find just that word that will spruce up an otherwise tedious document. The utility gives the user a list, from a selection of thousands of words, that mean the same thing as you have highlighted. If you've just drawn a blank and can't think of a word, let EZ Thesaurus help you find it.

EZ GRAPH

ONLY $69.95 COMING SOON

A professional graphics and statistical analysis package. Graph everything from your finances through stock market quotes. The program will operate within EZ CALC spreadsheet program or as a stand alone professional graphing program.

Help Calc Only 24.95

- ☐ **11 Frequently used Preprogrammed Templates for Home or Business**
- ☐ **Use with EZ CALC or VIP Professional**

VIP PROFESSIONAL is a Trademark of VIP TECHNOLOGIES.

Templates include:

- Check Register
- Depreciation schedules
- Investment Portfolio Analysis
- Name & Address directory
- Home Inventory
- Loan Amortization Schedules
- Personal Finance Statement
- and more

COMPUTER PALACE

MasterCard VISA AMERICAN EXPRESS

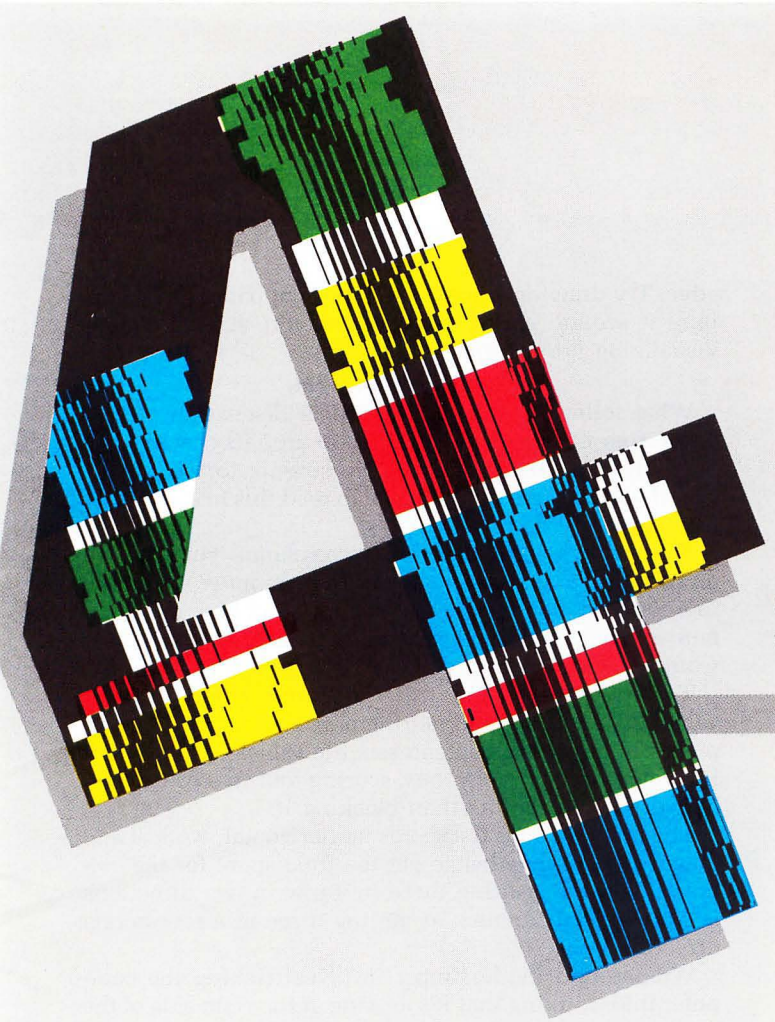
OPEN M-F, 9-6 Sat. 10-4 (Pacific Time)
710 McKinley, Eugene, Oregon 97402

USE YOUR CREDIT CARD & CALL
Toll Free 1-800-452-8013
★ **ORDERS ONLY, PLEASE** ★
There's never a penalty for using your credit card!
For Information, Call (503) 683-5361
Prices subject to change without notice.

SHIPPING INFO: Minimum $2 90 Ground, $4 75 Air Actual Cost depends on weight Call (503) 683-5361 for information
WARRANTY INFO: Everything that we sell is warrantied by the manufacturer If any item purchased from us fails to perform properly when you receive it, call us at (503) 683-5361 so that we can assist you No returned merchandise accepted without authorization Defective software will be replaced with another copy of the same program, otherwise, no software is returnable

• 2 Day Air Shipping **AVAILABLE** •

CIRCLE #121 ON READER SERVICE CARD



Score Four

Your opponent is the computer in this colorful strategy contest.

by Steven Grimm

Score Four is a challenging strategy game for Atari ST computers with color monitors. The object is to get four colored checkers in a row by dropping them into an upright board with eight columns. You can win by getting four in a row horizontally, vertically, or diagonally.

The ST is your ever-ready opponent. You can choose two levels of difficulty, depending on your mood or skill. In addition, a special "wraparound" mode can be activated to further increase the challenge.

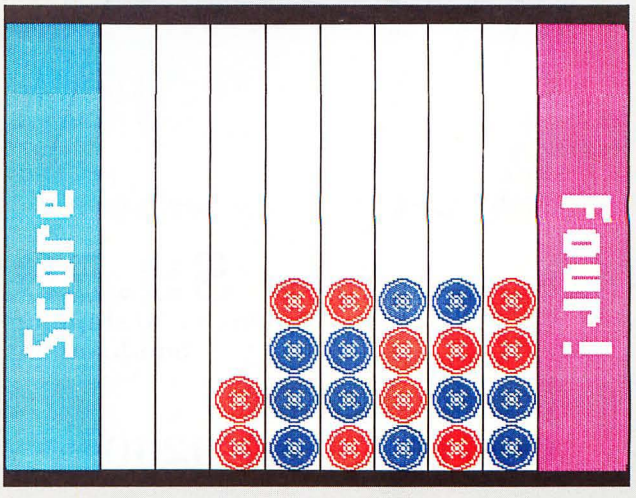
How to use *Score Four*.

To play **Score Four**, either type in the program in Listing 1 and compile it (using the Megamax compiler), or download the program file from a local bulletin board system. You can download it from the **ANALOG Computing/ST-Log** Atari Users' Group on Delphi if there are no BBSs in your area. Switch to low resolution using the "Set Preferences" command from the options menu. Find the SCORE4.PRG icon in your disk directory and double-click on it.

Once the program has loaded, you will see the title screen. Here, you select the game's difficulty level. Click the left mouse button anywhere above the two large skill level boxes to toggle the wraparound status. Enabling this option makes the game much more difficult—more on *wraparound* later.)

To begin the game, position the arrow in one of the two skill level boxes (easy or hard) at the bottom of the screen. Click the left mouse button when you're ready to start.

The computer will draw the **Score Four** playing board.



This consists of eight (initially) empty columns. To drop a checker into a column, simply position the cursor anywhere inside the column you want and press the left mouse button. A blue checker will fall from the top of the column.

The computer will, almost instantly, will drop a red checker next to yours. Now it's your turn again. Remember, you're trying to get four checkers in a row while preventing the computer from doing the same. Notice that the checkers stack up in the columns, making it possible to win vertically and diagonally.

Play continues in the same manner until you or the computer get four checkers in a row, or the entire board is filled, which results in a tie game.

Hard skill level and *wraparound* mode.

After playing **Score Four** on the easy level for a while, you'll begin to notice that it's (what else?) easy to get the computer to set up a diagonal win for you. The hard level uses a "lookahead" technique to make it much more difficult to trap the computer into letting you win. Lookahead will be explained below; it's not necessary to know how it works to enjoy the game.

When you play with wraparound activated, the sides of the board are no longer boundaries. You can win with two checkers in the bottom of the two left columns and two in the right, since the left edge is the same as the right edge. Try drawing a board on a piece of paper and wrapping it around a real cylinder if you're having trouble visualizing how wraparound works.

Getting technical.

What follows is a rather technical discussion of how **Score Four** decides which column to drop its checker into. You can skip this section and go straight to the conclusion if you like; it's not essential to read this to enjoy playing the game.

Score Four assigns a score to each column. This is based on two things: how many pieces the computer would get in a row if it moved into that column, and how many the human would get (in other words, how many the computer would block if it moved there).

The arrays block[] and get[], therefore, contain the point values for blocking and getting one, two, three and four pieces in a row. As you can see, the values in block[] are less than those in get[]; thus, scoring four in a row would be worth more points than blocking it.

The program adds the scores for horizontal, vertical and diagonal rows together to get the final score for the column. This way, getting three in a row in two directions is worth twice as much as getting three in a row in one direction.

Wraparound mode simply involves tricking the computer into thinking that it's looking at the right side of the board when it looks past the left edge, and vice versa. A simple add and subtract handle this; the operation takes place in the chekcheck() routine.

Lookahead is slightly more complicated than is wraparound. In the easy mode, the computer does nothing more than score the current board situation. Lookahead scans all the possibilities for the move *after* the one the computer is considering, and decides whether the move will be a bad one in the near future. This prevents the computer from making a move that would allow you to get four pieces in a row on the next move, for example.

Two more arrays, lblock[] and lget[], are used for lookahead. Notice that there are no point values for one and two in a row; these are of dubious value in the main arrays and utterly useless in the lookahead arrays.

The lget[] array is handled identically to the two main ones; it's added to the current column's point score. The lblock[] array, though, is *subtracted* from the score. This is because the lblock[] values are really the point values for the human's next move. The computer doesn't want to make a move that results in the human's getting four in a row.

Two-move (or more) lookahead could be added with relatively little difficulty. **Score Four**, however, is not complex enough to warrant the effort; most humans don't even look ahead one move, and a three-move lookahead capability would make the computer virtually impossible to defeat. The human should have a sporting chance.

Conclusion.

Score Four is an addictive game. It may seem easy, but the hard and lookahead options will challenge even the most expert of gamers for some time. //

Listing 1.
C listing.

```
/* Score Four by Steven Grimm
   (C) 1987, ST-Log */

#define HIDEC v_hide_c(handle)
#define SHOWC v_show_c(handle,0)

extern long xbios();

/* Define all the externals GEM needs to work nicely. */
int intin[128],intout[128],contrl[12],ptsin[128],ptsout[128],handle,
    work_in[11],work_out[57],pxy[8],pix[4],color[2];

/* Define a Memory Form Definition Block.  This is used by
   GEM to figure out how to blt an area from one form to
   another. */
typedef struct {
  long mpoint;          /* Pointer to data area */
  int  fpix;            /* Width in pixels */
  int  fhei;            /* Height in pixels */
  int  fwor;            /* Width in words (NOT bytes!) */
  int  form;            /* Format flag */
  int  fpln;            /* # of graphic planes */
  int res1,res2,res3;   /* Reserved space */
  } MFDB;

/* This is the checker's data.  It's a monochrome (1-plane)
   bitmap image, 2 words by 24 pixels, with a blank line at
   the top to make animation less difficult. */
int checker_d[50] = { 0x0000,0x0000,
   0x007e,0x0000, 0x01c3,0x8000, 0x073c,0xe000, 0x1cff,0x3800,
   0x13ff,0xc800, 0x37ff,0xec00, 0x2fff,0xf400, 0x6fbd,0xf600,
   0x5f7e,0xfa00, 0xdea5,0x7b00, 0xbfdb,0xfd00, 0xbfa5,0xfd00,
   0xbfa5,0xfd00, 0xbfdb,0xfd00, 0xdea5,0x7b00, 0x5f7e,0xfa00,
   0x6fbd,0xf600, 0x2fff,0xf400, 0x37ff,0xec00, 0x13ff,0xc800,
   0x1cff,0x3800, 0x073c,0xe000, 0x01c3,0x8000, 0x007e,0x0000 };

/* Point values for blocking n in a row or scoring it. */
long block[8] = { 0,1,100,1000,100000,100000,100000,100000 };
long get[8] = { 0,1,500,4000,500000,500000,500000,500000 };
long lblock[8] = { 0,0,0,350,10000,10000,10000,10000 };
long lget[8] = { 0,0,0,400,3000,3000,3000,3000 };
long *sctabs[4] = {block,get,lblock,lget};

/* Messages for win/tie states. */
char *winmsg[] = {
  "Tie game!       ",
  "You win!        ",
  "I win!          " };

/* Checker colors for win-show, player 1, player 2. */
int colors[3] = { 3,4,2 };

/* Title page text and colors for the letters. */
char *ttext = "Score Four!";
int tcols[11] = { 2,3,4,5,7,8,9,7,4,9,1 };

MFDB checker,screen;

int board[8][8];        /* Board data. 0=empty, 1=player 1, 2=player 2 */
int num[8];             /* Number of checkers in each row. */
int lookahead;          /* One move lookahead flag (hard mode). */
int wraparound;         /* Horizontal board wrap flag. */
int pieces;             /* # of pieces on board */
int xx,yy,_x,_y;        /* x,y,dx,dy for win-show. */

/* Initialize MFDB stati.  Uses an ST BIOS call to find the
   screen's location in memory. */
initmfdb()
{
  checker.mpoint=(long)checker_d;
  checker.fpix=32;
  checker.fhei=25;
  checker.fwor=2;
  checker.form=1;          /* Standard format */
```

```
  checker.fpln=1;
  screen.mpoint=xbios(2); /* Get physical screen loc. */
  screen.fpix=320;
  screen.fhei=200;
  screen.fwor=20;
  screen.form=0;          /* Device-specific format */
  screen.fpln=4;

  pxy[0]=0; pxy[2]=24; pxy[3]=24;
}

/* Look at a checker's color.  Return 0 if out of range. */
chekcheck(x,y)
int x,y;
{
  if (wraparound)     /* Wraparound check.  Goes to other */
  { if (x>7)       /* side of board if x is out of range. */
      x=x-8;
    if (x<0)
      x=x+8;
  }
  else
    if (x & 8)
      return(0);
  if (y & 8)
    return(0);
  return(board[x][y]);
}

/* Find out how many checkers are lined up in a certain direction. */
count(x,y,dx,dy)
int x,y,dx,dy;
{
  int chcol,num,tx,ty;
  chcol=board[x][y];
  num=0;
  /* look backwards:  */
  while (chekcheck((x -= dx),(y -= dy)) == chcol);
  tx=x+dx; ty=y+dy;
  /* look forward:    */
  while (chekcheck((x += dx),(y += dy)) == chcol)
    ++num;
  /* store values for win-show:  */
  if (num == 4)
  { xx=tx; yy=ty; _x=dx; _y=dy; }
  return(num);
}

/* Get a column's score.  Assumes that the piece has already
   been placed artificially (or normally).  Pass 0 for blocking
   or 1 for "getting", 2 or 3 for lookahead block/get. */
long getscore(x,blget)
int x,blget;
{
  long score,*aray;
  int tcol;
  aray=sctabs[blget];
  tcol=num[x];
  return(aray[count(x,tcol,1,0)]+aray[count(x,tcol,0,1)]+
         aray[count(x,tcol,1,1)]+aray[count(x,tcol,1,-1)]);
}

/* Figure out the computer's move for player n. */
compmove()
{
  register int i,col,j;
  long tscore,colscore;
  colscore = 0;
  col = 0;
  for (i=0; i<8; ++i)
  {
    /* if column is full don't bother checking */
    if (num[i] == 8)
      continue;
    board[i][num[i]] = 2;
    tscore = getscore(i,i);
    /* do lookahead if necessary  */
    if (lookahead)
```

```
    { for (j=0; j<8; ++j)
      /* column full condition  */
      { if (num[j] >= 7)
          continue;
        if (j == i)
          ++num[j];
        board[j][num[j]] = 2;
        tscore += getscore(j,3);
        board[j][num[j]] = 1;
        tscore -= getscore(j,2);
        board[j][num[j]] = 0;
        if (j == i)
          --num[j];
      }
    }
    board[i][num[i]] = 1;
    tscore += getscore(i,0);
    board[i][num[i]] = 0;
    if (((tscore == colscore) && (xbios(17) > 8388608)) ||
        (tscore > colscore))
    { col=i; colscore=tscore; }
  }
  return(put(col,2));
}

/* Draw a line. */
line(x1,y1,x2,y2)
int x1,y1,x2,y2;
{
  pix[0]=x1; pix[1]=y1;
  pix[2]=x2; pix[3]=y2;
  v_pline(handle,2,pix);
}

/* Let the human make a move. */
humove()
{
  int x,d;
 mouse:
  evnt_button(1,1,1,&x,&d,&d,&d);
  if ((x<52) || (x>268))
    goto mouse;
  x=(x-52)/27;
  if (num[x] == 8)
    goto mouse;
  evnt_button(1,1,0,&d,&d,&d,&d);
  return(put(x,1));
}

/* Put a checker in column n. */
/*       Return 1 if it wins. */
put(x,player)
int x,player;
{
  int y,maxy,ycnt;
  y=num[x];
  maxy=(8-y)*25-1;
  HIDEC;
  for (ycnt=0; ycnt<maxy; ++ycnt)
    plot(x,ycnt,player);
  SHOWC;
  board[x][y]=player;
  y=(getscore(x,1) > 500000);  /* Winning move! */
  ++num[x];
  ++pieces;
  return(y);
}

/* Plot a checker on the board. Pass player number. */
plot(x,y,player)   /* x,y.  y is BOTTOM of checker. */
int x,y,player;    /* y in pixels, x in columns */
{
  int topb,tops;
  x=54+x*27; topb=(y>24) ? 0 : 24-y;
  tops=(y>24) ? y-24 : 0;
  color[0]=colors[player]; color[1]=0;
  pxy[1]=topb;
```

```
  pxy[4]=x; pxy[5]=tops; pxy[6]=x+24; pxy[7]=y;
  vrt_cpyfm(handle,1,pxy,&checker,&screen,color);
}

/* Plot title page.  Returns skill level in 'lookahead', wraparound
   in 'wraparound'. */
title()
{
  char tchar[2];
  int i,temp,x,y;
  HIDEC;
  tchar[1]=0;
  /* clear screen to white... */
  v_clrwk(handle);
  /* set text alignment... */
  vst_alignment(handle,0,2,&temp,&temp);
  /* set "special effects" */
  vst_effects(handle,0x24);
  vst_height(handle,100,&temp,&temp,&temp,&temp);
  for (i=0; i<11; ++i)
  /* plot each letter in a different color */
  { vst_color(handle,tcols[i]);
    tchar[0]=ttext[i];
    v_gtext(handle,i*27+20,10,tchar);
  }
  vst_height(handle,6,&temp,&temp,&temp,&temp);
  /* reset everything:  */
  vst_color(handle,10);
  vst_effects(handle,0);
  v_gtext(handle,100,50,"by Steven Grimm");
  vst_color(handle,8);
  v_gtext(handle,56,60,"(C) 1987, ST-Log");
  vst_color(handle,9);
  v_gtext(handle,112,75,"Wraparound");
  vst_height(handle,13,&temp,&temp,&temp,&temp);
  vst_color(handle,12);
  v_gtext(handle,84,100,"Select skill level:");
  line(0,130,319,130);
  /* draw the boxes:  */
  line(0,199,319,199);
  line(0,130,0,199);
  line(319,130,319,199);
  line(160,130,160,199);
  line(200,74,208,74);
  line(200,74,200,82);
  line(208,74,208,82);
  line(200,82,208,82);
  vst_color(handle,4);
  v_gtext(handle,64,157,"Easy");
  v_gtext(handle,224,157,"Hard");
  pix[0]=201; pix[1]=75; pix[2]=207; pix[3]=81;
  wraparound=0;
  graf_mouse(0,&temp);
  SHOWC;
cheat:
  evnt_button(1,1,1,&x,&y,&temp,&temp);
  if (y<130)
  { HIDEC;
    evnt_button(1,1,0,&x,&y,&temp,&temp);
     wraparound = (! wraparound);
     vsf_color(handle,wraparound);
     vr_recfl(handle,pix);
     SHOWC;
     goto cheat;
   }
   lookahead=x/160;
   evnt_button(1,1,0,&temp,&temp,&temp,&temp);
}


/* Initialize the board and counters. */
initboard()
{
  int x,y;
  for (x=0; x<8; ++x)
  {
    num[x]=0;
    for (y=0; y<8; ++y)
      board[x][y]=0;
```

```
    }
    pieces=0;
    HIDEC;
    v_clrwk(handle);
    line(52,199,268,199);
    for (x=0; x<9; ++x)
      line(52+x*27,0,52+x*27,199);
    vst_effects(handle,0x11);
    vst_height(handle,50,&x,&x,&x,&x);
    vst_rotation(handle,900);
    vst_color(handle,5);
    vsf_color(handle,5);
    pix[0]=0; pix[1]=0; pix[2]=51; pix[3]=199;
    vr_recfl(handle,pix);
    v_gtext(handle,15,140,"Score");
    v_contourfill(handle,15,140,-1);
    vst_rotation(handle,2700);
    vst_color(handle,7);
    vsf_color(handle,7);
    pix[0]=269; pix[2]=319;
    vr_recfl(handle,pix);
    v_gtext(handle,304,55,"Four!");
    v_contourfill(handle,304,55,-1);
    SHOWC;
}

/* Display a win, after a short delay. */
win(stat)
unsigned stat;
{
    int i,j;
    char string[100];
    strcpy(string,"[0][");
    strcat(string,winmsg[stat]);
    strcat(string,"| |Play again?][Yes|No]");
    HIDEC;
    if (stat)
      for (i=0; i<7; ++i)
      { plotwin(0);
        for (j=0; j<15000; ++j);
        plotwin(stat);
        for (j=0; j<15000; ++j);
      }
    SHOWC;
    graf_mkstate(num,num,num,num);
    if (form_alert(1,string) == 1)
      return(1);
    exit(0);
}

/* Plot four checkers in a row for win display. */
plotwin(color)
int color;
{
    int x,y,i;
    x=xx; y=yy;
    for (i=0; i<4; ++i)
    { if (x<0)
        x += 8;
      if (x>7)
        x -= 8;
      plot(x,(8-y)*25-2,color);
      x += _x;
      y += _y;
    }
}

main()
{
    int but,x,y;
    appl_init();
    if ((int)xbios(4))
    {
      form_alert(1,"[3][You must use LOW resolution.  ][OK]");
      exit(0);
    }
     handle=graf_handle(&x,&x,&x,&x);
     for (x=0; x<10; ++x)
```

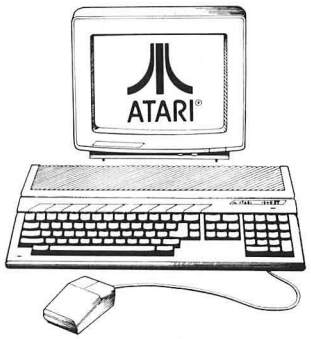
```
        work_in[x]=1;
      work_in[10]=2;
      v_opnvwk(work_in,&handle,work_out);
      menu_bar(0,0L);
      initmfdb();
      title();
      while (1)
      {
        initboard();
        while (1)
        {
          if (humove())
          { win(1); break; }
          x=1;
          /* get rid of extra button presses  */
          while (x)
            graf_mkstate(&y,&y,&x,&y);
          if (compmove())
          { win(2); break; }
          if (pieces == 64)
          { win(0); break; }
        }
      }
    }

    ,
```

COMPUTER CREATIONS

YOUR ATARI 520ST SUPPORT CENTER

ABACUS	
Paint Pro	36
Text Pro	36
Data Trieve	36
ACADEMY	
Typing Tutor/ Word Invaders	23
ACCESS	
Leaderboard Golf	26
Tournament Disk	15
ACCOLADE	
Mean 18 Golf	29
Sundog	26
ACTION	
Action Pak	32
ACTIVISION	
Mindshadow	33
Borrowed Time	33
Hacker	29
Music Studio	39
Paintworks	47
Little Comp. People	33
Hacker II	33
Tass Times/Tonetown	33
Champ. Basketball	29
ADVENTURE INTL.	
Spiderman	14
ANTIC	
C.O.L.R. Obj. Editor	23
Maps & Legends	27
Macro Assembler	60
Lattice C	114
Disk Doctor	23
A-Calc	45
Meta Pascal	75
CAD-3D	38
A-Ram	15
A-Seka	27
GST C Compiler	60
GST-ASM	45
Expert Opinion	75
Flash	30
Star Struck	19
Red Alert	19
Murray and Me	15
Mom and Me	15
Kermit & Remote Co.	19
Crystal	19
ARTWORX	
Bridge 4.0	19
Compubridge	19
Maillist	12
Strip Poker	25
Peggammon	12
Hole in One Golf	19
ATARI	
DB Man	103
DB Master	34
Joust	21
Star Raiders	21
BATTERIES INCLUDED	
Degas	26
Degas Elite	49
Timelink	33
Thunder	26
I/S Talk	33
I/S Paperclp Elite	Call
Homepak	33
BECKEMEYER	
Micro C-Shell	39
Micro C-Tools	19
Micro Make	27
RTX	53
Micro MTC Shell	79
Hard Disk Accelerate	30
BRODERBUND	
Mindwheel	29
Breakers	30
CENTRAL POINT	
Copy II	27
CLASSIC IMAGE	
Disk Library	38
Diablo	23
COSMI	
Super Huey	30
DAC	
Easy Accounting	49
DRAGON GROUP	
4 x Forth	79
Forth Accelerator	59
ELECTRONIC ARTS	
Financial Cookbook	36
Ultima III	39
Coloring Book	Call
Golden Oldies	23
Chessmaster 2000	Call
OGRE	Call
Autoduel	Call
Skyfox	Call
EPYX	
Winter Games	26
Rogue	26
Temple of Apshai	26
World Games	26
Champion. Wrestling	26
Super Cycle	26
FIREBIRD	
The Pawn	30
Starglider	30
FIRST BYTE	
Kid Talk	33
Speller Bee	33
Math Talk	33
First Shapes	33
HABA	
Habacom	19
Phonebook	39
Home Accountant	51
Mailroom	56
Checkminder	49
HIPPOPOTAMUS	
Hippoword	59
Hippoconcept	59
Hipposimple	33
Hippo Disk Utilities	33
Hippo Backgammon	27
Hippo Spell	27
Hippo Ramdisk	23
Hippo Comp. Almanac	23
Hippo Jokes & Quotes	23
Hippo Art I	27
Hippo Pixel	27
Hippo Fonts I	27
Hippo Laser	47
Hippo Eprom Burner	109
Hippo Sound Digitz.	109
Hippo Vision B & W	109
HOLMES & DUCKWORTH	
Tool Box	28
LOGICAL DESIGN WORKS	
LDW Basic Compiler	54
MARK OF THE UNICORN	
Final Word	99
Hex	27
PC/Intercomm	84
MEGASOFT	
A Copier	27
X-Rated Library	17



MICHTRON	
Kissed	26
DOS Shell	26
DFT	33
The Animator	26
Personal Money Mgr	33
Utilities	39
M-Disk	26
Mudpies	26
Softspool	26
Flip Side	26
Calendar	19
M1-Term	33
Gold Runner	26
Time Bandits	26
BBS	33
Cornerman	33
Mighty Mail	33
Lands of Havoc	13
Cards	26
Business Tools	33
Logo	33
Major Motion	26
Mission Mouse	26
M1-Dupe	20
BBS 2.0	54
Pro Football Wizard	27
Pinball Factory	27
Your Finance Future	26
MICROPROSE	
Silent Service	27
MI-GRAPH	
Easy Draw	54
MINDSCAPE	
Brataccus	33
NAVARONNE	
Astrology	29
Timekeeper	37
OMNITRENDS	
Universe II	47
OSS	
Personal Pascal	50
OTHER VALLEY SOFTWARE	
Delta Patrol	17
Monkey Business	17
PENGUIN/POLARWARE	
Crimson Crown	15
OO-Topos	15
Sword of Kadash	15
The Coveted Mirror	15
Transylvania	15
PHILON	
Fast Basic-M	99
PROCO	
Pro-Copy	19
PROGRESSIVE COMP. APPL.	
Graphic Artist	139
Font Editor	54
PRYORITY	
Forbidden Quest	27
QUANTUM	
ST-Talk	14
QUICKVIEW	
Zoomracks	54
REGENT	
Regent Spell	34
Regent Word	34
ROYAL	
EZ-Calc	54
SHANNER	
Soundwave SW-1	33
ST-Key	14
Macro-Manager	33
DB Calc II	29
SHELBOURNE	
Pool	24
SIERRA	
Kings Quest II	33
Ultima II	39
Winnie the Pooh	19
Black Cauldron	26
K/Q Hint Book	6
B/C Hint Book	6
Donald Duck	19
Cash Disbursements	67
SPINNAKER	
Homework Helper:Math	34
Treasure Island	27
SST	
Chat	15
SUBLOGIC	
Flight Simulator II	38
TDI	
Modula-2	59
Modula-2 Develp.Kit	109
UCSD Pascal	59
TELARIUM	
Amazon	Call
Nine Princes/Amber	Call
Farenheit 451	Call
Dragonworld	Call
TIMEWORKS	
Datamanager	54
Swiftcalc	54
Wordwriter	54
S/P Financial Plan	54
UNICORN	
Fraction Action	27
Decimal Dungeon	27
UNISON WORLD	
Printmaster	26
Art Gallery I	19
Art Gallery II	19
VIP	
VIP Professional	119
X-LENT	
Typesetter Elite	29
Rubber Stamp	25
Music Box	32
Megafont	25

PRINTERS

OKIDATA	
Okimate 20 & Plug N' Print	219
STAR MICRONICS	
LV-1210	189
NX-10	259
PANASONIC	
KX-P 10801	199
KX-P 10911	259
KX-P 3131	259
KX-P 1592	429
MODEMS	
Supra 300 ST	59
Supra 1200 ST	149
Avatex 1200	99
Avatex 1200 HC	139
Compuserve Start Kt	24
Microstuffer	59
Omega Terminal	20

ST ACCESSORIES

Disk Cleaning Kit	14
Dust Covers	7
Monitor Stand	12
Mouse Pad	8
Shanner Planner	29
The Easel	14
Disc Directory	20
Pocket Pak	10
The Library	34
Speed Pad	18
Disk File/30 Micro	10

ST BOOKS

Atari ST GEM Reference
Atari ST Internals
Atari ST Machine Language
Atari ST Tips & Tricks
Atari ST from Basic to C
Atari ST Basic Training
Atari ST Graphic & Sound
Atari ST Logo
Atari ST Peeks & Pokes
Atari ST for Beginners

CABLES

ST to Printer	15
ST to Modem	19

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

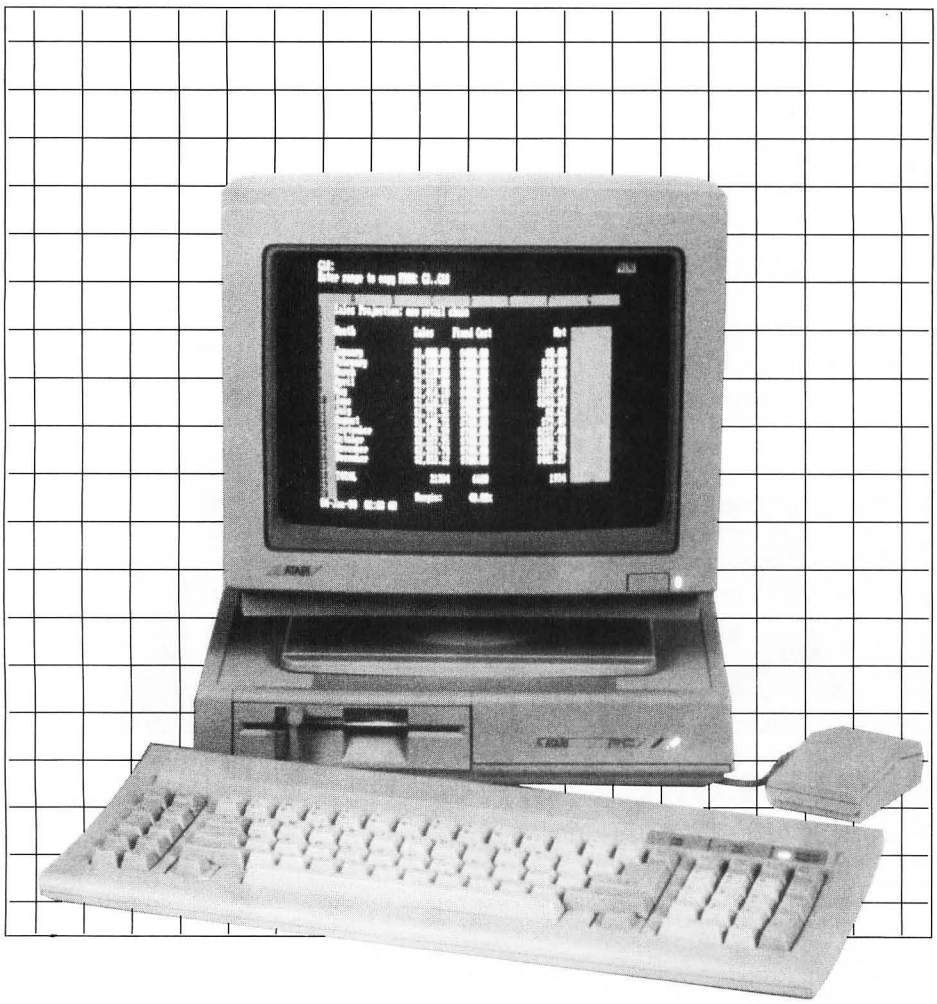
For information, order inquires, or for Ohio orders (513) 435-6868

VISA® MasterCard

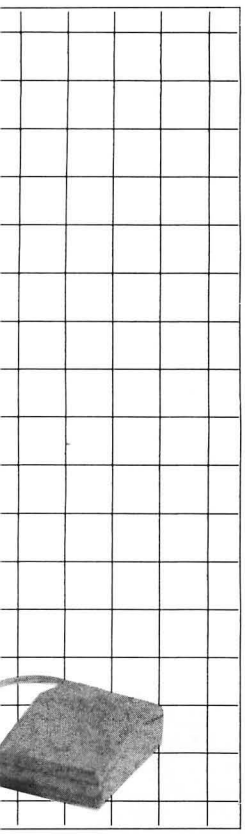
Order lines Open 9:00 a.m. to 8:00 p.m. Mon.-Fri.; 10 a.m. to 4:00 p.m. Sat. (Eastern Standard Time). Minimum $15 per order. C.O.D. (add $3.00). Please specify computer system. Call toll free number to verify prices and availability. Prices and availability are subject to change without notice. We ship C.O.D. to Continental U.S. addresses only! Please include 4% shipping on all Hardware orders (min. $4.00). Software and accessories add $3.00 shipping and handling in continental U.S. Actual freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico and APO. Ohio residents add 6½% sales tax. Canadian orders add 5% shipping, (min. $5.00). All other foreign orders, please add 15% shipping, (min. $10). 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. FOR YOUR PROTECTION WE CHECK FOR CREDIT CARD FRAUD. We do not bill until we ship.

CIRCLE #124 ON READER SERVICE CARD

Atari introduced the Mega ST, a compact version of the ST computer for the professional user.



Also new from Atari Corp. — the first IBM PC compatible with EGA (Enhanced Graphics Adapter) built in, for $699.



Atari news from the 1987 Winter CES

by Arthur Leyenberger

After the hectic and exciting Fall COMDEX (COMputer/Dealers Exposition) held just two short months ago, the 1987 Winter Consumer Electronics Show seemed somewhat subdued. Of course, both shows are held in "glittertown" (read Las Vegas), and it's hard to believe that *anything* in Vegas could be subdued. Nonetheless, CES featured a smaller than usual Atari exhibit, less software and hardware vendors showing their wares, and what seemed like a smaller attendance.

Atari declares war.

CES began with a bang at the Atari press conference. Atari announced a series of new products, consisting of an MS-DOS computer (the first of a complete line of IBM compatibles, we were told), three new ST computers, a mystery laser printer, a new video game machine and a cheap 1200-baud modem. All of these new products, as well as the existing ones, are to be manufactured in Atari's 200,000 square foot factory in Taiwan.

Michael Katz, Atari Vice President for Sales, declared war on the American computer competition by announcing the first IBM PC compatible with built-in EGA (Enhanced Graphics Adapter, an IBM video graphics standard). Called the **Atari PC**, it consists of a monochrome monitor, one 5¼-inch disk drive, 512K bytes of memory (expandable to 640K, the maximum free RAM an IBM PC or clone can directly address) and a separate keyboard. The **PC** will sell for $699.00 and features an Intel 8088 microprocessor (like the IBM PC itself). Unlike the IBM PC, however, the Atari's 8088 will run at either the standard IBM clock rate of 4.77 megahertz or almost double that, at 8.0 MHz.

Most IBM-compatible computers have built-in support for only the IBM monochrome graphics and the IBM Color Graphics Adaptor (CGA) standards. The **Atari PC** supports these, as well as the EGA graphics mode. It also has built-in serial (RS-232 for modems), parallel (for printers) and mouse ports, and comes with a bundled mouse and the GEM desktop from Digital Research (DRI). The **Atari PC** will be sold through mass merchants and computer specialty stores when deliveries begin this spring.

Interestingly, the GEM desktop used on the **Atari PC** is the "new improved" version, designed by DRI in response to a threatened lawsuit from Apple. For a complete comparison of the two versions, see last month's **ST User** column. The **Atari PC** was one of two inexpensive computers announced at the show (the other was the **Amstrad**) that bundle the GEM desktop and a mouse with the system.

The **Atari PC** will come with a single, 360K-byte 5¼-inch disk drive (as used by the IBM PC and clones). Fortunately, you'll be able to attach a double-sided, double-density ST disk drive directly to the **PC**.

Atari also showed three new versions of the venerable ST. Called the **Mega STs**, these feature a more compact version of the original 520 and 1040ST. The new model will be sold in 1-, 2- and 4-megabyte versions, and will come with a separate keyboard, battery-backed real-time clock and a bus for plugging in add-on boards. The smaller footprint was achieved by packaging the CPU (Central Processor Unit), floppy disk and power supply into one module that serves as a pedestal for the monitor. Atari will also sell a stackable hard disk unit for the **Mega** series.

According to Atari, the **Mega ST** will be available this spring only at computer specialty stores. It will not contain an RF modulator and is meant primarily for business users. Pricing and final availability have not yet been set, but you can be sure that Atari will follow its usual aggressive pricing strategy. Atari did estimate that the laser printer and one of the **Mega STs** (they didn't say which one) would sell for under $3000.00.

Although the new **Mega STs** looked "high-tech," and their detached keyboards had an improved feel, the key tops and key spacing have not been changed from either the 520 or 1040 models. The new series of **STs** are also said to have the blitter chip, although neither this nor the external bus mentioned above was shown at CES.

The Atari **Laser Printer** was also announced, but with very few details. Thus, it's difficult to believe that the printer is anywhere near ready for distribution. Conjecture says it will use the processor of the 4-megabyte **Mega ST** and 2 megabytes of memory (one for drivers and another for fonts). With no brains (processor) and no internal memory, it's easy to see how Atari can claim an under-$1500.00 price for its **Laser Printer**, which was said to connect to the ST via the DMA port, to insure fast printing speed.

Atari also announced another new video game. Looking like a "Miami Vice" prop with its sharp corners and round pastel art deco buttons, the mystery machine was only shown under glass. It reportedly

accepts 8-bit Atari game cartridges, a keyboard and a disk drive. This "XE System" resembles a similar product that James Morgan, CEO of the old Atari, introduced shortly before the fall of that Atari empire in 1984.

At the Atari exhibit.

The Atari exhibit consisted of the now familiar sub-booths shared by two or three software publishers each. In all, some forty vendors were there. Many of the companies had previously exhibited at the Fall COMDEX (see **ST-Log** 11, February 1987) and will not be covered here.

Electronic Arts has definitely seen the ST light; they announced several products for our favorite computer. **Autoduel** is a car battle adventure game created by Lord British of Origin Systems. **Autoduel** is like a marriage of the movie *Road Warrior* and the popular **Ultima** game series. Like **Ultima**, the game involves strategy role-playing in which the player must analyze various situations and deal with a large assortment of characters and circumstances. Like *Road Warrior*, the game is set in a ravaged future world in which only the most cunning automotive warriors survive. **Autoduel** will sell for $50.00, available in the second quarter of 1987.

Electronic Arts also announced that **Chessmaster 2000** will find its way to the ST real soon. Winner of the 1986 U.S. Open Personal Computer Chess Championship, **Chessmaster 2000** is now available for the ST for $45.00. Created by Software Toolworks, the game uses an algorithm whose performance was rated at 2,018 by the U.S. Chess Federation. **Chessmaster** claims to contain the largest opening library ever available on a microcomputer—over 71,000 lines.

The game offers twelve levels of play, that range from "Newcomer" to "Grandmaster." A teaching mode, showing all possible legal moves, and a hint mode are also provided to help you learn the game. One or two players can take part, and moves can be retracted as far back as you like. Especially impressive is the graphic portrayal of the chessboard. It can be viewed either in 2- or 3-dimensional display. The 3-D board can be rotated to any angle. Players can see all captured pieces and an on-screen chess clock.

Other Electronic Arts games to become available for the ST include **Arctic Fox, Skyfox, Ultima III: Exodus, Ogre, Software Golden Oldies** and **Starfleet I**. As sales of these products increase, Electronic Arts promises to bring out more software for the Atari ST.

Firebird, distributor of the outstanding graphics adventure game **The Pawn**, was showing a few new titles at CES. **Starglider** puts the gamer in command of an airborne ground attack vehicle which must ultimately do battle with the almost-invincible starship. Using 3-D animation and superb graphics, **Starglider** brings true arcade-quality action to ST gamers.

Golden Path, an animated graphic adventure game, goes beyond **The Pawn** in the quality of its "paintings." The central figure in **Golden Path** is a white-bearded sage who also happens to be a martial-arts guru. You use the mouse to move him through forty-five screens of smooth, life-like animation, to complete his quest. A book of lore is always available through an on-screen window; it provides clues to help keep you on the **Golden Path**. Available now, retailing for $45.00.

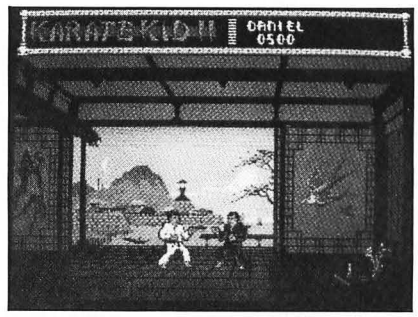
Firebird's other intriguing new title is **Universal Military Simulator**. With it, you can draw up your own battle scenarios by specifying locale, geographical features, armaments, troops, opposition and other variables. In addition to fantasies—such as having a Stone-Age man encounter a warrior from A.D. 2001—you can recreate any historical battle exactly as it occurred, or alter any of the original circumstances to explore what-if possiblities. Firebird provides several renowned historical battles on the disk and will offer battle library disks in the future. **Universal Military Simulator** will be available by the time you read this and will sell for $45.00.

MichTron was showing a number of new products. **Journey into the Lair** is a game played using a Pioneer Laser Video Disk player and an Atari ST. MichTron supplies the computer interface ($29.00), the video disk ($29.00) and the software ($49.00).

Journey into the Lair is a strategy adventure based on the arcade game **Dragon's Lair**. The video disk is actually identical to the one used in the arcade, but the play is slightly different. Because there's no quick jump facility in consumer disk players, there is a delay from the time a command is given till the disk player responds. Therefore, when an animation sequence is presented, you must select from a list of alternative moves in order to explore the realm of the castle.

Maneuvering Dirk the Daring through his perilous journey to save the beautiful princess Daphne, you watch the action (animated cartoon scenes) on the video screen and your score on the computer screen. Three levels of play are provided —apprentice, knight and lord.

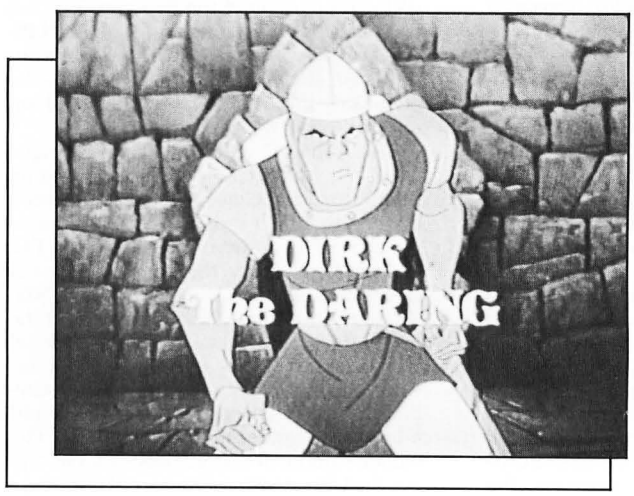
Journey into the Lair was created with MichTron's soon-to-be-released **VIVA** (Visual Interface Video Authoring). **VIVA** lets you access the laser disk in a variety of ways. Virtually any disk can be used, and "programming" consists primarily of storyboarding a sequence of video clips, associating these with key words or menu-based instructions, and storing that information within the computer. **VIVA** itself is a GEM-based product, to be available on the ST in a few months.



The complete *Karate Kid II*.

Karate Kid II was the second arcade game shown by MichTron. It's based on the movie and has very attractive graphics created by professional artists. Guide the young hero through fight after fight against increasingly powerful adversaries. The game includes one- and two-player options, joystick or keyboard control, realistic sound effects and a MIDI-compatible (with an attached synthesizer) soundtrack of the title song.

MichTron's third game at CES was **Shuttle II**, an action simulation of space travel. You control the space shuttle docking

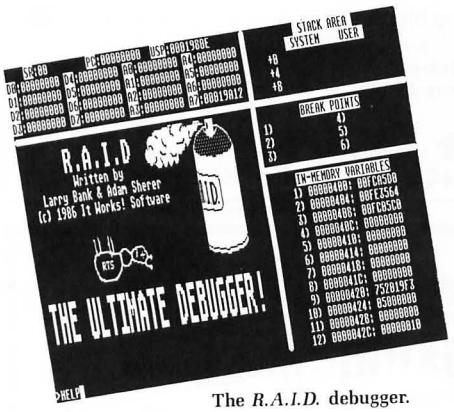


***Journey into the Lair* from MichTron uses a Pioneer Laser Video Disk to bring arcade strategy adventure into your home.**

with an orbiting satellite. Once docking is complete, maneuver the astronaut to make repairs and bring the satellite back to the shuttle, then fly home and attempt a perfect landing.

Also on hand were MichTron's **GFA BASIC**, a BASIC interpreter, and **GFA Compiler**, their BASIC compiler. **GFA BASIC** is said to rival assembly language in speed and to offer full access to the ST's features. Mouse commands make input easy, and there are also commands to create windows and dialog boxes from within your BASIC program. Other features allow structured programming techniques. In addition, the interpreter is compact, using only 55K bytes of memory on the ST.

GFA Compiler is a two-pass version that will convert your finished **GFA BASIC** programs into compact, faster-running .PRG files. Taken together, **GFA BASIC** and **GFA Compiler** offer a very attractive alternative to the original.

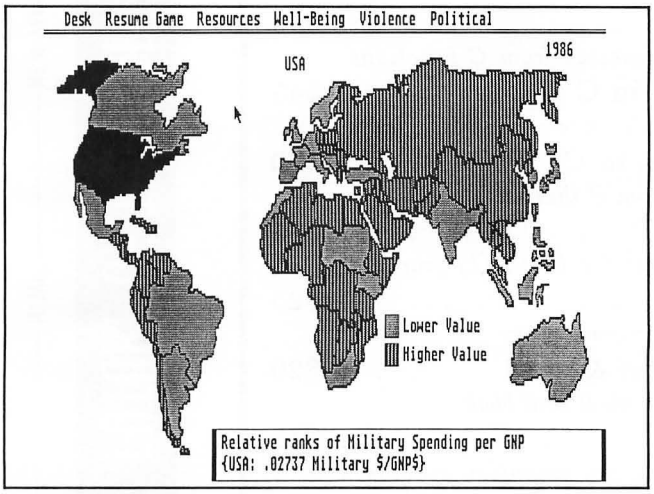


The *R.A.I.D.* debugger.

A second new programming tool announced by MichTron is **R.A.I.D.** The new debugger comes complete with a mini-assembler/disassembler, full screen editing capability and symbolic definitions display for monitoring your machine language and C programs. Other features include Copy, Fill, Move Block and resettable break points, to name just a few.

MichTron also announced **TRIMbase**, a complete data management system, said to be easy to learn yet powerful enough to handle large amounts of complex data. This relational database offers custom reports as tables or free form text. **TRIMbase** is GEM-based and offers pull-down menus, point and click controls and help options. Special relational functions (cross file interactions, condensing, merging and joining files) are provided.

MichTron's **Laser Driver** is a set of GDOS device drivers which support the Quadram QUADLASER, Hewlett-Packard Laserjet+ and Okidata 292 laser printers. GDOS is an extension to Atari's ROM-based GEM that lets programs use device-independent graphics data contained in META files, GEM's output interface. A monochrome screen dump for QUADLASER is included.



Wheelers and dealers of the ST world: if you liked the board game Diplomacy, try Mindscape's *Balance of Power*.

Mindscape announced that **Balance of Power** will be available for the ST by the time you read this, retailing at $50.00. **Balance of Power** allows users to make superpower decisions as leaders of the United States or the Soviet Union. The twofold goal is to complete eight years in office without initiating a nuclear conflict and to accumulate more prestige points than the opposing superpower. To win prestige, diplomatic tools (military aid, covert destabilization, treaties, military advisors and troops) are available.

Written by Chris Crawford, **Balance of Power** has received critical acclaim from the press and from thousands of Macintosh and IBM PC users. Crawford's earlier *tour de force* for the 8-bit Ataris, **Eastern Front**, was also a legend in its time.

Unison World showed **PrintMaster Plus**, a newly designed and enhanced version of their original package, **PrintMaster**. Unison World was sued by Broderbund Software for infringing on the "look and feel" of **The Print Shop**. The two companies settled out of court, and part of the agreement was that Unison World would change the "look and feel" of their program and would quit selling the original version. Damages will be determined in a trial later this year.

As a result of all this litigation, **PrintMaster Plus** got better. I always thought the original was better than **The Print Shop**, anyway—you could see your final design on-screen before printing it. Now you can create the various components of your signs, banners, stationary, etc., in any order. There's no need to follow a predetermined pattern through the various design options. In addition, you can preview your work at any time. Further, **PrintMaster Plus** will let you print two different graphics on the same page or banner. **PrintMaster Plus** sells for $40.00, is not copy protected and will be available in March.

Word Perfect Corporation was also at CES. **Word Perfect** has long been one of the top word processors for the IBM PC and MS-DOS clones. Unlike its PC predecessors, the ST version will include windows, pull-down menus and mouse controls—in other words, it's a fully GEM-based application.

Word Perfect is also known for its excellent documentation. The ST version will include graduated lessons, a thorough reference section, and more. **Word Perfect** for the Atari ST is scheduled for release in the second quarter of 1987 and will sell for a hefty $395.00. More on this product in ST news, page 15.

Other highlights.

CES itself may have seemed like a soft drink without the fizz, but there was plenty of excitement for **ANALOG** staff members who attended the show. For one thing, squeezing six people into a compact automobile for four days resulted in an opportunity for editors and staff to *really* get to know each other. It was also a show of firsts. . .for some, their first-ever visit to Las Vegas. For others, it was their first trip to a CES in Las Vegas. The more travel weary simply said it was their first trip to Vegas *this year*.

Among the highlights of our visit to the "city of sin" was a 2:00 a.m. trip to Hoover Dam. This was my second trip to the dam in the still hours, and I was awed by the enormity of this human-made tribute to engineering. Standing on the edge of the dam looking down toward the turbines far, far below—and the rushing water even farther below—brought technology into perspective for me. Whether it's consumer electronics or raw power harnessed from nature, we live during incredible times.

C-manship

This month, we cover file selector boxes and raster operations.

by Clayton Walnum

As mentioned before in this column, the GEM's AES contains a number of libraries, one of which is the form library. Last month, we had a brief exposure to the form library when we learned how to handle alert boxes, the simplest of the ready-to-use forms. Most the forms you'll be employing once you get used to programming with GEM will be *dialog boxes*. Dialog boxes are complex and can be put together in almost any form imaginable. One of these days, we're going to sit down and have a long talk about these puzzling creatures, but for now, there's still one other ready-to-use form that we haven't explored yet: the *file selector box*.

At first glance, one might think the file selector box is difficult to handle (from a programmer's point of view), what with those slider bars and the editable text fields, and the exit buttons. The truth is that file selector boxes aren't much more difficult to program than alert boxes, because, just as it does with alert boxes, GEM handles a lot of the busy work for us.

Picking a file.

Listing 1 shows how to use a file selector box within a C program. Type it in and compile it (the program was developed with Megamax-C; if you have a different compiler, you may have to make some changes).

When you run the program, you'll be presented with a file selector box that looks something like Figure 1. Choose one of the files, then press an exit button. The file selector box will be replaced with two lines of text informing you of the chosen file and the exit button you clicked.

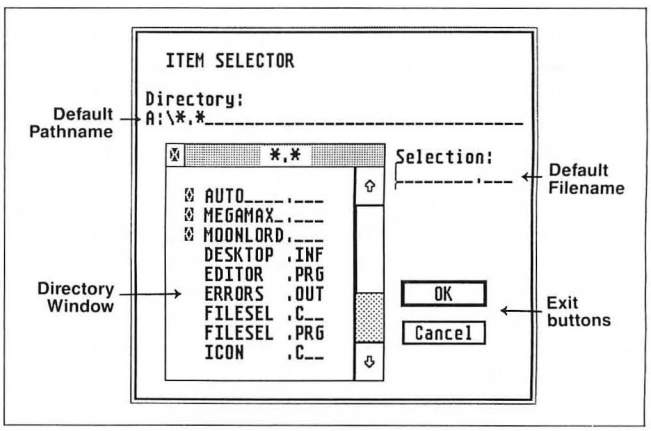


Figure 1. — FILESEL.PRG.

At this point, you're probably a little fuzzy on exactly what a file selector box does for you. What kind of information does it return? Even though the file selector box looks complicated, and allows the user to fiddle with scroll bars and buttons and text fields, all it really does is return a filename and the number of the exit button pressed. It's up to the programmer to decide what to do with the information. In most cases, the user will be selecting a data file—such as a document for a word processor—and you'll use the filename returned to open that file and read the data into the program.

Calling up a file selector box.

One simple call will get the file selector box up on your screen:

```
fsel_input (path,file,&button);
```

But there's a bit of preparation that must be done first.

In the above call, path is a pointer to a string where the default pathname is stored (that's the line at the top of the selector box; see Figure 1), file is a pointer to a string containing the default filename (the text field to the center right of the box), and the integer button will contain the value of the exit button chosen, where *0* equals the cancel button and *1* equals the OK button.

The pointers path and file actually serve a dual purpose. Both of the text fields they represent are editable. Upon exit from the selector box, they'll contain the strings typed by the user (if nothing was typed, they'll still contain whatever you put there). This is how we can find the file or path the user selected.

But there are still a couple of things you need to know before you can start using file selector boxes. For instance, how are the strings pointed to by path and file formatted? The answer can be found in the function sel__file() in Listing 1.

Selector box housekeeping.

The first thing we must do in sel__file() is declare the variables we need and set aside some space for filenames. It's important that you reserve enough memory. Otherwise, strings typed by the user may overrun their allotment and tromp all over other data. The storage area for the default pathname, path[50], is probably larger than we'll need, but it's better to be safe. Let's see what might happen if the array were smaller, say only 20 bytes (path[20]). Now, what if the file the user wants to select is found buried within two folders? We could end up with a pathname like: *A:\FOLDER.ONE\FOLDER.TWO\FILENAME.EXT. That gives us a pathname that's thirty-seven* characters long. Our storage area will hold only *twenty* characters. Watch out for that.

The storage for the default filename, file[13], isn't as tricky, since no filename will ever exceed thirteen characters (including the \0 terminator).

After we've set up our variables and storage space, we must do some initialization. First, we fill the default path and filename areas with nulls, getting rid of all the junk. We then ask the system for the default drive (the one the program was loaded from; any filename that doesn't specify a drive will use the default), convert it to ASCII and store it in the first element of path[] with the line:

```
path[0] = Dgetdrv() + 65;
```

Dgetdrv() is a GEM DOS call (*gemdos(0x19)* for those who are interested) and returns the number of the default drive as an integer where *0* means drive A, *1* means drive B, and so on. Since our pathname must be a string, we need to convert the drive number to the ASCII equivalent. And what's ASCII for A? *Sixty-five*, right? So all we have to do is add 65 to the drive number, then place this value in the first element of our string, and we're on our way to creating the default pathname.

We finish our pathname with the statement:

```
strcpy (&path[1],":\*.*");
```

The function strcpy() copies the string (including the null) pointed to by the second argument, to the string pointed to by the first argument. In the example above, the colon will be copied into the second element of our pathname, with the rest of the characters in the second string following. This is just one of many handy string-handling functions available with Megamax-C and the DRI compiler (most other implementations of C also support them). Others include strcat() and strncat() which concatenate strings; strcmp() and strncmp () which compare strings; strlen() which returns the number of characters in a string; and index() and rindex() which return a pointer to the first or last occurrence of a character in a string, respectively. The details of these functions can be found in your compiler manual.

Finally, in sel__file() we open the file selector box, then print the results of the user's selections.

Now that we've got the file selector box mastered, let's move on to something *really* challenging.

Raster operations.

Many of you may have heard the term *Bit Block Transfer*, or *BITBLT* as it's more commonly known. This is the name sometimes given to the VDI's raster operations. What's a raster operation? Simply, it's the movement of blocks of memory, usually from screen memory to someplace else in RAM or vice versa. Many of the programming techniques you'll be learning will require a good knowledge of raster operations. Rastering is used to draw icons and sprites, and also to update windows.

GEM's VDI contains a number of functions that help the programmer perform this memory juggling, but in order to take advantage of these functions, we must first have a way to describe the blocks of memory we want to move. We supply this information with a *Memory Form Definition Block* (*MFDB*).

The MFDB consists of ten words of information: the address of the memory we want to move; its height (actually, its height-1) and width; the coordinate system we're using (raster or normalized); and the number of bit planes that make up our screen. (There are also several words that are ignored, but must be present.)

C provides us with a handy way to group all this information into a single unit, the *structure*. Our MFDB, then, looks something like this:

```
typedef struct mfrmblk {
    long f_addr;
    int  f_w;
    int  f_h;
    int  f_wdwidth;
    int  f_stand;
    int  f_nplanes;
    int  f_r1,f_r2,f_r3;
} MFDB;
```

Here, f__addr is the address of the memory block, f__w and f__h are the width and height of the block in pixels, f__wdwidth is the width of the block in words, f__stand is the coordinate system (*0* for raster, *1* for normalized), and f__nplanes is the number of bit planes. The integers f__r1, f__r2 and f__r3 are reserved for future use and may be ignored (I usually initialize them to *0* just to be safe).

Filling in the blanks.

Confused yet? I thought you might be. All this talk of

coordinate systems and bit planes can be—if you've never been exposed to it before—daunting. Bit planes were mentioned in the last installment of **C-manship**, when we designed our own mouse forms, but now it's time to learn a little more about the way your ST's screen memory works.

The ST reserves 32K of memory for the screen, no matter what resolution you're in. In high resolution, the organization of this memory is simple: each bit in memory represents 1 pixel on the screen.

The first 640 bits (80 bytes) represent the first row of the screen; the second 640 bits represent the second row of the screen; and so on, for 400 rows. If we multiply 80 bytes per row times 400 rows, we get the magical number *32,000*, the size of screen memory. If a bit is on, the corresponding pixel will be a black dot; if a bit is off, the pixel will be white.

When you talk about low or medium resolution, however, you throw in extra complication: color. Now, we have to know more than just whether a pixel is on or off; we need to know its color. And we still need to get all this information into 32K.

In medium resolution, we're allowed four colors. It takes 2 bits to store this information (four possible combinations: *00, 01, 10, 11*) versus the 1 bit we needed to represent black and white, which means 32K of screen memory can hold only enough information for 128,000 pixels instead of the 256,000 pixels we had in monochrome. (Wow! A quarter of a million!) In order to compensate for this, the designers of the ST decided to halve the vertical resolution, giving us a screen 640x200. Eighty bytes (640 bits) times 200 lines times 2 bit planes per pixel gives us a total screen memory of 32K.

In low resolution, we have sixteen colors to work with. Since it takes 4 bits to represent sixteen combinations, we find that we must again cut the number of pixels in half, to 64,000. This time, the ST's designers made up the difference by halving the horizontal resolution (as well as the vertical), to give us a screen 320x200.

But that's not the end of the story. In the color modes, the screen memory is divided into bit planes (see Figure 2). You can think of the bit planes as transparencies laid one on top of the other. In order to get the color value for the first pixel on the screen, you must combine the first bit of each plane. The second bit of each plane forms the color value for the second pixel; the third for the third; etc. In medium resolution, there are two bit planes. In low, there are four.

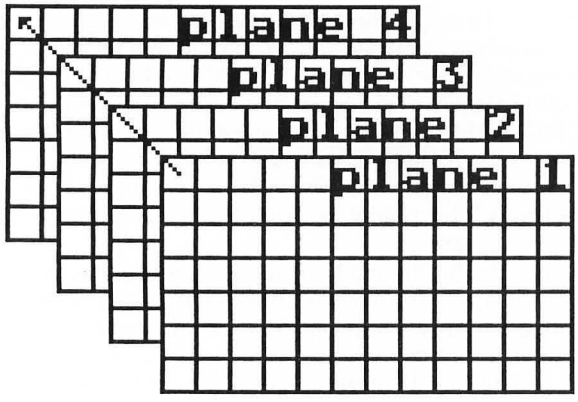


Figure 2.—Bit planes for low resolution.

Now that we understand all this nonsense about bit planes, what's with these coordinate systems? When programming in GEM, there are two coordinate systems you may choose between when you open a new workstation: *Normalized Device Coordinates* (*NDC*) or *Raster Coordinates* (*RC*).

The NDC system divides the screen into a grid that's 32,768x32,768 with the origin (point 0,0) in the lower left corner. Moving to the right from the origin increases the value of the X-coordinate, while moving upward from the origin increases the value of the Y-coordinate.

The RC system is the one we usually use on the ST, where the origin is in the upper left corner of the screen, and the width and height of the screen depend on the current resolution. The sample program (Listing 2) uses the RC system.

The next listing.

And speaking of Listing 2, it's time to type and compile. When you run the program, the infamous **ANALOG** icon (in multicolors) will appear. Use the mouse to point and click anywhere on the screen. The icon will move to that location. When you're through, press the right mouse button to return to the desktop.

The ANALOG icon.

The icon that you've been moving around the screen is an example of an *icon*, and is an image we've stored in memory. An icon is usually designed by drawing it with a graphics program, such as **Neo-Chrome** or **DEGAS**, then converting the image to its hexadecimal equivalent. This can be a complicated procedure, especially if you're dealing with a color mode and all those bit planes. Your best bet is to get hold of one of the public domain icon editors floating around. **ST-Log** will be publishing one in the near future. At any rate, the icon editor will take the image you've created and convert it to data which can be merged with your source code.

Take a look at Listing 2. About a third of the way down, you'll find the array, icon[]. See all that data? That's the hexadecimal representation of our **ANALOG** icon as it appears in low resolution (remember?—four bit planes). The data would look different in medium or high resolution, because we wouldn't be dealing with as many bit planes. In fact, in high resolution, there would be only one-fourth as much data, since there would be no colors to keep track of.

Now, take a look at the function do_icon(), where the main logic for the demo program is found. First, we change the mouse form to the pointing finger and initialize some variables. Then, after setting our control variable, repeat, to *true*, we enter the main while loop. Once in the loop, we adjust the mouse coordinates (X and Y) so the icon will be drawn in the right place. We have to do this, because the coordinates we get from the mouse are the location of the mouse's hot spot (oooh, I love it when we talk dirty); the coordinates we need for the raster functions are the upper left and lower right corners (actually, any diagonally opposed corners) of the block of screen memory. If we didn't do this extra calculation, the icon would always be drawn below and to the right of the mouse pointer.

We then save the coordinates so that, when we get a new mouse X and Y, we will be able to erase the first drawing. Finally, we turn off the mouse and call the function draw_icon() to actually draw the icon on the screen.

The raster details.

Which brings us to the point of this lengthy discussion (you knew there had to be a point, right?): the VDI call vro_cpyfm(). This function actually performs the rastering (there's a second VDI raster function, vrt_cpyfm(), which is very similar except that it's used to copy forms designed for monochrome onto a color screen). The function is called like this:

```
vro_cpyfm(handle,mode,pxy,
      &mfdb1,&mfdb2);
```

Here, the integer handle is the handle returned when you opened the virtual workstation; the integer mode is the raster writing mode; the pointer pxy is the address of an array of integers describing coordinates of the two rectangles; and &mfdb1 and &mfdb2 are pointers to the two MFDBs that describe the areas to be rastered. Gasp!

The parameter mode can be any number from *0* to *15*. The writing mode is the logical operation that's used to combine the source and destination values. There are sixteen logical operations available to us. In the sample program, we're exclusive ORing the source and destination. This way, an image can be easily erased by redrawing it in the same (exclusive OR) mode. The disadvantage to this mode is that, if you're not working with a blank screen, your image will be transparent, allowing the background to bleed through.

I'm not going to spend a lot of time describing the different writing modes. You should look them up in your manuals or experiment with them. You'll probably find that there are only a couple you'll use; the rest are there should you need them.

The pxy array holds our rectangles' coordinates: the upper left-hand and lower right-hand corners of both the source and destination rectangles. They should be stored in this order:

```
sx1,sy1,sx2,sy2,dx1,dy1,dx2,dy2
```

Now that we understand the vro_cpyfm() call, let's take a look at draw_icon() for the details. First, we must in-

itialize the two MFDBs. I don't think I have to say too much about this, since the MFDBs are fairly well described above, but there are a couple of things I'd like to clarify.

For one thing, if you look at the data for the icon in the source code, it would appear to be six long words wide or 192 bits—a big icon! Now, everyone who can tell me why our data is 192 bits wide, raise their hands. Those of you who are slinking down under your desks in embarrassment can relax; I'm not going to call on you. But do the words *bit planes* jar any memories?

"Yes!" you say. "Yes! That's why we've got 192 bits. We're dealing with four bit planes, and 192 divided by four is. . . is. . . is. . ."

Forty-eight.

"Yeah. . . thanks."

You're welcome. Now you know why we've made icn_w equal to *48* instead of *192*. But I've got another question for you. How come the icon, when it's on the screen, appears only 23 pixels wide?

"I don't know," you mumble, returning to your underdesk residence.

Well, climb back into the light, my friend; it's not your fault that you can't answer the last question. There's still something I haven't told you. Because of the way the raster functions move data, the width of a data block must be a multiple of *16* (this allows more efficient movement of data). In the case of our icon, we've had to pad the left and right of the image with "off bits" in order to bring the width up to the next highest multiple of *16* which is, of course, *48*. Now, we get to the screen MFDB. If you look at draw_icon(), you'll see that each member of the icon's MFDB, s_m, had to be initialized properly. Yet, for the screen MFDB, scr_m, we've only one line of code:

```
scr_m.f_addr = 0;
```

Why? Whenever the form address element (f_addr) of the MFDB is set to *0*, the system knows it will be dealing with the screen and will automatically handle everything for you. You don't have to fill in the rest of the MFDB. You *can* if you want, but—unless you're the type of person who enjoys painting houses with a single camel hair—why would you bother?

Off again.

I know I say this at the end of just about every **C-manship** column, but I'm going to say it again: *practice!* Everything you're learning builds upon what has gone before. Just like a course in mathematics, if you miss a lesson or don't understand it completely, you'll get more and more confused as you try to go on.

Try designing your own icons and raster them to the screen, experimenting with the different writing modes to see the results (there's an excellent illustration of the sixteen modes on page 228 of the *Programmer's Guide to GEM*, published by Sybex, for those of you who have that book). Of course, in order to experiment with the writing modes, you're going to need some sort of graphic in the background. Looks like you'll be getting some more practice with the VDI, eh? //

Listing 1.
C listing.

```
/*****************************************************************/
/*                    C-MANSHIP, Listing 1                       */
/*                       ST-Log #12                              */
/*                  Developed with Megamax C                     */
/*****************************************************************/

#include <osbind.h>

/* The usual required GEM global arrays */
int work_in[11],
    work_out[57],
    pxyarray[10],
    contrl[12],
    intin[128],
    ptsin[128],
    intout[128],
    ptsout[128];

/* Global variables */
int handle, dum;

/* Main program */
main ()
{
   appl_init ();       /* Initialize application.          */
   open_vwork ();      /* Set up workstation.              */
   sel_file ();        /* Go select file.                  */
   button_wait ();     /* Wait for a mouse button press.   */
   v_clsvwk (handle); /* Close virtual workstation.       */
   appl_exit ();       /* Back to the desktop.             */
}
```

```
/* Initialize a virtual worksation. */
open_vwork ()
{
    int i;

    handle = graf_handle (&dum,&dum,&dum,&dum); /* Get graphics handle. */
    for (i=0; i<10; work_in[i++] = 1);      /* Initialize GEM arrays.    */
    work_in[10] = 2;
    v_opnvwk (work_in, &handle, work_out); /* Open virtual workstation. */
}

/* Do file selector box. */
sel_file ()
{
  int button,    /* File selector button value. */
      i;         /* Loop variable.              */
  char path[50], /* Storage for filenames.      */
       file[13];

  for (i=0; i<20; path[i++]='\0'); /* Fill filename space with nulls. */
  for (i=0; i<13; file[i++]='\0');
  path[0] = Dgetdrv() + 65;        /* Convert default drive to char.  */
  strcpy (&path[1],":\*.*");       /* Complete the pathname.          */
  fsel_input (path,file,&button);  /* Open the file selector box.     */
  prnt_info (file,button);         /* Go print results.               */
}

/* Print out the user's choices. */
prnt_info (file,button)
char *file; /* Pointer to the chosen filename. */
int button; /* Value of the button pressed.    */
{
  v_gtext (handle,28,50,"The file you chose was: ");
  v_gtext (handle,220,50,file);
  v_gtext (handle,28,66,"And you pressed the ");
  if (button == 0)
    v_gtext (handle,188,66,"CANCEL button.");
  else
    v_gtext (handle,188,66,"OK button.");
}

/* Waits for left button to be pressed and released. */
button_wait()
{
    evnt_button (1,1,1,&dum,&dum,&dum,&dum);
    evnt_button (1,1,0,&dum,&dum,&dum,&dum);
}
```

Listing 2.
C listing.

```
/*************************************************************/
/*                  C-MANSHIP, Listing 2                     */
/*                      ST-Log #12                           */
/*               Developed with Megamax C                    */
/*************************************************************/

#include <osbind.h>

#define S_XOR_D 6
#define TRUE 1
#define FALSE 0
#define LEFT 1
#define RIGHT 2
#define HAND 3
#define OFF 256
#define ON 257

/* The required GEM global arrays */
int work_in[11],
    work_out[57],
    pxyarray[10],
    contrl[12],
```

```
    intin[128],
    ptsin[128],
    intout[128],
    ptsout[128];

/* Global variables */
int handle, dum;

/* Memory Form Definition Block */
typedef struct mfrmblk {
   long f_addr;     /* Addr of form data.                        */
   int f_w;         /* Width of the form in pixels.              */
   int f_h;         /* Height of the form in pixels.             */
   int f_wdwidth;   /* Width of the form divided by 16.          */
   int f_stand;     /* 0 = raster (RC); 1 = normalized (NDC).    */
   int f_nplanes;   /* Number of bit planes (1, 2 or 4).         */
   int f_r1;        /* The last three words are reserved.        */
   int f_r2;
   int f_r3;
} MFDB;

/* Data for the ANALOG "A" icon. */
long icon[] =
    {0x00000000,0x00000000,0x1FFF1FFF,0x1FFF1FFF,0xF000F000,0xF000F000,
     0x00000000,0x00000000,0x35552CCB,0x3C3823F8,0x58003800,0xF8000800,
     0x00000000,0x00000000,0x55554CCB,0x7C3843F8,0x58003800,0xF8000800,
     0x00000000,0x00000000,0xD555CCCB,0xBC3883F8,0x58003800,0xF8000800,
     0x00010001,0x00010001,0x57FDCFFF,0x3FFC07FC,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x5803C803,0x38020802,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x5803C803,0x38020802,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x5403CC03,0x3C020402,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x57F3CFF3,0x3FF207F2,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x555BCCCB,0x3C3A03FA,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x555BCCCB,0x3C3A03FA,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x55F3CDF3,0x3DF203F2,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x5503CD03,0x3D020302,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x5583CC83,0x3C820382,0x58003800,0xF8000800,
     0x00030002,0x00020002,0x5583CC83,0x3C820382,0xF800F800,0xF800E800,
     0x00030002,0x00020002,0x5543CCC3,0x3C4203C2,0xF800F800,0xF800E800,
     0x00030002,0x00020002,0x5543CCC3,0x3C4203C2,0xF800F800,0xF800E800,
     0x00030002,0x00020002,0x5563CCE3,0x3C2203E2,0x58003800,0xF8000800,
     0x00010001,0x00010001,0xFFC1FFC1,0xFFC1FFC1,0xF000F000,0xF000F000};
int icn_w = 48,  /* Width of icon.    */
    icn_h = 18;  /* Height-1 of icon. */

/* Main program. */
main ()
{
   appl_init ();           /* Initialize application.     */
   open_vwork ();          /* Set up workstation.         */
   graf_mouse (OFF,&dum); /* Shut off mouse.             */
   v_clrwk (handle);       /* Clear the screen.           */
   graf_mouse (ON,&dum);  /* Bring the critter back.     */
   do_icon ();             /* Go draw icon.               */
   v_clsvwk (handle);      /* Close virtual workstation.  */
   appl_exit ();           /* Back to desktop.            */
}

/* Initialize a virtual workstation. */
open_vwork ()
{
   int i;

   handle = graf_handle (&dum,&dum,&dum,&dum); /* Get graphics handle. */
   for (i=0; i<10; work_in[i++] = 1);     /* Initialize GEM arrays.    */
   work_in[10] = 2;
   v_opnvwk (work_in, &handle, work_out); /* Open virtual workstation. */
}

/* Main program loop */
do_icon ()
{
   int button,  /* Mouse button pressed.   */
       x,       /* Mouse X coordinate.     */
       y,       /* Mouse Y coordinate.     */
       ox,      /* Old Mouse X coordinate. */
       oy,      /* Old Mouse Y coordinate. */
       repeat;  /* Loop flag.              */
```

```
   graf_mouse (HAND,&dum);                  /* Switch mouse forms.       */
   x = 50 ; y = 50;                         /* Initial location of icon.*/
   repeat = TRUE;                           /* Get into WHILE loop.      */
   while (repeat) {                         /* Begin WHILE loop.         */
      x -= 30; y -= 20;                     /* Adjust for mouse coords. */
      ox = x; oy = y;                       /* Save old coordinates.     */
      graf_mouse (OFF,&dum);                /* Turn off mouse.           */
      draw_icon (icon,S_XOR_D,icn_w,        /* Go draw icon.             */
           icn_h,x,y,x+icn_w,y+icn_h);
      graf_mouse (ON,&dum);                 /* Turn on mouse.            */
      button = 0;                           /* Get into WHILE loop.      */
      while (button == 0)                   /* Begin WHILE loop.         */
         vq_mouse (handle,&button,&x,&y);   /* Get mouse status.         */
      if (button == LEFT) {                 /* If left button pushed...  */
         graf_mouse (OFF,&dum);             /* Turn off mouse.           */
         draw_icon (icon,S_XOR_D,icn_w,     /* Erase old icon.           */
          icn_h,ox,oy,ox+icn_w,oy+icn_h);
         graf_mouse (ON,&dum);              /* Turn mouse back on.       */
      }
      if (button == RIGHT)                  /* If right button pushed...*/
         repeat = FALSE;                    /* get out of loop.          */
   }
}

/* Perform raster operation. */
draw_icon (data,mode,width,height,dx1,dy1,dx2,dy2)
long data[];
int mode,    /* Raster writing mode.                                    */
    width,   /* Icon width.                                             */
    height,  /* Icon height.                                            */
    dx1,     /* Upper left X coordinate of destination rectangle.       */
    dy1,     /* Upper left Y coordinate of destination rectangle.       */
    dx2,     /* Lower right X coordinate of destination rectangle.      */
    dy2;     /* Lower right Y corrdinate of destination rectangle.      */
{
   MFDB s_m,    /* Form definition block for source.                    */
        scr_m;  /* Form definition block for screen.                    */
   int pxy[8]; /* Coordinates for source and destination rectangles.   */

   s_m.f_addr = (long) data;            /* Store addr of icon data in MFDB. */
   s_m.f_w = width;                     /* Store width of icon in MFDB.     */
   s_m.f_h = height;                    /* Store height of icon in MFDB.    */
   s_m.f_wdwidth = width/16;            /* Store icon width/16 in MFDB.     */
   s_m.f_stand = 0;                     /* Raster coordinates.              */
   s_m.f_nplanes = 4;                   /* Low resolution (4 bit planes).   */
   s_m.f_r1 = s_m.f_r2 = s_m.f_r3 = 0;/* Zero out reserved words.          */
   scr_m.f_addr = 0;                    /* Set up screen MFDB.              */
   pxy[0] = 0;        /* Upper left X coordinate of source block.        */
   pxy[1] = 0;        /* Upper left Y coordinate of source block.        */
   pxy[2] = width;    /* Lower right X coordinate of source block.       */
   pxy[3] = height;   /* Lower right Y coordinate of source block.       */
   pxy[4] = dx1;      /* Upper left X coordinate of destination block.   */
   pxy[5] = dy1;      /* Upper left Y coordinate of destination block.   */
   pxy[6] = dx2;      /* Lower right X coordinate of destination block.  */
   pxy[7] = dy2;      /* Lower right Y coordinate of destination block.  */
   vro_cpyfm(handle,mode,pxy,&s_m,&scr_m); /* Do the raster operation.  */
}
```

●

ST Music Box

XLENT SOFTWARE
P.O. Box 5228
Springfield, VA 22150
(703) 644-8881
Medium resolution $49.95

by Charles F. Johnson

XLent Software's entry in the low-end MIDI sequencer market is called **ST Music Box**. This program uses either the ST's internal sound chip or an external MIDI synthesizer to play music. No real-time recording is supported; all notes must be entered one at a time using the "smart" editor in the program. When you've finished a score and saved it to disk, another program lets you print it to an Epson, NEC/Prowriter, or Gemini 10X printer. The **ST Music Box** reviewed here only works with a color monitor, but XLent promises a monochrome version soon.

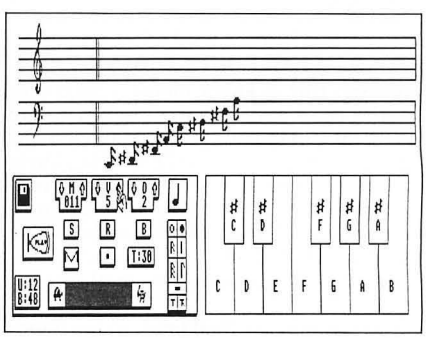
Entering music data with this program is a slow and cumbersome process. The editing screen displays treble and bass staffs, an icon box/control panel, and a one-octave piano keyboard. To put a note on the staff, you must select its time value (eighth note, quarter note, etc.), then select its octave, then click on the piano note that represents it. Compare this with another low-end sequencer, **Music Studio** from Activision—there, you simply drag the note onto the staff. **ST Music Box** does have some nice capabilities, however, such as the ability to change tempos, instruments (patches), portamento settings and transpositions at the beginning of every measure.

One well-meant feature of the note editor quickly becomes annoying: it won't let you move to the next measure until you've entered the proper number of notes and rests to fill the current one. While this does prevent you from making a mistake, it also prevents you from deliberately entering the wrong number of beats (which I have been known to do). The editor has some other severe drawbacks. Notes cannot be tied together, and there is a limit of twenty-four notes per measure per voice. There are functions for copying, moving, deleting, inserting, etc., but they only work on entire measures at a time.

ST Music Box does not use the standard GEM interface of drop-down menus and dialog boxes. Instead, function keys are used to select most operations (shades of IBM). Pressing a function key brings up what XLent calls a "window" (not a GEM window), which usually presents you with another list of choices—all accessed by pressing another function key! On an ST, there's no excuse for this kind of confusing interface; TOS contains built-in, easy-to-call routines to handle all kinds of user input. Worse, after exiting one of these "windows" the screen redraw is excruciatingly slow—much slower than a normal GEM redraw. Things like this can really disrupt the creative concentration needed to compose music. . .

The program allows you to record up to eight tracks (playable over four MIDI channels), so it is possible to enter a fairly complex score. But all the tracks must be the same length; tracks that are shorter often end up with the last note played ringing through the remainder of the longer tracks (this is called a "MIDI hang"). As the music plays, you see a screen with some quarter notes dancing around. I could find no real purpose for this display. And don't touch that mouse while the music plays—any mouse movement has a severe effect on the timing of the playback. To stop the music, you press the CONTROL key. . . I wonder if XLent noticed that the ST has a key labeled UNDO? Until **ST Music Box**, I'd never seen a program that used the CONTROL key by itself to perform a function; CONTROL is used almost universally as a sort of "shift" key, in conjunction with another key. This kind of nonstandard interaction with the user only serves to confuse, and could have been avoided quite easily.

The Printer program lets you print the scores saved to disk, with all or selected voices showing. As with the Music Box, all program operations are performed with the function keys. You can add text to a printout at any position on the workscreen; this can be used to print lyrics along with melodies. Unfortunately, no attempt is made to connect notes properly with beams; each note is printed individually. The resulting printout in no way resembles a professional score, and is unsuitable for anything but casual home use.



Commercial programs should always have extensive error-trapping routines; one

of a programmer's main jobs is to keep the user from accidentally screwing things up. I really started to dislike this program after it trashed one of my disks during a save operation. I tried to save a sequence I'd worked on for several minutes, and forgot that the (write-protected) master disk was still in the drive. First I got a TOS error box, saying the drive could not be accessed, and giving me buttons to continue or abort. The only problem was, **ST Music Box** had turned off the mouse cursor, and I couldn't select either button. I put in another disk and hit RETURN (the default button in this box was "Continue"), and **ST Music Box** proceeded to mangle my disk directory.

Good MIDI software is finally beginning to appear for the ST. **ST Music Box** has already been surpassed by programs which do more, cost about the same, allow real-time note entry (in other words, you just play your synthesizer), and are relatively bug-free as well. This program has a half-finished look, as if options were thrown in as an afterthought, not as a result of careful design. I can't really recommend **ST Music Box** to anyone. This is a mediocre product. //

*Charles F. Johnson is a musician by trade, drawn into the Atari computers by **Star Raiders** about five years ago. When he got the BASIC cartridges, he was seriously hooked. He currently lives in Los Angeles with his wife and the most intelligent cat in the world.*

EZRAM 520

TERRIFIC PERIPHERALS INC.
Brookline, MA 02146
(617) 232-2317
All resolutions $199.00

by Charles Bachand

Long ago, a champion emerged from the enchanted kingdom of Sunnyvale. A powerful hero, he came carrying a mighty sword and a shield engraved with the name 520ST—*a name that soon came to command respect wherever it was heard. He ruled the land of Computerdom for many long and happy months, until one day, from the far-off land of his own birth, there came a new and more powerful knight, known as the* 1040ST. *The new champion's strength lay in the fact that he could hold twice as much as 520ST, because he had been bestowed with a double quantity of the Rare and Magical Elements—those things that have all but defied description and have come to be known only by the mystical name* RAM. *Not to be overshadowed by the appearance of the newcomer, 520ST sought from the land of Cambridge, Massachusetts the help of a Wizard. This mage, known to the people as* Terrific Peripherals, *was trained in the modern and mysterious arts of RAM and RAM upgrades.*

Sorry, I got a little carried away there—must be that fantasy novel Clayton's got me reading. Anyway, if you haven't already guessed, this is about memory upgrades for the Atari 520ST computer. There are basically three ways you can expand the memory size of your 520ST to 1 megabyte of RAM. The most expensive method is to trade in your 520ST for a 1040ST. Since most of you will toss this idea aside ("What does he think I'm made of—money?"), let's take a look at the other two methods left open to us.

The cheapest method (and, in my mind, what also has to be the most dangerous) involves soldering sixteen RAM chips in "piggyback-style" on top of the original set of chips in the computer. Why is this dangerous? If one stray spark of static electricity—from something as minor as walking across the carpet, for example—should touch a dynamic RAM chip, you might as well just throw it away. Even the heat from the soldering iron used in the installation of these chips can fry them fast. The money that you'll spend getting your computer fixed can best be put toward the purchase of the third alternative: a 512K memory upgrade board.

The **EZRAM 520** is a "piggyback" board, which should not intimidate anyone, since no soldering is required. To install it, we must first disassemble the computer and remove the printed circuit board. When you examine the circuit board, you'll notice there's a section on it that's isolated from the rest by a metal shield. In my particular case, the top of this shield was soldered in place and had to be unsoldered before proceeding. (Where did I put those blasting caps?) Once the shield is removed, you'll see a large 40-pin chip mounted in a plastic socket. This chip generates the video signals—both black-and-white and color—that finally end up appearing on your monitor screen. Carefully remove this chip from its socket and insert it into the 40-pin socket on the **EZRAM 520**. The memory board itself is then plugged into the socket on the 520ST that became vacant when we first removed the chip.

It's a little easier to visualize this with diagrams—of which there are many—than by reading the long-winded sentences that I can come up with. The rest of the electrical work consists of routing the ribbon cable protruding from the side of the memory board over to the Memory Management Unit (a square 68-pin chip used to decode the address lines, as well as to provide memory refresh to the RAM chips). This cable is terminated in a plug that is pushed snugly on top of the socket holding the MMU, thus eliminating the need for any soldering whatsoever. Quick, simple and very neat.

My only complaint had to do with the documentation. It had a lack in only one area, reminding me of the old nursery rhyme, "Humpty Dumpty." Whoever wrote the instructions forgot to tell the user just how to put the ST computer back together again. Luckily for me, I've had plenty of experience in this area; the computer is back in one piece again. I've since checked with the people at Terrific Peripherals. They are aware of the oversight and have already done up new documentation.

The **EZRAM520** has been in one of our STs for a few months now, and we have had absolutely no problems with it. I heartily recommend this upgrade to anyone who can use a screwdriver and a pair of long-nose pliers—and who takes the necessary steps to reduce the possibility of static electricity being encountered during the board's installation.

And so, the brave 520ST became as powerful as the 1040ST that would have replaced it. They now live together in peace and harmony. But there are rumors of new heroes soon to emerge in this land, with more power and more RAM than ever before imagined. We can only hope that the Wizard's magic will, once more, be sufficient to the task. //

Developments in the Atari ST world.

by D.F. Scott

This month, a complete and extensive overview (another one) of Atari's new machine, a glance at four other new Atari machines, and a celebration of being debt-free. Our sources this month include: Shiraz Shivji, Vice President for Research and Engineering; and Neil Harris, Marketing Communications Director.

Jack to IBM: "PC-ing you."

Atari Corp. interrupted a two-year spate of 68000 developments to slip into their line a new series of IBM PC-compatible computers, starting with the Atari PC—an 8088-based, 4.77 MHz (switchable to 8.0 MHz) machine that comes either in a plain grey package without a monitor, or with the Enhanced Graphics Adapter (created by IBM) and a monochrome monitor. These configurations will be retailing for $499.00 and $699.00, respectively. Reports are that a '286- and '386-based PC will be added to the line. Each PC adheres to what might be called the "Atari Expansion Principle," which will be described later; further details on the PCs will be revealed in later installments.

Mega news.

The long-awaited official announcement of Atari's new 68000 machine came at last January's Winter Consumer Electronics Show in Las Vegas. The Mega ST workstation will be sold in configurations of 1-, 2- and 4-MB RAM. The Mega features a small CPU cabinet (similar to the Apricot or Apple IIgs), with spaces for internal disk drives (microfloppy and Winchester fixed), a detachable keyboard, and a driver for an external bus.

In order to expand on that last term, we'll take a detailed look at the interior of the Mega (code name: "Middle Box"), provided by Atari's chief engineer, Shiraz Shivji. Atari says the Mega is the computer that the people wanted. If so, the people must have wanted hardware expandability.

Shivji tells us, "The way we're building this Middle Box is that the motherboard, which is on the bottom, has essentially two layers. It's like a double-story building; you have another level where you can put another card. The motherboard has stakes sticking up which contain the full 68000 bus. Expansion will be done by plugging something into the stakes that are on the bottom board."

In the past, an "open architecture" machine has typically had slots for several expansion cards—usually about eight—fitting perpendicularly to the motherboard. Atari's engineers opted instead to make space for a single expansion board just larger than a card would be, but smaller than the motherboard. This is the "Expansion Principle" that will make the entire new Atari line unique, for better or worse. However, if the Mega-expansion card industry takes off the way it did for the Apple II and IBM PC, hardware hackers, and many other users, will want more than one expansion outlet.

So, Shivji tells us, Atari will introduce the "expansion box." He explains, "If you want to have a lot of extra memory, and a lot of other things, we will also offer this: you can have the second card on the top; and we'll have bus drivers that will take the 68000 bus outside the machine. This can then connect to another box that has much more memory—up to 8 megabytes—or anything else that can sit on the 68000 bus."

But why an external box; why not just provide eight or so internal expansion slots? The answer to that question leads into the matter of how the Mega ST earned its project name Middle Box. It implies that there's something on the Bottom and Top of the product line. In terms of cost and configuration, and not value, the current ST could be called the "Bottom Box" (don't take it personally). As for what the "Top Box" might be, regular readers of **Status report** will already know that I'm referring to the 68020 machine which Atari's prospectus called "a UNIX-based, multi-user, multi-tasking microcomputer system."

The term *multi-user* implies networking; and the nickname *Middle Box* suggests the Mega ST's place in the network. We predicted in our first **Status report** (**ANALOG Computing**, issue 48, November 1986) that if Atari was serious about selling a low-cost, high-power machine to business through Value-Added Reseller outlets, it would have to find a way to sell several of them; and in fact, networking capacity will be a major selling point for the Mega ST.

Re: NICs, LANs, CSMA/CD, etc.

Shiraz Shivji introduces us to what can tentatively be called the Atari NIC—Network Interface Card, the cornerstone of Atari's new machine-intercommunicability architecture. "The way we figure things will happen," says Shivji, "is as follows: I think we're going to supply a networking card as an option. We're interested in a lower-cost version of EtherNet, using standard RG-58, RG-59-type cable, 75

ohms, coaxial, which is readily available. I think we'll run it at about 4 megabits per second.

"A lot of people are using this version," continues Shivji. "It's called 'CheaperNet.' It uses the same principle—a CSMA/CD-type approach."

Here's what he means: EtherNet is perhaps *the* major networking scheme available for minicomputers today. It was developed by Xerox and can run on a fiberoptic cable, although coaxial is usually preferred because, as Shivji pointed out, it is so widely available. A company called 3Com has developed a coaxial cable version of this Local Area Network system, called EtherLink, for the Intel-based computers (IBM). 3Com is currently enjoying strong financial success, especially in the over-the-counter stock market.

CSMA/CD stands for "Carrier Sense Multiple Access with Collision Detect," and no, it has nothing to do with player/missile graphics. In a networking scheme, there are several receiving terminals linked by cable through an electronic carrier wave—not an audible wave as over a phone line, but more like the signal sent through a parallel printer cable.

The host unit sends bits of data to each terminal, but it has to have some way of deciding what data to send to what terminal, and when. This decision process is called the "contention scheme"—and that's what CSMA/CD is. The process is shared between host and terminal: when the terminal has a couple of cycles open and is ready to receive, it sends a carrier signal to the host—a sort of tap on the shoulder. Once a carrier signal is received, the host responds with a package of bits to the terminal.

Now, in cases where a cable is shared by terminals, one carrier signal cannot overlap another, or the host will become confused. Thus the "Collision Detect" principle: the terminal actually monitors the signal coming back for a wave overlap; if it finds one, it clears its end of the line.

As for the speed mentioned by Shivji, a "raw bit transfer rate" of 4 megabits per second (4Mbps) is about medium speed for an LAN system. The least expensive networking card now available for the IBM PC and Apple Macintosh (Corvus Omninet) runs at just over 1Mbps; a fiberoptic cable system can run at 10Mbps and over. The speed predicted for Atari's NIC, as well as its being run on cards, suggests a linear bus scheme for the network—a minor sacrifice of speed for convenience and low cost. In a linear bus system, host and terminals are connected in a daisy-chain arrangement, with in and out plugs on each NIC card.

This leads us to the hardware of the card. The host unit will be UNIX-based, so the terminal had better be, also. However, contrary to what was suggested here last month, GEM *isn't* being subtracted from the new system's ROM. At first we'd had indications—since GEM wasn't designed for a multi-user environment, and since additions or patches can rarely be made to GEM without several of Apple's lawyers looking over the programmer's shoulder—that it would be dropped from the new system. The networked UNIX environment may or may not involve GEM; the two just don't seem made for each other. Nonetheless, GEM—revised or not—*will* be in the system ROMs of the Mega. So full ST compatibility is assured.

On the IBM PC, MS-DOS (which is disk loaded) has had to be modified to handle networking. If the Atari system is to avoid having a disk-based networking TOS, the new operating system would most likely have to go on the NIC card, thus making the card larger.

Hence, the "top board" concept—it's perhaps the only reasonable way networking could be implemented. The expansion goes outside to a separate box where, most likely, the coaxial cables would plug in. The top board area might contain expansions to the Mega to give it UNIX terminal capacity, as well as more power for the video; but the actual networking would take place outside, where everything is much easier to access.

Hop on the bus.

Besides the NIC and the video expansion card, there's the matter of that extra 8 megabytes of memory mentioned by Shiraz Shivji. The former ST TOS couldn't address that much; so there are bound to be improvements in TOS somewhere.

Explains Shivji, "We'll have the full 16 megabytes of address space available, of which the internals—the area where the screen resides and the internal RAM—can be 4 or 8 megabytes. So you'll have the second 8 megabytes, but part of this is occupied by the ROM. There'll be, approximately, up to 1 megabyte of ROM. We have a little over 7 megabytes available for more RAM, if you need that much RAM.

"Or you could put peripheral chips on the outside. If you have an IEEE-488 chip, you can put it right on the bus. For example, there are standard chips from TI and Motorola that sit on the bus and do IEEE stuff."

"IEEE stuff" means, among other things, communicating with professional digitizing, drafting, engineering and plotting peripherals—also known as Hewlett-Packard territory. You can see where this design philosophy could lead: a Motorola CPU running an AT&T multi-user operating system driving a bus that allows a TI chip to run HP equipment, all of which could connect to an IBM-type LAN—but all in a proprietary fashion. All the big guys, as well as the potential innovators and entrepreneurs who've been yearning for the opportunity to produce 68000-based hardware—all of them are brought together by the Atari system and, because of the uniqueness of the scheme, by Atari alone.

Note that 1 megabyte of ROM Shivji mentioned. If TOS consumes 192K of ROM, just imagine what would fit in a megabyte! As for hard disk drive storage, Atari has announced its Winchester half-height hard drive, which promises to be much faster than its SH204 20Mb drive. And about that keyboard: for those of you who've felt that typing on the ST was like surfing in a creek, you too can rejoice. Although the keyboard's internal feedback mechanism should be the same, the new design will be detachable, lightweight and have a crisper touch.

Says Shivji, "The ST was designed to be put in many different packages from the start. That's why the keyboard talks to the main unit through a serial bus. It's just a little detachable keyboard, with a telephone-type [coiled] connector. . . but that doesn't change the electrical part of it. We will have several versions of key switches, which will give different feels."

To answer another question we're all asking: the Mega *will* contain that elusive blitter chip. What's holding up that first chip ship, Shiraz? "We're waiting for parts from our supplier. As soon as we get parts, I think we'll be ready to ship." In the meantime, Shivji's division is continuing to perform software checks on the prototype chips.

And, as if the above weren't enough (yes, Virginia), there is a clock chip! Battery-powered, linked to the control panel, just the way we wanted it. Shivji also mentioned the possible addition, through the expansion box, of extra serial ports—which suggests the possibility of a multi-function card. This would be the ultimate peripheral driver.

How will the machine be sold? Atari's VAR Sales Director, Sig Hartmann, is still negotiating for a national sales outlet, with a big advertising budget and a service department. Like it or not, the introduction of the Atari PC line may be the key to introducing and selling the ST series in the major VAR chains. The ST has gained extra press coverage, as well as prestige, as a result of the PC announcement. Remember, the two machines can communicate with each other via UNIX/XENIX and, perhaps, via Atari NICs—so selling in bundles becomes quite feasible.

In the meantime, the specialty store channel is still open, but it may not be the preferable one. First, there's the servicing problem: any service contracts would have to be made by the dealer, and dealers won't offer service contracts if they can't get parts when they need them. Second, the new ST would face competition in specialty stores. . . from the old ST. There's still speculation over whether Atari will phase out the old ST.

Atari's Neil Harris tells us that the old software-bundling policy will be scaled down, to include only BASIC (perhaps MetaComCo's new version), Logo and maybe **ST Writer**. The reason, Harris says, is that since there's a plethora of ST software now available, Atari should be the last company to compete in this area by offering comparable products for free. In other words, software bundling was a good policy when there were hardly any other software choices around. Now that the bundled packages can be surpassed in quality, more power to the third party! Atari will remain demurely in the background, to emerge only occasionally for a friendly sparring match with, but never a full-scale war against, other software manufacturers. This may be bad news for Atari's Software Division chief Gershon Blumstein, who doesn't want his group producing mediocre material.

Atari: in-the-black-Inc.

Now for a quick update of Atari's financial condition. The third-quarter '86 results are in, and once again sales have more than doubled over those of the same quarter in '85.

Specifically, sales for the quarter ending September 30 were $59,878,000, versus $28,985,000 for the same quarter of the previous year. The public share offering on the American Stock Exchange, coupled with these tremendous sales figures, enabled Atari to pay off in full all that it owed to Warner Communications—all $36.1 million of it—and still rake in a $4,486,000 net profit for that three-month period.

Keep in mind that WarnerCom owns 7.1 million of Atari's 23,762,000 shares—nearly 30% of the company, substantially more than was offered to the public. Third-quarter net income is 41 cents per share; and stock now trades for about two dollars a share more than the initial November 7 offering price. So even Warner is once more raking in the green.

Atari President Sam Tramiel, in a press release, claimed that "demand for the Atari video games and the ST computers exceeded our expectations." Both assertions are debatable, especially since a recent *BusinessWeek* article stated ST unit sales to be 150,000—as in the prospectus—and not the 250,000 previously quoted by Atari sources.

Conclusion.

The **Reader comment** sections in **ANALOG Computing** and **ST-Log** last January contained some of the best letters I've read in either magazine; I'd like to take this space to address the issues raised by two of them.

Robert L. Newman of San Antonio, Texas, in **ANALOG** , brought up the issue of Atari service. He feels there is no way the serious computer purchaser will believe in the ST if the corporation isn't willing to back it up with a national servicing program similar to the old Atari's program. Mr. Newman should know; he's a service technician for Atari and Commodore products. When the Tramiels were at Commodore, he had the same problem with them.

If Atari is ever to be believed by its buyers, VAR dealers, specialty dealers, shareholders and servicemen, it needs to rearrange some of its priorities. A corporation cannot survive on the enthusiasm generated by new product announcements alone; it must deliver on its promises. It must get parts out on time. Working to pay off debt is no excuse for a poor service program. If the debt-ridden Atari under James Morgan could deliver the parts, so can the debt-free Atari under Tramiel and sons.

[Atari Corp. is aware of users' complaints regarding service and support. Neil Harris assured our staff at CES that the company has added personnel in this department. According to Harris, in the last six months customer service has made a "180-degree turn." Thanks for the comment, Mr. Newman. We're hoping that conditions have indeed improved. —Ed.]

Robert J. Fusillo's comments in **ST-Log** hit me, in a sense, where I live. Like him, I bought the ST for its visual and aural power. But, like many of the "Log-icians" who write for **ST-Log**, I'm a hobbyist programmer—more hobbyist than programmer most of the time—and in our technical enthusiasm, when we love something, we take it apart. When we get too technical, we can antagonize users like Mr. Fusillo, who purchased his ST as a writing, artistic and musical tool, not an electronic tinkertoy.

Mr. Fusillo feels the "68000 generation" will be comprised primarily of users, people who aren't interested in ST internals and who couldn't care less about LAN specifications. I agree, but only partly. Certainly the ST was designed to be used by nontechnical people. But it's also a programmable machine. There are people who drive automobiles and leave the mechanics of their machine to mechanics. They can't be overlooked. But I haven't met a Ferrari owner yet who didn't brag about every little detail of the car's engineering, and who didn't spend a good deal of time under the hood, even when the car was in perfect condition.

My feeling is that the 68000 generation is comprised of all kinds of people. And to try to restrict the classification any more than that would be to deny a substantial portion of that generation its right to entertainment in these pages. Thanks for an excellent letter, Mr. Fusillo.

Finally, my very special thanks to Shiraz Shivji, the genius behind the ST, for giving us a spectacular look into the future. That's **Status report** for this month. I'll see you on Delphi. //

Assembly line

A column to teach assembly language for the 68000.

by Douglas Weir

This is the first article of what I hope will be a series devoted to explaining the basics of 68000 assembly language programming to ST users. The joys of assembly language are different from those of the higher-level languages. You could say that working in C is like being in the Navy—you can sail anywhere you like, but you can also run aground, and sink. With Pascal or Modula-2, you're in the Air Force (above the details, and sometimes in the clouds); in BASIC, you poke around with the Coast Guard.

But when you program in assembly language, you're in the Army. You get the foot soldier's close-up view of what's going on, and sometimes getting results can seem like hand-to-hand combat.

So, why bother with assembly language?

It's a question you will often hear asked, and you will always hear an answer for it. But it won't always be the same answer. Here are some of mine.

First: even a little experience with the computer's way (and not Pascal's, or BASIC's, or even C's way) of storing data or handling files can give you a lot of insight into the behavior of compilers and interpreters. Whether you realize it or not, when you do a POKE or a PEEK in BASIC, or look for the "null" byte at the end of a C string, or grapple with Pascal's idiotic EOF convention, you're working at the assembly language level.

Second: you don't have to be a hardware hacker to program in assembly, but you can't help but learn something about how the hardware works. There are more similarities among today's popular microprocessors than there are differences, so what you learn about the 68000 will give you at least some insight into how a 6502 or 8086 or 80386 works. In many ways, once you've really learned one assembly language, you know them all, and it's basically a matter of looking up new instructions and addressing modes to accomplish the same tasks, using (for the most part) the same principles.

Third: there are still plenty of programming tasks that can only be accomplished in assembly language. You suddenly find that you want to go behind the operating system's back, or access features of the hardware that aren't available through your favorite language, or a program has to be made to run as fast as possible, or to fit in as small a memory space as possible—in all these cases, assembly language routines or entire programs are usually the only answer.

Fourth: it's fun. And think of the satisfaction of actually being able to make sense out of dc.w $ffff and bra.s zippety_zap and rol your_own,d0.

I'm going to assume that you've already done some programming in one or more of the high-level languages, such as BASIC, Pascal or C. It's not impossible to learn assembly as a first programming language—it may even be desirable—but nobody does it. I'll often try to illustrate things by referring to these other languages. Also, I know that there are many who are interested in learning assembly primarily in order to write short routines within larger programs, and I'll try to supply material in this area, too.

Assembly language isn't as hard as some would have you believe. It's not the easiest thing in the world, either. A lot depends on what kind of program or routine you're writing. But, whatever your assembly language goals, you're lucky to be working with the 68000. Its combination of power and simplicity make it the ideal learning machine.

To get started, you'll need an assembler and a linker. There are many of these, and I don't have all of them. But I'll try to guarantee that every program presented here will be compatible with AS68 and Link68, the assembler and linker supplied with the Atari Developer's Kit. (The other candidate for "standard" status, Metacomco's MacroAssembler, varies wildly—I emphasize *wildly*—from version to version. I have versions 10.195, 10.200 and 10.204, and the only dependable copy has been 10.195. I use 10.195 regularly, but I wouldn't trust anything later without a lot of testing.)

You'll also need an editor or word processor to type in programs. I use **1st Word**, but any program that outputs plain ASCII files (without special formatting and style codes, etc.) should be fine.

Finally, although it's not an absolute necessity, you should have a copy of the Motorola *M68000 Programmer's Reference Manual*. There are many 68000 books available, and many are very good—I hope to mention some of them as time goes on. However, none of them has all the information that the Motorola book has, and you never know what you might want to look up. It has a nice separate-page format for each of the 68000 instructions; you'll use it a lot. Currently there are two versions available: the fourth and fifth edition. For our purposes, either one is fine (the fourth edition, if you can find it, is cheaper).

For a real understanding of what assembly language is and how it works, we have to take a close look at the environment it exists in: namely, the computer.

Computer organization.

Originally, I was scheduled to write an article this month called "Computer Organization." It was to show how to arrange an ST, power supplies and all, on and around a normal desktop. However, the background research involved became too complex, so I suggested that I write an assembly language column instead. At a spe-

cial **ST-Log** board meeting, this was reluctantly agreed to.

Usually, *computer organization* refers to the parts of a computer system and the way they work together. For our purposes we can divide a system like the ST into three main parts: *memory* (where data and programs are stored), *microprocessor* (the "brain" of the computer—the 68000 chip in the ST), and *Input/Output (IO) devices* (such as the monitor, disk drives, etc.). Then there are those darned power supply units—where do you put them, anyway?

In this installment, we'll be discussing the first two parts only, computer memory and the microprocessor itself.

Memory.

To begin with, we can imagine a notebook containing several pages of graph paper. Suppose that the squares of the graph are big enough to write numbers in, so that there are only, say, 100 squares on each sheet. If we now have a list of numbers we want to save—for example, the ages of all of **ST-Log**'s hundreds of staff personnel—we could write these into our book, one number per square. Each number (in most cases consisting of more than one, and less than three, digits) would be written in its own square.

If we wanted to refer to a particular number, we could say that it was in a certain square in a certain row on a certain page in our book.

This may not be such a bad system for storing people's ages, but suppose we were keeping a list of the word-counts of **ST-Log** articles. These numbers would vary widely. Some would be small enough to fit into our squares quite nicely, but others (such as the size of my upcoming "War and Peace Character Generator") would probably be much too big to fit all of the digits into one square. What to do?

One easy solution would simply be to get graph paper with bigger squares. But let's assume we can't do that. Instead, we have to think of a system for splitting up the numbers among more than one square. In other words, we could say that the first square on page 1 holds part of the first number, and the second square the rest of it; the next two squares hold the next number, and so on. Now all we need is a rule for splitting the numbers.

Let's do something simple. Since it might be hard to squeeze more than four digits into one square, we'll let each square hold four digits only. That means two squares taken together can hold an eight-digit number. If the first six squares in our notebook contained the digit-groups:

0123 0016 0001 0000 0000 0001

—then our first number would be *1,230,016*; our second would be *10,000*; and the third number would be *1*. The point here is that the first number would *not* be *123*, nor the third number *1*—you would have to take a pair of squares together to get a valid number. More than that, you would have to get the right pair. For example, taking the second and third squares together would give the number *160,001*, which is invalid.

To read numbers stored in a book set up in this way, you must know (1) how many squares to read together, and (2) which squares are the "beginnings"—as opposed to "middles"—of numbers (we'll call these *boundary* squares). And all of this complication is caused by the fact that one square can hold only a limited number of digits.

Let's stop here for a moment. Our notebook of graph paper is still some distance away from being an accurate model of computer memory. But we're close enough to the real thing to point out some interesting characteristics of how memory works and how a microprocessor uses it.

First of all, consider how neutral our storage system is. Those numbers could represent anything. They could be telephone numbers, shoe sizes, scores from hockey games, or encoded bossa nova rhythms—you can't tell from the numbers themselves. Furthermore, there's nothing to tell you that pairs of squares have to be combined in order to get valid numbers. Something else—in the case of a computer, a program consisting of the proper instructions—is needed to make sense out of this material.

This brings us to the next point. It is very easy to read invalid numbers, by getting the number boundaries wrong (i.e., reading the wrong two squares together). There's a way we can check on this, however. If there are ten squares in a row (meaning five numbers), and if each row of squares is numbered from *0* through *9*, then we know that each valid pair begins with an even-numbered square (assuming *0* is an even number). Now, as long as we stick to our two-square storage system, we can always be sure of reading a valid number, simply by always starting with an even-numbered square. We will see that the 68000 uses a similar system to make sure that it is reading memory correctly.

Things can get even worse. After several pages of word-counts, suppose we now want to store some SAT scores. These are never bigger than four digits (if yours is, write to me in care of this magazine, immediately). It would be a shame to waste all those even-numbered squares just so we could use the odd-numbered ones. So, for the SAT scores, we'll go back to our old system, and call each square a complete number. We'll just have to remember where our list of word-counts ends and the SAT scores begin. And after that, I'd like to enter my collection of Hungarian farm implement serial numbers. Since it turns out that these are often up to sixteen digits long, they will be stored in groups of four squares. Again, we'll have to remember where this section begins and ends, to be sure of reading valid serial numbers.

Sounds complicated, doesn't it? But this is the way computer memory is organized and reorganized by programs, to make efficient use of the space available. As we will see, some special formats (very similar to the examples we've been talking about) are built into the 68000; others can be temporarily created within an assembly language program.

Finally (this is the third point), let's take another look at how we locate our stored numbers. We can refer precisely to the location of any four-digit group by giving page number, row number on the page, and square number within the row. But what about separate digits within a square? We don't have a handy way of specifying the location of a single digit within a group (for the moment, we'll disregard the fact that there doesn't seem to be any reason why we'd want to). Computer memory has a similar limitation, as we'll see.

Programs.

That's how data looks in memory—more or less. What about programs? Where are they, and how does a 68000 read the instructions contained in them and then carry out those instructions? Now we're approaching the basic concepts of what assembly language is all about.

Every microprocessor has a special internal counter that always refers to some location in the computer's memory (if it doesn't, then we're in trouble—how much trouble, we'll soon see). Using our notebook analogy, a typical reading from this counter might be *1 0 0*, which would mean "page 1, row 0, square 0." This counter is the heart of every microprocessor. Assuming that everything else is normal, the microprocessor will always be somewhere in the midst of a forever-repeating process.

This is what happens: the internal counter is checked, and the memory location it refers to is read. In our example, the microprocessor would read the contents of square 0 on row 0 on page 1 (we'll assume it was constructed to read one-square numbers rather than groups of squares together). The four-digit number from this square is assumed to be a code that tells the microprocessor what to do next—in other words, it's an *operation code*. Before it actually executes this operation, however, the microprocessor updates the internal counter so that it refers to where the next code is located.

Unless it's told differently, the microprocessor assumes that the next code is located right after the last one, so it simply adds 1 to the counter, which would now read *1 0 1*. So far, everything described has been automatic; nothing the programmer does affects it. Now, however, it's time to execute the current operation code. This is where the programmer comes in; the code is part of a program he or she has

written. Typical operations are: add two numbers, read a location in memory, etc. When the operation is completed, the cycle is repeated; the next code is automatically read ("fetched") from memory, the counter is updated, and so on. This process continues until either something goes wrong or the computer is switched off.

So a program is stored in memory, just like data. What's more, it looks just like data. A program is simply a series of numbers, like our list of SAT scores. The difference is that the numbers making up a program are used by the microprocessor in a special way. They tell it what to do, step by tiny step, in order to perform meaningful (we hope) tasks, such as drawing a picture on the screen, targeting an ICBM, and so on.

Obviously, only certain numbers will "make sense" to the microprocessor as valid operation codes. So what happens if, for some reason, it starts reading from an area of memory that contains, not operation codes, but, instead, SAT scores?

If the microprocessor happens to be a 68000, it will realize that it has a meaningless code, and will automatically substitute in its internal counter the starting location of a special program, written for just such an emergency. To make a long story short, on the ST this small program will end up printing four cute little bombs on your monitor screen (it never happened to me, of course). The program tried to "execute an illegal instruction." Result: it "crashed."

Since a microprocessor is always, normally, fetching instructions from memory and executing them, it's clear that your computer is always in the middle of running some program or other. If nothing else, your ST is running the desktop program. All of this depends on that automatic counter we described, which always tells the microprocessor where the next instruction is. For obvious reasons, this counter is called the *program counter*.

To run a program, you (1) put it in memory somewhere, and (2) load the program counter with the location of its first instruction, then—let 'er rip. The microprocessor's automatic ever-repeating instruction-fetch/instruction-execute cycle will do the rest. If not, then we can be sure *we* did something wrong, not the computer.

Now if you think of programs written in BASIC, it's clear that very few of them start execution at the first line and continue sequentially down to the last line, where they end. Real programs have various commands that allow you to "goto" subroutines or loop through sections of code a certain number of times. Similar instructions exist in assembly language. The difference is that these instructions directly load new values into the program counter, so the margin for error is not great. The program counter may start off with the correct value, but erroneous "branch" or "jump" instructions can change that very quickly.

Assembly language.

The pool of all available operations a microprocessor can execute is called its *instruction set*. When you program in assembly language, you use this instruction set directly. Obviously, this is different from programming in BASIC or Pascal, but what is the difference, exactly?

If you were to program by writing down the actual numeric codes used by the 68000, you would be using *machine language*. If you wanted the computer to do something useful, such as print several words on the screen, you would have to figure out what sequence of machine operations would accomplish this, then look up the numeric codes for these operations, and finally input them into memory.

If there were places in your program where the normal sequence of operations was changed (for example, in loops or where execution goes to a subroutine—like using GOSUB in BASIC), you would have to figure out the new value to be loaded into the microprocessor's program counter to accomplish the branch. Since this value would be the location of some other operation code elsewhere in the program, the value would have to be recalculated every time the number of operations (and therefore the distance in memory space) between the branch and the destination changed.

None of this is any fun, but that's the way they used to do it. The first giant step in programming efficiency came when someone got the idea (back in the early 1950s) to write a program that would accept a text file using English-like acronyms to refer to operation codes and output another file containing the numeric codes themselves. Even nicer, all of the tedious location calculations were handled by this program. Now, when you wanted to jump or branch somewhere else in the program, you didn't bother with explicit locations at all. Instead, you gave the destination a label, and you jumped to the label. Every time the text file was translated into operation codes by the translator program, the *value* of the label (i.e., the memory location it referred to) was recalculated, and that value was used in any jump operations which referred to the label. BASIC programs have similar labels—namely, the line numbers of the programs.

Translator programs of this sort are called *assemblers*, and the set of acronyms and rules for using them is called *assembly language*. You probably know that some sort of translation process also takes place when a Pascal source file is turned into a program, or when you run a BASIC program. But the translation done by an assembler is the simplest kind of all. There is practically a one-to-one correspondence between what you see in an assembly language source listing and the program as it exists in memory. On the other hand, the distance between a Pascal source file and the resulting program is very great indeed. It's not difficult to write a program that takes another program file and reverse-translates it back into assembly language (such a program is called a *disassembler*), but I've never heard of one which would output a Pascal or C source file, and the difficulties involved in writing one would be considerable.

The translation performed on assembly language is called, logically enough, *assembly*. When a C or Pascal or Modula-2 source file is translated into a .PRG file, the process is called *compilation*, and the programs that accomplish this, *compilers*. Compiling is much more complicated than assembling, because the compiled languages are so much further removed from the native machine language of the microprocessor. This is because compiled languages are designed for use by humans. Machine languages describe operations that were designed by engineers, to be performed by machines. Assembly language tries to bridge the gap.

BASIC is different from all of these. When you run a BASIC program, you have to be "in" BASIC. On the ST, this means you must first run the program called BASIC.PRG. Then you can load a .BAS file and run it. A .BAS file looks something like a C or Pascal source file, in the sense that it consists of English-like commands and keywords, and data. But you never get a .PRG file from a .BAS file, so how does the program run?

Actually, a BASIC program is translated into machine language, but the translation is done by BASIC itself, and on the fly, line by line. Suppose the Soviet Minister of Really Heavy Industry were to give a speech at the United Nations. The Minister speaks in Russian. There are two possible ways an English-speaker could listen to the speech. He (or she) could put on a pair of headphones and listen to a simultaneous translation, or wait for a printed English version of the speech. In the latter case, you have a permanent copy; unless you're a connoisseur of Russian bureaucratese, you can dispense with the original. But, to get a simultaneous translation, you must always have the original speech.

The same is true of BASIC as compared to compiled languages. BASIC takes a program, reads it line by line and interprets its statements into routines that the computer executes. No machine codes are generated: a .BAS program says "do this" in a certain line, and BASIC.PRG does it. The next line says "do something else," and BASIC does this, too—and so on. This process is called *interpreting*, and BASIC is thus an *interpreter*. BASIC has its own lit-

tle internal programs that perform the actions specified in the programs.

We know what the relative advantages and disadvantages of assemblers and compilers are. The former allow programmers to write smaller, faster, more efficient and more flexible programs. The latter make writing programs easier (for the most part). Interpreters, too, exist mainly for the convenience of programmers. The nice thing about them is that you can make changes in a program and try them out immediately without going through all the intermediate steps required with a compiler. On the other hand, interpreted programs tend to be slow (the interpretation process takes extra time), and they require the interpreter to be run.

Binary numbers.

Before we take a look at the inside of a 68000, we have to refine our picture of computer memory somewhat.

It would be very nice if computers worked with decimal numbers like the ones in our imaginary notebook. But it was much easier for engineers to design memory circuits with two states (on/off) than with ten. Numbers written with combinations of only two digits (usually *1* and *0*) are said to be *binary*, and one digit of such a number—a binary digit—is called a *bit*.

If we ignore what the actual values of these binary numbers are (don't worry—that'll come soon enough), it's very easy to substitute them into our model computer memory: we'll just say that a digit can only be a *1* or a *0*, rather than any decimal digit from *0* through *9*.

Now let's make one more change. It turns out that we can get paper with larger graphs after all—large enough to write eight digits in one square. Voila—instant 68000 memory.

Call the groups of eight digits (remember, we're calling them bits now) *bytes* instead of squares. Call a pair of 2 bytes (which will consist of 16 binary digits, or bits) a *word*. Call two words taken together (a total of 4 bytes, or 32 bits) a *longword*. That's all there is to it.

You'll remember that we specified memory locations by giving page number, row number and square number. This rather resembles the computer's internal way of referring to memory locations. However, the programmer's view of things is a little different. As far as we're concerned, memory locations are specified in a much simpler way: the bytes are numbered 1-up from 0, beginning at the beginning of memory. A number which represents a memory location is called an *address*.

Binary numbers work on the same principle as decimal numbers. A number like *1603* uses four of only ten basic digits to represent a much greater value. The *1* really means "1000," the *6* means "600," etc. The position of each digit in the number indicates a power of ten, so the notation *1603* stands for the following computation:

```
   1     * (10**  3) =  1000
 + 6     * (10**  2) =   600
 + 0     * (10**  1) =    00
 + 3     * (10**  0) =     3
                        1603
```

(The ** means "to the power of," so *10 ** 2* is like saying "ten squared.") The powers of 10 increase by 1 with each new digit-position leftward.

Suppose we have a binary number *11001*. We can find out its decimal equivalent in the same way:

```
   1     * (2**  4) =  16
 + 1     * (2**  3) =   8
 + 0     * (2**  2) =   0
 + 0     * (2**  1) =   0
 + 1     * (2**  0) =   1
                        25
```

Thus, binary *11001* equals decimal *25*. Here, each new digit-position to the left means that the power of 2 is increased by 1. To count from *1* to *10* in binary, you would say "1, 10, 11, 100,101, 110, 111, 1000, 1001, 1100." (Be sure to repeat this aloud.)

Clearly, a number expressed in binary will usually require more digits than the same number in decimal. A byte of 8 bits in 68000 memory will have a maximum value of *11111111* binary, or *255* decimal. Two bytes taken together as a single binary number will have a maximum value of *1111111111111111*, or *65535*. When we combine 2 adjacent bytes in this way, we're doing the same thing as when we combined two squares in our notebook, each containing four decimal digits, to make one eight-digit number. The only difference is that now we're using binary numbers.

Binary numbers are cumbersome—there's no two ways about it. Fortunately, even assembly-language programmers don't normally have to worry too much about binary representation of data. The assemblers we'll be using allow the programmer to write numbers in decimal—conversion to binary, if necessary, is handled by the assembler. Still, there's a much handier way to write binary numbers which we'll learn next month.

Next month:
Inside the 10000100110100000.

We're now ready to understand the internal workings of the 68000 itself, and that's the main item on next month's agenda. We will also type in and run our first assembly language program. Until then, practice your binary multiplication tables—if you're that kind of person.

The Pawn mover

An interview with Anita Sinclair.

by Clayton Walnum

*When **The Pawn** first hit the ST software stands about a year ago, the Atari community started buzzing. Shouts of "You gotta see this!" rang through computer shops as wide-eyed people crowded around STs displaying some of the most dazzling graphics ever to be combined with a text adventure. Anxious fingers jabbed at keyboards, impatiently advancing the game from room to room, pausing only when a whirring disk drive informed them that a new scene was being loaded. With the picture on-screen, few could resist playing with **The Pawn**'s famous "windowshade" graphics. Leering demons' images scrolled smoothly up and down amidst mumbles of "Amiga, schmiga."*

Of course, that was just the start. Once into the game, adventurers soon realized there was more ticking under the hood than fancy graphics, which, though created by talented artists and delightful to the eye, still owed a great deal to the built-in capabilities of the ST—capabilities there for the tapping by any with the ingenuity to make the best of them.

*Pictures soon become, like your reflection in a mirror, all too familiar; the game must, in the end, speak for itself. Anita Sinclair, the head of the team responsible for **The Pawn**, knows this, and as she demonstrated the game for us—all the while annotating her actions with enthusiastic ebullience—it was the game, rather than the graphics, that brought a proud glow to her face: the sophisticated parser, the magical plot, the imaginative characters.*

We watched with bemused appreciation as she manipulated the game with a speed and assurance gained through hundreds (thousands?) of similar demonstrations. She tortured the parser ruthlessly, placing one game object within another, over and over—nesting them so deeply Houdini would've had difficulty untangling them. And then, with the input of one sentence (albeit one that resembled an English professor's nightmare), the parser retrieved the most deeply buried object. No human mind could've sorted out that labyrinthine syntactic construction.

*One thing was certain: here was a woman who loved her work. **ST-Log** managed to get Anita away from the bustle of a Consumer Electronics Show to talk for a few moments about programming, magic, artificial intelligence and the important role women have begun to play in the computer and software industry. The following is the result of that chat.*

> *"...the Infocom system was the best—until we came along."*

CW: *Which came first? **The Pawn** or the company, Magnetic Scrolls?*

AS: They both came together, really. The team we got together in England consisted of three people. One was very good at front-end design and graphics; one was extremely good at writing parsers; and the other was extremely good at writing databases. We were all adventure freaks, so it just came together. We'd known each other a long time, and we all were free of commitments. So we said, "Okay, let's set up some competition for Infocom." That was three years ago.

CW: *How do you think Infocom feels about that?*

AS: I think they appreciate it. They have not had any real competition till now. And it's quite fun for us. I can see a battle developing where, in a year's time, Infocom will come out with a parser better than ours, and a year after that, we'll come out with one better than theirs. It gets a lot of good publicity bouncing back and forth across the Atlantic. It must've been very difficult for them not to have had any competition. We've seen every adventure that's come into the market, and the Infocom system was the best—until we came along.

CW: *Where did the idea for **The Pawn** come from?*

AS: We were sitting in a McDonald's on Baker Street, trying to work out a plot. We had to have something really powerful as a game, so we could try out our system on it. We decided that the best way to do it is to have no direct goal, have a sort of open game, where many things can happen, many variations. The game just sort of transpired. The name came a lot later.

We designed the scenario then started putting characters in. One of us thought, "Okay, let's have this Honest John character,"—in fact, he came from me, based on a friend of mine who's a used car salesman. In England, you don't trust car salesmen—and the character Honest John emerged. Kronos came along because we wanted to make sure our system could handle spells and sorcerers and the like. Then we needed a character who does a lot of talking,

so the guru happened. The game wasn't a contrived thing; it developed. The original version had about fifty rooms and a very simple scenario. Now it's got about one hundred rooms and a very complex scenario.

> *"...in the game, you are a pawn ...being manipulated..."*

Once we had the game, we had to find a name for it. **The Pawn** seemed logical since, in the game, you *are* a pawn. You have no reason for being there. The minute you start doing things, you are being manipulated by people.

CW: *So the plot was developed by many people. There wasn't someone responsible for just the story and another for the graphics and another for the programming.*

AS: Right. With the follow-up for **Guild of Thieves**, it was very much one person who sat down and did it. With **The Pawn**, we all put in the ideas. It does suffer from the fact that it *evolved* rather than *was created*. But it's got character. For example, it's quite fun to go down into Hell and find a full-blown refrigeration system down there. I mean, what's it doing in Hell?

It's a fun game and can be played on any level. The humor is there; the humor came from all of us. Actually, a lot of the humor in the original was cut out of the final version. Having just seen **Leather Goddesses of Phobos**, I think maybe we should have left it in. **Leather Goddesses** is going to be such a hit.

CW: *Infocom's really not the first to release a sexually oriented adventure—although they seem to be claiming that. There was one from Sierra called* ***Softporn Adventure****.*

AS: The **Softporn Adventure** came out many, many years ago. I think I played the **Softporn Adventure** before I played any of the Infocom games on micros. It was wonderful, an amazing game. It was a real seller in England. Everyone wants to live out their fantasies.

CW: *Where did the idea for* ***The Pawn****'s scrolling graphic displays come from? Was that borrowed from the Amiga?*

AS: No. It just seemed the only way to do it. We've got a text adventure that's extremely verbose and we've got all these pictures. So what do we do? We played around with the idea of the text disappearing and the graphics emerging, but it didn't hang together. The player wants the graphics instantly. We made it so he can just pull them down. To take them away, just push them off. It seemed logical. If you look at most graphics and text adventures, they've got the graphics on top of the screen and a barrier of sorts below. We've got that, too, except we allow you to move the barrier where you want.

CW: *Most of the other games just turn the graphics on and off. Your way is a bit more elegant.*

AS: Well, we go for front-end design. I'm in charge totally of the front-end design, and I really believe that you've got to make things cute—you've got to have something that's instantly appealing. In England, when we demonstrated **The Pawn** at a Commodore user show, one of the magazines wrote afterwards that "the much publicized **Pawn**...is a venetian blind simulator." And it was really wonderful. It fits it. People come along and say, "Where are the graphics up and down controls?"

CW: *How long did* ***The Pawn*** *take to write?*

AS: About three years, plus all the front-end design. That took an extra six months on the ST version and six months more on the Amiga version—a long, long time. We haven't developed a game; we've developed a system, and that involves writing a lot of code. When we went into it, we said, "We have to have a system."

CW: *What makes* ***The Pawn*** *exciting for you as a programmer?*

AS: After all these years!...I spent a lot of time debugging **The Pawn**; there's something really fine about getting in there with debugging tools and just playing with objects. When you've got the debugging tools that I've got, you can edit objects and change objects and invent objects. You can say, "Okay, now this one's a character, so he'd better start walking around the game" — and he does! That's what I find really exciting.

Obviously, the game holds no interest for me anymore. I've played it maybe three thousand times. I could probably solve **The Pawn** with my eyes closed. Other people's experiences can be interesting. Sometimes you talk to somebody who's tried something you never thought of, and it turned out to be just the right thing. You sit back and say, "How the hell did you do that?"

About a year and a half ago, a guy came by from one of the top universities specializing in artificial intelligence, and we showed him our new character system. He was very cynical about it. He said you can't do anything like this on a micro; you've got to have 200 megabytes of RAM and several Crays. He just typed away, talking to one of the personalities in the game and coming up with some very technical responses. He typed in *I think, therefore I am;* the program came back a bit later on with, *Oh? Do you?* We sat there laughing, with tears running down our faces. We just couldn't believe it. The guy got up and walked out. He knew we weren't doing anything exceptionally clever.

CW: *What makes* ***The Pawn*** *exciting for the player?*

AS: Primarily, the lack of restrictions. There's only one puzzle that actually blocks you, and once you solve it, you can move around the entire game. Also, the fact that you don't have to think much about your phraseology. Sure, there are restrictions; we can't have a 10,000-word vocabulary. We don't have space for it. But we allow people to type things the way they would say them in English. Players will also enjoy the scenario. As it develops, you get very confused as to what's going on. There are other people to speak to, and they tell you to do this or that.

The game actually begins where the book leaves off. But the personalities that come across in the book help you to understand what's going on. One of the things you might discover is the fact that Kronos has made this pact with the devil, a nasty pact. He wanted some magical powers and agreed that he'd exchange his soul for them. In the book, this is all laid out for you. You *can* discover it in the game if you read things.

The book will also tell you that Kronos is trying to wiggle out of the deal. There's a subclause in the contract stating that if Kronos can get three other souls, the devil will release him. Kronos is a very good negotiator, so the devil is anxious to get a hold on Kronos before he can get the other souls. Of course, you soon realize that *you* might be one of the souls that Kronos wants to deliver to the devil. There are a lot of other characters Kronos could pick, so you have to decide if you should point Kronos in another direction or kill him.

> *"We haven't developed a game; we've developed a system..."*

CW: *That kind of leads up to the game's difficulty. How do you think it rates?*

AS: If we were working with Infocom's standards, it would be expert level. It is very difficult. You can score two hundred points fairly easily, especially with the manual's help section. To score three hundred fifty points, you have to be extremely good. Once, when someone was trying out a new adventure writing system, they said "Can someone come up with a puzzle, like getting past the dragon in **The**

Pawn? You know, a puzzle that'll take at least six months to solve!"

Ninety-eight percent of the people who play adventures never finish them; they get the satisfaction of scoring the easy points. A novice adventurer would probably spend thirty hours to score the first two hundred points. If you were a hardened adventurer, it would probably take a couple of hours. There were a couple of guys—really ace adventure players—who finally won. It really annoyed me.

CW: *Did they use the clue book?*

AS: They did, but the clues don't help you on the really difficult puzzles. We give you pointers toward the puzzle's solution, but I don't think it's fair to spell it out in black and white, because the clue book should really be for the novice. The whole fun of playing an adventure is thinking about it and working it out in your mind.

> *"The ST is an extraordinary machine."*

CW: *What did you use to develop the game?*

AS: We used several Apple IIs, partly because it was the only thing we could afford, but primarily because all of us in the office were Apple II owners at some point in our lives. We knew that if the disk dropped bits we could hack into it and get the data back, and we knew the Apple II inside and out. It was important to use a computer we knew. We then upgraded ourselves, about a year ago, to STs and we just purchased a microVax.

The game is written in our own language, which is semi-interpreted, semi-compiled. And, obviously, we've written all the utility tools we need to go with that.

CW: *What was it like to move to the ST?*

AS: From the Apple, it was wonderful. It used to take us an hour and a half to actually edit and compile the source. On the ST, it takes about fifteen minutes. The ST is an extraordinary machine. I've been working on the Amiga for the last three or four months—and I've grown to like it in its own bizarre fashion—but if you want to sit down and do something with the ST, especially graphics, all you have to do is find out where the screen starts, and there it is. With the Amiga, you have to write display lists for the coprocessor; you've got to try to explain to the machine that you want this fantastic graphic, and then it won't let you do it.

For instance, take up and down scrolling on the Amiga. The Amiga operating system calls can only cope with windows scrolling up from the bottom. They won't do it from the top down. So we had to write our own system. That's really ridiculous. Windows shouldn't work only one way just because the Amiga operating system works that way. We don't all want to use the operating system. That's the nice thing about the ST: it doesn't have any front end to it, really. It's got GEM—but we can live without GEM.

CW: *One last question. What's it like to be a woman programming in such a male dominated environment?*

AS: It's actually wonderful. I mean, all my programmers are men, and they respect my judgment just as I respect theirs. I think all, bar one, are better programmers than I am. I know it and they know it. I'm very willing to ask questions.

It's extraordinary to be a woman, because it opens so many avenues. If I go to a party and there's a group of men I need to do business with, I can just walk straight up and talk to them. There's no barrier for a woman. If you're a man, there has to be an exchange of business cards and all that.

I love it from that point of view; I wouldn't want to be a man.

A woman in a male environment gets more attention. If I were running a knitting business, it wouldn't be the same. A man in a knitting business would be as much a celebrity as I am in the computer business. I've never encountered any chauvinism or anybody who's turned my product down because I'm a woman.

And I've never encountered anyone who's underrated me. //

INDEX TO ADVERTISERS

ATARI USERS' GROUPS

Santa Clarita Valley Atari Computer Enthusiasts (SCVACE)
19449 Nadal, Canyon Country, CA 91351
Meetings; newsletter. President: Mark Ostrove.

Pinellas Atari Computer Enthusiasts (PACE)
5290 70th Ave., No., Pinellas Park, FL 33565
Meetings; newsletter: *Pacesetter*.
President: Gust Bangas.

Blackhawk Atari Computer Enthusiasts (BACE)
220 Sherman Ave., Waterloo, IA 50703
Meetings; newsletter. President: Allan Freeman.

Suburban Chicago Atarians (SCAT)
P.O. Box 72266, Roselle, IL 60172
Meetings; BBS; newsletter. President: Bob Dillon.

Atari Exchange of Louisville (AEL)
P.O. Box 34183, Louisville, KY 40232
Meetings; BBS; newsletter. President: Mark Stroud.

Serious Atari Interests of Louisiana (SAIL)
407 Marmande St., Houma, LA 70363
Meetings; BBS; newsletter. President: Ed Spoon.

New Orleans Atari Users Group (NOAUG)
P.O. Box 73236, Metairie, LA 70033
Meetings; BBS; newsletter. President: Dave Porter.

Baltimore Atari Computer Enthusiasts (BACE)
6502 Smokehouse Ct., Columbia, MD 21045
Meetings; newsletter: *BASE Station*.
President: Paul Freeman.

Coastal Area Atari Users' Group (CAAUG)
P.O. Box 5098, Biloxi, MS 39534-0098
Meetings; newsletter: *FOCUS*
President: David Warner.

Charlotte Atari Users' Group (CAUG)
5911 Brookhaven Rd., Charlotte, NC 28210
Meetings; newsletter. President: Lex Thomas.

Minot Atari Computer Users' Group (MACUG)
P.O. Box 3034, Minot, ND 58702-3034
Meetings; newletter.

Miami Valley Atari Computer Enthusiasts (MVACE)
P.O. Box 24221, Huber Heights, OH 45424
Meetings; newsletter. President: Stan Mahurin.

Central Wisconsin Atari Users Group (CWAUG)
217 S. Schmidt, Marshfield, WI 54449
Meetings; BBS; newsletter. President: Tom Ptak.

Bitburg In Germany Atari Computer Enthusiasts (BIG ACE)
P.O. Box 4735, APO NY 09132-5375
Meetings; newsletter. President: David Hinds.

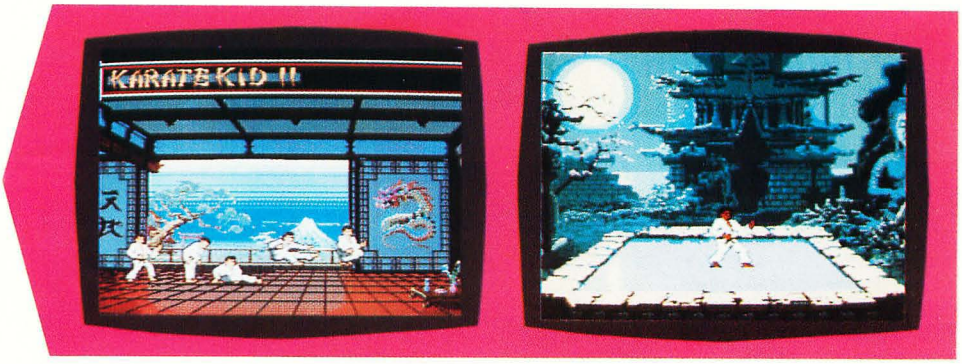
ATTENTION USERS' GROUPS

If you would like your organization to be listed here, send information (and newsletter, if appropriate) to **ST-Log** Group Listing, P.O. Box 23, Worcester, MA 01603. **ST-Log** is not responsible for errors.

STIMULATION
FROM THE WORLD'S LARGEST ST PUBLISHER

STupendous!

Karate Kid Part II
Guide Daniel through fight after fight against ever more powerful advesaries. Then face the evil Chozen himself in the Castle of King Shohashi, where you must discover the secret of the drum - or die! But the fighting isn't all: Catch flies with chopsticks and break ice with your bare hand in full-color, animated graphic sequences. Daniel LaRusso faced the challenge and became a hero. What will you do?
Color only. $39.95

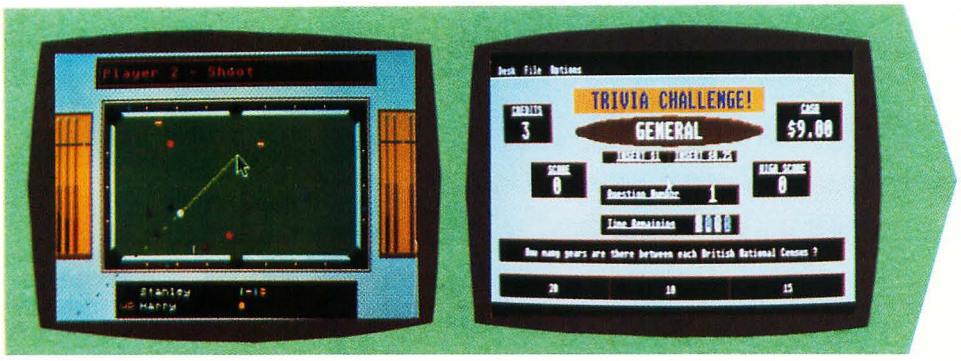


STaggering!

Shuttle II
It's the 1990's and this new Shuttle can take off and fly with the minimum of controls. Decide the launch and landing details then take the seat of Flight Commander for liftoff. Find a fualty Satelite and go out with your jet pack and haul it in. Then return to Earth. Can you keep from burning up? Can you land without crashing? Find out now in this fantastic simulation!
Color only. $39.95

STunning

Pinball Factory
Play great pinball and even design your own screens. Place tabs, bumpers, and ball traps to get the action just right, then use the powerful drawing system to put in the walls. Add flair with a graphic logo. You can even set the scoring, gravity, speed and elasticity! Play the best pinball game you can imagine! .There are even tilt controls to let you "bump" the machine!
Color only. $39.95



STriking!

Eight Ball
The poolhall favorite is here! Realistic graphics and ingeniously simple mouse control make this one a sure bet.
Color or Monochrome. $29.95

Trivia Challenge
Nearly 4000 questions on International Sport, The Arts, Pop Music, Science and General knowledge. Arcade style play keeps the game quick. You can even put in your own questions!
Color or Monochrome. $39.95

Dealer inquiries welcome • Visa and Mastercard accepted • Add $3.00 shipping and handling to each order.

MichTron

576 S. TELEGRAPH, PONTIAC, MI 48053
ORDERS AND INFORMATION (313) 334-5700

CIRCLE #136 ON READER SERVICE CARD

THE STARS

THUNDER *New!*

The writer's assistant

"extremely fast and flexible . . . destined to make a lot of noise!" — COMPUTES ST

3 PROGRAMS IN ONE!

① SPELLING CHECKER 50,000 words
- Real-time mode: program alerts you when you've made a mistake
- Stand-alone mode: have Thunder! check the text after you've finished writing
- Select the correct spelling from a list of similar words — Thunder makes the change instantly

② ABBREVIATION EXPANDER saves you thousands of repetitive keystrokes
- Simply enter a pre-set abbreviation and Thunder! expands it to full form

③ DOCUMENT ANALYZER gives you a statistical report including number of words, average sentence length, FOG score and much more!

ONLY $49.95 * NOT COPY PROTECTED

TIMELINK

Electronic calendar/diary system

"you'll find Timelink indispensable!" — ST WORLD

Your essential tool for planning and record keeping!
- Windows for each day, week, month and year — put any combination on the screen simultaneously
- Instant and automatic updating between frames
- Built-in database capabilities let you keep track of expenses, billable hours and other timely data
- Add/Subtract, Time Between Dates and other features help plan ahead

ONLY $49.95 * NOT COPY PROTECTED

DEGAS

Design and entertainment graphic arts system

"If you want the best graphic arts package for the Atari ST, get DEGAS!" — ANTIC MAGAZINE
- total flexibility: create and store your own custom brush patterns, colors, fill patterns and more
- powerful TEXT capability lets you combine words and pictures
- choose the ideal letter size and weight from the built-in DEGAS TEXT FONT — or design your own font
- draw an accurate LINE, CIRCLE, DISK, BOX or FRAME by simply setting two points — DEGAS does the rest automatically
- MAGNIFY any picture or portion
- MOVE or COPY any figure, anywhere on the screen
- advanced graphic design tools include MIRROR, SHADOW, and AIRBRUSH
- works in all three Atari ST resolution modes

ONLY $39.95 * NOT COPY PROTECTED

COMING SOON • PAPERCLIP ELITE!

ISGUR PORTFOLIO SYSTEM *New!*

Investment manager

"It is dangerous to look at any software package and say, 'This is the one', but this one comes very close." — COMPUTERIZED INVESTING

Helps you manage any combination of investment portfolios more efficiently — stocks, bonds, options, commodities, mutual funds. For casual investors and professionals.
- Designed by Lee Isgur, top Wall Street analyst and V.P. of PaineWebber NY.
- Program updates your portfolios automatically with the latest market data — pre-set for all the popular on-line financial services
- Powerful features show you the most profitable decisions: "Suggest", "Raise Money", "Gains/Losses" and many more
- Analyze, compile and display your investment data in new ways, to gain new insights

ONLY $199.95 * NOT COPY PROTECTED

I*S TALK *New!*

Powerful, easy telecommunications

". . . complete telecommunications that offers you three levels of user definable macros and a 50,000 word memory-resident spelling checker." — PERSONAL COMPUTING

Takes the hassle and uncertainty out of telecommunicating
- Multitude of exceptional features are clearly organized on the GEM user interface
- Call up a HELP window for every major function
- Totally flexible data routing to any combination of devices — disk to modem, modem to memory, screen to modem . . .
- 50,000-word memory-resident, real-time spelling checker to ensure error-free outgoing messages

ONLY $49.95 * NOT COPY PROTECTED

DEGAS ELITE *New!*

For those who want advanced features

The professional ST graphic package. All the powerful design features that made DEGAS a critical success. Plus, fantastic new capabilities.

"Degas Elite is state of the art software!" — ANALOG

ADVANCED FEATURES:
- Up to 8 MULTIPLE WORK SCREENS let you work on several pictures simultaneously
- CUT and PASTE between pictures in different screens
- SCALE, FLIP and ROTATE whole pictures or portions
- Incredible new advanced features including; color animation, block move, text size and font style, color fill patterns, new drawing capabilities, stretch, rotate and much, much more!

ONLY $79.95 * NOT COPY PROTECTED

BATTERIES INCLUDED

BATTERIES INCLUDED, an ITM company, 30 Mural Street, Richmond Hill, Ontario, Canada, L4B 1B5 (416) 881-9941, Customer Information (416) 881-9816

If you can't find this product at your local retailer, you may order it direct from us at the full suggested list price plus $5.00 for postage and handling. For product orders please call 1-800-387-5707 (U.S. only). With all Batteries Included products you can always have the latest version of your program by returning the original disk and $10.00. Write to us for our full color catalog of products for the APPLE, APPLE MACINTOSH, ATARI, ATARI ST, COMMODORE, COMMODORE AMIGA, AND IBM SYSTEMS.

*All prices shown in U.S. funds. Retailers may sell for less.

If you can't find this product at your local retailer call our toll-free number to order: *1-800-387-5707*