ST-LOG

THE ATARI ST MONTHLY MAGAZINE

U.S.A. $3.50
CANADA $4.75

APRIL 1988

ISSUE 18

Tom Hudson Takes a Close Look at the MEGA ST

The Atari ST Buyer's Guide

Mouse-Ka-Mania!

Atari 8-bit Emulator-Part 2

0 73361 64736 8

NOW PLAYING AT A SOFTWARE DEALER NEAR YOU

Available for Amiga, Commodore 64, IBM PC, Apple IIgs, and Atari ST, which are trademarks respectively of Commodore-Amiga, Commodore Electronics, Ltd., International Business Machines, Apple Computer Inc., and Atari Inc.

Cinemaware Corporation, 4165 Thousand Oaks Blvd., Westlake Village, CA 91362

CIRCLE #101 ON READER SERVICE CARD

ST-LOG
THE ATARI ST MONTHLY MAGAZINE

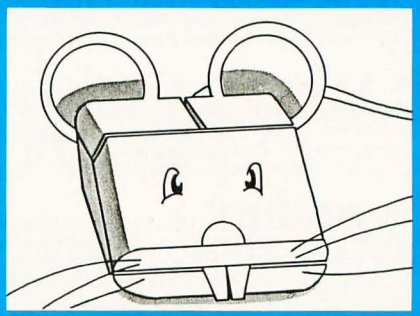
ISSUE 18
APRIL 1988

FEATURES

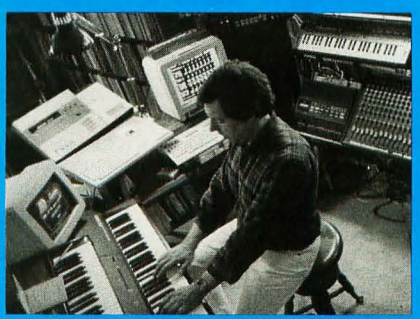
REVIEWS

COLUMNS

12



21



54

ST-LOG (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. U.S. subscription $28 for one year (12 issues), $52 for two years (24 issues, $76 for three years (36 issues). Foreign subscriptions: add $7 per subscription per year. Single copy $3.50 (add $1 for postage). Change of address: six weeks advance notice, and both old and new addresses are needed. POSTMASTER, send change of address to: **ST-LOG** Magazine, P.O. Box 16928, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

REVOLUTIONARY NEW PRODUCT

SWITCH BACK

REQUIRES at least 1 meg. of RAM
(or a MEGARAM or Polydisk Cartridge)

- Imagine Saving almost any game at any point, then being able to return there as many times as you like.
- Imagine the Ultimate Back-up Utility that actually UNPROTECTS programs as it copies them. Lets protected programs be stored as files, run from a hard disk or even be transmitted over a modem.
- Imagine saving three or more protected single sided disks on just one double sided disk.
- Imagine Instantly switching back and forth between two different programs, games, utilities or business applications.

Now Stop Imagining and get Switch/Back.
It can do all this and more.

Switch/Back is a revolutionary new hardware and software package that lets you get more from your ST, MUCH MORE.

Switch/Backs gaming features lets you instantly save most games then continue playing. If you get in trouble you can switch back to where you were as many times as you like.

BACK-UPS –Switch/Back can work with your favorite back-up program and allow you to save whole protected disks to files for archival purposes. It can also automatically unprotect a program and save it as standard file. This method works on hundreds of ST programs and it allows you to run the files directly. Its perfect for running protected programs off a hard disk. It creates standard TOS files, that can be stored together on disks or even transferred by modem.

SWAP – Switch back lets you load just about any two programs into your ST and switch instantly between them. It works with games, business programs, utilities, compilers, etc. Although only one program is running at a time, the other is available instantly, right where you left off.

The Switch/Back hardware plugs into your printer port for easy use (It has a pass through connection for your printer too.)

Switch/Back requires at least One Meg of memory (Or a Polydisk or Megadisk) **ONLY $69.95**

ST Protection Techniques

Finally ST Copy protection techniques are revealed. This complete book and disk package details the state of the art in ST Protection methods and much, much more.

The Software included with the book provides many powerful features like the AUTOMATIC PROGRAM PROTECTOR. This easy to use Utility allows you to protect just about any ST program. You can choose a combination of protection methods like encryption, checking custom disk formats, password protection or a limited use option that makes the program self-destruct after running a preset number of times.

The book includes topics such as Phreaking, Logic Bombs, Hardware data keys, the legal aspects of piracy and software protection, Custom disk formats, Pirate Bulletin boards and much more.

In addition it contains reviews of the popular ST back-up programs and detailed explanations of ST disks and drives.

ST Protection Techniques (Book and disk package) **Only $39.95**

The worlds most inexpensive clock cartridge. Finally its affordable to keep your time and date accurate. 3 year battery included. **ONLY $24.95**

WATCHCART ST

MEGADISK Ultra high speed solid state disk drive • 500% Faster than a Hard Disk • Provides almost instant booting • Like a RAM disk that's always loaded with your favorite programs and ready to use • One megabyte of Solid State storage • Built in battery back-up in case of power failures

MEGADISK is actually one megabyte of RAM that simply plugs into your cartridge port. It acts as an added disk drive that's ultra fast and always ready for use. Like a Hard disk, MEGADISK won't loose its memory when your computer is turned off. It comes with its own power supply and battery back-up system so its independent of your computer.

Megadisk can be configured according to your needs. • Set it up as one large disk • An 800K double sided disk and a 200K hardware print buffer • Or as two 400K single sided disks and a print buffer

Megadisk will work fine with your current system whether you have a hard disk and two drives or you're just getting started.

Megadisk is perfect for those who want the high speed of a hard disk for a lower price. Its even better for power users or software developers who may already own a hard disk and two drives but want extra speed and power. Megadisk can also emulate other cartridges for testing and back-up. In addition Megadisk can be used with Switch/Back to allow you to instantly jump between two full size one meg applications. **$299.95**

Megadisk Clock Option – Adds a Clock/calendar card to your Megadisk cartridge. Contains replaceable Three year battery 29.95

Polydisk Polydisk is a 512K version of a Megadisk. Polydisk gives you the same fast boot features, the high speed access, and the print spooler. Polydisk has a power supply (like Megadisk) but does not contain a battery back-up.

Note: Those with only 512K of main memory can use Switch/Back with a Polydisk, just like those with one Meg.

Polydisk (512K Solid state drive) **Only $199.95**
(Clock option card is also available for Polydisk $29.95)

COLOR COMPUTEREYES™

Incredible COLOR video digitizer. • The first and only full color digitizer for the ST • Uses standard video inputs like video camera, VCR, or video disk. • Works in all ST resolutions, Low res provides 16 shade black and white or full color pictures. • Pictures can be used with Degas, Neochrome, Powerprint and others. • Automatic calibration of contrast, brightness and white balance. • Plugs into cartridge port for easy set-up. • Capture your picture or that of your favorite star. **ONLY $199.95**
SPECIAL OFFER – Buy both Computereyes and Powerprint and SAVE 20.00 from the total.

PowerPrint ST **BLOW YOURSELF UP**

Imagine your picture on a 6 foot poster. Create a business graph that can cover a wall. Quality output for posters, t-shirts, news letters, and more.
POWERPRINT

Whether it's a photo digitized with ComputerEyes, a masterpiece created with Degas, or the winning screen from your favorite game, POWERPRINT can print it with unequaled clarity and resolution. PowerPrint supports **ALL** ST resolutions. It prints multiple sizes up to **GIANT WALL SIZED POSTERS.** Print 16 shades for incredible detail. Print the whole screen or **ZOOM** in on just the part you want. POWERPRINT offers unique effects, including rotate, mirror and inverse options. Selective shading option allows you to print multi-color pictures on any printer by printing one color at a time (using color ribbons). Powerprint lets you capture and print almost any ST screen. Works with Star, NEC, Citoh, Gemini, EPSON, XM8048 and compatible printers. **ONLY $39.95**

Digisound ST High Quality sound digitizer for the ST. This powerful hardware and software package lets you sample real world sounds and play them back on any Atari ST. Add special effects like Echo, Reverse, looping, pitch manipulation, mixing and envelope control. Turns your Atari keyboard into a musical instrument to play songs with your digitized sounds (also works with any MIDI keyboard). Digisound makes it simple to add sound to your own program, too! Unleash the incredible sounds in your ST with DIGISOUND. Supports sampling from 5 to 40Khz, DIGISOUND is the choice of the professionals. DIGISOUND was used to create the voice in Chessmaster 2000, and other commercial programs.
DIGISOUND **ONLY $89.95**

DIGISOUND PROFESSIONAL

All the excellent features of DIGISOUND plus these great extras

LOGARITHMIC SAMPLING – Special hardware extends the sound quality far above the other ST sound digitizers. Logarithmic sampling and playback (external amplifiers only) greatly extends the dynamic range while reducing distortion and noise.

Internal Real Time Mixing – Input from a stereo and a microphone so you can sing over a tape. **$149.95**

DIGIPLAYER The High powered digisound software can now be obtained by those who already own a digitizer for the ST. Compatible all cartridge based digitizers. Extend the power of your digitizer with Digiplayer.
Only $49.95

ALPHA SYSTEMS

24 HOUR HOTLINE – VISA & MasterCard Welcome

216-374-7469

Customer Service line (216) 467-5665. Call or write for free catalog.

Order by phone or send check or money order to:
ALPHA SYSTEMS 1012 Skyland, Macedonia, OH 44056
Include $3.00 shp. & hdlg. (US & Canada). Ohio residents add 5½% sales tax. Foreign orders add $8.00

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.

Name _____

Street Address _____

City _____ State ____ Zip ____

ATTACH LABEL HERE

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

P.O. BOX 16928, N. HOLLYWOOD, CA 91615

ST-Log Staff

Editors/Publishers: Michael J. DesChenes, Lee H. Pappas. *Managing Editor:* Kathy Wesner. *Technical Editors:* Charles Bachand, Clayton Walnum. *East Coast Editor:* Arthur Leyenberger. *Midwest Editor:* Mathew J.W. Ratcliff. *West Coast Editor:* Charles F. Johnson. *Contributing Editors:* Ian Chadwick, Arnie Katz, Bill Kunkel, Maurice Molyneaux, Steve Panak, D.F. Scott, Douglas Weir. *Graphics:* Sherry Morin. *Typography:* Edythe Stoddard. *Advertising Manager:* J.E. Publishers Representatives. *Accounting/Circulation:* Robin Levitsky. *Contributors:* Frank Cohen, Andy Eddy, Tom Hudson, Darek Mihocka, Dan Moore, Daniel A. Silvestri, David Small, Roderick W. Smith, Scott Wasser. *Production:* Donna Hahner, *Vice-President, Production.*

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

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** P.O. Box 16928, North Hollywood, CA 91615. Or Call (818) 760-8983

Correspondence concerning a regular column should be sent to our editorial address, with the name of the column included in the address. We cannot reply to all letters in these pages, so if you would like an answer, please enclose a self-addressed, stamped envelope.

Advertising Sales

J.E. Publishers Representatives — Los Angeles: (213) 467-2266.
San Francisco: (415) 864-3252. Chicago: (312) 445-2489.
Denver: (303) 595-4331. New York: (212) 724-7767.
6855 Santa Monica Blvd , Suite 200, Los Angeles, CA 90038.

Address all advertising materials to: Janice Rosenblum — Advertising Production, **ST-LOG**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

Permissions

Subscriptions

ST-LOG, P.O. Box 16928, North Hollywood, CA 91615; or call (818) 760-8983. Payable in U.S. funds only. U.S.: $28.00-1 year; $52.00-2 years; $76.00-3 years. Foreign: add $7.00 per year per subscription. 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

Hey, long time, no see.

That's what you're thinking, right?

It *has* been a long time, no doubt about that—too long for anyone's comfort, least of all for you, our faithful readers. But the wait is over. You can relax now: we come bearing good news.

For those of you who aren't aware of what's been happening, an explanation is owed. **ST-Log** has been in transition for the past several months. The magazine is now under new ownership, and based in Beverly Hills, California, rather than Worcester, Massachusetts, where it has been for the past few years. The increased financial backing provided by the new owners will bring many new and exciting improvements to the magazine, not the least of which are increased distribution and better service. In fact, if you previously had trouble finding copies of the magazine on your local newsstand, please let us know where you live, so we can remedy the situation.

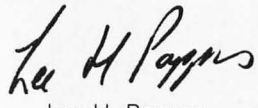
One thing that will *not* change, however, is the editorial content of the magazine. We'll still be providing you with the best programs for your machines, as well as up-to-the-minute news and reviews. You'll continue to see familiar names such as Charles Johnson, Art Leyenberger, Clayton Walnum, Steve Panak, Ian Chadwick, Matthew Ratcliff, Andy Eddy and Maurice Molyneaux, as well as our other contributors.

Most of the changes to the magazine will be artistic in direction. For instance, I'm sure all of you have noticed that the magazine is now printed on a slicker paper. We hope, in the months to come, to improve even more upon the magazine's design by adding more color and incorporating more creative layouts. Over the past few years, we've provided you with the classiest magazine possible; now we want to provide you with the classiest *looking* magazine as well.

Those of you who have experienced subscription problems will be delighted to know that a new subscription fulfillment service has been selected for the magazine. All the subscription mix-ups that plagued us (and you) in the past will soon be corrected—just another example of the improved service you will experience under the new owners.

Your patience over the past few months has been more greatly appreciated than you will ever know, and we at **ST-Log** are looking forward to many more years of offering you the kind of Atari coverage you've come to expect from us.

Yes, it's been a long wait. But the new **ST-Log** is what we've *all* been waiting for.



Lee H. Pappas
Publisher
ST-Log

ST-Log's new customer service number is (818) 760-8983. You may also write to:

ST-Log
P.O. Box 16928
North Hollywood, CA 91615

ANALOG Computing
P.O. Box 16927
North Hollywood, CA 91615

The above phone number and addresses are for subscription matters only.

FAST JUST GOT FASTER...

AND EASIER.

LDW BASIC is a powerful, integrated programming tool that lets you edit, compile, and execute your BASIC programs without exiting to the desktop. It also creates a program file which can be executed directly from the desktop. You will never have to load the BASIC language interpreter or run-time module to execute your program.

LDW BASIC Rev. 2.0 is very easy to use, yet it's more powerful than any other development system for the Atari ST. It supports a full set of high level GEM access statements and functions which let you:

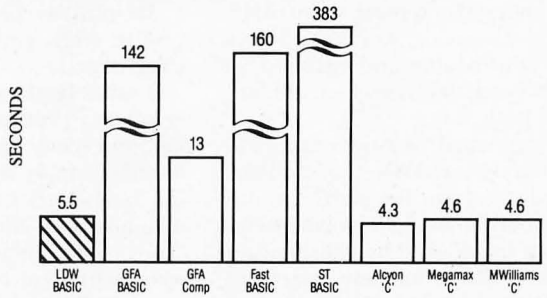
- Create and use your own windows, menus, dialog boxes, buttons, edit fields and check boxes.
- Use desk accessories while running your BASIC program.
- Design your own mouse pointer shapes and icons.
- Trap GEM events.

You get all this and much more. No more PEEKs/POKEs to access GEM. Among the supported statements are: ACTIVEW, ALERT, ASK FILE, ASK MOUSE, BOX, BUTTON, DIALOG, DRAW ICON, EDIT FIELD, EVENT, INFOW, MENU, MOUSE, ON DIALOG, ON MENU, ON MOUSE, REDRAW, TITLEW, WINDOW and about 40 more new statements.

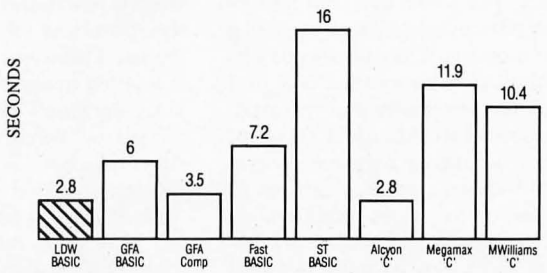
LDW BASIC Rev. 2.0 is a stand-alone development tool, but it can also compile any program written using the old ST BASIC interpreter or the new ST BASIC interpreter. It is also functionally compatible with BASICs for the Macintosh.

For more information or to obtain a listing of benchmarks contact Logical Design Works, Inc., (408) 435-1445
(In Europe) 022/31 97 52

Compare the following benchmarks and see why LDW BASIC Rev. 2.0 leaves the competition behind!



SIEVE 10 ITERATIONS



CALC — BYTE MAGAZINE BENCHMARK MAY '85

ONLY $89.95

Please add shipping & handling costs

U.S.	2.00
Canada	5.00
Outside U.S. and Canada	10.00

Contact your local Dealer or send check or money order to:

Logical Design Works, Inc.
780 Montague Expwy., Suite 403
San Jose, California 95131 U.S.A.

VISA, MasterCard or C.O.D. accepted. California residents please add applicable sales tax.

CIRCLE #103 ON READER SERVICE CARD

Reader comment

So where's my blitter?

Atari has received a lot of flak over the past year or so for not getting their new products (the MEGA, SX212, XEP80, laser printer and PC clone) on the market as quickly as promised. The issue has been discussed in excruciating detail on all of the major BBSs and in the various Atari (and generic) computer magazines. Much of the discussion over these delays has been quite heated and resulted in some rather harsh words and a lot of bad feelings on both sides.

So what happened? Is Atari entirely to blame for the delays? Were they deliberately lying to us from the start? Or, are there contributing factors involved here, factors being overlooked by Atari enthusiasts in their desire for new and more powerful Atari systems? Let's take a quick look at a few facts.

(1) In a world of giants like IBM, Apple (and even Commodore), Atari is still a very small company. They have well under 1,000 employees here in the U.S. and only a handful of developers and research-oriented personnel worldwide. Except for their warehouses and an overseas assembly plant, the *entire* company resides in a not-very-large, two-story complex in Sunnyvale, California. So, while each individual employee may be quite talented and productive, they simply can't match the output of companies twenty to one-hundred times their size. Large numbers of talented personnel and development centers take large amounts of money, which brings us to fact number 2.

(2) While Atari has performed a major miracle in turning itself around and is enjoying continued growth in their quarterly earnings, their total profits (while a *huge* improvement over earlier years) are well behind their competition's. Apple and IBM record their sales figures in the hundreds of millions of dollars each quarter. In contrast, Atari's last quarter showed a $13-million profit. Therefore, Atari's profits, while growing, are still comparatively small.

It takes large amounts of money to develop new products and get them to market, and ready cash is always something in short supply for any company still on the "comeback trail." And make no mistake about it, Atari will remain on that "trail" for a few years yet. In an effort to ease their tight money belt, Atari made a stock offering last year and recently had a release of bonds in Europe. They've also started to expand their market base with the purchase of the Federated line of stores. There are also rumors that Atari is looking to expand their product line beyond the small PC market, though details are scarce. Personally, I'm impressed that Atari has done as well as it has, considering its financial position.

(3) Atari has only *one* production center, located in Taiwan. While fully automated and considered the most efficient in the industry, there's only so much a single factory can produce. New products means retooling and redesigning the assembly line, and this takes time and money. To resolve this roadblock, Atari is currently looking for a location to build a new production center. Rumors place this new plant in Europe (where the majority of ST sales are), Mexico and here in the good old U.S. of A. (my favorite).

Even with a new or expanded facility ready and waiting there can be problems. Once a design has been finalized and the assembly line is ready to roll, it takes only one tiny little item to royally "gum up the works." Stop and think about it. What is a computer? It's a case, power supply, motherboard, keyboard and operating system—plus one more little item, without which it's just another pile of parts. In fact, there can be a whole bunch of the little buggers, which brings us to problem number 4.

(4) This particular problem has probably caused Atari more headaches than anything else over the past year, and it's a problem that few of the other major computer manufacturers have to put up with. Unlike Apple, IBM or Commodore, Atari is 100 percent dependent on outside sources for *all* their computer chips, both standard and custom. For standard "production" chips (such as the 68000, 6502 and the various RAM chips), this is rarely a problem. These chips are widely used and readily available at affordable prices.

But it's a very different story when you get to the various "custom" chips that Atari designed for the MEGA and other new products. If you've just designed a new "super chip" and you're an Apple, IBM or Commodore staff member, you simply send the designs over to your fabrication plant and tell them, "I want it in production by the end of the month, or

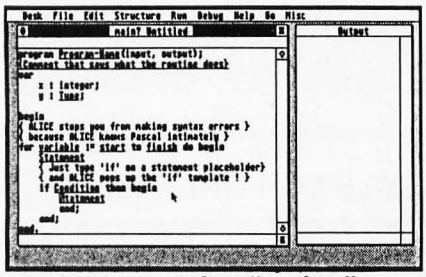
(continued on page 9)

We were going to tell you all about

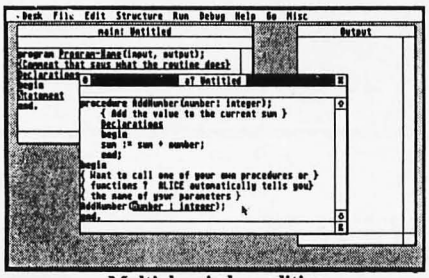
ALICE

The Personal Pascal™

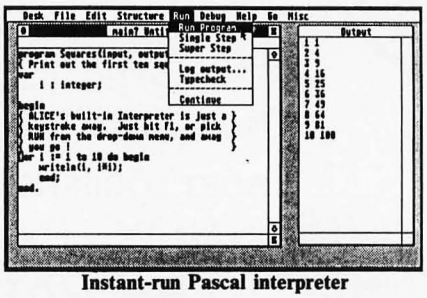
but we'd rather show you . . .



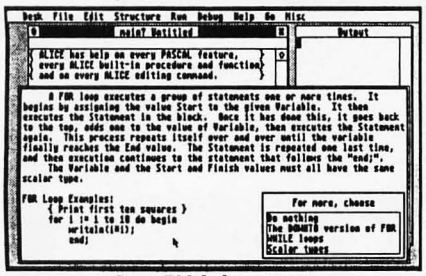
You program from "templates"



Multiple window editing



Instant-run Pascal interpreter



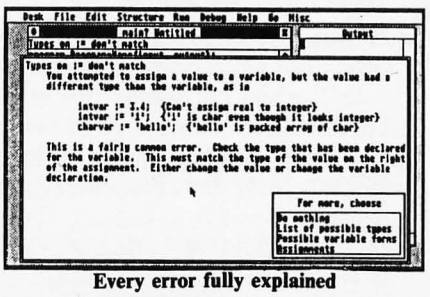
Over 700 help screens

Programming Made Easy

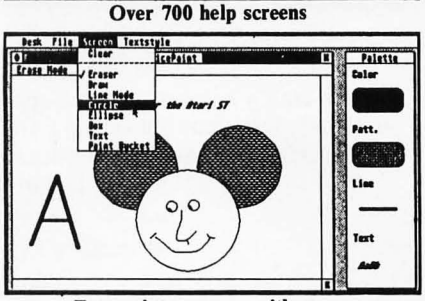
If you're into programming, or if you *want* to be, you won't find a better system for your Atari ST than ALICE: The Personal Pascal. Already popular on the IBM-PC, ALICE now brings easy, interactive programming to the Atari ST.

ALICE is, without question, the best way around to learn about computers and programming. Beginners can even order our ALICE based textbook for only $19.95 with ALICE.

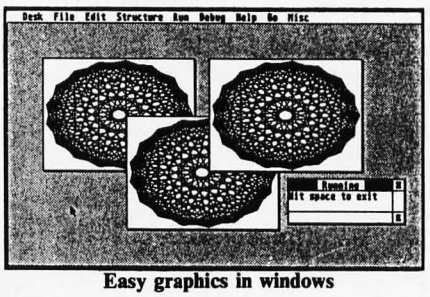
Even if you're an expert, ALICE makes it easier to write, test and especially *debug* programs. ALICE's extensive Pascal improvements include a GEM interface even beginners can use and most of the extensions of Turbo Pascal. The surrounding examples just give a glimpse at how easy it is to put programs together using ALICE.



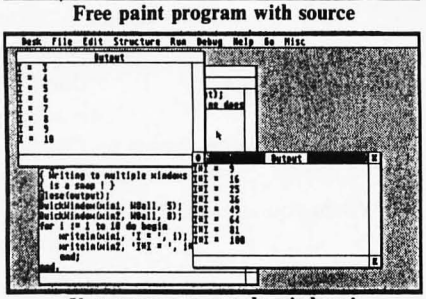
Every error fully explained



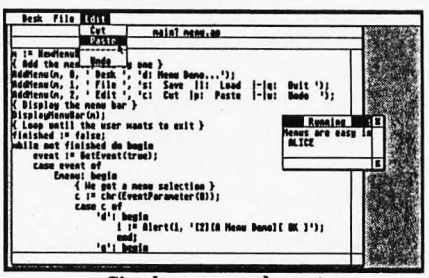
Free paint program with source



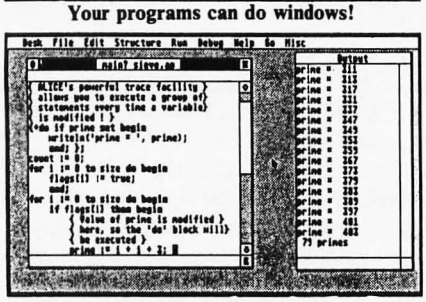
Easy graphics in windows



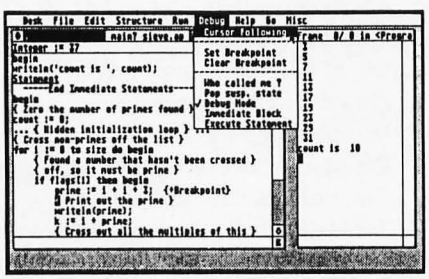
Your programs can do windows!



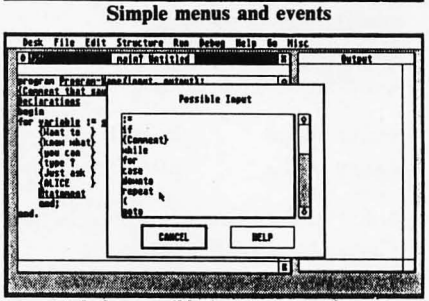
Simple menus and events



Powerful variable trace



Breakpoints, Single Step



Ask for possible input at any time

"If I needed to learn Pascal all over again, or were going to teach a course in the language, I can't imagine using any program other than ALICE."

— Adam Green, **Infoworld**

"ALICE may be the most advanced programming environment currently available for the PC."

— Michael Covington, **PC World**

"If you enjoy programming languages, this comes pretty darn close to being as much fun as a video game."

— Gene Wilburn, **Computing Canada**

Visit your dealer or send us $79.95 (USD) plus $5 for shipping and handling. Major credit card orders may phone collect. Dealer inquiries invited. IBM-PC version $95.

Looking Glass Software

Looking Glass Software Limited
124 King St. N. Waterloo, Ontario
N2J 2X8
519/884-7473

ATARI ST

PUBLISHING PARTNER™

CREATES LIKE A PUBLISHING COMPANY WITHOUT THE OVERHEAD

SOFT LOGIK NEWS

AVAILABLE FOR ONLY $149.95

PUBLISHING PARTNER ™ HELPS YOU CREATE!

You'll benefit by using your Atari ST to create professional quality journals, newsletters, ads, business cards, certificates, letterheads, logos,art designs, bar graphs, flow charts, *even bumper stickers* and all the forms you or anyone would ever need. Create just like a professional publishing company without the overhead! Publishing Partner is actually three progams in one--Word Processor, Page Layout, and Forms Creator. Expand your potential in becoming a better writer, artist and designer with your Publishing Partner.

WORD PROCESSING

What you see is what you get!
Combine text and graphics easily and quickly from existing or newly created documents. Position entire paragraphs or individual words exactly where you want them. Create one, or multiple color separations ready for printing.
Production time will never be the same--it will be much shorter!!
Just take a look at some of its features!

* GEM based
* Justifies right or left as you type
* Edit Multiple Columns on One Screen
* Search and Replace
* User Definable Page Size
* **Bold**, underline, ^{super} and $_{subscript}$
* *Italicize*, **shadow**, **outline**
* **Reverse Image**
* Backslant Characters
* Mirror Image / Invert Image
* Auto Headers/Footers, Page Numbers
* Easily Move Text
* Reads & Merges other files together
* Sets Tabs
* Macros
* Vertical & Horizontal Printing

PAGE LAYOUT

Whatever you require--cutting artwork from other programs, custom logos, unique borders, unusual mastheads, digitized photos--Publishing Partner is your solution. After all, it was specially designed for you--the home and/or office Atari ST user--by the pro's who realize that there's more to your computer than just typing letters.

* Vert. and Horz. rules--Exact Alignment
* Auto Text flow for columns/pages
* Layout Multiple Columns of text
* Change columns on finished page
* Display entire page, 50%, or 25%
* Easily Re-position Text and Graphics
* Alternating Headers/Footers
* Mix Type, Fonts, and Sizes *anywhere*
* Adjust Sizes from 2pts to 144 pts (2")
* Adjust line spacing (leading) by points
* Import other program's graphics
* Rotate Graphic Images
* Multiple Patterns, Shades and Colors
* Enlarge & Reduce Graphics/Exact fit
* Cut, Paste & Crop Graphics
* Tool box function/Unlimited patterns
* Boxes, Circles, Arcs, Polygons, etc.
* Insert lines directly on the page
* Adjust Character Spacing
* Use Hairlines to separate columns

FORMS CREATOR

You can create a variety of forms quickly and easily with your Publishing Partner. For example,you can create your own:

Letterhead
Invoices
Purchase Orders
Labels
Bumper Stickers
Business Cards
Certificates
General Ledger
Shipping and Receiving
Routing Slips
"While you were out" Phone messages
Templates
Price Estimate forms
Requistion forms
Shipping Logs and much more!

Publishing Partner supports most dot matrix printers, including the Epson ™ Star ™ and Okidata ™ Printers. Also supported are any Postscript output devices such as the Apple Laserwriter ™ New print drivers are constantly being released, *so please call to make sure your printer is supported.*

FOR MORE INFORMATION OR TO PLACE AN ORDER, CALL (314)894-8608.
DEALER INQUIRES INVITED.

CIRCLE #105 ON READER SERVICE CARD

Soft Logik Corp.™

4129 OLD BAUMGARTNER * ST. LOUIS, MO. 63129 * CALL (314) 894-8608

 continued from page 6

you're fired!" In other words, you have a *big* say in how fast and hard that chip gets worked on.

In Atari's case, all they can do is deliver the design to an independent company and timidly ask, "Can you *please* try and get it done soon?" Sometimes this works out if the chip design is a simple one and the fabricator isn't too busy at the time. Unfortunately, this is rarely the case. Production of complex ICs (such as AMY and the Blitter) is often plagued by problems and all too often takes a back seat to other projects the fabricator feels are more important or profitable. And, since there's a limited number of independent fabricators around today, it's not just a simple matter of changing fabricators whenever a delay arrives. Add in the additional delays of having to start all over again with the new company, and you only compound the problem.

This lack of control over the fabrication and production of custom ICs has proven to be the major stumbling block to the timely release of Atari's new product lines. Lacking a fabrication and production center of their own has placed Atari in the embarrassing position of having to publicly apologize for repeated delays and, all too often, higher prices. In an effort to eliminate this problem, Atari is considering two options: buy a controlling interest in a small existing fabricator or build their own fabrication center on the property they own close to the Sunnyvale headquarters.

The final decision is still unknown to those outside of Atari, but I'm sure it will be determined by the amount of available cash for the investment. In either case, we can be pretty sure the resulting facility will be small and as "state of the art" as Atari can afford to make it. The resulting plant will also have to be large enough to meet Atari's current needs, and expandable enough to handle the development work for Atari's expansion plans. Unfortunately, it will be a least a full year before we can expect to see either choice in full production, so we can expect some problems to remain until then. Frustrating I'm afraid, but unavoidable given the present situation.

Well, now we've seen some of the factors involved. So what's the result? Were the delays unavoidable? Is Atari blameless? In my opinion, not totally. Though many of the delays have indeed been beyond Atari's control, others should have been foreseen. Granting that no crystal ball is perfect, Atari was still well aware of the constraints they had to operate under and should have been more conservative when they gave out the release dates on their new products.

There's also Atari's (and Mr. Tramiel's) tendency to announce products not yet off the design board. This tends to raise both hopes and frustrations, with one quickly followed by the other. These, when combined with the inevitable delays associated with new hardware, have caused a serious erosion of Atari's credibility industry-wide. In an effort to combat this erosion, Atari has announced a new "buttoned lip" policy in which *no* new products (or developments) will be announced or discussed until actually ready for production. I hope they can manage

(continued on page 20)

Easter Super Printer Package Sale

Atari ST

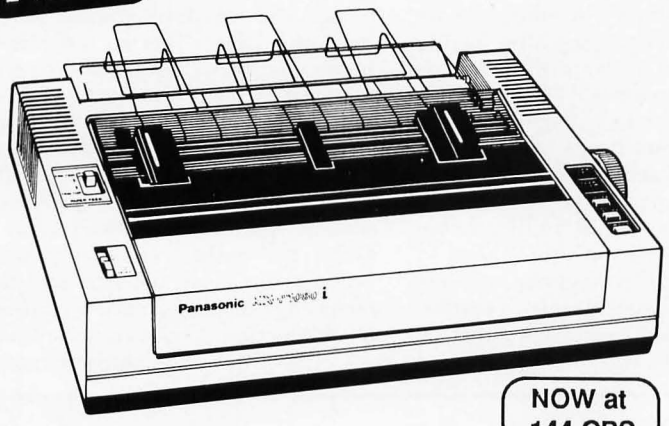
Panasonic Office Automation

8 Bit Atari

&

ComputAbility

NEW



NOW at 144 CPS

Panasonic Office Automation **KX-P1080i-II**

Super Printer Package
***for 8 Bit Atari computers**

with Xetec Graphic AT **$215**

with Supra 1150 interface **$225**

*Package price includes Delivery in continental U.S.A.

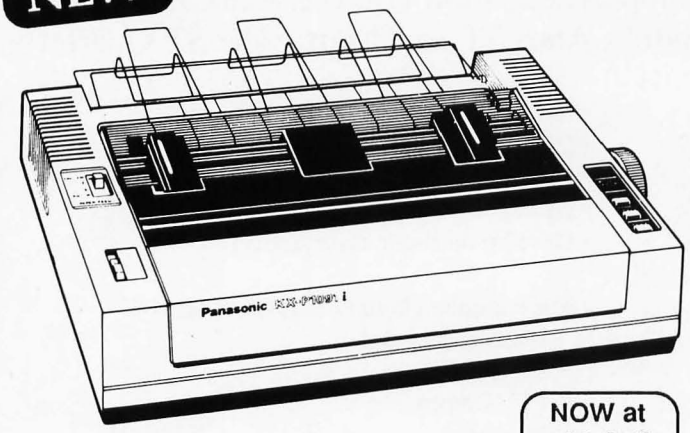
Panasonic Office Automation **KX-P1080i-II**

Super Printer Package
***for All Atari ST computers**

with ST Printer cable **$199**

*Package price includes Delivery in continental U.S.A.

NEW



NOW at 192 CPS

Panasonic Office Automation **KX-P1091i-II**

Super Printer Package
***for 8 Bit Atari computers**

with Supra 1150 interface **$245**

with Xetec Graphic AT **$235**

*Package price includes Delivery in continental U.S.A.

Panasonic Office Automation **KX-P1091i-II**

Super Printer Package
***for All Atari ST computers**

with ST Printer cable **$219**

*Package price includes Delivery in continental U.S.A.

* Price for APO & Non Continental U.S.A. orders. See Special order information in our 2 page spread

Mon-Fri 9am-9pm CST
Sat 11am-5pm

order Call Toll Free
800-558-0003

SINCE 1982
ComputAbility
Consumer Electronics

No surcharge for Mastercard or Visa

Inquiries, or for Wisc. Order
414-357-8181

CIRCLE #107 ON READER SERVICE CARD

The Atari ST

Some wise buys for the smart ST shopper.

by Arthur Leyenberger

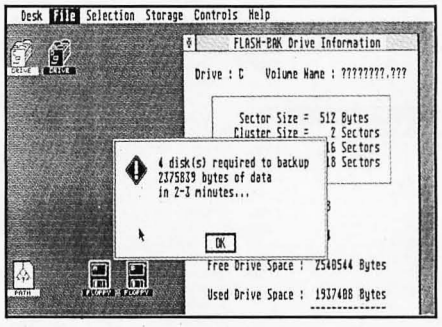
The new year is now upon us and with the new year comes the prospect of exciting developments for Atari ST owners. If you've read the last two episodes of "ST user" you know that each successive year brings a slew of new products for the Atari ST owner. This year, we hope, will be no different from years past. In this Buyer's Guide we will take a look at the bountiful harvest of recently introduced products for the Atari ST, and reexamine the bevy of solid performers from previous years.

As is our policy—stated up front for all to see—we make no claims that this list of products is comprehensive. There are so many good products out there, that we cannot possibly include them all. Further, we may overlook your particular favorite piece of hardware or program. If you think we missed the boat let us know. Sometimes there are products that escape even our ever-watchful eyes. On the other hand, what we do promise is a sincere effort to pick and describe the "best of the best" of ST products. If it's mentioned here, you know it's good. So, with caveats emptied, forewarnings forewarned and word processors at the ready, we present to you the 1988 **ST-Log** Buyer's Guide.

Utility software.

For me, the best backup program for the ST is still **Copy II ST** by Central Point Software. They continue to update the program at least twice a year and the company has the experience of making this kind of product for several types of microcomputers. Copy II ST works with one or two disk drives, either single- or double-sided. It provides a fast sector copier and a bit copy utility for making duplicate disks. The program is completely mouse-driven, so it's easy to use and retails for $40. Copy II ST is intended only for making *archival* copies of disks for your *own* use.

Migraph, the makers of Easy Draw, have a useful utility program called **Label Master Elite**. It's a GEM-based label maker and mailing list program that retails for $40. Labels can be printed with, or without, designs in a variety of formats, including mailing labels, 3½-inch disk labels, and 3x5 and 4x6 index cards. The program comes with 100 graphic designs and the graphics editor lets you create your own designs, cut and paste between designs, or use Printmaster designs.



Flash-Bak.

The mailing list manager allows you to search, sort, delete and modify records, merge databases and create freestyle labels. It also has a 48-character comment field. Other features of Label Master Elite include the ability to work with most printers—including 9- and 24-pin printers—and the power to print directories and disk labels.

There are several useful utility programs for ST users who own hard disk drives. Beckemeyer Development Tools has supported the ST from almost the beginning. Their first ST product was Micro C-Shell, a UNIX-like command line interpreter which not only was a boon to program developers and end users alike, but

Buyer's Guide

also showed us Beckemeyer's ability to create and support quality system software.

The Hard Disk Toolkit from Beckemeyer is a program that provides fast and reliable hard disk backup, plus verify and restore operations. The backup program automatically formats the disk as it copies and the user can choose to backup any file, folder or an entire disk. Hard Disk Toolkit retails for $30.

Another excellent Beckemeyer product is **The Hard Disk Accelerator**, a software package that makes the ST perform faster by speeding up the hard disk access. Common tasks, like starting a program or searching a database, can often increase in speed by 100 to 300 percent. The Hard Disk Accelerator is a "disk-caching" program, which means that it keeps a part of the program that's normally stored on the hard disk in memory. The net result is a reduction in the number of disk accesses. Hence, fewer accesses means faster speed.

The Hard Disk Accelerator works with programs of all sizes, and can be easily configured to use as much or as little RAM as you want. The program is not copy protected and lists for $40.

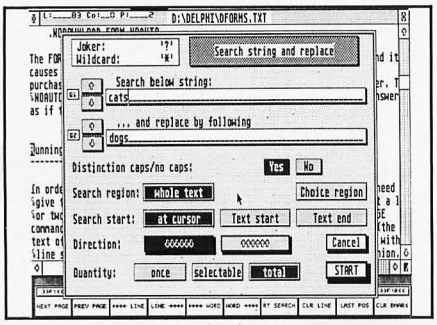
Eidersoft is a British company that recently started marketing ST programs in the U.S. So far, their products are top-notch. One particularly useful pair of products is **Flash-Bak** and **Flash-Cache**. These two programs are packaged together and provide hard disk backup and disk-caching functions.

From my preliminary tests, Flash-Bak is the fastest hard disk backup program for the ST. The program is GEM-based, can use two floppy drives alternately for backup, and also allows the user complete control over all aspects of operation, such as verification, encryption, data compression, etc. Moreover, Flash-Bak is able to selectively backup files based on criteria selected by the user (i.e., date stamp, wild card match, etc.).

Flash-Cache is the RAM cache utility that holds a part of the most recently accessed part of the hard disk in RAM. This has the practical effect of speeding up disk-intensive operations such as program loading/saving, program compilation and data manipulation. Caches can be any size to suit your system. One very important extra function in Flash-Cache is that it allows the "folder heap" (number of folder accesses) to be enlarged, thereby avoiding the infamous Atari TOS "40-folder syndrome." Flash-Bak and Flash-Cache retail for $80 and they are well worth the price for owners of hard disk drives.

Tempus, another product from Eidersoft, is a GEM-based text editor for programmers. It offers just about every feature you would expect in a top-of-the-line product, plus a few more. The most outstanding feature of this product is its speed—which happens to be blindingly fast. Compared to a word processor like 1st Word, Tempus is four to ten times faster in loading files, scrolling the document from top to bottom and searching and replacing on a single character. Compared to a text editor like MicroEMACS, Tempus is up to five times faster for the same operations.

Other features of Tempus include medium- and high-resolution support, mouse or keyboard operation, the ability to have up to four files on-screen at once, auto indentation for structured programming, on-line help menus, twenty programmable function keys, and more. A particularly useful feature for programmers is the editor's ability to automatically create a cross reference list, with line and column



Tempus.

references, to multiple search strings. Once the search string is specified, the table is created very fast, then subsequent clicks on the right mouse button instantly moves the cursor to the next string match in the document. Tempus retails for $50 and is by far the best ST text editor currently available.

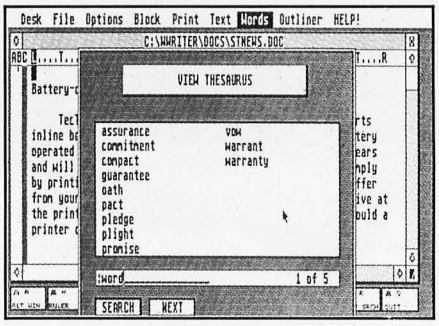
MichTron is another of the handful of companies that have supported the ST from the start. One of their titles is a multiple desk accessory called **Cornerman**.

Cornerman offers the user ten separate functions under one accessory name, including: a complete ASCII reference table, with decimal, hexadecimal, character and mnemonic information for all 256 ASCII codes; a 16-digit calculator containing binary, octal, decimal and hex modes, three summing memories and printing tape display; a notepad with full editing, word wrap and automatic date and time stamp; a phone dialer with autodial capability; a phone log that will automatically transfer information from the dialer; a fifteen-puzzle game; two clocks—one is digital, the other analog; a complete setup module for customizing dialer, RS-232, clock, calculator and window position parameters; a print utility; and a DOS window to be used with MichTron's DOS Shell. All of these features for a list price of just $50.

Another useful utility program is **Write 90** from XLent Software. This program's primary function in life is to print text and VIP spreadsheet files sideways. Write 90 offers five different character sizes, on-screen preview of sideways printed files, the ability to use continuous or single-sheet paper, and supports printers such as Epson, NEC and others. Write 90 lists for $30.

It's difficult to describe the most revolutionary product for the ST in just a couple of paragraphs. Suffice it to say that **PC Ditto** from Avant-Garde Systems allows you to turn your ST into a PC clone. Why in the world would anyone want to do that? If you use a PC at work and have an ST at home, PC Ditto allows you to run hundreds of PC software packages on your ST. No kidding!

First, you need to have an RGB monitor since Ditto does not yet support monochrome. Second, it really helps to have a 5¼-inch external disk drive for your ST (Paradox and IB Systems are two companies that make them), because almost all PC software is sold on this size disk. Aside from these two points, the program works quite well and is fairly simple to use. Of course, the PC programs don't run as fast on the ST as they do on a PC, but for most programs, it really doesn't matter. PC Ditto sells for $90.



Word Writer.

Application software.

There are three ST word processors that are, at this time, worthy of your consideration. One of them is Regent Software's **Regent Word II**. Regent Word II is a full GEM-based implementation of their previous word processing program, Regent Word. Moreover, many new features have been added and the program runs faster. Space does not permit a listing of all of the program's features, but they include a built-in spelling checker, a multiple margin command that lets you easily create indents and outdents, word count, a foolproof and easy to use block command, and the ability to design and use up to nine fonts.

Regent Word II has some additional features that make life in the "write lane" easy, including automatic reformatting, on-screen display of line and page numbers, and a brief—but thorough—manual. Regent Word II sells for $80.

Another excellent word processing program for the ST is **Word Writer** by Timeworks. By the looks of the packaging, one unfamiliar with the software might judge this to be a top drawer program. Indeed it is. The quality is apparent in the operation of the program, as well as in the documentation accompanying it. Word Writer is a full-featured, GEM-based word processor that includes an integrated outline processor, three spelling checkers, extensive help screens and a built-in print spooler. Other features include on-screen display of text attributes such as underlining, boldface, italics and super/subscripting.

In addition to being a stand-alone word processor, Word Writer interfaces with other Timeworks products such as their database program, Data Manager, and spreadsheet, SwiftCalc. Text files from 1st Word can be imported and printer drivers are provided for a variety of printers. Word Writer ST retails for $90.

Abacus not only excels in the technical ST book category—they're clearly the most prolific publisher—but they also have a very good word processor. **Text Pro** is a GEM-based word processor with all the mouse-driven features most of us have come to know. However, once you learn the program, you can use keyboard commands, instead of mousing around. In addition to having almost all of the required features of a good word processor, Text Pro has a few additional ones as well.

Like several other Abacus products, Text Pro was originally written by Data Becker, a German software publishing company. The program is a second generation word processor for the Atari ST. It can print two-line headers and footers, display output on the screen, easily print multiple-column text and perform logical hyphenation, making it one of the best ST programs currently available. In addition, such features as sorting, indexing and table of contents make Text Pro worthy of your attention when choosing a word processor. Text Pro retails for $50.

If you do any kind of writing at all you'll want to use a spelling checker. Two of the word processors mentioned above include spelling checkers, but for my money, the best one is still **Thunder!**. Originally published by Batteries Included, Thunder!—like all BI products—is now published by Electronic Arts. What makes it the best spelling checker for the ST? Features, features and more features.

Thunder! comes with two versions of the spelling checker program. One is a desktop accessory program that can be used anytime from within a GEM program and really contains three programs in one: a 50,000-word, real-time spelling checker; a word expander/corrector; and a set of writing analysis tools. The other,

a word expansion feature, lets you predefine how specific strings of characters should be expanded. You can, for example, define your initials to stand for your entire name. Then, whenever you type your initials, your full name will appear.

The writing analysis tools consist of standardized measurement statistics that rate the "readability" of your prose. They include two readability indexes that measure the grade level you write at, and the number of characters, syllables, words, sentences and paragraphs in your document.

Thunder!'s spelling checker lets you add your own words—a must for any quality spelling checker. Separate dictionaries can be set up to speed up the already-fast checking function. One particularly useful feature of the desktop accessory is that once it's loaded, you can "turn it off," thereby freeing up the RAM for other programs. Most other accessories require you to reboot to rid them from memory.

The other version of Thunder!'s spelling checker is a stand-alone program that can be used with any file. It lets you check files created by non-GEM programs, as well as any using GEM. Thunder! retails for $50.

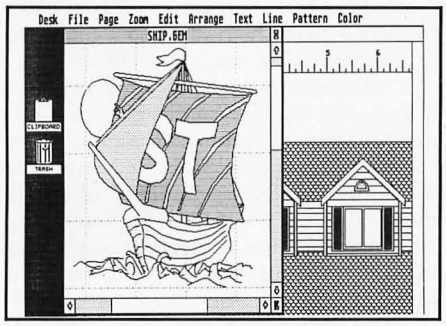
Desktop publishing is a combination of word processing, page layout and drawing that lets you design, compose and paste up the contents of a printed page for forms, newsletters, charts, etc.

Publishing Partner by SoftLogik Corp. is the best desktop publishing program available for the ST. With it, you can create a document of up to 99 pages on a 1040ST and about 27 pages on a 520ST. Type can be as small as 6 points, or as large as 144 points, in half-point increments. Multiple columns are easily created and text can be entered directly, or imported from an ASCII file. Text can be routed from column to column, or page to page, and a large variety of text attributes can be used. Output can go to Epson, Apple Laserwriter, Postscript or a GDOS supported printer.

Publishing Partner is a full GEM application. Drop-down menus make learning and using the program simple; but, the power of the program is far from simple. You can load any Neo-Chrome or DEGAS pictures *and* enlarge or reduce them for exact fit. Several drawing tools—circles, boxes, lines, etc.—can be used from within the program. Different line types and line widths can be specified, and any of forty-eight different fill patterns can be used.

Publishing Partner sells for $150. An impressive program, it combines the features of a drawing, word processing, graphics and publishing program into one complete package. Other programs claiming to be desktop publishing programs pale in comparison.

If you need a little something more than a paint program, something that offers a greater degree of precision, you should consider Migraph's **Easy Draw**. Billed as a professional drawing program for the ST, Easy Draw is an object-oriented program that lets you create custom business graphics, presentation materials, multi-



Easy Draw.

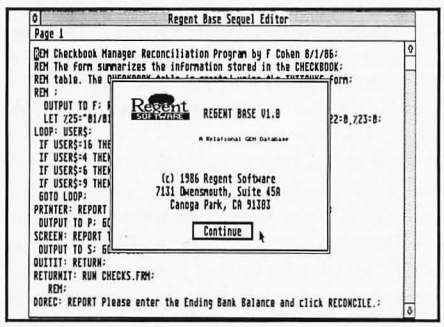
dimensional illustrations and line drawings. The program offers eleven drawing tools and thirty-nine patterns to help create anything from technical drawings to simple illustrations. Drawing is easy with features like multiple windows, full GEM interface with drop-down menus and mouse action, zooming, clipboard art, pre-defined patterns, object rotation and multiple font selection. Easy Draw sells for $80.

EZ-Calc from Computer Palace is a good spreadsheet program that lists for $70. Since it's GEM-based, it uses the mouse for all commands. It requires less memory than other similar programs, so there's more room for your formulas and data. Features include: a 300-column by 999-row maximum worksheet matrix, a built-in ten-key calculator, on-line help windows, a built-in sort routine, the ability to have up to ten function key macros, a split-screen capability, a notepad and graphing. The graphing is especially nice because it can be performed immediately from the worksheet, by selecting a drop-down menu command. EZ-Calc is quite easy to use and compares favorably to other ST spreadsheet programs.

SwiftCalc is Timeworks's spreadsheet program that can be used by itself or interfaced with their Data Manager ST and Word Writer ST products. SwiftCalc features include: a maximum worksheet size of 8200 rows and 256 columns; the ability to print the spreadsheet normally or sideways; a selection of graphic formats (pie charts, vertical bar charts, scatter diagrams and line diagrams); built-in mathematical and financial functions; and full support of the GEM interface. In addition, on-line help and sample spreadsheets are provided. Couple these features with excellent documentation and you have SwiftCalc, which retails for $80.

Timeworks's database program, **Data Manager ST**, also interfaces with the other two Timeworks products (word processor and spreadsheet). Data Manager is a GEM-based application program that comes with sample databases that illustrate mailing lists, sales analysis and other types of databases. The program also features extensive on-line help screens, a print spooler, report writing capabilities, full math capability, and graphics and label-making functions. Data Manager ST retails for $80.

One of the most popular ST database programs is **Regent Base**, a GEM-based product from Regent Software. This popularity is due, no doubt, to the program's dual identity: it's both an easy-to-use, mouse-driven database program and a feature-laden relational database program/language. The advantages of a database programming language are power and flexibility. You can design your database, reports, functions, printing routines and applications exactly as you want them.



Regent Base.

GEM input and output boxes, titles, mouse-controlled buttons and other screen parts normally associated with a database program are easy to create. Regent Base sells for $100.

Another valuable program is Electronic Arts' **Financial Cookbook**. For $40, this program includes dozens of financial "recipes" for calculating everything from mortgages to IRAs. It's been around for years, but I still recommend it highly.

My favorite terminal program is still the no-frills but very practical **ST Talk** by QMI. It's an inexpensive, non-GEM program that fulfills most of my day-in and day-out terminal program needs. The program still costs less than $20—one of the best values around. (Watch for a new version called ST Talk Professional, which should be available soon.)

Entertainment software.

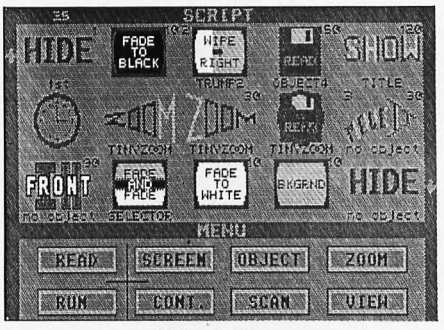
DEGAS is one of the best values in ST graphics software. It was the first full-function ST paint program and one of those products that every ST owner should buy. DEGAS lets you paint in any of the three ST graphics modes and save your files to disk. In fact, the file format used by DEGAS has become an ST standard. DEGAS is now distributed by Electronic Arts and retails for $40, although most stores sell it for under $30.

Soon after DEGAS was complete, author Tom Hudson began thinking about future features for the program. The result, **DEGAS Elite**, is also distributed by Electronic Arts. DEGAS Elite contains all the functions of the original program, plus major enhancements and additional features that make it, in my opinion, the best graphics paint program available for the ST.

Some of the features of DEGAS Elite include: use of eight multiple work screens; the ability to cut and paste all or part of these screens to each other or to disk; the ability to create and use clip art; block operations such as stretch, rotate, horizontal or vertical skewing and distortion. Of course, these neat features involve color animation. Complete control of the direction, speed and color of the animation is at the user's command. DEGAS Elite retails for $80.

One of the first programs to let you unleash the serious animation capabilities on the ST was **Aegis Animator** from Aegis Development. Using DEGAS or Neo-Chrome pictures, the program creates three types of animation: cel, metamorphic and cyclic. Cel animation moves bit-mapped images as a unit. Once cels are clipped and saved, placed on the screen, sequenced and given a duration, the resulting illusion of movement looks quite real.

Metamorphic animation involves creating an image and specifying the changes it will take on. As the picture goes from one image to another and back, the illusion of movement is created. Color cycling creates movement by rotating colors (like DEGAS Elite and Neo-Chrome). These



Make It Move.

techniques may be used individually or together to build professional quality animations. Aegis Animator is a first-class product—from the packaging right down to the manual. It sells for $80.

If you use any of the ST graphics programs, you may be interested in an animation program that makes your art work come to life. Called **Make It Move**, this $50 program from MichTron is really more of an easy-to-use slide sequencing and manipulation program than an animation program. Nonetheless, Make It Move is great for tasks like creating video titles, making presentations, and putting some polish on your collection of ST graphic images. The program is mouse-driven and doesn't require any special programming skills. Features include screen wipes in any of four directions, fades and quick cuts, each with user specified durations. Objects may be zoomed, panned, hidden and moved.

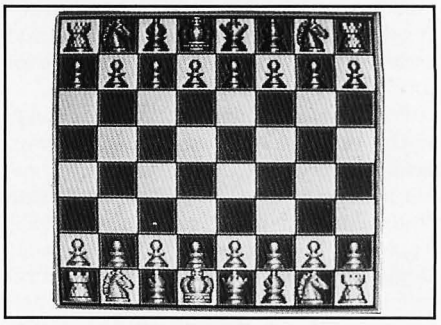
Printmaster Plus from Unison World is a graphics program that allows the user to be creative. With it, you can easily create calendars, flyers, stationery, banners, invitations, signs and greeting cards with many types of printers. The program is menu-based, allowing you to preview your design before printing it. You can also save your graphics to disk. In addition, Printmaster Plus has been totally redesigned to give better graphic output and work with add-on font and graphic disks. Printmaster Plus lists for $40.

Games.

Flight Simulator II from subLOGIC has been around for several years in 8-bit format. Now that the program is available for the ST, users will benefit from an improved visual display with much better graphics (as it should be on the ST), a better control interface for flying the plane from either the keyboard or the mouse, and the ability to get in the air faster with less frustration. Although you may not be a *Top Gun*, Flight Simulator II is, and it can be bought for $50.

Electronic Arts is a name that's not new to seasoned Atari users. They earned their reputation in the 8-bit world by producing excellent games and simulations. Now, they've finally come around to producing software for the ST. One of the best Chess programs around for any computer is **Chessmaster 2000**. Winner of the 1986 U.S. Open Personal Computer Chess Championship, Chessmaster 2000 is based upon the algorithm that was rated at 2,018 by the U.S. Chess Federation. With over 71,000 opening positions, Chessmaster contains the largest move library ever available on a microcomputer.

The game provides twelve levels of play, ranging from "Newcomer" to "Grandmaster." A teaching mode that shows all possible legal moves and a hint mode are provided to help you learn the game. One or two players can play and moves can be retracted as far back as you like. One of the most striking features of the game is the graphic portrayal of the chessboard. It can be viewed in either two- or three-dimensional display. The three-dimensional board can be rotated to see the play from any angle. Additionally, the game allows the players to see all captured pieces and view an on-screen chess clock. Chessmaster 2000 sells for $45.



Chessmaster 2000.

Firebird, makers of the outstanding graphics adventure game, The Pawn, now has a title that is as good, if not better than, their first entry. **Golden Path** is an animated, illustrated graphic adventure game that goes beyond the Pawn in the quality of its graphic "paintings." The central figure in Golden Path is a white-bearded wise man, who also happens to be a martial arts guru. The player uses the mouse to control this surrogate character through forty-five screens of smooth, life-like animation, while solving his quest. A book of lore is constantly available for examination through an on-screen window that provides clues in developing the wisdom necessary to fol-

low the golden path unerringly. Golden Path retails for $45.

The wait is over. Mindscape has finally published **Balance of Power** for the ST. Balance of Power gives the user the power to make decisions as he plays the role of either the President of the United States or General Secretary of the Soviet Union. The player's goal is twofold: complete eight years in office without initiating a nuclear conflict and accumulate more prestige points than the opposing superpower. To win world prestige, diplomatic tools such as military aid, covert destabilization, treaties, military advisors and troops are available to influence friendly and unfriendly nations.

Written by once-Atari game designer Chris Crawford, Balance of Power has received critical acclaim from the press and thousands of Macintosh and IBM PC users. You'll recall that Crawford's earlier work for the Atari 8-bit, Eastern Front, was also a tour de force in its time. Balance of Power retails for $50.

Mindscape has another dynamite game that was the first of the CinemaWare series of games to be developed for the ST. Called **S.D.I.** (Strategic Defense Initiative), it's a science fiction adventure involving a young, beautiful Russian cosmonaut and a handsome American scientist in the midst of a global war of the super powers. Mindscape calls it a story and a flight simulator rolled into one. Over twenty-five screens accompany the story line, requiring two disks for the entire game. The game has a movie-oriented theme and is played in real-time, making it—for the most part—a decision-making game of strategy. S.D.I. retails for $50.

If you enjoy a quality shoot-'em-up, you'll probably like **Goldrunner** from MichTron. It's a *fast* arcade-style game, similar to Galaxian, except the fighter isn't stationary. The entire playfield scrolls vertically, both forward and backward, and you can fire in the direction you're traveling in. The game may be controlled by either a mouse, joystick or keyboard; there's no need to select which one you want to use, since all are active at once. I especially like the opening title which says: "If it moves, shoot it. If it doesn't move, shoot it." That's as good a description of the game as I could give. Goldrunner lists for $25.

GATO from Spectrum Holobyte is a one-player submarine simulation that challenges you with a series of World War II missions. The missions range from search-and-rescue to convoy support to specified targets. Ten different levels of difficulty are provided and the use of either mouse or keyboard control makes the game easy to play. GATO is an entertaining simulation and the best of its kind for the ST. Retail price is $40.

MicroLeague Baseball is a computer simulation of major league baseball that uses your managerial abilities, together with the actual statistics of real players from twenty-five teams. One or two players can play and each can choose how the particular batter will hit and run, and how the defense will pitch and field the ball.

Winter Games.

You can also make personnel changes (i.e., relief pitchers, pinch hitters and pinch runners). MicroLeague Baseball is neither an arcade game or a statistical simulation, but a graphic-oriented baseball game in which you, the manager, make decisions and see them carried out on the field. If you love baseball, you'll definitely enjoy MicroLeague Baseball from MicroLeague Sports Association. Retail price is $50.

Winter Games has been very popular on a host of computers and now comes to the ST. Seven true-to-life sporting events, from bobsledding to ski jumping to figure skating, challenge the user. You'll need skill and stamina to succeed. The Biathlon and four other events are also included. An opening ceremony—complete with national anthems—greets up to eight players.

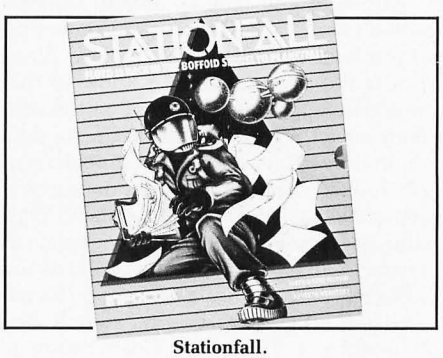
World Games is a continuation of Epyx's "Game" series in which you can compete with up to eight players in eight events. Cliff diving, sumo wrestling, barrel jumping, bull riding, weight lifting, giant slalom, poll vaulting and a hop, skip and jump contest should keep you and your friends entertained for hours on end. Both Winter Games and World Games sell for $35.

Infocom ranks as the undisputed leader in text adventure games. Their entire catalog of several dozen titles is available for the ST, in many categories like science fiction, fantasy and adventure. Each title is labeled with one of the following grades: Introductory, Standard, Advanced or Expert. The grades are based on the difficulty level of the game—although all titles require some degree of puzzle solving—and are written in the spirit of fun.

My long-time favorite Infocom text adventure has been **Planetfall** because it contains a combination of adventure, science fiction and humor. The sequel, called **Stationfall**, is equally challenging and fun. A good choice for the novice text adventurer might be **Seastalker**. It offers a good challenge and—like all Infocom games—is a quality product.

(It's getting pretty difficult to select only a few favorites from all of the excellent Infocom titles. . .)

Another title I like is **Trinity**. A cross between the Twilight Zone and Alice in Wonderland, Trinity leads you into an alternate universe where magic and physics coexist, and every atomic explosion that has ever occurred is inexplicably connected. The chilling climax of the story takes place in the New Mexico desert on July 16, 1945. You arrive minutes before the most fateful experiment of all time: the world's first atomic explosion, code named Trinity.



Stationfall.

Infocom recently released a title in a strictly humorous vain with a different twist. Called interactive short stories, or more specifically, **Nord and Bert Couldn't Make Head Nor Tail of It**, this is a game where nothing is quite as it seems. Eight short stories are provided, each involving a different type of wordplay. You'll be challenged to come up with the clichés, spoonerisms and other verbal trickeries needed to complete the puzzles. Fortunately, Infocom offers built-in hints (for the first time) that can be called upon when the rough get going.

Every ST owner should have at least one

Infocom game in their software library. List prices of these text adventures range from $40 to $60 each.

Accessories.

There are a number of accessories for the ST user that seem frivolous. On the other hand, some serve quite a useful function and are worth their price. One such product is called **The Last Stand** from O&R International. Billed as the "complete desktop computer center," this is an attractive two-level shelf made out of plastic that can support two disk drives and a monitor.

The ST is stored under the bottom shelf when not in use and can easily slide forward for use. Your drives sit on the bottom shelf with room to spare for disks or other computer paraphernalia such as a modem or hard disk drive. The top shelf supports your monitor, and all cabling is neatly kept behind the unit. The Last Stand retails for $45.

Black Box Corp. has dozens and dozens of products to make your computer life easier. The majority of these products are hardware switch boxes, such as **ABC-25 Switch**, a switch that allows two computers to talk to the same printer. Another useful switch model allows two computers to communicate through the same modem.

Although you may be able to find assorted flavors of A-B boxes and some other products for a little less money, Black Box is the only company I know of that provides support before and after the sale. They have the most complete line of data communication and computer devices in the industry, provide telephone support and publish a catalog, complete with periodic updates. From years of experience as a user, I can tell you that the products are quite reliable and function exactly as advertised. The Mercedes of switch box companies is Black Box Corporation of Pittsburgh.

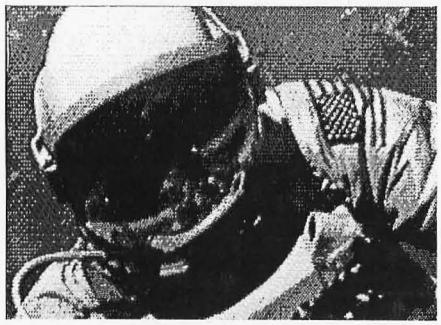
You may not know it, but your computer and peripheral equipment are at the mercy of the power company and Mother Nature. The power company can supply you with voltage that's either too high or too low. Temporary power surges or spikes can seriously damage your hardware, as can undervoltage which occurs during a brownout. Mother Nature likes to throw lightning storms across the countryside. Although telephone and power lines are designed to prevent lightning from traveling along the wires and into your home, it occasionally happens.

To get protection from these potential catastrophes you need a surge suppresser, which is a device that's inserted between the AC outlet and your computer. By limiting extra high or extra low voltages, it protects your computer's electronic components from being "fried." These products come in a variety of shapes and sizes, but one company that offers a complete line of products that have been tested to work as advertised is Panamax.

Panamax makes a variety of different products that include power line conditioning, telephone line protection, multiple outlet strips and brownout (undervoltage) protection. In addition, there's a lighted on/off toggle switch, reset button, circuit breaker, replaceable fuse and separate surge and brownout indicator lights. Some models have circuit breakers, alarms and other features to help you manage your power protection needs. Panamax's models sell for $69 to $149.

Hardware.

If you happen to have *both* a monochrome and color monitor for your ST, you can make things easy on yourself by getting **Monitor Master** by Practical Solutions. Monitor Master allows you to connect a monochrome and RGB monitor to your ST, and includes separate audio and composite video output jacks. Changing



ComputerEyes.

from monochrome to color, or vice versa, is as easy as pressing a button on the unit. Further, the composite video output is great for providing a video signal to a VCR. Although the composite signal is only available on 520s and 1040s with an RF modulator, the audio is available on all machines. Monitor Master sells for $50.

Digital Vision was the first to develop and market a video digitizer for the 8-bit. Although it could only digitize black-and-white images, it was seen as a unique and low cost means of capturing graphic images.

Now, Digital Vision has a video digitizer for the ST called **ComputerEyes**. This system includes both hardware and software and supports all of the graphic capabilities of the ST. The hardware interfaces easily to any source of standard NTSC color or black-and-white composite video. These sources include video cameras, VCRs, video laserdiscs and other computers. All capture and display functions are mouse controlled under GEM and images may be captured in either monochrome or full color.

The ComputerEyes software performs automatic calibration of brightness, contrast and color balance. Once the image has been captured, brightness, color content and contrast can be easily adjusted to suit your needs. The images saved by ComputerEyes are compatible with NeoChrome, DEGAS, DEGAS Elite and other graphic programs. ComputerEyes retails for $250.

If you've been waiting for the quintessential desktop accessory cartridge for the ST, **DeskCart!** from Quantum Microsystems Inc. (QMI) may end your search. Deskcart! contains a battery-backed-up real-time clock and a plethora of desktop accessories on one cartridge, all of which consumes a meager 75K of precious RAM.

The fourteen accessory programs include: a calendar that runs to the year 2040; an appointment book with alarm; a notebook which is really a mini-word processor that allows you to create, edit and save multiple 12-page notebook files; a card file that allows nine lines of data; a very good calculator; a typewriter that lets you use the ST to address envelopes or print other short pieces; an address book; a VT-52 terminal emulator; keyboard macros; a RAMdisk of any size on any drive; disk utility functions; a print spooler; a control panel similar to the ST's control panel; a screen dump; and memory test. DeskCart! retails for $100.

The best hard disk for the ST has been —and still is—the **Supra 20-Megabyte hard disk**. I've used the 20-meg drive for almost a year and can still says it's an excellent piece of hardware. The 20-megabyte drive retails for $699 and can be purchased direct from Supra or from retailers. All Supra ST hard disks connect to the ST's DMA port and come with a boot program that allows the ST to boot directly from the hard disk.

Other things.

Under our miscellaneous category we thought it appropriate to include Atari user group memberships. There are many fine groups around the country and their memberships average about $20.

These groups have monthly meetings,

publish newsletters and often have libraries of public domain software. User groups always have knowledgeable people who are willing to share information. With the new, "lean" Atari, user groups often provide a good source of Atari-specific information.

Both beginning and expert ST users may also want to take a look at the books and magazines now available. The predominant book publisher in the ST field is Abacus Books and they have more than a dozen different titles that will fill just about every need. I'll just mention a few that may be of interest.

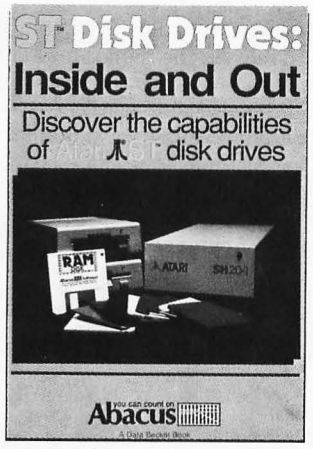
Atari ST Graphics and Sound is a book that teaches you how to create graphics and use the built-in sound facilities of the ST. Examples are written in Basic, C, Logo and Modula 2, so there's something for everyone.

ST Tricks and Tips covers such topics as: using GEM from BASIC, combining BASIC and machine language, creating a RAMdisk and print spooler, and automatically starting TOS applications. If you would like to learn more about programming, this book will be a great help through the use of sample programs and tips for both the new and expert programmer.

ST Internals is a well organized, useful compilation of technical ST information. A clear description is given for such ST "parts" as the mouse, keyboard, 68000 processor, custom chips, I/O ports (RS-232, parallel, cartridge, DMA, floppy disk, MIDI and video) and operating system. In addition, this volume contains information about GEMDOS, graphics and the BIOS (Basic Input/Output System).

ST Disk Drives—Inside and Out covers topics like files, file structures and data management. The book also discusses all aspects of floppy, hard and RAMdisks as they relate specifically to the ST. The book also lets you explore ST disk drives from a programming or technical perspective.

There are a few programming books that are useful for the person who wants to learn a new language. ***Atari Basic Training Guide*** is a functional and educational introduction to ST Basic. From problem analysis to BASIC commands to algorithms, the book provides the fundamentals of programming in an easy to understand format. ***ST BASIC to C*** is an ideal book if you already know BASIC, but want to learn how to program in C. This book compares BASIC programs and their C equivalents, so you can make the transition easily and rapidly.



ST Disk Drives—Inside and Out.

Atari ST Machine Language is devoted exclusively to programming the 68000 microprocessor. Starting from scratch, with chapters on microcomputer fundamentals and hardware fundamentals, this book leads the programmer, step by step, into the world of machine language programming.

Another Abacus book, ***ST GEM Programmers Reference***, provides detailed information on GEM, with examples written in C and 68000 assembly language. An overview of such topics as VDI, AES, GDOS and GIOS is included. In addition, material on programming in GEM, with explanations on using the editor, C compiler, assembler and linker of the development system, are included.

All these books are currently available and sell for $20 each, except *Basic Training Guide*, which costs $17, and *ST Disk Drives*, which retails for $25.

Everyone needs an extra joystick or two and Epyx has an excellent one for right-handed people. Called the **500XJ Joystick**, this stick is made for fast action because it fits your (left) hand so well. The fire button is located right where your trigger finger falls, so you can blast the aliens or select from a menu fast and accurately. The stick has a short throw for more precise control and you can actually hear and feel every move of the stick. If you need a new joystick, you should definitely check out the under $20 500XJ from Epyx. The 500XJ also comes with a 5-year or ten-million-shot warranty.

There are several on-line information services available to the ST user. To get started using the vast resources of CompuServe Information Service, the **CompuServe Starter Kit** is a good value. For $39.95 you receive five free hours of connect time and a thorough manual. CompuServe offers computer and general interest news and information, and the user is also able to electronically correspond with other system users.

Another information service is the **Atari Users' Group SIG on Delphi**. Run by ANALOG Publishing, Delphi is a great way to participate in the world of Atari computing. Delphi is a full-service, on-line communication and information network. Access charges average as low as 10 cents per minute from most parts of the country and there's no extra charge for high-speed access.

The Atari SIG on Delphi provides a host of services and features. You can send and receive messages from and to other Atari users worldwide in the Forum. The database area includes thousands of free programs that can be downloaded, including the programs presented each month in **ANALOG Computing** and **ST-Log**. The conference feature allows you to meet electronically with other Atari users. In addition, the ANALOG Publishing staff is available on-line to answer your questions and share information.

```
Name: ST_CITADEL_I.1A
Type: PROGRAM
Date: 21-SEP-1987 12:40 by YELVINGTON

STadel I.1a (Citadel) is the latest version of the powerful discussion/BBS
system. This version includes CBS-net compatible networking: Shared discussion
rooms, netmail, and sysop-requested file transfers among Citadels. It also
includes a UUCP-compatible networker for sending/receiving mail and files via
Usenet (and Bitnet, CSnet, etc., via the various gateways). Other enhancements
of this version include more file-transfer protocols, autonet rooms, and
"floor" organization options. STadel is from David (orc) Parsons of
Minneapolis; source code is available from the author (see note in runtime
archive).

Keywords: APPLICATION, CITADEL, BBS, STADEL, UUCP, USENET, ARC

Contents:

  1  RUNTIME.ARC (Size: 136704 Count: 17)
  2  HELPFILE.ARC (Size: 26240 Count: 13)
  3  DOCS.ARC (Size: 58368 Count: 14)

ACTION> _
```

Atari Users' Group SIG on Delphi.

To use either of these information services you'll need a modem, a device that allows your computer to electronically interact with another computer, via phone lines. There are several good ones to choose from, but one of the best values around is **Avatex's 1200-baud modem**. It retails for under $130 and is Hayes compatible.

An unusual, but useful, product is the **X-10 system** from X-10 America, Inc. This is a system that's used to remotely turn AC devices on and off, either directly or via a timer. Assorted modules control such things as lights, appliances and other electrical devices by means of commands sent through the AC wiring in your house. A computer peripheral, called the **X-10 Powerhouse**, attaches to the RS-232 port of your ST and can be programmed to control the various AC devices in your house.

The X-10 Powerhouse unit is self-

powered, and once programmed, can be disconnected from the computer, thus freeing that port for other uses. The X-10 Powerhouse is available from MichTron for $25. The software needed to control the Powerhouse is called **Echo** and is also available from MichTron. It lists for $40. Echo is a GEM application, so all inputs are entered via the mouse and any desktop accessories are always available. ///

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

Companies Mentioned in This Article

Abacus
P.O. Box 7211
Grand Rapids, MI 49510

Aegis Development, Inc.
2210 Wilshire Blvd. Suite 27
Santa Monica, CA 90403
(213) 392-9972

Avant-Garde Software
381 Pablo Point Drive
Jacksonville, FL 32225

Beckemeyer Development Systems
478 Santa Clara Ave.
Oakland, CA 94610
(415) 452-1129

Black Box Corporation
P.O. Box 12800
Pittsburgh, PA 15241.

Central Point Software
9700 SW Capitol Highway, Suite 100
Portland, OR 97219
(503) 244-5782

CompuServe Communications
5000 Arlington Center Blvd.
Columbus, OH 43220
(614) 457-0802

Computer Palace
710 McKinley
Eugene, OR 97402
(800) 452-8013

Delphi
(800) 544-4005
(617) 491-3393 (in MA)

Eidersoft USA, Inc.
P.O. Box 288
Burgettstown, PA 15021
(412) 947-3739

Electronic Arts
2755 Campus Drive
San Mateo, CA 94403
(415) 571-7171

Firebird Licensees, Inc.
P.O. Box 4874
North Central Avenue
Ramsey, NJ 07446
(201) 934-7373

Infocom, Inc.
125 Cambridge Park Drive
Cambridge, MA 02140
(800) 262-6868, ext 17Y

MichTron
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700

Micro League Sports Association
2201 Drummond Plaza
Newark, DE 19711
(800) 752-9225

Migraph, Inc.
720 S. 333 St. (201)
Federal Way, WA 98003
(206) 838-4677

O&R International, Inc.
165 N. Balboa St. C-14
San Marcos, CA 92069

Panamax
150 Mitchell Blvd.
San Rafael, CA 94903
(800) 472-5555).

Practical Solutions, Inc.
1930 E. Grant Rd.
Tuscon, AZ 85719.

Quantum Microsystems, Inc. (QMI)
P.O. Box 179
Liverpool, NY 13088.
(315) 451-7747

Regent Software
7131 Owensmouth, Suite 45A
Canoga Park, CA 91303
(818) 882-2800

SoftLogik Corp.
4129 Old Baumgartner Rd.
St. Louis, MO 63129

Spectrum Holobyte
2061 Challenger Drive
Almeda, CA 94501
(415) 522-3584

subLOGIC
713 Edgebrook Drive
Champaign, IL 61820

Supra Corp.
1133 Commercial Way
Albany, OR 97321
(503) 967-9075

Timeworks, Inc.
444 Lake Cook Rd.
Deerfield, IL 60015
(312) 948-9200

Unison World
2150 Shattuck Ave., Suite 902
Berkeley, CA 94704
(415) 848-6666

XLent Software
P.O. Box 5228
Springfield, VA 22150

Reader comment *continued from page 9*

this, but past experience indicates it'll be an uphill fight.

But, by the same token, I don't think Atari deserves all the flak they've been catching. Granted, there have been some major delays in their new product line, and there are a *lot* of unhappy Atari owners out there wanting these new products. But be realistic folks, try to see the issue from *both* sides. Even Apple and IBM, with all their cash reserves and personnel, have had major product delays. The problem is by no means unique to Atari.

So what can be done about the current "War of Words" between the company and its enthusiasts? For our part, I suggest a little patience for a change. Voice your opinions, suggestions and even your gripes—but try to keep it reasonable. Atari has done incredibly well in getting the XE and ST series out on the market and some problems and delays have to be expected with any new product developments. Keep in mind the limitations they have to operate under and think of ways they can get around them.

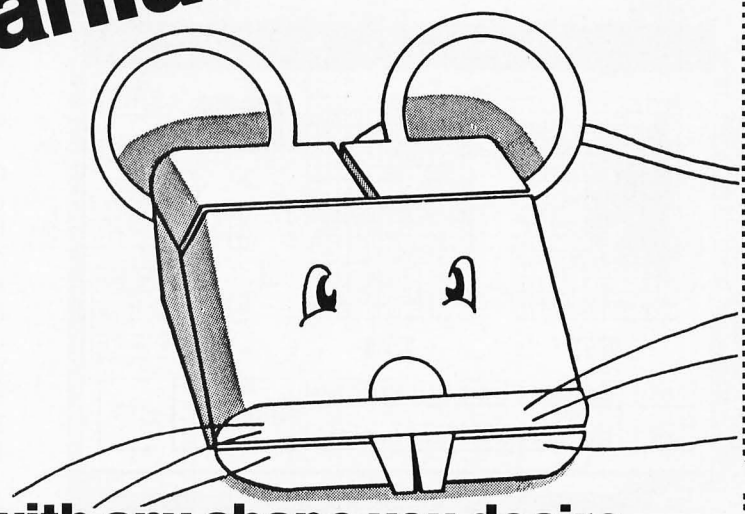
As for Atari, I can only suggest a little more restraint in their product announcements and fewer "development leaks" designed to excite user interest without having hardware ready to support those leaks. I also urge an all-out effort to get that new fabrication center finished — ASAP!

We're all in this together, people. In the early days we did our part by supporting the company long after the rest of the industry had given it up for dead. In response, Atari gave us the most powerful 8-bit and 16-bit computers in the industry and released them at prices we could afford. Hang in there folks, things will get a *lot* better once Atari has its fabrication plant up and running. Remember, that with each quarter, Atari grows larger and stronger, which can only help them make the needed changes and acquisitions for a still more productive future. Once that starts, *watch out!*

Your computer curmudgeon,
Gregg Anderson
Rapid City, SD

Mouse-Ka-Mania!

A SPECIAL INCLUSION



Replace the arrow cursor with any shape you desire . . . and animate it too!

by Charles F. Johnson

How would you like to be able to change the shape of the GEM mouse cursor, customize it with a sophisticated editor, and install the new mouse cursor as the GEM default in place of that omnipresent arrow? Would you like to run your favorite GEM programs with your custom mouse in spite of their (and GEM's) best efforts to reset it to the arrow? And what if I threw in the ability to have mice in *two colors* in medium and low resolution? Then, for the *piéce de résistance,* how about if I let you *animate* those customized mice—up to thirty-two frames, each with its own color information? If all this sounds like fun, read on. . . **Mouse-Ka-Mania** is for you!

Mouse-Ka-Mania is a GEM desk accessory written entirely in 68000 assembly language. The program grew from a suggestion that Maurice Molyneaux (author of the "Step 1" series) made to Clayton Walnum, **ST-Log**'s Technical Editor. Clay passed the suggestion on to me (an "accessory that lets you edit the mouse shape," he said) and the rest is, well, history, kind of. (See Maurice's "Mice-Conceptions" accompanying this article, for some excellent tips on creating your own mice.)

Mouse-Ka-Mania is a mouse editor that can be run from within any GEM program that allows a menu bar; but it also has several capabilities that set it apart from other mouse editors. It lets you replace the system's arrow cursor (you know, the one you see *every* time you turn on the ST) with any mouse-begotten shape you can dream up. And, it lets you design an animation loop of up to thirty-two frames, which will cycle at any speed.

The mouse· editor part of **Mouse-Ka-Mania** works in medium or high resolution. However, in low resolution you can still load a custom mouse shape (single-frame or animated) from disk, install or remove it as the default, and adjust its colors.

How to work this thingie.

The **Mouse-Ka-Mania** accessory is included on this month's disk edition of **ST-Log**, under the filename MOUSEKA.ACC. You can also find the program on the ANALOG Publishing Atari SIG, on the Delphi information service. Simply copy this file to the main directory of your boot disk. (If you have a hard disk, you'll probably want to copy it to drive C.) And remember, GEM only allows you to install six accessories at a time.

Now, turn off your computer and reboot with the disk containing MOUSEKA.ACC. When the GEM desktop appears, you'll find **Mouse-Ka-Mania** listed among the accessories in the "Desk" drop-down menu. Click on its name and the editor will appear.

An ST mouse cursor is composed of two elements: data and mask. The data is the actual shape of the cursor (usually in black or the same color as the text). The mask is used to ensure that the cursor is still visible if it passes over an area of the same color as the data. For example, examine the standard ST "arrow" cursor as it passes over a selected object, and you'll see a 1-pixel-wide area of white space surrounding the cursor. This is the mask.

The **Mouse-Ka-Mania** editor lets you edit both data and mask for the mouse cursor on the same screen (see Figure 1). The left button turns on the squares in the grids, and the right button erases them. You can hold the button down and drag the mouse to draw continuous lines and other shapes. When you draw on either data or mask grids, it shows up on both. If you're drawing on the data side, you'll see a solid fill pattern on both grids; if you're drawing on the mask grid, the cells will be filled with a different pattern (and a different color in medium resolution).

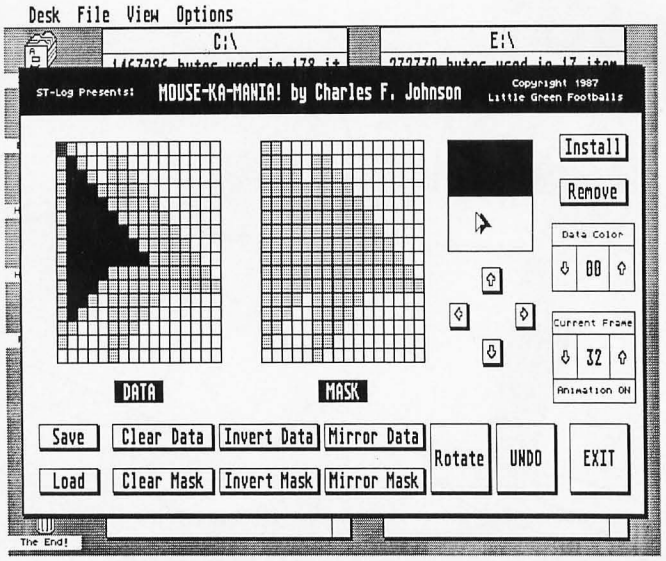


Figure 1.

To understand how the editing grids work, picture them this way: each one consists of the grid itself, with two transparent plastic cels laid over it. The shapes of the data and mask are drawn on these transparent cels. On the data grid, the data is drawn on the top cel, overlaying the mask underneath. On the mask grid, the mask overlays the data. Hence, if you draw in the same place on both data and mask grids, then erase the mask, you'll see the data appear from underneath the mask as if it were being uncovered.

The "hot spot" is the active point of the mouse cursor. On the system's default arrow cursor, it's located at the tip of the arrow. (Makes sense, I guess.) This is how GEM knows what you're pointing at as you move the mouse around the screen and select objects by pressing the buttons. **Mouse-Ka-Mania** lets you move the hot spot to any point in the data grid, allowing you to construct odd shapes like upside-down arrows. The hot spot is set on the data grid only, by one of two methods. The first method is to hold down the CONTROL key on your ST's keyboard and click the left mouse button on the square where you want the hot spot to appear. The second method is to press both mouse buttons at once, but there's a trick to this. If the square you want to move the hot spot to is already turned on, you should first press the left button (so **Mouse-Ka-Mania** doesn't turn the square *off*), then tap the right button and release it. If the square isn't selected, first press the right button, then tap the left button. Doing it this way will insure that you don't accidentally draw or erase squares in the data grid.

Just to the right of the editing grids is a rectangle with the top half filled in. This is the "change box." Whenever you move the mouse cursor into this box, it will instantly change to the form you're currently editing, giving you an easy way to see what your mouse is up to.

Down the right side and along the bottom of the editing box are rows of objects which can be selected with either mouse button. These button objects provide the editing commands to manipulate your rodent in lots of ways, including inverting, mirroring and clearing either data or mask. There's a "Rotate" button which causes the mouse form (both data and mask together) to rotate 90 degrees in a clockwise direction. There's even an UNDO command which will reverse your last action. UNDO only works on the very last thing you did, whether drawing or using an editing button.

The SAVE and LOAD commands call up the dreaded GEM file selector, to enable you to store your mouse shapes or animation sequences on disk. **Mouse-Ka-Mania** uses default extensions of *.DAT* for single-frame mouse data files and *.MKM* for animation data files. When you type in a filename you don't need to add the extension; just type the first eight characters and **Mouse-Ka-Mania** will automatically tack the .DAT or .MKM onto the end. Of course, if you *want* to type *.DAT*, or use some other extension, you can still do that. If the filename you use already exists on the disk, **Mouse-Ka-Mania** will warn you with an alert box that asks if you want to overwrite the existing file.

Underneath the change box are four small buttons containing up, down, left and right arrows, in a diamond formation. These buttons allow you to scroll the mouse shape 1 pixel in any of those directions. If the shape scrolls off the editing grids, it will wrap around to the other side.

In the upper right corner, next to the change box, are two buttons labeled "Install" and "Remove." "Install" lets you install the current shape on the grids as the system mouse cursor, replacing the default arrow pointer entirely. Your mouse will stay installed no matter which program you run. Anytime a program or the desktop tries to change the mouse to the arrow form, **Mouse-Ka-Mania** will intercept and replace it with the cursor *you* want. "Remove" unhooks the custom mouse from the system and lets the arrow reign supreme once again in the mouse kingdom.

Underneath these two buttons is the color box. This box lets you adjust the color of either the data or the mask. When you first summon **Mouse-Ka-Mania** from the desk menu, the top line of this box will display the words *Data Color*. Clicking on this top line will toggle it between Data Color and Mask Color, and the color value below will be displayed accordingly. Click on the up and down arrows on either side of the current color value to raise or lower the value, and move the mouse into the change box to see what it looks like. If you find a color combination you like, install it as the system default. . .and presto! Colored mice, flying around your ST's screen!

Go ahead and play around. . .draw some shapes on the editing grids; try clicking the different editing buttons and notice their effects. Alter the colors for data and mask, and move the mouse into the change box to check out your creations. Install it and you can use your mouse shape to draw with. When you've got a feel for how **Mouse-Ka-Mania** works, come back to this article and we'll dig into the subject of animation.

Okay, now that you're a maestro of mouse manipulation (I make no apologies for my shameless alliteration), let's examine the animation control box, located just underneath the color box. When **Mouse-Ka-Mania** first appears, the top line of the animation control box should display the words *Current Frame*. If you click on this top line, you'll advance

through the other two animation options: "# of Frames" and "Delay." The up and down arrows change the values displayed in the center of the box.

"Current Frame" is the mouse shape you're presently editing, which can be a value from *1* to *32*, and "# of Frames" is the number of consecutive frames that define an animation "loop." This will usually be set to the same number as the last frame in an animation sequence. "Delay" is the number of vertical blank interrupts (about 1/60th of a second) to wait between frames of an animation sequence. The bottom line of the animation control box toggles between "Animation OFF" and "Animation ON" when you click on it. Note that, if a mouse shape isn't installed, turning animation on will have no effect.

If you hold either SHIFT key while clicking on the up or down arrows to change the "Current Frame," the shape that's on the editing grids will be copied into the new frame. This lets you easily generate variations on a shape. If you want to copy a shape to a nonconsecutive frame, you should save it to disk first, then load it into the new frame. (A RAMdisk is ideal for this purpose.) You'll also notice that the data and mask colors are remembered for each frame of an animation sequence, allowing you to create fancy color cycling effects.

In low resolution, **Mouse-Ka-Mania** works somewhat differently. To be more specific, the editor doesn't work at all . . .there simply isn't room on the low resolution screen to do it justice. Instead, you'll see the dialog box in Figure 2, which allows you to load, save and remove a mouse form, and also change the colors in much the same way you do with the editor in medium resolution. In low resolution, changing the colors instantly changes the mouse's color; there's no need to install it after a color adjustment. The color numbers used by **Mouse-Ka-Mania** correspond to the colors in the control panel in the manner shown in Figure 3.

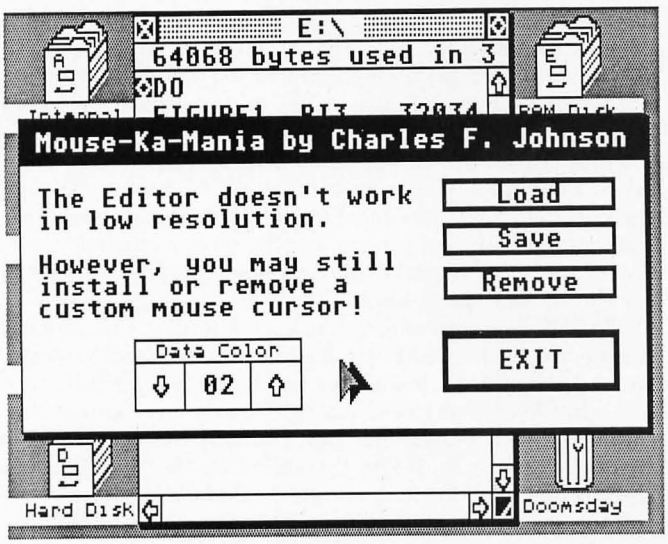


Figure 2.

You can load an animated mouse file in low resolution, but you can't edit it in any way. If you want to adjust the colors for the individual frames of a low resolution mouse animation, you must do it in the medium resolution editor and save it to disk. This isn't too much of a problem, especially if you use a RAMdisk. Some impressive color cycling effects are possible in low resolution, with sixteen colors to choose from.

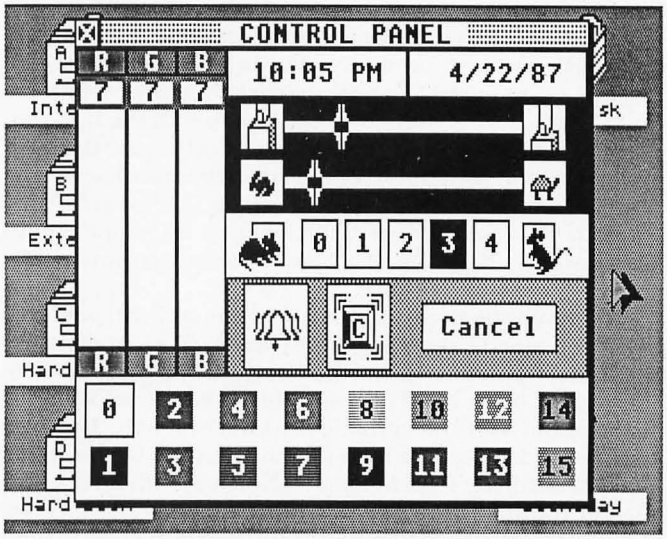


Figure 3.
Boot up with a custom mouse.

This handy-dandy little accessory will also allow you to install a custom mouse (single-frame or animated) when the accessory is loaded, at boot time or after a resolution change. Since the number of colors you can work with is different in low, medium and high resolution, you may install different mice for each resolution.

To install a mouse as the boot default, copy the mouse data file you've saved with **Mouse-Ka-Mania** to the main directory of your boot disk, and rename it either MOUSELOW.DEF, MOUSEMED.DEF, or MOUSEHI.DEF, depending on which resolution you want to use it with. Alternatively, you can just save the mouse shape or animation sequence to the main directory under one of those names.

The little mouse who could.

The ST's "desktop" program is really in love with that arrow cursor; it tries to keep setting it back every chance it gets. You may occasionally notice the arrow cursor appear for a brief flickering moment, but, once you've installed a **Mouse-Ka-Mania** mouse, the ST will never be able to totally replace it with the arrow again—at least until you remove it.

Mouse-Ka-Mania performs this feat of legerdemain by stealing the ST's vertical blank interrupt vector, at location $70 in the zero page. Sixty times a second (seventy in monochrome), the system's sprite buffer is checked to see if something (either GEM or an application) has put that old arrow cursor in there, and, if so, it's replaced with your custom mouse. This sprite buffer isn't a documented location, so **Mouse-Ka-Mania** uses a trick to find it. In its initialization phase, immediately after loading, **Mouse-Ka-Mania** calls the AES function graf_mouse, to set the mouse to the arrow cursor. Then it searches memory for the word values

that define the arrow shape. When it finds all thirty-two words, it has found the place where the ST keeps its mice.

Those of you with some ST programming knowledge may be wondering why I didn't use the vertical blank queue list, instead of stealing the main VBLANK interrupt vector. The vertical blank queue contains a list of routines which the system will call on a vertical blank interrupt. A pointer to this list is contained in the system variable _vblqueue, located at $456. To install a vertical blank routine of your own, you simply put the address of your routine in the first zero entry in this list. The OS then automatically executes your code as part of its main VBLANK interrupt routine, saving all registers and generally taking care of the housekeeping for you. This would have made things a bit simpler, but I didn't use this list because of something I discovered while testing **Mouse-Ka-Mania**.

Apparently, there are quite a few programs out there (even some commercial ones, like Word Writer ST) that don't use the "officially approved" method of installing a routine into the vertical blank list. There are usually eight vectors in the list, and the ST's OS normally uses the first one. The legal (and sensible!) method is to search from the beginning of the list, until you find an open vector, and put the address of your routine there. The offending programs simply use the second vector, assuming it will always be open—and clobber anything else that happens to be there. Needless to say, this kind of sloppy programming can wreak havoc on the system if some other program has already installed a routine into the second slot in the VBI list.

Therefore, since **Mouse-Ka-Mania** is an accessory—and must be able to peacefully coexist with as many different types of applications as possible—I decided to install my routine by plugging its address into the main vertical blank interrupt vector at $70, and then having my code fall through to the normal VBI routines. So far, I've found no programs incompatible with this approach.

Problems, problems.

As is often the case, I programmed the main routines of **Mouse-Ka-Mania** very quickly (actually, in less than a day), then spent three more weeks tracing down problems like the one above. I discovered another interesting quirk in the ST's operating system while developing **Mouse-Ka-Mania**. ("Interesting quirk" is a nice euphemism for "bug," don't you think?)

When you do a resolution change from low to medium, or vice versa, using the desktop's set preferences menu, the interrupt vectors in page 0 are not reinitialized—although accessories and the DESKTOP.INF file *are* reloaded, and the memory they used previously is cleared. Sooo. . . if I installed a custom mouse with **Mouse-Ka-Mania**, then changed resolution, the vertical blank interrupt vector I had stolen was suddenly pointing at empty memory. This led to some very picturesque system crashes.

This problem had me stumped for almost a week. How could I tell if the user was changing resolutions from the desktop? I tried using the vector at location $46E, about which Abacus's *Atari ST Internals* says: "Vector for changing the screen resolution. A branch is made via this vector when the resolution is changed." Sounds like it should work; I just steal this vector too, and turn off my vertical blank if a resolution change is in progress. Unfortunately, Abacus forgot to mention that this vector is only used if you're switching from *monochrome* to *color*, or vice versa, not when switching between low and medium resolution on a color monitor.

Finally, I found an approach that solved the problem: when **Mouse-Ka-Mania** is initialized, it searches through the desktop program's object trees in memory, until it finds the buttons in the set preferences dialog box labeled "Low," "Medium" and "High." (This happens right after the search for the mouse sprite buffer; see above.)

Then, during the same vertical blank routine that watches for the arrow shape, **Mouse-Ka-Mania** also checks the status of these buttons. If you choose a button which will cause a resolution change (for instance, if you're in medium resolution and you click on the "Low" button), **Mouse-Ka-Mania** instantly detects it. If you then click on the "OK" button, the vertical blank routine simply de-installs itself, preventing the resolution change from upsetting the ST's delicate temperament.

This method—in programmer's lingo—is a "kludge" of the first water. It irks me to use it, but in this case I had no choice, because GEM does *not* inform accessories of a resolution change!

The defeat of an old adversary.

In the course of developing **Mouse-Ka-Mania**, I discovered a way to get around one of the most frustrating programming problems I've encountered on the ST: mouse button bleed-throughs. This problem occurs when you're using VDI to check the mouse position and button states, then call some AES function (such as the file selector box).

The AES part of GEM seems to "remember" some button presses, even though they've already been accepted by the VDI. So, if you call an AES function to get input from the mouse, the AES will occasionally react as if you had pressed the button—even though that was *ages* ago. Real trouble is in the works if you happen to be over the spot where an application's "Exit" button will be when the AES grabs that spurious button press. You may find yourself suddenly returning to the desktop! What we have here is a communication failure between the AES and VDI parts of GEM.

Official GEM doctrine says, "Don't mix AES and VDI input." When writing a GEM-based program that depends on user input, you're supposed to stick with one or the other. Well, for several reasons (the foremost being speed), that was an unacceptable solution to use in **Mouse-Ka-Mania**. The AES function for reading the mouse, graf_mkstate, is sluggish in comparison to VDI's vq_mouse.

In the **Mouse-Ka-Mania** editor, VDI routines are used to track the X/Y position of the mouse and read the state of the mouse buttons. The AES form_do call, which manages input in dialog boxes, is *not* used. But, when you save or load a mouse shape, I call the GEM file selector, which uses AES input routines. When the AES took over at this point in early versions of **Mouse-Ka-Mania**, strange things would happen. Files would be selected by accident, or the console bell would ring endlessly as the stupid (pardon me; it's really just ignorant) AES thought the mouse button was being pressed outside the file selector's dialog box area. This drove me to distraction, and after *quite* a bit of experimentation, I came up with old Doc Johnson's prescription for VDI/AES mouse button bleed-throughs.

Before doing any VDI mouse input, use the VDI call vex_butv (exchange button vector) to point the system's mouse button interrupt routine to an RTS, and save the original address of the button routine. While this vector points at the RTS, the AES remains blissfully ignorant of any and all mouse button presses. What's more, the VDI function for checking the mouse position and buttons (vq_mouse) still works! Then, before calling any AES functions which read the mouse (like form_do, fsel_input, graf_mkstate, or evnt_button, just to name a few), replace the original button vector that was saved after the first vex_butv call Then the AES input functions work fine again, with no button bleed-throughs. I've run a lot of tests, and, so far, this fix has worked 100 percent of the time.

Mega-mania.

I recently had the chance to test an early version of **Mouse-Ka-Mania** on a new Mega ST4 (the 4-meg model). Of course, it didn't work. In the original version, I was using some undocumented memory locations which changed in the new TOS ROMs. Luckily, I was able to spend enough time with the new machine to analyze the problems, and the fixes I came up with will work on both of the current versions of ROM TOS. (**Mouse-Ka-Mania** doesn't use undocumented variables anymore, except when searching for them.)

Enjoy **Mouse-Ka-Mania**—you'll never have to look at that drab, dull arrow pointer again! //

Charles F. Johnson is a professional musician and, now, a semi-professional computer programmer/reviewer/author. He lives in Los Angeles with his wife Patty and Spike, the world's most intelligent (and gluttonous) cat. Charles is a SYSOP on the Analog Publishing Atari SIG on Delphi; his user name is CFJ.

Mice-Conceptions

by Maurice Molyneaux

It was a dreary morning, and I woke up feeling like a disk that had spent its life in a Commodore 1541 disk drive. I was in a foul mood and looking for something to complain about. The easiest target was my faithful ST. I thumbed my nose at it, angry because I couldn't design my own mouse pointers, as Amiga users can. BASIC programs to do this are useless, because the new pointer will not remain installed when you leave ST BASIC. Drat! Black arrow again!

Not being a master programmer, I couldn't write such a utility myself. I called **ST-Log** and told Clayton Walnum I'd like a mouse redesigner that would work from the desktop; no stupid ST BASIC programs! Clay gave my suggestion to Charles Johnson, and the rest, as Charles said, is history. Oh, the idea was mine, but the product is solely the mouseterwork of CFJ. Make no mice-take about it, this is his baby. Me? I was just the midwife (so to speak).

In addition to the "Mouse-Ka-Mania" program, this month's disk version of **ST-Log** contains a file of mouse pointer shapes, if you don't want to design your own. If you don't know how to use the accessory yet, read Charles's "Mouse-Ka-Mania." When you're done, I'll give you some pointers on mouse-terful mice design.

Warning: the MICE.ARC file is only a few K in size, but note that, when you de-arc it, despite the fact that each pointer is only 74 bytes, each will effectively consume 1K of disk space. This is because the smallest amount of space a file can be under GEMDOS is 2 physical disk sectors of 512 bytes each, or 1K. When de-arced, the total number of mouse data files will take up that much disk space, so de-arc the file on another disk if you haven't room on your current one.

Some practical limitations first. The pointer can be no larger than 16x16 pixels. You're better off not trying anything too complex, but you can draw simple shapes with

reasonable detail. Hands, feet, simple faces, bizarre arrows, insects, little mice (of course), and all manner of other things.

Second to consider: the part of the shape you actually point with, the "hot spot," should be fairly obvious. The hot spot on an arrow-shaped pointer should be at the tip, *not* the tail end. Fingers should follow the same rule. The ST's bumblebee pointer has the hot spot at its center; hardly an obvious location.

For the modified bee (included in the MICE.ARC file), I moved the hot spot to the front of its head. Cross hair or box cursors usually have a hot spot in their center area. If you create a pointer like this one, it's a good idea to leave a "hole"—in both the data and mask—around the hot spot, through which you can see what it is you're pointing to.

If you're making initials, a word, or any shape that doesn't have a spot that could clearly and obviously be made the hot spot, you might consider moving it as far as you can to one edge of the grid, and drawing a small "+" cross hair in the empty space and make the hotspot the center of it.

Third, keep your pointer simple. It's going to be small, so keep clarity in mind. To keep details like the legs on the ant pointer clear, I intentionally didn't connect them to the body. This separation makes it obvious they are legs.

On most pointers, the mask is 1 pixel wider around the edges than the data, allowing the data to be seen when it moves over a part of the screen of the same color. (Note the mask around the usual arrow; with it, you can see the black arrow against black text, etc.)

The mask can also be used to fill in the parts of the pointer that aren't filled with the data color (as with the white spots in the bee). By leaving holes in both the data and mask, you can create see-through parts in your pointers. This simple effect should *not* be underestimated. Load and install MAGGLASS as your pointer, and move it back and forth over some text. Notice that you can see through the "lens," and it almost looks as if the image is indeed magnified. This interesting trick is accomplished by carefully choosing which parts of the pointer to make transparent. I didn't make the whole lens see-through, because that spoiled the effect.

Furthermore, you can use the mask to "hide" little details in your pointer. Install the SC1224 pointer and move it over white parts of the screen. Nothing special, huh? Okay, move it in front of the black viewing square on the accessory. See, the letters ST are on the screen. I drew the letters with the mask color and left the rest of the monitor "screen" see-through. In some cases, you can avoid making the mask bigger than the data, but this is dependent on the pointer shape and which resolution it will be used in.

Each of the ST's three display modes has its advantages and disadvantages. In low resolution, it's easy to make a pointer without an overlapping mask. You can even make a pointer that consists only of the data color or the mask color alone, if you carefully select one of the sixteen colors that isn't normally used by the desktop or other programs. Using color fourteen for the data with no mask will be perfectly visible on the default desktop.

You can also use the mask color like a second data color. If you run GEM programs that change the color registers you've assigned your mouse pointer, then you might have a few problems. I like to install the finger pointer, using color fourteen for the mask and fifteen for the data, making fourteen a flesh tone and fifteen a darker hue of the same color. This is a neat-looking, humanlike hand floating about on the desktop! (Note: if you change some of the default colors using the control panel, you'll have to make sure the panel accessory is present whenever you boot, or the system will go back to default colors.)

This won't work well in medium resolution, where you only have four colors. In medium resolution, it's best to use a color like black for the data and make your mask color three—which isn't used by the desktop—and change that to whatever tone you wish. Medium resolution presents its own proportional problems, because the pointer will be just as tall as in low resolution, but half as wide.

Many times, a pointer that looks fine in low or high resolution looks weird in medium. But you can take advantage of this "stretched" look. For the K'tinga pointer (a Klingon battle cruiser to you Earthings), the taller-than-wide ratio of medium resolution makes the pointer look more like the long-necked ship it's supposed to represent. It doesn't look quite so hot in the other resolutions.

High resolution presents its own problems. You really can't make a clear pointer if you don't use both the data and mask. This is because everything in monochrome is made up of pixels of black and white. The gray desktop is just a pattern of alternating black-and-white dots. To draw a black mouse with no white mask would result in the pointer blending into the background at the edges when on the desktop, disappearing completely in front of anything black, but looking sharp in front of white. If you have a solid, single-color pointer, it's a good idea to make the mask just slightly bigger than the data, so it will show up against any conceivable black-and-white combination.

If you plan to design a pointer to be used in all resolutions, I suggest you give it an overlapping mask, and make it a shape that will look good in all resolutions. (Remember, it "stretches" in medium resolution.) Plus, you should design it for use in black and white, even though you might change its colors in low or medium resolution.

Now, go out there and start mouse-ing around. I expect some real mouse-terpieces from you mice-creants! Don't mice-understand me: I'm only engaging in some mouse-chief. And, if you can read aloud this mice-ellaneous act of writer's mice-conduct, then, truly, you have really said a mouse-ful!

No doubt, I'll be mice-quoted . . . ///

Allergic to all things Commodore, Maurice Molyneaux is an author/artist who—when not writing for ***ST-Log****—continues to struggle with a recalcitrant 8-year-old science fiction novel, paints, illustrates and uses his ST for "every conceivable task." His interests include classic cel animation as well as the computer variety, and he draws the meanest "Star Trek" pictures on microcomputers. His Delphi username is MAURICEM.*

C-manship

Those mysterious rectangles.

by Clayton Walnum

Those of you who read the last installment of **C-manship** (that's all of you, right?) are no doubt a little—or, more likely, *a lot*—perplexed about all this rectangle business. Don't feel bad. Not only is it a complex topic, but it's also virtually impossible to find complete documentation on it anywhere. Most of the books I've seen merely gloss over the subject, as if the reader were born with an intimate knowledge of GEM's rectangle list.

Well, friends and neighbors, I, for one, was not born with that knowledge. I've spent the last couple of months in research, trying to dig out all the facts I could about rectangle lists, not only because I wanted to clarify the issue for myself, but because I wanted to put together a decent tutorial to help you, the reader, understand this mysterious process.

The typing part.

This month's demo program can be found in Listing 1. You should now type it in and compile it, since all the following discussion will be based on it. The functions main(), open_vwork(), do_move() and set_clip() are identical to the functions of the same name in issue 16's listing. If you have that listing, you don't need to retype the above functions; just copy them in from the old listing. Note that the program was developed using the Megamax C compiler. If you have a different compiler, you may need to make some modifications.

You can also find the listing on this month's disk version and in the Atari SIG on Delphi.

Rectangles revealed.

I recently spoke with Frank Cohen of Regent Software, who told me that one method he used to sort out this rectangle nonsense was to update each rectangle returned from the redraw message with a different fill pattern. I told him I thought that was sheer genius (well, slightly clever, anyway) and as soon as I hung up the phone, I set about stealing his idea.

Steal it, I did (with his blessings, I hope). The demo program uses this method to graphically illustrate the process of walking the rectangle list. The figures on the following pages take you step by step through the tutorial here. These screens were taken from a monochrome monitor, so if you have a color system, you may get slightly different results.

The tutorial.

When you run the program, you'll first see the screen shown in Figure 1. Three windows have been opened, and since the windows need to have their work areas drawn, GEM has sent the program three redraw messages—one for each window. Because I inserted a call to Cconin() in the rectangle list processing loop, nothing further will happen until you press RETURN.

But before we let GEM do its thing, take a look at the top line of the screen. Here, you'll see the handle of the window for which the redraw message was sent and the number of the rectangle from the rectangle list we're currently working on. In Figure 1, we see that window 1 is waiting for a redraw and that we're about to process rectangle 1 from the list.

We can't blindly go ahead and fill in window 1's workspace, because part of that space is covered by the other two windows; we don't want to erase them. For this reason, when GEM created the redraw message for window 1, it took that window's workspace (only those areas *not* covered by the other two windows) and divided it into a series of non-overlapping rectangles. It then loaded the coordinates and sizes of those rectangles into the rectangle list, where we can get at them for processing.

Now press RETURN once. The first rectangle from the list will be processed, the screen will be updated, and the

program once again waits for a key press. But before we give it that key press, let's take a closer look at what happened with that first rectangle.

If you think back to the last **C-manship**, you'll remember that, when processing the rectangle list, we're always dealing with two rectangles: one returned in the redraw message and one retrieved from the list. In the case of the rectangle we just processed, the rectangle received from the redraw message was window 1's complete work area. The rectangle returned from the list was the one you just saw filled in (the light gray rectangle at the top of Figure 2). The first thing we had to do when we got these rectangles was check if they overlapped, with this call:

```
rc_intersect ( rec1, rec2 );
```

What I didn't mention last time was that, after the call, the rectangle found in rec2 may not be the same one we started with; it'll actually be a rectangle representing the intersection of the two original rectangles.

Now, do our first two rectangles intersect? Sure enough! We know we've found an area that must be updated, and we send the rectangle found in rec2 to draw_interior(), our actual drawing routine. Note that, in this case, the rectangle returned in rec2 was the same rectangle we started out with, because its entire area is in intersection with the one returned in the redraw message.

Look at the information at the top of the screen. We're now ready to process rectangle 2 from window 1's list. Press RETURN, and this area will be updated. Keep pressing RETURN, filling in each of window 1's rectangles, until you hear the computer's bell ring, indicating you've reached the end of the rectangle list and completed the processing of the first redraw message.

Now, the information at the top of the screen will show that we're about to process a redraw message for window 2. Before you press RETURN, see if you can figure out how many rectangles it'll take to do the job. (Hint: nothing should be drawn in window 3's workspace.) Figured it out? Everyone who guessed that we need to update two rectangles may look into a mirror and tell himself how clever he is. Press the RETURN key twice to update window 2. The bell should ring after the second press, leaving us ready to process the redraw message for window 3.

How many rectangles for window 3? One. Window 3 is the topmost window, so has nothing covering it; we need only fill in the entire work area.

Now your screen should look similar to Figure 2. Grab window 3 by the mover bar, and drag it into the upper right corner of the screen, as shown in Figure 3. Notice that even though the image of the old window 3 is still on the screen, we didn't have to redraw the window in its new position. Since its complete image was on the screen, GEM did the job for us, blitting the image to its new location.

After blitting the window image, GEM then redrew window 2, *not including* the workspace. Why not the workspace, too? Before we moved window 3, window 2 was partially hidden, so *after* the move, GEM didn't know what we want-

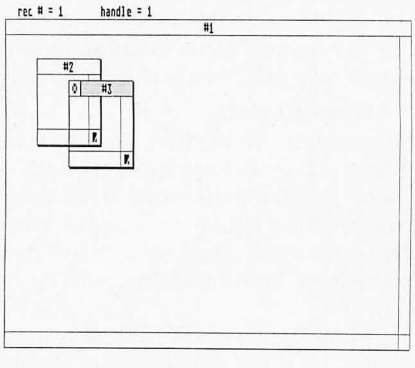


Figure 1.

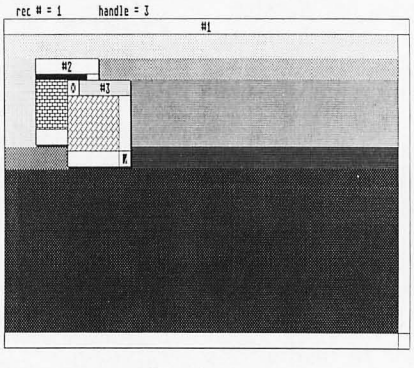


Figure 2.

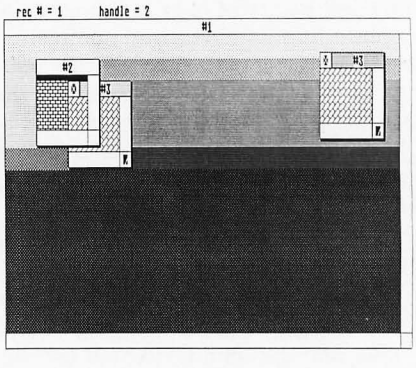


Figure 3.

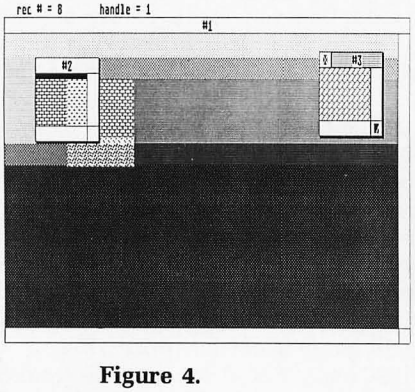


Figure 4.

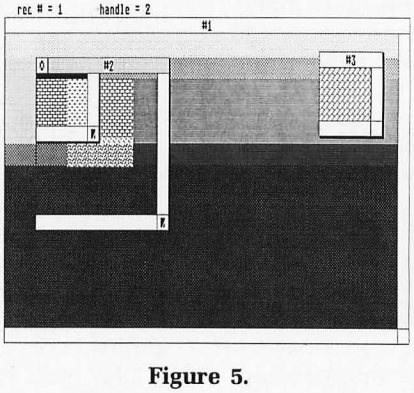


Figure 5.

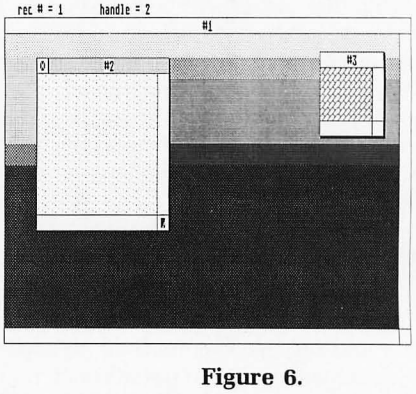


Figure 6.

ed to put in the uncovered area. It happily dropped the whole mess—including a rectangle list—into our laps. Our status line at the top of the screen now tells us that we've received a redraw message for window 2, and we're waiting to process rectangle 1.

Now think for a minute (aw, come on, it won't hurt much). What portion of window 2 is going to be redrawn when we press the RETURN key? Any guesses? The entire work area, you say? Wrong! Why take the time to update the entire work area, when only a small section—the area covered by window 3—needs it? Because it's easy? Well, yes, but it's just as simple to do it right, because, after our call to rc_intersect, we have the area of intersection—the exact rectangle we need—in rec2. Press RETURN to see this rectangle get its due.

Since GEM wants to get rid of the rest of window 3's old image, we now have a redraw message for window 1. Press RETURN. Hmmm. Nothing happened. If you look at the rectangle number in our status display, though, you'll see that something *did* happen, because we're now on rectangle 2. So what happened to rectangle 1? We processed it just like all the others, but because it didn't intersect the dirty area (the rectangle returned in the redraw message), we skipped over it. If you look at the screen, you can see *that* rectangle didn't need to be redrawn.

Continue pressing RETURN until the bell rings, watching to see which rectangles are redrawn and which are skipped. Your screen should end up resembling Figure 4.

As you may recall, another way to generate a redraw message is to increase the size of a window. Grab window 2's sizer button (gently, we don't want any bruising) and enlarge it as shown in Figure 5. Predictably, a redraw message for window 2 is sent. How many rectangles this time? Only one.

Sorry, but I guess it was a trick question. Go ahead and press RETURN. The entire work area of window 2 is redrawn, leaving your screen looking like Figure 6. Strange, considering I just said it was a waste of time to redraw portions of a window that didn't need it. That top left-hand corner didn't need to be redrawn, did it? The fact is, whenever you enlarge a window or make it uppermost, the rectangle returned in rec2 will be the entire work area. I never promised GEM was consistent.

Next step in our experimentation: move window 2 to the lower right corner of the screen, below window 3 (not overlapping), then press RETURN until the bell rings, watching as window 1 is updated. Figure 7 shows the results. Now click the mouse pointer on window 1's workspace, making it uppermost. Press RETURN and your screen will look like Figure 8. Using window 1's sizer button, reduce the window to its smallest size, as shown in Figure 9.

Note that, when we reduced window 1, GEM cleaned up the desktop on its own, leaving only the work areas of windows 2 and 3 for us to worry about. As they stand now, both windows 2 and 3 contain information left behind by window 1. Press RETURN twice to update both windows.

Move window 1 around the desktop, without overlapping any of the other windows or going off screen. We get no

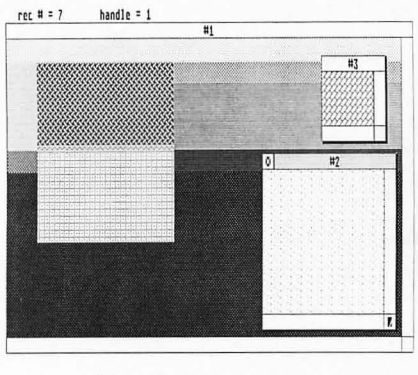


Figure 7.

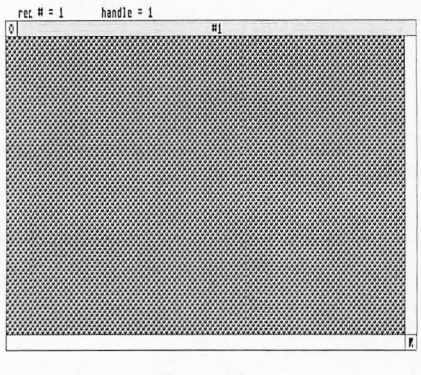


Figure 8.

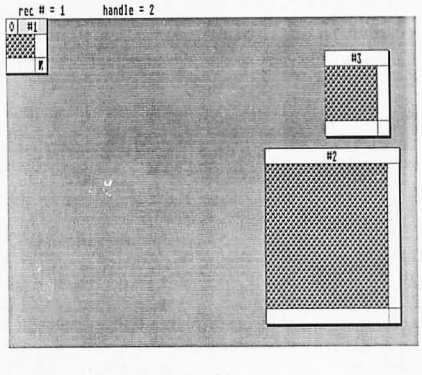


Figure 9.

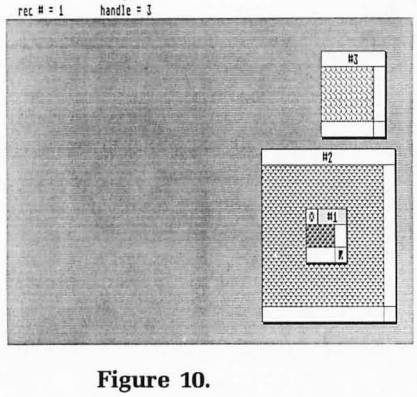


Figure 10.

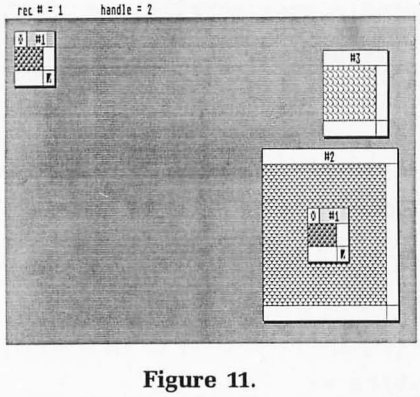


Figure 11.

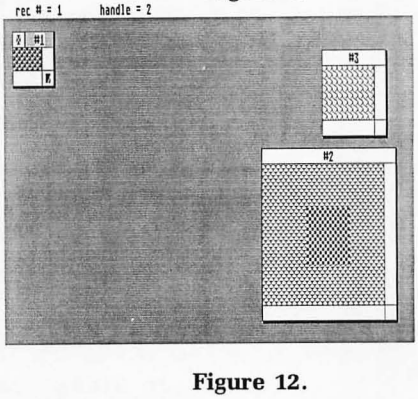


Figure 12.

redraw messages. GEM is delighted to blit window 1 from one place to another with no help from us.

Now place window 1 in the center of window 2, as shown in Figure 10. Still no redraw messages. GEM blitted the window to its new location then erased the old image, just like before. Windows 2 and 3 don't get redraw messages because none of their *visible* data has been corrupted. Sure, we covered some of it up, but that's not a problem until we move window 1 out of the way again. Do so now, as shown in Figure 11. Whoops! GEM did the blit all right, but it left a mess behind. Press RETURN to conclude this fascinating journey through GEM's rectangle lists, leaving the screen shown in Figure 12.

Out of the fog.

I hope the above experiments have taken some of the mystery out of GEM's rectangles and how the system works. Play around with the program all you want, moving and changing windows, all the while watching to see how GEM sets up its rectangle list and how the program processes it. There are an infinite number of possibilities. It'll be a *long* time before you exhaust them.

If you want the program to run without waiting for a key press, remove the call to Cconin() found in the function do_redraw(). The information printed at the top of the screen won't do you much good if you do, though. You'll never be able to read it as fast as our program can update those rectangles.

Sidelines.

Before we close up shop for this month, there are a couple of things in Listing 1 we ought to go over.

This is the first time we've handled the *WM_TOPPED* message in a program. We get this message from GEM whenever the user clicks the mouse over an inactive window. All it takes to "top" the window, make it uppermost and active, is the call:

```
wind_set (msg_buf[3], WF_TOP, 0, 0);
```

Here, msg_buf[3] is, as usual, the window's handle; WF_TOP is the function we want wind_set() to perform, defined in GEMDEFS.H as *10*; and the two zeros are dummy arguments.

Another nuance worthy of note is the way we're using arrays to cut down the window handling code in the program. Wherever we perform the same function concurrently on all windows, we can use a FOR loop, the control variable of which becomes an index into an array (one element for each window).

If you look at the function open_window() in Listing 1, you'll see the loop that opens the windows and places their handles into the array w_h[]. We need only one set of wind_create() and wind_set() calls. However, when we actually open the windows, we use a separate wind_open() call for each window. Why? Because each window is opened at its own set of coordinates, and for the sake of clarity, I decided to "hard code" those coordinates into the function calls, rather than use arrays.

Another day, another dollar.

Next month our subject will be. . .Yes, you guessed it! Windows, again. Maybe we'll figure out how to use those sliders and arrows to change the contents of a window's workspace. Sounds like a good idea to me. How about you? ///

Listing 1.
C listing.

```
/*************************************************/
/*            C-manship, Listing 1               */
/*                 ST-Log #18                    */
/*          Developed with Megamax C             */
/*************************************************/

#include <gemdefs.h>
#include <obdefs.h>
#include <osbind.h>

#define TRUE  1
#define FALSE 0
#define PARTS NAME|CLOSER|MOVER|SIZER
#define MIN_WIDTH 64
#define MIN_HEIGHT 64
#define PATTERN 2
#define BELL 7
#define HIGH 2

/* 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, fullx, fully, fullw, fullh, wrkx, wrky,
```

```
    wrkw, wrkh, res, char_w, char_h, box_w, box_h, fill;

int msg_buf[8];
int w_h[3];

char *titles[] = {"#1", "#2", "#3"};

main ()
{
   appl_init ();        /* Initialize application.        */
   open_vwork ();       /* Set up workstation.            */
   do_wndw();           /* Go do the window stuff.        */
   v_clsvwk (handle);   /* Close virtual workstation.     */
   appl_exit ();        /* Back to the desktop.           */
}


open_vwork ()
{
   int i;

   /* Get graphics handle, initialize the GEM arrays and open  */
   /* a virtual workstation.                                   */

   handle = graf_handle ( &char_w, &char_h, &box_w, &box_h);
   for ( i=0; i<10; work_in[i++] = 1 );
   work_in[10] = 2;
   v_opnvwk ( work_in, &handle, work_out );
}


do_wndw ()
{
   /* Clear screen. */
   graf_mouse ( M_OFF, 0L );
   v_clrwk ( handle );
   graf_mouse ( M_ON, 0L );

   /* Find screen resolution. */
   res = Getrez();

   /* Initialize and open our windows. */
   open_window();

   /* Change mouse to arrow, and initialize fill style. */
   graf_mouse ( ARROW, 0L );
   fill = 0;

   /* Receive event messages until the window closer is clicked. */
   do {
      evnt_mesag ( msg_buf );
      switch ( msg_buf[0] ) { /* msg_buf[0] is message type. */

         case WM_MOVED:
         case WM_SIZED:
            do_move();
            break;

         case WM_TOPPED:
            wind_set ( msg_buf[3], WF_TOP, 0, 0 );
            break;

         case WM_REDRAW:
            do_redraw ( (GRECT *) &msg_buf[4] );
            break;
      }
   }
   while ( msg_buf[0] != WM_CLOSED );

   /* Close and delete the windows. */
   close_window();
}


do_move()
{
   /* Set window at new location.  Also disallow any      */
```

```
   /* window sizes less than our minimum allowable size.  */

   if ( msg_buf[6] < MIN_WIDTH )
      msg_buf[6] = MIN_WIDTH;
   if ( msg_buf[7] < MIN_HEIGHT )
      msg_buf[7] = MIN_HEIGHT;
   wind_set ( msg_buf[3], WF_CURRXYWH,
              msg_buf[4], msg_buf[5], msg_buf[6], msg_buf[7] );
}


draw_interior ( clip )
GRECT clip;
{
   int pxy[4], y, x;

   /* Turn mouse off prior to drawing. */
   graf_mouse ( M_OFF, 0L );

   /* Calculate clip rectangle and turn clipping on. */
   set_clip ( TRUE, clip );

   /* Get coordinates of window's work rectangle. */
   wind_get ( msg_buf[3], WF_WORKXYWH, &wrkx, &wrky, &wrkw, &wrkh );

   /* Set the color and fill style. */
   vsf_interior ( handle, PATTERN );
   fill += 1;
   if ( fill>24 )
      fill = 1;
   vsf_style ( handle, fill );
   vsf_color ( handle, BLACK );

   /* Draw the background in the window's work area. */
   pxy[0] = wrkx;
   pxy[1] = wrky;
   pxy[2] = wrkx + wrkw - 1;
   pxy[3] = wrky + wrkh - 1;
   vr_recfl ( handle, pxy );

   /* Drawing over, so turn the clipping */
   /* off and turn the mouse back on.    */
   set_clip ( FALSE, clip );
   graf_mouse ( M_ON, 0L );
}


do_redraw ( rec1 )
GRECT *rec1;
{
   GRECT rec2;
   int rec_cnt, y;

   /* Init rectangle count, and set y coord for text. */
   rec_cnt = 0;
   if ( res == HIGH )
      y = 15;
   else
      y = 8;

   /* Lock screen for update. */
   wind_update ( BEG_UPDATE );
   /* Get first rectangle from list. */
   wind_get ( msg_buf[3], WF_FIRSTXYWH,
              &rec2.g_x, &rec2.g_y, &rec2.g_w, &rec2.g_h );

   /* Loop through entire rectangle list, */
   /* redrawing where necessary.          */
   while ( rec2.g_w && rec2.g_h ) {
      test_print ( "handle", msg_buf[3], 150, y );
      rec_cnt += 1;
      test_print ( "rec #", rec_cnt, 20, y );
      Cconin();
      if ( rc_intersect ( rec1, &rec2 ) )
         draw_interior ( rec2 );
      wind_get ( msg_buf[3], WF_NEXTXYWH,
```

```
                &rec2.g_x, &rec2.g_y, &rec2.g_w, &rec2.g_h );
   }
   /* Unlock screen after update. */
   wind_update ( END_UPDATE );
   Cconout ( BELL );
}


set_clip ( flag, rec )
int flag;
GRECT rec;
{
   int pxy[4];

   /* Convert rectangle to pxy coords. */
   pxy[0] = rec.g_x;
   pxy[1] = rec.g_y;
   pxy[2] = rec.g_x + rec.g_w - 1;
   pxy[3] = rec.g_y + rec.g_h - 1;

   /* Turn clipping on or off. */
   vs_clip ( handle, flag, pxy );
}


open_window()
{
   int x;

   /* Find the size of the desktop's work area. */
   wind_get ( 0, WF_WORKXYWH, &fullx, &fully, &fullw, &fullh );

   /* Create the windows. */
   for ( x=0; x<3; ++x ) {
      w_h[x] = wind_create ( PARTS, fullx, fully, fullw, fullh );
      wind_set ( w_h[x], WF_NAME, titles[x], 0, 0 );
   }

   /* Draw the windows. */
   wind_open ( w_h[0], fullx, fully, fullw, fullh );
   wind_open ( w_h[1], 50, 65, 100, 100 );
   wind_open ( w_h[2], 100, 90, 100, 100 );
}


close_window ()
{
   int x;

   /* Close and delete the windows. */
   for ( x=0; x<3; ++x ) {
      wind_close ( w_h[x] );
      wind_delete ( w_h[x] );
   }
}


test_print ( label, number, x, y )
int number, x, y;
char *label;
{
   char t[100];

   sprintf ( t ,"%s = %d%s", label, number, "     " );
   v_gtext ( handle, x, y, t );
}
●
```

COMPUTER GAMES +

(714) 639-8189

VIP PROFESSIONAL
$99.95 PRO
$119.95 GEM

520ST
COMPLETE SYSTEM
Monitor, disk drive, computer
COLOR $679.95
B & W $549.95

COLOR MONITOR	299.95
MONO	169.95
SINGLE SIDED DRIVE	129.95
DOUBLE SIDED DRIVE	199.95
20 MEG HARD DISK	529.95
COMPUTER & DRIVE	399.95

ASTRA'S RAM EASY
$124.95
1 Meg ST upgrade

DB MAN
$89.95
(Best ST Data Base)

NEW!
XE DISK DRIVE XF551
$179.95

GUARANTEED LOWEST PRICES — CALL FOR TITLES NOT LISTED

RECENT AND SOON TO BE RELEASED ST PROGRAMS

BREACH	27.95
BARBARIAN	27.95
EMPIRE	37.95
DEFENDER OF THE CROWN	32.95
MIDI MAZE	27.95
VIDEO VEGAS	24.95
PAPER BOY	27.95
SENTRY	29.95
FAERY TALE ADVENTURE	34.95
ALIEN FIRES	27.95
BARD'S TALE	32.95
F-15 STRIKE EAGLE	27.95
WIZARD'S CROWN	24.95
MARBLE MADNESS	24.95

REMARKABLE EUROPEAN ST GAMES

GAUNTLET	27.95
ROAD RUNNER	27.95
ARKANOID	24.95
XEVIOUS	29.95
METROCROSS	29.95
MOUSE TRAP	24.95
INDIANA JONES	29.95
SOLOMON'S KEY	29.95
EAGLE'S NEST	27.95
PROHIBITION	24.95
EXTENSOR	24.95
HADES NEBULA	24.95
LIBERATOR	22.95
TYPHOON	24.95
ALTAIR	24.95
CRAFTON & XUNK	24.95
TRACKER	29.95
SILICON DREAMS	24.95
JEWELS OF DARKNESS	24.95
KARATE MASTER	22.95
PASSENGERS IN THE WIND	29.95
KNIGHT ORC	29.95
ANNULS OF ROME	29.95

ADVENTURES

S.D.I.	32.95
ALTERNATE REALITY-CITY	24.95
ULTIMA II	38.95
ULTIMA III	38.95
ULTIMA IV	38.95
PHANTASIE	24.95
PHANTASIE II	24.95
PHANTASIE III	24.95
MERCENARY	26.95
MERCENARY - 2nd CITY	14.95
THE PAWN	29.95
BORROWED TIME	29.95
MINDSHADOW	29.95
SPIDERMAN	9.95
UNIVERSE II	34.95
SUNDOG	24.95
GUILD OF THIEVES	29.95
BRATACUS	32.95

ARCADE / ACTION

WAR ZONE & FIRE BLASTER	27.95
SPACE STATION & PROTECTOR	27.95
WANDERER (3D!)	25.95
STAR GLIDER	29.95
DEEP SPACE	19.95
SUPER CYCLE	14.95
MAJOR MOTION	24.95
TIME BANDIT	24.95
STWARS	24.95
PLUTOS	19.95
Q BALL	19.95
TRAIL BLAZER	27.95
AIR BALL	24.95

STRATEGY / SIMULATION

HEX	29.95
BALANCE OF POWER	32.95
COLONIAL CONQUEST	24.95
ROADWAR 2000	24.95
AUTODUEL	32.95
SKYFOX	14.95
ARCTIC FOX	27.95
FLIGHT SIMULATOR II	32.95
SUPER HUEY	25.95
GATO	32.95
SILENT SERVICE	24.95
SUB BATTLE	24.95
HACKER	17.95
HACKER II	24.95
SHANGHAI	24.95
CHESSMASTER 2000	29.95
BRIDGE 5.0	22.95

SPORTS

GRIDIRON FOOTBALL	32.95
MICROLEAGUE II - BASEBALL	39.95
CHAMP. BASEBALL	29.95
CHAMP. WRESTLING	29.95
CHAMP. BASKETBALL	24.95
ST KARATE	19.95
WORLD CHAMP. KARATE	29.95
KARATE KID II	24.95
WORLD GAMES	24.95
WINTER GAMES	24.95
LEADER BOARD	24.95
TOURNAMENT DISK	14.95
10th FRAME BOWLING	24.95

PRODUCTIVITY

FLASH BAK / FLASH CACHE	49.95
PUBLISHING PARTNER	94.95
EASY DRAW	64.95
GRAPHIC ARTIST	124.95
AEGIS ANIMATOR	48.95
ST TALK 2.0	19.95
PERSONAL PASCAL	49.95
GFA BASIC	49.95
GFA COMPILER	49.95
TYPING TUTOR	24.95
WORD WRITER ST	44.95
DATA MANAGER ST	44.95
SWIFT CALC ST	44.95
REGENT WORD II	48.95
REGENT BASE 1.1	79.95
DEGAS ELITE	39.95
N-VISION	29.95
PAINTWORKS	24.95
MUSIC STUDIO	32.95
PRINT MASTER PLUS	24.95
ART GALLERY I	19.95
ART GALLERY II	19.95
DAC EASY ACCOUNTING	44.95
WORD PERFECT	229.95
MAGIC SAC	99.95
PC DITTO	74.95
DESK CART	69.95

XE / XL SOFTWARE

GAUNTLET	24.95
DEEPER DUNGEONS	14.95
ARKANOID	24.95
HEAD OVER HEELS	22.95
DRUID	22.95
LIVING DAYLIGHTS	24.95
ALT REALITY-CITY	24.95
ALT REALITY-DUNGEON	24.95
UNIVERSE	34.95
BOUNTY BOB STRIKES BACK	29.95
RETURN OF HERACLES	9.95
ULTIMA I	39.95
ULTIMA III	19.95
ULTIMA IV	39.95
THE PAWN	27.95
WOMBATS I	29.95
INFILTRATOR	22.95
BOP & WRESTLE	22.95
ACE OF ACES	22.95
HARDBALL	22.95
FIGHT NIGHT	22.95
LEADER BOARD	27.95
PHANTASIE	27.95
FLIGHT SIMULATOR II	37.95
AUTO DUEL	37.95
PARTY QUIZ	14.95
BASIC XL	36.95
BASIC XE	46.95
ACTION!	46.95
MAC 65	46.95
TOOL KITS	18.95
WRITER'S TOOL	38.95
ATARI WRITER PLUS	39.95
TYPING TUTOR	17.95
SAT COMPLETE	27.95
PRINT SHOP	29.95
GRAPHICS I	14.95
GRAPHICS II	14.95
GRAPHICS III	14.95
SYNCALC or SYNFILE	32.95
VIDEO TITLE SHOP	22.95
DELUXE JOYSTICK (was $35)	9.95
ATARI JOYSTICKS	2 / 9.95

DISK DRIVES

ASTRA THE ONE	(XE/XL) 249.95
ASTRA 2001	(XE/XL) 279.95
ASTRA HD + (ST 20 meg & d/s drive)	849.95
INDUS GT	179.95
IB IBM ST DRIVE	219.95

PRINTER INTERFACES (XE / XL)

MICROPRINT	29.95
MPP-1150	39.95
PR CONNECTION	59.95

PRINTERS

STAR NX-10	169.95
STAR NB24-10	449.95

MODEMS

AVATEX 1200	89.95
AVATEX 1200HC	119.95
PRACTICAL 2400	199.95

COMPUTEREYES ST 179.95
IMAGE SCAN . . . 79.95

PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

ATARI

714-639-8189

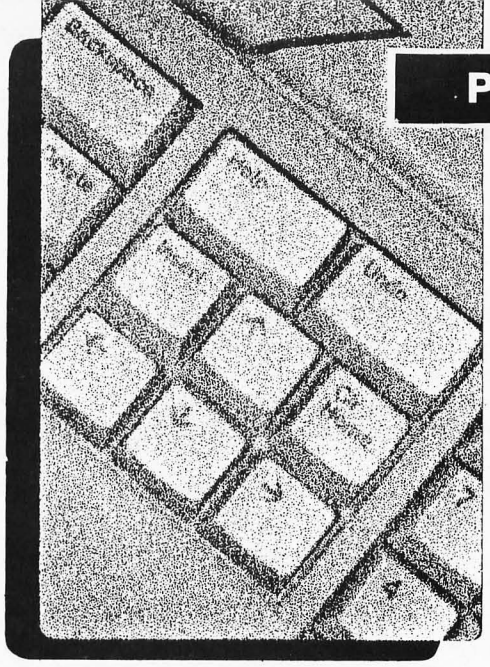
ORDERS ONLY PLEASE
1-800-443-8189

SHIPPING: Software - free shipping on U.S. orders over $100, otherwise $2.50 U.S., $6.50 outside U.S. Hardware - depends on weight, call for quote. Charge cards + 3%. C.O.D. orders are welcome, add $1.90 for UPS + 3%.

COMPUTER GAMES + • Box 6144 • ORANGE CA 92667 • (714) 639-8189

CIRCLE #109 ON READER SERVICE CARD

Assembly line



Inside subroutines.

by Douglas Weir

I mentioned last time that there's a way of looking at values on the stack without disturbing the stack itself. Doing this requires learning a new addressing mode, but it will be the next-to-last of all of them. This new mode, combined with a couple of other powerful instructions, will make it easy for us to write well-behaved, modular subroutines. We're now getting to a point where some of the things we do in assembly language can give us insight into how and why certain high-level languages—I'm thinking of C in particular—do some of the things they do.

This month's program, after you've assembled and linked it, will display four strings and quit. It's not a very exciting performance, but the simplicity of what's going on will let us concentrate on what's really important: the inner workings of a subroutine.

First, though, I'd like to tie up my discussion of loops from last time by introducing the 68000 instruction that is most often used in looping situations: dbra, or "decrement and branch."

A decremental journey.

In our last program, we wrote a loop used to scan through a number of keypress possibilities. The number of keys we were willing to consider was loaded into a register. Each time we went through the loop, we subtracted one (by means of a subq instruction) from the contents of this register. Then we tested the result of this subtraction with a bne instruction. If the count was down to zero, that meant the loop was finished; otherwise, we branched back to the top of the loop and went through the whole process again, and so on.

That's fine, but there's a quicker way of doing the same thing. The 68000 has a set of instructions specially designed to handle everything required to manage a loop. The instructions execute in two parts. Each begins by performing a test (i.e., looking at the state of one of the bits in the Condition Codes register). If the result of the test is "true," execution of the instruction ends and the 68000 proceeds to the next instruction. However, if the test result is "false," the second "half" of the instruction is executed. One (1) is subtracted from a specified data register (the first of two operands for this instruction), and now this result is tested.

If the contents of the register have been exhausted, the instruction ends. If the contents of the register have not been exhausted, the 68000 branches to a label (which is the second operand specified for this instruction).

The most-used of these instructions is, as I've mentioned, the dbra. Actually, its proper name is dbf: "if 'false' is not 'true,' decrement and branch." And what does that mean? Let's look at a simple example.

```
        move.w  #10,d0      load loop count
loop:   nop                 body of loop
        dbf     d0,loop     keep going...
        bra     out         till finished
```

We start by loading d0 with the immediate value *10*. (Note, by the way, that we're loading it as a word value.) The next instruction, which is labelled loop, does nothing: it's our old friend nop. Nevertheless, this is the body of our loop. The next instruction contains all the loop control code we need. First, it checks the specified condition. What's the condition? That's easy—it's false (that's what the f stands for).

So the 68000 goes through a chain of reasoning something like this: "Let's check the condition. Is false true? No, it's not. So I can't drop out of this instruction yet. Now I'll subtract one from the register d0. Ten minus one equals nine, so the loop counter isn't finished yet either. That means I'll have to take the branch . . . Let's see, I'm supposed to branch to loop. Here goes. What's this—a nop? Okay, you want nothing, you got nothing. Now here's a dbf. Let's check the condition. Is false true?. . ." And so on. As you can see, there's a philosophical side to assembly language.

The main point of dbf is that the initial condition always tests out false—false can never be true, after all. So the decrement is always performed, and the loop always continues until the loop counter register is exhausted (unless, of course, you insert some other, unrelated code in the middle of the loop that causes a branch). The dbf instruction, therefore, gives you something very much like a FOR loop in BASIC:

```
for i = 10 to -1 step -1
print "howdy"
next i
```

or Pascal:

```
for i:=10 downto -1 do
writeln('howdy');
```

or C:

```
for (i =10; i > -1; i--)
printf("howdy\n");
```

With dbf, it's almost as if the first test isn't there—only the "decrement and branch" part of the instruction is in operation. For this reason, most 68000 assemblers accept the alternate form, dbra (i.e., decrement and branch—for the strictly correct dbf—until false is true, decrement and branch).

This may seem like a lot of theoretical rigamarole to go through just for an automatic loop counter instruction. But consider one of dbra's cousins, dbeq:

```
        move.b   #key, d0    char we're looking for
        move.w   #10,d1      loop count
        movea.l  #table,a0   base of table
loop:   cmp.b    (a0)+,d0    check char in table
        dbeq     d1,loop     if not found, continue loop
        nop                  do nothing
```

The first instruction loads a byte value into d0. Let's assume it's an ASCII code we'll be searching for. (We'll also assume that key is defined with an equ directive elsewhere in the program.) The next instruction loads the loop counter with the immediate value *10*. Then the base address of a table of byte values (also declared elsewhere) is loaded into register a0, and we're ready to start the loop.

This time there's no nop. Instead, we test whether the byte value, whose address is currently in a0, is equal to the byte value in d0. This is done with a cmp instruction (see issue 16's **Assembly line** for a detailed explanation of cmp). If the two values are equal, then the zero bit in the Condition Codes register will be set to *1* when the instruction ends.

It's precisely this bit that dbeq tests first (*eq* in opcodes usually stands for an "equal" condition—in other words, is the zero bit set?—meaning that two values were just compared and were found equal). The 68000 reasons it out like this: "Hmm, a dbeq. Let's check the condition. Is the zero bit set? No, it's not (that means the last two values compared were not equal). Okay, so I'll do the decrement. Ten minus one equals nine. There's still plenty left in the register. So I'll do the branch. Right—back to loop. Get the byte a0 is pointing to. Okay, got it. (Don't forget to increment a0.) Now compare it to the byte in d0. Next instruction: here's a dbeq. Let's check the condition. Is the zero bit set? Yes, it is (that means the two values were equal). So that's that—I can forget about the rest of this instruction. Let's go to the next. What's this—a nop?. . ." Well, you get the idea.

Thus, an instruction like dbeq gives you a built-in compound condition for continuing the loop: keep looping until a match is found, or the loop counter is exhausted:

```
for (i = 10, match = FALSE; !match && i > -1; i++)
match = table[i] == KEY;
```

You will find that dbra is by far the most-used member of this group of instructions (they can be found under the general heading dbcc in the Motorola manual), however, you should not forget the extra efficiency offered by the others.

The minus syndrome.

Now let's go back and tidy up a couple of details. No doubt you've noticed that in the high-level language versions given, the loop counts down not to *0*, but to *-1*. In fact, that's exactly how the dbcc instructions work. They test for -1, not 0, in the specified counter register. In other words, loops controlled by these instructions will always execute one iteration more than the value initially loaded into the register. Our little test loop above will execute *11*, not *10* times, as written. In order to get around this quirk, you must either use loop count values that are always one less than they should be (e.g., *9* if you want the loop to execute *10* times, etc.), or do something like the following:

```
        move.w   #10,d0      load loop count
        bra      test        now start
loop:   nop                  do nothing
test:   dbra     d0,loop     keep going. . .
        bra      out         till finished
```

This is the same code as before, except the dbra instruction is labelled and a branch to this label is inserted to kick off the loop. The result is that the "extra" decrement-and-branch occurs before the loop really starts, and the body of the loop is only executed 10 times after all.

This is a favorite trick of mine, but there's a drawback: you waste a little time when you do things this way. One unnecessary bra and dbra are performed for every loop (not every iteration)! There are two schools of thought: assembly language is so fast that you usually can't tell the difference in these situations, but then many people go to the trouble of writing assembly language just to get away from this sort of redundancy (which is what compilers are past masters at). And, there are time-critical applications where the extra code does make a difference.

I'm in favor of a little redundancy in the interest of clarity and consistency. By doing loops this way—usually—I can use loop count constants which *say* what they *mean*. Whichever method you choose, you should try to be consistent, yet never forget that there's always another way of getting the job done.

It's important to note that a word value is loaded into the loop count register. That's because the dbcc instructions decrement and check the register as a *word*, not as a *longword* or a *byte*. (This means, by the way, that a loop can only execute a maximum of 32768 iterations using these instructions.) Once you've written a mistake like this into a program, it can be very difficult to find.

Finally, notice what a powerful instruction dbeq (for example) is. Using it, we can write five lines of assembly code that are practically equivalent to two fairly dense lines of C code—not a bad ratio, considering the difference in language levels.

The plural of move is movem.

This month's other new instruction, movem, is not nearly so complicated. So far we've been using the stack to pass arguments to GEMDOS routines (via trap #1), and (implicitly) to save subroutine return addresses. We haven't saved any register values on the stack. This is easy to do, however, and is one of the stack's main uses. For example, we could write:

```
move.l   d0,-(a7)          save d0
move.l   a0,-(a7)          save a0
move.l   #message,-(a7)    get string address
move.w   #9,-(a7)          code=display string
trap     #1                do it
addq.l   #6,a7             pop arguments
movea.l  (a7)+,a0          restore a0
move.l   (a7)+,d0          restore d0
```

This bit of code would simply use GEMDOS function number 9 to display a string (labelled message), similarly to the sort of thing we've done more than once before. However, the current values of registers d0 and a0 are saved before the trap and restored afterward. If GEMDOS uses these two registers, it won't make any difference to us—we'll still have their current contents when the saved values are popped from the stack. We just have to make sure that we push and pop the values in exact opposite order.

Saving and restoring registers is such a common use for the stack that a compound form of this instruction has been included in the 68000 set: movem, for "move multiple." With one execution of movem, you can save one or all of the data and address registers (including a7) in any combination on the stack. With another execution, you can restore all these values. Using movem, the above code fragment would look like this:

```
movem.l  a0/d0,-(a7)       move 'em out
move.l   #message,-(a7)    get string address
move.w   #9,-(a7)          code=display string
trap     #1                do it
addq.l   #6,a7             pop args
movem.l  (a7)+,a0/d0       move 'em back in
```

Suppose you wanted to save more registers:

```
movem.l  a0-a2/a3-a4/a6/d0-d4,-(a7)  save 'em
(. . .do something here. . .)
movem.l  (a7)+,a6/a3-a4/d0-d4/a0-a2  get 'em
```

A sequence of registers to be saved is indicated by a dash: a0-a2 means a0, a1, a2. Groups of sequences are separated by slashes. A group can consist of a single register. (Note that you don't have to worry about the order in which you name registers.) The assembler, when generating the machine code for movem, generates a second word of code that consists entirely of bit-flags representing the registers. Each register has its own bit. If it's set, the register is pushed (or popped); otherwise, the register is left alone. The order of pushing and popping is always the same, and isn't dependent on the order in which you happen to write the registers.

By the way, if you're using the Motorola 68000 manual as a reference, you'll probably get very confused if you look at its little diagram of the format of a movem instruction (at the bottom of page B-71 in the 5th edition). The diagram seems to be saying that the instruction is only one word long, and that the word somehow does double duty as both a machine instruction and a set of bit-flags for the registers. That would be a truly remarkable feat of data packing. I had to look up movem in *Programming the 68000* by Steve Williams (Sybex) to make sure that the entire instruction (with the register list) is two, not one, words long. So, if you're interested in this sort of thing, be warned: the Motorola instruction format diagrams are not models of clarity.

A see-through address mode.

The last new topic I want to discuss this time is that promised address mode that will let us peek into a stack, without popping any items. Consider the situation immediately after we've jumped to a subroutine by executing a bsr instruction. What is the stack pointer pointing to? That's easy—the return address, a longword value. We know this because the return address was automatically pushed onto the stack by the bsr, and that was the last instruction executed.

Furthermore, we know that at anytime after entering a subroutine in this manner, the execution of an rts instruction will immediately return us to where we came from (as long as we haven't fooled with the stack in the meantime).

Suppose we wanted to move the return address into another register, without disturbing the stack. We already know how to do that:

```
move.l   (a7),d0           get return address
```

Register a7 is pointing to the value, so we simply use the Address Register Indirect mode to read it via a7 and move a copy of it into d0. This mode (which I sneaked into issue 16's program) is the same thing as Address Register Indirect With Predecrement or Postincrement, only without the decrement or increment. The value of the register is used as an address, and the value remains unchanged.

Now suppose a word value had been pushed on the stack just before the bsr was executed. At entry into our subroutine, a7 points to a longword that's the return address. The word value pushed before the bsr is therefore in the two bytes just after the return address. If we added 4 to the current value of a7, it would then be pointing to our word. But we don't want to alter the value of a7.

When you think about it, writing:

```
move.l   (a7),d0           get return address
```

. . .is just another way of saying:

```
move.l   0(a7),d0          get return address
```

. . .that is, we're adding a zero offset to a7 to get an "effective address" that's used to read a value. Can we use offsets other than zero?

```
move.w   4(a7),d1          get the word that was pushed
```

. . .yes, we can. In this line of code, the 68000 adds the offset value 4 to the value of a7, and uses this new effective address to read the word value. The value of a7 remains the same; it's still pointing to the return address, ready to execute an rts at any time. This new addressing mode is called Address Register Indirect With Displacement (the latter word is Motorola's term for offset).

The offset value is a signed 16-bit value, which means that it must be greater than or equal to -32768, and less than or equal to 32767. (The Megamax C compiler allocates space for global variables and strings using this mode. It equates the variable names to offset constants, then uses the offsets with an address register to reference the variables. Strings

get negative or positive offsets, and globals the other. . . I forget which. This explains the 32K-byte limit Megamax enforces on data segments.)

The term "effective address" is an important one. So far, we've been able to imagine that every address we've used in a program actually exists somewhere—either as a value in a register, or as a constant generated by the assembler from a label. This time, however, the address we use to read the pushed word is. . . nowhere. The 68000 takes the value in an address register, adds a specified offset value to it, then uses this new temporary address to access memory in some way. The result of adding the offset to the register value is the effective address of either the source or the destination operand (here it was the source). After the instruction is executed, the effective address goes away—just like the result of a cmp intruction. We'll be using the concept of effective addresses often in coming installments.

The program.

The program should now be very easy to understand. At the bottom is declared a table of four strings. Actually, it would be more meaningful to describe it as an array of strings. Above the strings are their addresses, arranged so that we can easily index through them with an address register.

The program begins by pushing the address of the table of string pointers on the stack, followed by the number of strings. Then we jump to the subroutine display.

The subroutine first pushes four registers (as longwords) onto the stack. That means the return address is now located 16 bytes beyond the current position of the stack pointer (a7), which was decremented by *16* to accommodate all the pushed data. The word item—count—that was pushed just before the bsr is located 4 bytes beyond the return address, making an offset total of *20* to be added to a7, if we are to read count. Again, strings, which was pushed before count, is 2 bytes beyond the latter (remember: count is only a word value), and an offset of *22* is therefore needed to access it.

Registers a1 and d1 are loaded with strings (the base of the string address table) and count, respectively. We don't have to worry about data that might be lurking in the high bytes of d1: since the dbra instruction only uses the low word of the specified register, we can ignore the high word.

The loop is straightforward enough. The dbra is branched to at the start, thus ensuring that the loop will execute only four, not five, times. Note that registers a1 and d1 are used in the loop—not a0 and d0. That's because GEMDOS calls can affect the contents of a0 and d0: the routines use these registers, but (to save time) don't bother to preserve their old values.

When the loop ends, we restore the registers we saved at the beginning of the subroutine. After this is done, the return address is again the next item on the stack. Execution of rts pops it into the program counter, we return to the main program, and immediately execute GEMDOS call number 0: terminate program.

Next time we'll continue our discussion of subroutines with (among other things) one of the most complicated of all the 68000 instructions: link. //

Listing 1.
Assembly listing.

```
          text                              code segment
          move.l  #strings,-(a7)            get string addresses
          move.w  #count,-(a7)              get string count
          bsr     display                   go display strings
          addq.l  #6,a7                     pop args

          move.w  #0,-(a7)                  code=exit program
          trap    #1                        do it

display:
          movem.l a0-a1/d0-d1,-(a7)         save registers
          movea.l 22(a7),a1                 get base string array
          move.w  20(a7),d1                 get count
          bra     d_test                    now start
d_loop:
          move.l  (a1)+,-(a7)               get string
          move.w  #9,-(a7)                  code=display string
          trap    #1                        do it
          addq.l  #6,a7                     pop args
d_test:
          dbra    d1,d_loop                 go till end
d_exit:
          movem.l (a7)+,a0-a1/d0-d1         restore registers
          rts                               and return

          data                              data segment
          even
count     equ     4
strings   equ     *
          dc.l    str1                      address of 1st string
          dc.l    str2                      address of 2nd string
          dc.l    str3                      address of 3rd atring
          dc.l    str4                      address of 4th string

str1      dc.b    'Once upon a midnight dreary,',10,13,0
str2      dc.b    'As I pondered, weak and weary,',10,13,0
str3      dc.b    'Over many a quaint and curious',10,13,0
str4      dc.b    'Volume of forgotten lore...',10,13,0
```

Golden Path

FIREBIRD
71 Franklin Turnpike
Waldwick, NJ 07463
Low resolution $44.95

by Andy Eddy

I don't know where the fanaticism comes from, but there have been a truckload of games with an oriental flavor—Shanghai (Activision), ST Karate (Eidersoft), Karate Kid II (MichTron), Kareteka (Broderbund), *et al*. Now we can add another to the pile: Firebird's **Golden Path** a graphic, mouse-controlled adventure.

Your on-screen counterpart is Y'in Hsi, the infant son of the Golden Emperor, T'ang Yin. Ch'un Kuei, who is a ruthless warrior and the mortal enemy of T'ang Yin, has successfully slain T'ang Yin in battle, leaving Y'in Hsi—too young to be aware of the ties to his father—to be raised by monks.

When he reaches his sixteenth birthday, the monks explain to Y'in Hsi his history, as well as the fate of his father. To make a long story short—Firebird went to the trouble to include this whole tale as a novella (similar to the one with their previous paramount game, Starglider), so why should I spoil it? T'ang Yin has left a ring and a scroll (called the Book of Knowledge) for his son, as well as a legacy. Upon placing the ring on his own finger, Y'in Hsi is converted by the ring's magic into a frail, old man. He becomes, in essence, what his father would be at that moment. He must travel the "Golden Path of Enlightenment" and avenge the death of his father. By doing this, he will reclaim his lost youth.

Golden Path is a clean programming effort—something we've come to expect from Firebird and their parent British company, Rainbird. The cursor (a small, oriental calligraphy character) is smoothly manipulated by the mouse, and the character on-screen maneuvers in the direction of the cursor. Certain movements, like picking up or throwing objects that are on the screen, storing items in your pockets, and offensive/defensive actions (like kicking, punching and blocking) are all mouse controlled, as well.

Each screen is a different location, where you're likely to find various characters and objects to help you toward the game's conclusion. Your task is to discover which items cause the characters whose paths you cross to give you what you need, or to assist you with the proper action. A spokesman at Firebird's New Jersey office told me that the game can be completed from start to finish in 20 minutes, provided you do everything *just right*. Of course, that's the challenge: finding just the right moves in order to succeed.

The most difficulty comes from the fact that you only have four pockets—each one able to hold only one object—so things have to be accomplished in some sort of order. And, most importantly, the game cannot be saved to disk on the fly, as is normal with adventures, so what you do is critical. It can be incredibly frustrating to be cruising along at a good clip and run into a snag that depletes your character's energy level.

Energy is displayed on-screen by a vine. The larger the vine, the stronger Y'in Hsi is; when the vine withers to nothing, he drops to the ground, a dead man. All is not lost, though: you can reincarnate Y'in Hsi with a tap on the left or right button, the latter bringing a more difficult scenario.

You do have a number of aids on your side, to help you in your mission of discovery. For starters, at the bottom left corner of the display is an icon resembling a gold-trimmed tome (the previously mentioned Book of Knowledge), which will open when you come across something new. This can include a character you haven't seen before, an object for you to pick up, or the description of a new location. You can also click the pointer on the book at any time for a status report on where you stand.

On the bottom right, there's a scaled-down picture of your location. If you click on this, you'll get a rundown of how fruitful your mission is thus far. You will be told how many screens you've seen (out of a possible thirty-seven), how many of the myriad of characters you've met, and how many "steps" you are along your journey. The number of steps can be considered your score, because, when you have completed all of the steps necessary, you've won.

Unfortunately, you can't dawdle over the puzzles at hand. If you do, you'll be greeted with harrassment, in the form of an annoying little gnome called Hoppy. You can dispatch him with a well-timed strike, but it's better to just exit the screen you're in and return. In time, you'll discover what each room has to offer and what each item is used for. At that point, Hoppy won't interrupt you too often.

I found **Golden Path** (which comes on two disks) a difficult foray, frustrating and deeply challenging. It requires deft movements, intuitive thinking and, most of all, a strong sense of patience. It's not as addictive as Starglider, nor as witty as The Pawn, but—in what's becoming a Firebird trait—it has an air of freshness about it, breaking some new ground. I've been waiting for Firebird to trip up and release a real loser of a game, but a recent press release states that **Golden Path** for the ST was awarded the title of Best Adventure Game at this year's Summer CES (Consumer Electronics Show) in Chicago. It's not surprising. //

Andy Eddy works as a cable TV technician in Connecticut, but has been interested in computers since high school. While his family's Atari 800 is four years old, he has been avidly playing arcade games since Space Invaders and is a former record holder on Battlezone.

1st Convert Utility

Changes a standard ASCII file to be reformatted in 1st Word or Word Writer ST.

by Roderick W. Smith

How many times have you loaded a text file into 1st Word or Word Writer ST, only to find that the person who wrote it did not use 1st Word text format? The first time you did this, you may have tried reformatting the document, but to no avail.

Now, though, there is an answer: **1st Convert Utility**. This program, written in Personal Pascal, will take a standard ASCII file and convert it so that it can be reformatted in 1st Word or Word Writer ST.

To run the program.

If you're a disk subscriber, simply click on the 1ST_CNVT.PRG icon; the program will load and run. If you don't have the program in a compiled form, though, you'll have to compile it yourself with Personal Pascal. My apologies to those without this language: I refuse to program in ST BASIC; Logo is inadequate; and Hippo C nauseates me. Since Pascal is the only other language I have, I used it.

If you have Personal Pascal, enter the editor and type in the program listing. Press the F9 key, and the program will be saved to disk and compiled automatically. Then you can run it. (Note: if you're using a RAMdisk, you should copy the source file to floppy or hard disk before running the program; certain typos might cause your ST to lock up!)

When the program runs, a dialog box will come up with a copyright message, three option buttons, and the usual "OK" and "Cancel" exit buttons. If your file is from a "generic" text editor, choose the "ASCII" source. Just about everything except space characters will be passed through as is. Paragraphs will be defined by two blank lines—so if there are no blank lines in the file, 1st Word will see it as one very large paragraph unless you do some manual editing.

If your source file is from ST Writer, choose the middle button. The codes for new paragraphs, line centering, underlining, superscripting, subscripting and page ejects will be recognized and converted to their 1st Word equivalents. Other control codes—such as margin settings, headers and footers—will be ignored, along with all the text after such codes on the same line. Finally, if your file was generated using WordStar on an MS-DOS or CP/M machine, select the third button. The codes for underlining, boldfacing/double-striking, superscripting, subscripting and forced page breaks will be converted to their 1st Word counterparts. Most "dot commands" will be ignored.

After you click on the OK button, you'll be presented with an alert box informing you that you must select the source file. Click on the "Ready" button, and a file selector dialog box will appear. Click on your source file, then on OK. Another alert box should appear, informing you that a 1st Word filename will be requested. When the file selector box appears, a filename will already be entered. This filename is the same as the one you selected as the source file, but with a *.DOC* extension. Click on OK if this is acceptable; or edit the pathname and filename fields. Do not enter the same filename as the original. When you click on OK, your file will be copied and converted. This may take a while if the document is of any length, so be patient. Make sure you have enough space on your disk to hold another copy of your document! When the conversion is finished, you will be asked if you wish to convert another file. A word of warning: do not remove your destination disk from the drive until the program is finished; Pascal appears to close the files only after the program is done.

You can now load 1st Word or Word Writer, and view or modify the converted document(s). Note that, especially with ST Writer files, the formatting may not be pretty—but that's not a big problem, since you can reformat the document at will. If you convert an ASCII file and find that 1st Word thinks it's one large paragraph, try deleting trailing spaces at the ends of all the paragraphs to eliminate the problem.

What the code does.

This section is intended for those interested in the actual workings of the program. Others need not read on.

The program begins—as does any other Pascal program which utilizes GEM—with a number of CONSTant and VARiable declarations and includes from the Pascal disk. The function TYPE_TRANSFER is the code that is necessary to put a dialog box on the screen. It consists of a call to NEW_DIALOG, followed by a series of calls to ADD_DITEM and SET_DTEXT, which add an item and specify what text that item is to contain, respectively. DO_DIALOG is all the programmer has to do to actually put the dialog box on the screen and allow the user to interact with it—GEM does the rest. The calls to OBJ_STATE determine which of the three buttons the user selected. END_DIALOG removes the dialog box from the screen, and DELETE_DIALOG deletes it from memory.

Procedure GETINFO returns the filenames for the source and destination files. At each call, the default pathname (DEF_PATH) is used, but the filename for the source (FILE_SIMPLE) is set to null. The procedure looks for a period in the input filename and, if one is found, uses its location to construct the default filename for the destination file.

Procedure HANDLELINE is the bulk of the program, and it contains many sub-subprograms. The first of these, NEW_LINE, simply starts a new line. TOGGLE_CODE toggles an internal flag (STATUS, the second parameter) representing what text style is currently being written. The first of its parameters is a CONSTant which, effectively, represents 1 bit of 6 which are used by 1st Word in determining text styles. This value is either added to or subtracted from STATUS, which is then written to disk. NEW_PAGE writes a blank line, which is necessary to avoid formatting difficulties should the user reformat the document, and then the code for a new page. FIX_CHAR strips the high bit from alphanumerics. This is required because WordStar sets this bit in the last character of each word in a file, making the file unreadable in 1st Word—and, for that matter, in most other word processors. DO_SIMPLE is the procedure called when the user requests an ASCII conversion; it merely replaces standard spaces with 1st Word's "variable spaces."

DO_STWRITER is considerably more sophisticated, and contains several subprograms of its own. Procedure NEXT_LINE is essentially identical to a READLN call, but works with ST Writer files. These files use the NUL character (ASCII code *0*), instead of the usual carriage return. UP_ARROW is called whenever the code for begin superscript/end subscript is encountered. It has to determine whether subscripting is on or not, then toggle either subscripting or superscripting. DOWN_ARROW is UP_ARROW's counterpart for begin subscript/end superscript. CENTER_LINE is called whenever the center line character is encountered. It reads in the entire line, determines its length, writes out blanks, then writes out the text. Note that the procedure does not support any text enhancements except underlining. DO_STWRITER itself is nothing more than a CASE statement which calls earlier procedures.

DO_WORDSTAR is considerably less sophisticated than DO_STWRITER; this is because 1st Word's file format is more akin to WordStar's than to ST Writer's. Subprocedure CHECK_DOT checks to see if a period occurred in the first column. If so, it's a "dot command," and is either ignored or signals a page break. Like DO_STWRITER, DO_WORDSTAR is really a CASE statement which calls earlier subroutines, mostly TOGGLE_CODE.

PAST75 is a procedure which was originally designed for ST Writer files, but ended up being used for all files, simply out of caution. It is called only if the program's internal column counter exceeds 75. It's basically a miniature HANDLELINE, but it stops execution and puts in a RETURN when a space is read. This is the first place that the means for coding paragraph ends becomes readily apparent. 1st Word, throughout a paragraph, places what might be called a "null space" at the end of each line; but there is no such null space at the end of a paragraph.

The HANDLELINE main loop is mostly a WHILE loop which reads a character, strips the high bit for alphanumerics, calls a character handler, depending upon the input source, and advances the column counter. The procedure then rids itself of the end-of-line (EOL) by the READLN statement. The final IF statement determines whether the program should put out the null space which was mentioned above. If the column number is *1* (that is, if the line is blank), if the source is an ST Writer or WordStar file (which already have the null space at the end of the line), or if the next character is an EOL (that is, if the next line is blank), no space character is output.

The program's main procedure basically glues all these together and handles some preliminary initialization details. GEM is notified of impending use of its resources, the mouse is set to the arrow form, and a default pathname (the current drive's root directory) is defined. Then begins a loop which ends when the user does not wish to convert another file. STATUS is set to *128* (1st Word's code for normal text), the type of source file is determined, and both source and destination filenames are requested. Then the work begins: the mouse form is set to the "busy bee" and files are reset. If the file is an ST Writer file, the first thirty-four characters—which are junk as far as 1st Word is concerned—are read and ignored. Then, until the end of the file is reached, the HANDLELINE routine is called. Finally, the mouse is returned to its arrow form, the user is asked about further conversions, and, if no further conversions are wanted, GEM resources are de-allocated and the program is exited.

Concluding remarks.

While controlling GEM will never be easy, Personal Pascal makes it at least somewhat simple, with redesigned GEM and AES calls. Even outside the GEM interfacing, the advantage of Pascal—or, for that matter, any other structured language—over something like BASIC in this program is also fairly clear, if you think about actually writing the program. I could contemplate what I wanted the program to do in very broad terms at first—handle a line here, put up a dialog box there—and only after writing this outline did I need to actually code the details, which I saw to be much simpler when viewed away from the context of the program as a whole. The clumsiness of subroutines in BASIC makes

this more difficult to do, and the temptation to use GOTOs, if not resisted, results in code that's more difficult to follow, even to the person who wrote the program. In any event, I hope **1st Convert Utility** will be used well by people—both in terms of converting files and in terms of learning about programming, if that's an interest of yours. //

Roderick W. Smith is currently attending Oberlin College in Oberlin, Ohio, where he's majoring in psychology. He mainly uses his ST for word processing and communicating with Oberlin's VAX, but enjoys programming, as well.

Listing 1.
Pascal listing.

```
program simple_to_1st_Word (simple, FirstWord);
   {Program to convert from straight ASCII to 1st Word file format}
   {by Rod Smith, December 1986; Revised March 1987}

const
   {$I GEMCONST.PAS}
   STRETCH_SPACE = 28; {1st Word's "stretch" space}
   INDENT_SPACE  = 29; {1st Word's "indent" space}
   VAR_SPACE     = 30; {1st Word's "variable" space}
   FIX_SPACE     = 32; {1st Word's "fixed" space}
   F_BOLD        = 01;
   F_ULINE       = 08;
   F_SUPSCRPT    = 16;
   F_SUBSCRPT    = 32;
   ASCII         = 1; {simple ASCII file}
   STWRITER      = 2; {STWriter file}
   WORDSTAR      = 3; {WordStar file}

type
   {$I GEMTYPE.PAS}

var
   file_simple, file_1st, def_path: string;
   simple, FirstWord:               text;
   source, status, i:               integer;
   junk:                            char;

{$I GEMSUBS.PAS}

{************************************************************************}

function CurrentDrive: integer;
   {returns the current disk drive}
GEMDOS ($19);

{************************************************************************}

function type_transfer: integer;
   {returns the type of transfer}
var dialog: Dialog_Ptr;
    title, author,
    copy1, copy2, copy3,
    prompt,
    simple, ST_Writer, WS,
    OK_btn, cancel_btn:  integer;
begin
   dialog := New_Dialog (13, 0, 0, 38, 14);
   title := Add_DItem (dialog, G_Text, None, 1, 1, 36, 1, 0, BLACK * 256);
   Set_DText (dialog, title, '1st Convert Utility', System_Font, TE_Center);
   author := Add_DItem (dialog, G_Text, None, 1, 2, 36, 1, 0, BLACK * 256);
   Set_DText (dialog, author, 'by Rod Smith', System_Font, TE_Center);
   copy1 := Add_DItem (dialog, G_Text, None, 1, 4, 36, 1, 0, BLACK * 256);
   Set_DText (dialog, copy1, 'Portions of this product are', Small_Font,
              TE_Center);
   copy2 := Add_DItem (dialog, G_Text, None, 1, 5, 36, 1, 0, BLACK * 256);
   Set_DText (dialog, copy2, 'copyright (c) 1986 OSS and CCD.', Small_Font,
              TE_Center);
```

```
copy3 := Add_DItem (dialog, G_Text, None, 1, 6, 36, 1, 0, BLACK * 256);
Set_DText (dialog, copy3, 'Used by permission.', Small_Font, TE_Center);
prompt := Add_DItem (dialog, G_Text, None, 1, 8, 36, 1, 0, BLACK * 256);
Set_DText (dialog, prompt, 'The source file is from:', System_Font,
          TE_Center);
simple := Add_DItem (dialog, G_Button, Radio_Btn | Selectable,
                     2, 10, 10, 1, 1, BLACK * 256);
Set_DText (dialog, simple, 'ASCII', System_Font, TE_Center);
Obj_SetState (dialog, simple, Selected, FALSE);
ST_Writer := Add_DItem (dialog, G_Button, Radio_Btn | Selectable,
                        14, 10, 10, 1, 1, BLACK * 256);
Set_DText (dialog, ST_Writer, 'STWriter', System_Font, TE_Center);
WS := Add_DItem (dialog, G_Button, Radio_Btn | Selectable,
                 26, 10, 10, 1, 1, BLACK * 256);
Set_DText (dialog, WS, 'WordStar', System_Font, TE_Center);
OK_btn := Add_DItem (dialog, G_Button, Selectable | Default | Exit_Btn,
                     10, 12, 8, 1, 3, BLACK * 256);
Set_DText (dialog, OK_btn, 'OK', System_Font, TE_Center);
cancel_btn := Add_DItem (dialog, G_Button, Selectable | Exit_Btn,
                         20, 12, 8, 1, 1, BLACK * 256);
Set_DText (dialog, cancel_btn, 'Cancel', System_Font, TE_Center);
Center_Dialog (dialog);
if Do_Dialog (dialog, 0) = cancel_btn then Halt;
if Obj_State (dialog, simple) = Selected then type_transfer := ASCII;
if Obj_State (dialog, ST_Writer) = Selected then type_transfer := STWRITER;
if Obj_State (dialog, WS) = Selected then type_transfer := WORDSTAR;
End_Dialog (dialog);
Delete_Dialog (dialog);

end; {type_transfer}

{*************************************************************************}

procedure GetInfo (var file_simple, file_1st, def_path: string);
   {returns the filenames for the source and destination files}
var junk:     integer;
    sans_ext: string;
begin {GetInfo}
   file_simple := '';
   junk := Do_Alert ('[0][Filename of the source file...][ Ready ]', 1);
   if not Get_In_File (def_path, file_simple) then Halt;
   junk := pos ('.', file_simple);
   if (junk > 1)
      then sans_ext := copy (file_simple, 1, junk - 1)
      else sans_ext := file_simple;
   file_1st := concat (sans_ext, '.DOC');
   junk := Do_Alert ('[0][Filename of the 1st Word file...][ Ready ]', 1);
   if not Get_In_File (def_path, file_1st)
      then GetInfo (file_simple, file_1st, def_path);
end; {GetInfo}

{*************************************************************************}

procedure HandleLine (source: integer; var status: integer);
   {procedure to do the real work of the program}
var new_char: char;
    column:   integer;
{-------------------------------------------}
function new_line (spaces: integer): integer;
begin
   if (spaces > 0)
      then writeln (FirstWord, ' ')
      else writeln (FirstWord);
   new_line := 1;
end; {new_line}
{-------------------------------------------}
procedure toggle_code (code: integer; var status: integer);
begin
   if ((status & code) = code)
      then status := status - code
```

```
      else status := status + code;
   write (FirstWord, chr(27), chr(status));
end; {toggle_code}
{-----------------------------------------}
procedure new_page;
begin
   writeln (FirstWord); {avoids formatting problems later}
   write (FirstWord, chr (12)); {new page marker}
end; {new_page}
{-----------------------------------------}
function fix_char (new_char: char): char;
var testchr: integer;
begin
   testchr := ord (new_char);
   if ((testchr > 160) and (testchr < 255)) then begin
      testchr := testchr - 128; {strip the high bit}
      new_char := chr (testchr);
   end; {if}
   fix_char := new_char;
end; {fix_char}
{-----------------------------------------}
procedure do_simple (new_char: char);
begin
   case ord (new_char) of
      FIX_SPACE: write (FirstWord, chr (VAR_SPACE));
      else:      write (FirstWord, new_char);
   end; {case}
end; {do_simple}
{-----------------------------------------}
procedure do_STWriter (new_char: char;
                       var status, column: integer);
const NUL       = 00; {code for <RETURN>}
      CENTER    = 03; {code for center line}
      EJECT     = 05; {code for page eject}
      PARAGRAPH = 16; {code for new paragraph}
      UNDERLINE = 29; {code for underlining on/off}
      SUPER     = 27; {"up-arrow"}
      SUB       = 28; {"down-arrow"}
{.........................................}
procedure next_line (column: integer);
var counter: integer;
    junk:    char;
begin
   repeat
      read (simple, junk);
   until (junk = chr (NUL));
   column := 1;
end; {next_line}
{.........................................}
procedure up_arrow (var status: integer);
begin
   if ((status & F_SUBSCRPT) = F_SUBSCRPT)
      then toggle_code (F_SUBSCRPT, status)
      else toggle_code (F_SUPSCRPT, status);
end; {up_arrow}
{.........................................}
procedure down_arrow (var status: integer);
begin
   if ((status & F_SUPSCRPT) = F_SUPSCRPT)
      then toggle_code (F_SUPSCRPT, status)
      else toggle_code (F_SUBSCRPT, status);
end; {down_arrow}
{.........................................}
procedure center_line (var column, status: integer);
var count, i: integer;
    letter:   char;
    line:     string;
begin
   letter := 'A'; {dummy constant}
   line := '';
```

```
   while (letter <> chr (NUL)) do begin
      read (simple, letter);
      line := concat (line, letter);
   end; {while}
   count := (66 - length (line)) div 2; {# spaces to pad}
   for i := 1 to count do write (FirstWord, chr (INDENT_SPACE));
   for i := 1 to length (line) do
      if line [i] = chr (UNDERLINE)
         then toggle_code (F_ULINE, status)
         else write (FirstWord, line [i]);
   writeln (FirstWord);
   column := 1;
end; {center_line}
{.....................................}
begin {do_STWriter}
   case ord (new_char) of
      FIX_SPACE: write (FirstWord, chr (VAR_SPACE));
      NUL:       column := new_line (1);
      CENTER:    center_line (column, status);
      EJECT:     new_page; {page eject}
      PARAGRAPH: write (FirstWord, '     '); {^P, new paragraph}
      UNDERLINE: toggle_code (F_ULINE, status);
      1, 2, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20,
      21, 22, 23, 24, 25, 26:
                 next_line (column); {control characters; ignore line}
      SUPER:     up_arrow (status); {either start sup-scripting, or end sub-}
      SUB:       down_arrow (status); { "     "   sub-    "       "   "  sup-}
      else:      write (FirstWord, new_char);
   end; {case}
end; {do_STWriter}
{-----------------------------------------}
procedure do_WordStar (new_char: char;
                       var status, column: integer);
const BOLDFACE   = 002; {^B, boldface toggle}
      DBL_STRIKE = 004; {^D, double strike toggle}
      UNDERLINE  = 019; {^S, underline toggle}
      SUPSCRPT   = 020; {^T, superscript toggle}
      SUBSCRPT   = 022; {^V, subscript toggle}
      DOT        = 046; {'.', for 'dot commands'}
      GARBAGE    = 138; {don't know what it is, but appears after new page}
      MET_RET    = 141; {WordStar's <RETURN>}
      MET_SPACE  = 160; {WordStar's "stretch" space}
{.....................................}
procedure check_dot (column: integer);
var code: string [2]; {'cause codes are 2 chars. in length after '.'}
begin
   if (column <> 1)
      then write (FirstWord, '.')
      else begin
         readln (simple, code);
         if ((code = 'PA') or (code = 'pa'))
            then new_page; {page break}
      end; {else}
end; {check_dot}
{.....................................}
begin
   case ord (new_char) of
      FIX_SPACE:  write (FirstWord, chr (VAR_SPACE));
      MET_SPACE:  write (FirstWord, chr (STRETCH_SPACE));
      MET_RET:    column := new_line (0); {w/in paragraph newline}
      UNDERLINE:  toggle_code (F_ULINE, status);
      BOLDFACE,
      DBL_STRIKE: toggle_code (F_BOLD, status);
      SUPSCRPT:   toggle_code (F_SUPSCRPT, status);
      SUBSCRPT:   toggle_code (F_SUBSCRPT, status);
      DOT:        check_dot (column); {check to see if it's a 'dot command'}
      GARBAGE:    ; {null statement; ignore character}
      else:       write (FirstWord, new_char);
   end; {case}
end; {do_WordStar}
```

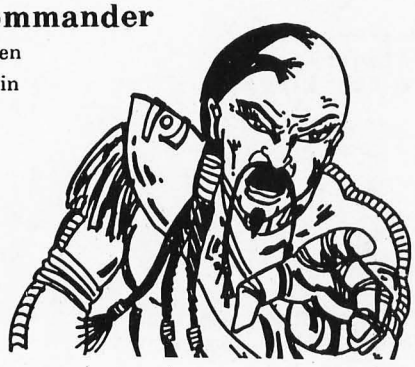
```
{-------------------------------------------}
procedure past75 (var status, column: integer);
   {goes until next space, then starts a new line}
   {always returns a '1'}
var next_char: char;
begin
   next_char := 'A'; {dummy constant}
   repeat
      read (simple, next_char);
      next_char := fix_char (next_char);
      if (next_char <> ' ') then case source of
         ASCII:    do_simple (next_char);
         STWRITER: do_STWriter (next_char, status, column);
         WORDSTAR: do_WordStar (next_char, status, column);
         end; {case; if}
   until ((eoln (simple)) or (next_char = ' '));
   write (FirstWord, chr (VAR_SPACE)); {makes next line part of same para.}
   if (not eoln (simple)) then writeln (FirstWord); {start a new line}
   column := 1;
end; {past75}
{-------------------------------------------}
begin {HandleLine}
   column := 1;
   new_char := 'A'; {dummy value}
   while not eoln (simple) do begin
      read (simple, new_char);
      new_char := fix_char (new_char);
      case source of
         ASCII:    do_simple (new_char); {simple file}
         STWRITER: do_STWriter (new_char, status, column); {STWriter file}
         WORDSTAR: do_WordStar (new_char, status, column); {WordStar file}
      end; {case}
      column := column + 1;
      if column > 75 then past75 (status, column);
   end; {while}
   readln (simple); {start the next line}
   if ((column = 1) or (source <> ASCII) or eoln (simple))
      then writeln (FirstWord)
      else writeln (FirstWord, chr (VAR_SPACE));
end; {HandleLine}

{*********************************************************************}
{*********************************************************************}

begin {simple_to_1st_Word}
   if Init_GEM >= 0 then begin
      Set_Mouse (M_ARROW);
      def_path := Concat ((chr (CurrentDrive + ord ('A'))), ':\*.*');
      repeat
         status := 128;
         source := type_transfer; {what type of original file?}
         GetInfo (file_simple, file_1st, def_path);
         Set_Mouse (M_BEE);
         reset (simple, file_simple);
         rewrite (FirstWord, file_1st);
         if (source = STWRITER)
            then for i := 1 to 34 do read (simple, junk); {dispose of junk}
         while not eof (simple) do
            HandleLine (source, status);
         Set_Mouse (M_ARROW);
      until (Do_Alert ('[2][Convert another file?][ Yes | No ]',1) = 2);
      Exit_GEM;
   end; {if Init_GEM...}
end. {simple_to_1st_Word}
●
```

The Krellan Empire wants YOU...

STAR FLEET II™

Krellan Commander

by Trevor Sorensen
& Mark Baldwin



from

interstel™ corporation

In **STAR FLEET II-Krellan Commander**, second in the STAR FLEET series of advanced space strategy simulations, your mission is to explore the star systems of the United Galactic Alliance, looking for planets to conquer and Alliance warships to destroy. Your ships are equipped with the latest technology, and on board are detachments of the feared Krellan shock troops, used to board and capture enemy vessels. At higher ranks, you command dozens of warships and a million combat-hardened warriors to bring the Alliance to its knees!

Join the Krellan forces in destroying the enemy!

Coming soon for the Atari ST, IBM PC & Compatibles, Amiga, and others.

Accept the challenge...

EMPIRE™

Wargame of the Century™

Version 2.0

by Walter Bright
& Mark Baldwin

from

interstel™ corporation

EMPIRE-Wargame of the Century, from the STAR FLEET series, is a strategic simulation of global conflict, conquest, and empire building between two or three human or computer opponents. As an Alliance task force leader, you must explore and conquer new territory, and produce the land, air and sea forces needed to bring the planet under your domination. Command armies, fighters, aircraft carriers, destroyers, submarines, and more. All your strategic skills are required, and there can be only one winner!

Accept the challenge to control an **EMPIRE**!

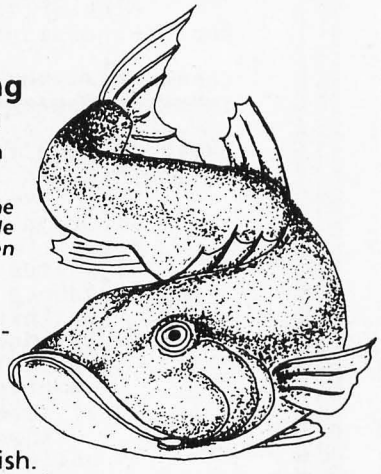
Available now for the Atari ST: $49.95.
Coming soon for the IBM PC & Compatibles, Amiga, and Apple II.

New from Interstel, for a change of pace...

Get Hooked on **Gone Fish'n**™

A Bass Fishing Simulation

by Roger Damon



"Since the bass emerged from his primordial piscatorial tree, he has been susceptible to the plastic worm. He has been susceptible to some struggling morsel kicking around on the surface. He has been susceptible to open and close his mouth on every creation that wanders by and even remotely resembles protein. Bass, after all, are bass." --Roger Damon on The Art of Bass Fishing

This is the philosophy behind ***Gone Fish'n,*** a realistic simulation of the popular sport of Bass fishing. Outstanding color graphics and sound effects give you the feel of actually being out there!

Weather and Fishing Reports help you solve the dilemma of when to work and when to fish. Choose from eight different lakes, each with unique features. Out on the lake, use your depthfinder to find underwater structure and fish. Once you pick a spot and cast your line, YOU control rod and reel action. Catch a fish and its weight is added to your total. A real lunker might win you a place in the Bass Fishing Hall of Fame! Ready for some competition? Try your luck in a three-day tournament.

Gone Fish'n is just like real Bass fishing - except you can fish from your favorite easy chair!

Available now for the Atari ST: $45.00.
Coming soon for the IBM PC and others.

interstel™ corporation
P.O. Box 57825
Webster, Texas 77598
(713) 486-4163

To order these Interstel products: Visit your retailer or call 800-245-4525 (in CA call 800-562-1112) for VISA and MC orders. Or, contact **Electronic Arts**, P.O. Box 7530, San Mateo, CA 94403.

Distributed by
ELECTRONIC ARTS™

CIRCLE #114 ON READER SERVICE CARD

WordPerfect

AN *ST-LOG* EXCLUSIVE PRODUCT PREVIEW

by D.F. Scott

Last year, Atari and Microsoft signed a pact which would let Microsoft port its Macintosh version of Write over to the ST, and let Atari market it under the Fuji symbol. Atari viewed this as an official endorsement of the ST by the world's largest and most respected software conglomerate; Microsoft has, over time, openly implied it was all a waste of time.

Then along came a small software company in Orem, Utah, whose only claim to fame was that it produced the best-selling software package in the U.S.—for a few years running. Its name *was* SSI Software. But, perhaps in a move to thwart phone calls asking how to overcome a bug in Gemstone Warrior, the company renamed itself in 1986, after its premiere software feat, to become WordPerfect Corp.

Now Atari is playing down its own Write product—which is on revision seven, going on eight, in the labs—and pushing WordPerfect instead. It might seem to some that the introduction of mass market software to the ST realm would initiate the inevitable collapse of the one- or two-man software production houses.

WordPerfect Corp. does not feel, however, that it can just march into the ST software market as the all-powerful conqueror. In fact, in conversations with the company's representatives, one might receive the impression that it is making a dangerous move in an already saturated market.

At the last Dallas AtariFest, WordPerfect Corp.'s marketing stance was, at once, defensive. To describe themselves and their program, they used the term *unlimited* repetitively. Ease of use has been their trademark, but that quality—with respect to the ST word processing foray—isn't as unique as it is in the IBM superpower melee. WPC had to find distinctions to draw between themselves and other companies; so they began by introducing a totally *new* WordPerfect, written entirely in assembly language and sharing only the name with its predecessors.

The WordPerfect package contains *six disks;* its structure is so large that WPC's advertising suggests using a Supra 20-meg hard drive to run the program. Printer drivers alone consume two single-sided disks; and, besides the program disk, there are separate spelling checker and thesaurus disks, plus a tutorial disk. The spelling checker contains 115,000 words, some of which are divided into separate "specialty" categories, with room for additions. The thesaurus disk displays both synonyms and antonyms.

This is the first WordPerfect to be GEM-dedicated. It uses the standard GEM windows and menu bar, leaving desk accessories available. The information line has been moved to the bottom, and registers the current cursor location.

WPC has spawned its own Atari Products Division to handle this one product. Todd Ashman, the division manager, recently discussed his company's marketing philosophy: "We're going to be targeting the high-end market in word processing. 'Full-featured,' to us, is something that has much more than just a 50,000-word dictionary, outlining capabilities and merge. What we incorporate in WordPerfect is speed. It's fully functional from both the keyboard and the mouse.

"Some of the outstanding features that we have incorporated include on-screen columns—the ability to have up to five columns on-screen, just as they'll print to the printer. We've got macro capabilities which are unlimited and unlimited document size—we're not limited to RAM. Table of contents and index generation is done for you. Also, footnotes and endnotes are unlimited—you can have one footnote, for instance, that's 16,000 lines long.

"To show you the flexibility [of footnotes], we automatically renumber them as you edit, delete or add them. You can change the spacing between the notes, spacing within the notes, [decide whether to use] lines separating text and footnotes—so you've got full control over footnotes.

"A lot of the difference that you'll see," continues Ashman, "is in the high-end market. The people we've talked to have at least three or four word processors. One does one thing very nicely, another does another thing very nicely—there's no one that does it all. That's the kind of market we're going after. We're priced a little high for this particular market, but peo-

ple need to realize, if they want this type of software, it requires an *investment* in support of this type of development. We're going to have a toll-free technical support line, so they can call us direct. We'll support up to 200 different printers, 10 different laser printers—we'll support Atari's laser printer when they get that out."

The program also uses its own interpretive macro language, through which multiple keystrokes may be represented as statements and engaged in one chain via a single keystroke. The language structure is such that macros may be invoked, or invoke others, conditionally.

The "merge" feature might also be employed as a storage and retrieval database of sorts, as Ashman explains: "Our merge is much more than just 'take an address file and make a form letter.' That's easily done with WordPerfect. What we allow you to do is create a form on your screen, and do a form fill-in input from the keyboard. You can also embed macros in the merges and make it a conditional type of merge. It's very simple if all you want to do is merge an address file with a form letter; very complex if you get into assemblage of documents where you're switching primary and secondary files."

Individual paragraphs may be numbered or labeled in a variety of ways, for use with the built-in outlining system, or for the alphanumeric sort procedure. Paragraphs may represent "index cards," in a way, and may be sorted and stored as such. Internal math tables may also be generated as columns within a document. For instance, the user may select which items to add and subtract from each other for the "total" column.

The full GEM character set is supported, including accented characters for foreign languages. In fact, WordPerfect employs a Flash-like translation table, allowing the user to map which control key invokes which special character. Functions or macros may be invoked using alternate (ALT) keystrokes, as well as through the menu bar.

Internal memory is managed in such a way that the *last three deletions* may be retrieved from the brink of extinction, and documents which overfill the user data buffer automatically spill over to disk. Files created with the last-most-recent IBM version of WordPerfect, (version 4.1) may be converted to ST format, using the program's internal translator.

Perhaps the most controversial part of this program is its suggested retail list price—$395. While one may wonder why WPC feels the ST user will gladly spend a majority of the cost of a 520 monochrome system for just a word processor, WPC is busy using its PC-realm sales tactics to encourage dealers *not* to sell at suggested list. To be precise, they're telling dealers to hold *permanent 65-percent-off sales.*

Ashman explains their line of reasoning: "Obviously, the price is substantially higher than some of our competitors, but we don't feel they are in the same league as we are—feature-wise or performance-wise. The street value of the products will be substantially less; we let the dealers take a look at how much support they want to give, and then they can determine how much they want to charge.

"Take, for instance, our IBM PC product. The suggested retail is $495. You can pick up several magazines and find it priced anywhere from $200 to $250. That represents substantially more than 50 percent off. So, when you're talking suggested retail at $395, what it's really going to be selling for will be substantially lower than that. The market that the price will actually address is about the $180 market.

"Even at $395," states Ashman, "I feel that it's worth it, because I've worked with it, I know it; I've worked with some of the other packages that are currently out on the market, and what we offer and what they offer are substantially different. It will take a little bit of time for the end user to realize the difference. Right now, if we were selling our product by price alone, it wouldn't have achieved the number one position it has in the IBM market."

It was only a matter of time before the "high-end" software market met the ST. It will be interesting to see how a relatively large corporation fares against the ever-growing array of word processing dynamos conceived by individual computer artists in their backyards. We *can* predict this: With WordPerfect, expect to smell smoke, to see comparison lists ablaze within advertisements, and to set aside a Saturday to attend a 65-percent-off sale at a software dealer near you. //

D.F. Scott is an artist, writer, educator and programmer living in Oklahoma City. He is currently engaged in the study of quantum physics, computing and other ways in which elementary particles interact with each other. Otherwise, he fills infinite pieces of paper.

Ian's Quest

ST news, information and opinion.

by Ian Chadwick

By the time you read this, Atari may have released their IBM clone, the PC1. Then again, they may not have—it's hard to keep up with the pushme/pullyou nature of their product announcements. Rumor suggests that some of the higher echelon Atari execs aren't anxious to sell in the PC market. And, that some of them are not happy about manufacturing the machine in the first place. They're afraid it will take away from their efforts to market the ST.

Dealers, on the other hand, seem eager to get it. I've been told by several dealers that they expect to move a lot of them. Some of these dealers are also (Gasp) Commodore dealers who sell the Commodore PC10, PC20 and so on, in reasonable numbers. The Commodore isn't a bad clone, but is a tad more expensive than many of the gray-area imports (those no-name Korean and Taiwanese clones). Other dealers may offer clones of various names and brands, but, in any case, no one has ever suggested the PC clones reduce potential sales for their Amigas—the markets are too different.

There's plenty of sales potential in those PC/MS-DOS machines. In this two-university city with dozens of colleges and technical schools, teachers are saying to their students, "You should (or need to) buy a computer."

They don't, however, mean buy an ST, or an 800XL, or a C64, or even a Macintosh—they mean *buy an IBM clone*. Period. They have shorn versions of WordPerfect, dBase, 1-2-3, and others for sale in the textbook stores—all running under an MS-DOS environment.

They have all sorts of courseware programs and books, all for the same system. They've got doodley-squat for the ST; and they don't care one whit either. It's PC/MS-DOS, take it or leave it.

Atari huffs and puffs and gets in a tizzy about selling to the "Big Blue" market. But face it folks, the money is in MS-DOS. Whether you (or they) like it or not.

So me, I'm looking forward to getting one of these machines. I already have a laptop Toshiba T1100 on which I do a lot of my writing now. I couldn't live without it. My ST can read its 3½-inch disks too, so I can pass data files back and forth with ease, and use my ST's hard drive for Toshiba support.

But, I want a desktop model; something with slots so I can add a 2-megabyte RAM card, a hard disk, and so on, and not have to swap disks between incompatible machines. I trust the PC1 will have those options—I'd hate to think Atari is going to release a crippled machine. Consider that even here in Canada, where the devalued peso barely buys what it oughta, I can get an XT clone with one drive, AT-style keyboard, 640K RAM and monochrome video board for $650 and I can

add boards and drives to my heart's content. Atari has to compete with this.

Finding out exactly what the PC *does* have is tough. People up here hem and haw when I ask how many card slots it has. Or RAM. Or expansion bus lines. What's the big secret? Maybe no one knows! Maybe it's not even in production yet!

Anyway. . .turning to things ST, my friends at Micro D Distributing arranged for me to borrow a copy of PC Ditto from Avant-Garde Systems to review and test, since I have a lot of PC software on 3½-inch disks. I dropped in on the closest Electronic Playworld where Eric set it up and ran it for me.

A kid looked up from the display of Commodore PC-10s while I loaded some software. He'd been pricing clones and asking about compatibility for the past few minutes. He was a student at the University of Toronto (Engineering, which is almost like a real subject) and needed a PC for his lab work.

"Is that Sidekick running on that 1040?"

"Yes it is," I said in my most innocent voice.

"I didn't think it was translated over to the ST," he said as awed sounds issued from his mouth. Other shoppers, sensing action, peered our way. A few shuffled over toward the end of the store where the STs are set up.

"It's not," I replied while loading WordPerfect 4.2 with a text file.

"Then how are you doing that?" (Intimations of magic and skulduggery here. . .)

"Why, I'm using PC Ditto, a PC/MS-DOS emulator. . ."

You get the drift, I'm sure. I ran through the dozen or so programs I had with me, including a few public domain things I got from CIS. Yes indeed, prestidigitation, folks. Nothing up my sleeve, but all the same, here we see a *game machine* acting like a Baby Blue. Hmmm. . .

So, here I stood, loading this, trying that. I got GEM 2.0 running on an ST—that's the version after the legal conference between Apple and Digital Research: the one that's not supposed to look like a Mac, but looks like Smalltalk no matter *what* anyone says (yet I never saw Alan Kay suing Steve Jobs about it). But, dear readers, I digress. . .

Why, you ask, would anyone want to run PC software when they have an ST? Well, for one, there are a *lot* of good programs out there—1-2-3, WordPerfect, Grammatik, Ventura, dBase III, AutoCAD, to name a few. On the other hand, a lot of PC software is more expensive than an entire ST system. But, for those of us with both systems, it's quite a boon. If you hook up a 5¼-inch drive to your ST, you can even run a lot of the protected programs without any trouble. There are even (Gasp) some good games for the PC which will probably come out for the ST about the time Halley's comet returns.

For me, with my Toshiba T1100 and amber monitor, the advantage is obvious. I can run programs in color using the ST monitor. Also, a 5¼-inch drive for my Toshiba retails at $88 here, while an ST drive is about $300—a distinct savings. The flip side is that it's *slow*—I mean glacial. About as fast as a C64. Obviously, I don't want to run anything where speed is an issue. But then if my T1100 needs to go in for repairs anytime, I can use my ST in the interim and not lose valuable work time. So the cost is certainly worth it.

There are some caveats, however. First, I don't believe it works with the monochrome monitors—color only. Second, you need a DOS disk from which to boot. PC and MS-DOS are proprietary programs, just like other software. You can't walk into a computer store and ask for a copy of DOS to go along with your PC Ditto. You'll have to buy it. (One advantage of owning a PC or clone is that you already have this disk and don't need to shell out another $200 or so).

When I visited Atari Canada and brought up the topic of the PC1 and PC Ditto, I got a pretty cool reception. Some of those same people who were dumping on MS-DOS as clunky, slow, inane and hostile, were praising CLIs (Command Line Interpreters) for the ST a while back. For the uninitiated, CLIs give you text command input for commands, instead of GEM. For example, rather than double-click on drive A to see its contents, you type *DIR A:*. Sound familiar? A lot like MS-DOS? Maybe *exactly* like MS-DOS? Some are, at least. Others are of the UNIX persuasion: nasty, obfuscatory, hostile and generally inelegant (compared to which MS-DOS is a veritable delight of ease and charm).

The ST is a nice machine, but it's not going to make it in the business world. No way, no how. IBM architecture has its foot in that door for life. And they're not exactly stagnant: they've grown from PC, through XT and AT to some amazing 16mHz 386 machines that just roar. And then there's OS2, the new operating system due sometime before the next ice age. Things change in that world, but the sales curves just keep going up. *And*, believe it or not, IBM and other manufacturers have rejuvenated the PCjr concept—the "home" PC market—with what appears to be success. Can the ST compete in the home market if Big Blue Brother and its minions come forth in force? Sigh. . .

Okay, so I see people having two distinct machines: a business and home machine. Certainly, most ST games can run circles around the usual lot of PC offerings (despite an unfortunate penchant for arcade style, as opposed to strategic entertainments where the mind is the workhorse, not the trigger finger). PC Ditto is the bridge between the two, so that you can enjoy the benefits of both worlds. The PC1 and its ilk are the serious solution for long-term efforts (assuming, as I stated above, Atari is intelligent and foresightful enough to offer a machine that can compete with the others).

Now for some whining, crying and sniffling. I'm writing this column on my T1100. Why? Not because WordPerfect is so much better than any current ST word processor, but because I had a little accident. Let me tell you about it.

I recently got a copy of MichTron's Tune Up, a hard disk optimizer. This program basically reads the files off the disk and writes them back in contiguous order, rather than scattering them all over the disk. This makes reading and writing operations a lot faster. I've used this sort of program in the PC environment, so I knew what to expect. Sort of.

I read the manual first. It's short. *Very* short. Okay, nothing new, no warnings I need worry about. I ran the program. After trying a few basic operations, I optimized my hard disk (segmented into four drives C to F). I don't have a RAM-disk at present, but I do have DeskCart. Looks fine. . .working. . .a few error messages, but it keeps on chugging. Fine. It's over. I reboot the system.

DISK ERROR! DISK ERROR! I couldn't reload my hard drive. No way, no how. I tried everything. No go. Somehow, the program managed to zap the drive. I even had a terrible time getting the &±%$# thing formatted—it took several hours and many, many attempts. Everything lost, wiped out, destroyed, *gone*. Imagine my delight.

Of course, I have backups of my program files and applications. *But*, what I hadn't done for a few months—due to an overload at the ex-office—was backup my

own files. Everything I had downloaded from CIS was gone. All my programs in GFA BASIC gone. My Midway game, and all the data files that went with it, gone. My Flash DO files gone. My romance novel gone. My sci-fi short stories gone. All the stuff that really mattered: GONE.

Now, to be fair, I took this little problem to a few other folk. Julius at Atari Canada sat down and ran the program on his own system without a single hitch. No one else had the same experience. Which led me to wonder: *Why me?* Was it because I had four disks partitioned? Or did DeskCart somehow interact with it in a nasty fashion? (For a reason that escapes me, some of the disk copy programs like Procopy will not work with DeskCart plugged in.)

Nonetheless, caveat emptor. I'd suggest that you remove any and all carts from the port first, and—just in case—make *sure* you have a backup copy of *everything* before you start. For those of you who want this sort of optimization, but don't want to risk my experience, you can simply back up your hard disk and reformat it, then copy each file back, one at a time. They'll be written contiguously. And, of course, the real lesson is: *Always back everything up first!*

An Atari twist: I saw a little item in the local paper, *Toronto Computes*, so I thought I'd bring you my experiences with the same problem.

Quite a long time ago, my 520 was upgraded locally to 1 meg. It started refusing to boot, then sometimes bombed out in the middle of things. One day, it finally refused to start up at all. So I took it to Keith Hope at BI (ah, the good old days) and he took it in hand, literally. He held it up and twisted the case. Voilà! It worked again. Needless to say, I questioned him on this matter. His reply: "The metal shield is in two parts, joined only by pressure contacts. It tends to oxidize and break contact between the halves. The twisting cleans it enough to work again."

"Isn't there a somewhat better solution?" I asked. He suggested that I solder one or two of the twist connections so that the connection was—more or less—permanent. He did so for me and I've never had this problem since. I might add that the *real* solution involves Atari making a more competent shield that screws down, rather than this stupid twist-tab idea. But, hey, who's listening to me? //

Ian Chadwick is a Toronto-based freelance writer with a newly-empty hard disk drive. If anyone has PD software—especially GFA BASIC programs—to replace those now lost, he would appreciate any you can send to him at: 47 Oakcrest Ave, Toronto, Ont., Canada M4C 1B4. Gracias.

New England's First Atari Fair labeled a success

by Maurice Molyneaux, Andy Eddy and Charles F. Johnson

October in New England brings the first touch of crispness to the air, signalling the start of the seemingly short stroll into winter. It also sets the foliage in the surrounding hills ablaze with bright reds and oranges, like a fiery DEGAS backdrop overseeing the small towns.

In 1987, October also brought the first official Atari Fair in New England, an event that took place over the Columbus Day weekend (October 10-11) at the Centrum in Worcester, Massachusetts. Sixty-two booths provided much to see and do for thousands of Atari enthusiasts from all over the Northeast and Canada, from looking at the latest in computer hardware, to blasting fellow show-goers via the networked wonder of MIDI-Maze.

The action really began on Friday, October 9, when the various exhibitors began setting up their booths amid confusion and handshakes galore, as those present got acquainted or reacquainted with each other. Through it all, whizzing about like a high-speed assembly language routine caught in an infinite loop, was ANALOG Editor/Publisher Lee Pappas—doing his best to get everything up and running smoothly.

To complicate matters, the rock group Heart was performing in the Centrum's auditorium both Friday and Saturday night, which resulted in a lot of anxious security guards trying to shuffle everyone out of the show both evenings. They must have feared "groupies" would sneak through the show to get backstage. The guards seemed unwilling to believe that those wearing staff or guest badges were just trying to take care of business and *not* trying to get on stage with Heart's Wilson sisters (though the thought had crossed our minds)!

The show opened to the public at 10 a.m. Saturday, and eager Atarians swarmed the many displays, keenly interested in seeing products they had previously known of only through rumors or announcements. The only real disappointment was how little was shown for the 8-bit machines; for the most part, the ST was the show's focus.

Closest to the entrance was the booth occupied by Supra Corp., where various peripherals were being demonstrated. Of particular note were the 2400-baud modem, 30-meg ST hard disk drive (sporting a DMA port for daisy-chaining additional devices), 20 megabyte *internal* hard disk (hard card) for the Mega ST, and a special new drive that can store *ten megabytes* of data on a 5¼-inch floppy disk! This high-density drive, which was shown utilizing Konica 480 TPI disks, operates at speeds comparable to a hard disk. Even though the disks are expensive, the ability to *swap* 10-meg disks is an attractive selling point—not to mention its obvious usefulness for backing up hard disks.

Astra Systems was on hand, displaying their combination hard/floppy ST disk unit. Sophos Chess was also present, showing their Chess software for the ST.

The **MIO** box for the XL/XE and a new ST hard disk were among the items shown by ICD. The MIO, as you may already know, allows simple "card" expansion of Atari XL and XE computers. ICD's 20-megabyte hard disk for the ST is notable because it fits *under* the monitor and can be tilted (acting as a stand). Further, it provides a DMA port for daisy-chaining other peripherals, and also sports two SCSI (Small Computer Systems Interface) ports for attaching non-ST specific devices.

Eidersoft had their large booth lined with computers showing an ST-ware cornucopia, from Triangle games and the **Pro Sound Digitizer** to the soon-to-be-released **Quantum Paintbox**—which, through programming tricks and screen interlacing, is capable of displaying some 4,096 colors.

Eidersoft also demonstrated another pending release from their '88 catalog: the **Minicomm** desk accessory. This program lets the user go on-line from within any GEM program, and offers many options that were only previously available through full-featured terminal software. One of the most sought after features of

The Northeast Atari Fair

Minicomm is *background file transfer*, a process that frees you to run other applications *while* moving files, much like a printer buffer.

Virtusonics was one of the few vendors showing a new 8-bit product. Their **Virtuoso Desktop Performance Studio** package contains features for animating graphics, creating music, telecommunications, and much more. The program can be used for entertainment only, or for creating images and music for videotapes, etc. The Virtuoso package requires a minimum of 64K of RAM, so many older Ataris are not equipped to use it. Virtusonics plans to license drivers so that work created with Virtuoso can be used in other programs. Plans to port Virtuoso to the ST are underway.

Data Pacific was showing off the famous **Magic Sac** Macintosh emulator for the ST, as well as the long awaited **Translator One** box, which makes it possible for the ST 3½-inch drives to read, write and format actual Macintosh disks. (Previously, programs had to be ported from a Mac, or downloaded from a BBS, because ST drives couldn't read the Mac disk format). The Translator One plugs into the MIDI ports on the ST, and lets you use either internal or external drives in Mac mode.

An ST running Magic Sac demonstrated a Macintosh program called Mac-a-Mug, which builds facial images ("mug shots," if you will) from a large variety of noses, eyes, mouths, and other computerized body parts. The software for the Magic Sac is up to revision 4.52 now; proof of Data Pacific's high level of support.

Practical Solutions was on hand to demonstrate their popular **Monitor Master** switchbox (for toggling between color and monochrome monitors without cable switching) and their upcoming **Video Key**, which converts the ST's RGB output to a composite signal, allowing the user to hook up composite monitors, or even videotape computer graphics. Another new item was the **Mouse Master**, a box that allows users to plug two joysticks *and* the mouse into an ST. The flip of a switch changes from mouse to joystick mode. 1040ST owners, rejoice!

JNL Technologies—perhaps the youngest exhibitor in computer history—was showing **Monitor Box**, which combines a monitor switchbox with a converter providing both composite video/audio and RF (television) output.

At the show's center was the large Atari display. ST and XE machines were on hand, as was a **Mega ST4**, hooked up to the Atari **SLM804** laser printer. Also present, though overlooked by many, was the entry-level **Atari PC**. It should also be noted that while the laser printer and the Atari PC were on display, Neil Harris confirmed that neither were locked into actual release dates.

Also at Atari's booth was the **SX212** 1200-baud modem. A beta version of the upcoming **SX Express** telecom program (by Keith Ledbetter) for 8-bit users of the SX212 was shown echoing the **GEnie** connection from a neighboring ST. Running on one XE was a stunning graphic demo with a pulsing Atari logo floating over a *scrolling* 16-shade GTIA field. Amazing what can be done on an 8-bit!

The long-promised Atari **XEP80** 80-column 8-bit adapters were also seen and sold at the show. However, there was no sign of the (then-just-released) **XE Game System**, or any of Atari's other video game consoles.

Thomas Carbone and Bill Leslie of Omnitrend Software were on hand, selling copies of their complex **Universe II** game, as well as their newest offering, **Breach**, a single-player tactical combat game in which the user must see his "squad leader" through many complex missions with varied objectives. It comes with a goodly amount of scenarios, in addition to a scenario editor and the promise of upcoming scenario disks.

Frank Cohen, president of Regent Software, presided over the Regent booth, where many of their ST programs were on display, including **Regent Word II, Regent Base** and **The Informer**, Regent's latest application. It's a multi-table database which allows the user to import presentation graphics from DEGAS or Neo-Chrome.

Hartech USA caught many an eye as they showed and sold their line of **Atari calculators.** Ranging in price from $5.95 to $24.95, Hartech offered over a dozen models from solar-powered "credit card" models to handheld printing calculators featuring 32-step auto-recall. All are handsome, inexpensive and bear the Atari stamp.

ANTIC Publishing's booth gained much attention with demonstrations of the many new graphics programs from The Catalog's ST line. From the 512-color pictures of **Spectrum 512** to the video-style graphics of **CyberPaint**, there was a lot to see. Impressive—but somewhat slow—was Tom Hudson's eight-minute, 4-meg **Spider Patrol** running on a Mega ST4. Looking like a computer movie, Spider Patrol got a lot of deserved attention. It certainly showed what you can do with 4 megs of RAM and the **Cyber Control** animation scripting program.

Atari Explorer had a booth, and one point gave away some Firebird software packages to new subscribers. Other publications represented were ***ST Express, ST World*** and ***ST Applications***.

Our own ANALOG Publishing had two booths. The ANALOG Sales booth was clearing out old 8-bit software and equipment at ridiculously low prices, while visitors to the main ANALOG booth were purchasing subscriptions, back issues of **ANALOG** and **ST-Log**, as well as ANALOG's Atari 8-bit Extra and pocket reference cards. Also on display in the ANALOG booth were animations by Maurice Molyneaux, which attracted crowds unaccustomed to seeing such a computer-generated "cartoon."

Digital Vision, maker of the popular **ComputerEyes** video digitizer for both 8-bits and STs was showing off the ST version, along with software that could increase the demand for ComputerEyes even more. A program called **Digispec** now allows the (color) ST ComputerEyes to digitize full-color pictures into Spectrum 512 format, creating results far better than previously possible. The marriage of ComputerEyes and Digispec—consummated just two days before the show—is indeed a powerful one.

Migraph's booth was frequented by users interested in programs like **EZ Draw** 2.0 and the newer **Drafix**. Migraph's release of new modules for use with EZ Draw 2.0 have made EZ Draw more desirable than ever because they allow for more CAD-type graphics, as well as some desktop publishing capability.

Next to Migraph was QMI (Quantum Microsystems Inc.), makers of **DeskCart** and **ST Talk**. The most often heard question in that vicinity was, "When is **ST Talk Professional 2.0** coming out?" The response from QMI chief John DeMar: "early November." If so, telecommunications users will have another quality choice for software.

QMI also showed their driver software for the Mitsubishi graphics tablet. This product will take the place of the mouse, permitting the user to input data into almost any application (such as DEGAS or CAD 3-D, even the desktop) with a stylus; a much-preferred method for computer artists and engineers.

Avant-Garde Software was demonstrating their PC emulator program, **PC-Ditto**. When run on an ST, it lets the user run most IBM PC-compatible software—with a slight loss of speed.

SoftLogic, the current leader of the ST desktop publishing pack, demonstrated their **Publishing Partner** program. More and more fonts and clip art for Partner are coming every month—many from outside firms—much to the delight of the program's users, and a testament to the program's acceptance in ST-land.

Delphi, the network housing ANALOG's on-line Atari Users Group, had a booth where users could peruse the system (free of the usual on-line fees). While many users checked out the news, entertainment and sports, some of the Delphi regulars hopped into the ANALOG SIG, and read waiting messages in the forum, or chatted with other users on-line about the show itself. On more than one occasion, users' questions were relayed to a manufacturers' booth, with the answer shuffled back to the Delphi terminal for re-transmission.

MichTron's booth was filled with sights and sounds as many of their programs were played and purchased. Their oldest and newest were displayed, including the dazzlingly fast **Goldrunner** and the Marble-Madnessesque **Airball**.

Terrific Peripherals showed their ST RAM expansions and battery backup clocks. The guts-exposed ST on display offered a kind of visual support that most of those who have taken the "do-it-yourself" route wished they had experienced before.

One of the show-stoppers was the booth where **Word Perfect** for the ST was being premiered. Many users were impressed that a full-blown "professional" word processor was at last a reality for the ST. Although boasting a hefty price tag (suggested retail $395), Word Perfect offers file compatibility with *all* versions of the program, and calling of all commands through either GEM menus or keyboard strokes. The program sports a built-in 115,000-word dictionary/spell checker, as well as normally unavailable enhancements like an on-line thesaurus, footnotes and endnotes, plus the ability to generate tables-of-contents and indexes. If you couldn't afford the program, you could at least get a free Word Perfect baseball cap at the show.

True BASIC is the name of the program *and* the company that produces it. Created by two of the fathers of the original BASIC language (John Kemeny and Thomas Kurtz), True BASIC is an updated and revised version of the language, designed to encourage "structured" programming. Also available on IBM PC compatibles, the Macintosh and Amiga, True BASIC offers modularity and graphics support. The ST version has complete AES/VDI library functions, and includes a GEM-based text editor. At the show, True BASIC, Inc. distributed a "flying ring" (like a hi-tech frisbee) that was designed with the IBM version of True BASIC!

Megamax, Inc. showed a beta test version of their new updated C language system, now called **Laser C**. This is a much enhanced descendant of their popular Megamax C. Laser C does away with Megamax's 32K limitations, and features a very nice graphic shell to aid in the compiling process. Also included are a DRI-compatible linker and debugger, improved Resource Construction program, and improved floating point libraries. A variable sized RAM cache lets the editor, compiler, linker and debugger remain in memory simultaneously if you wish, speeding development even further.

There was a lot of action on the MIDI front, as the ST computer is quickly gaining acceptance in musical circles as a powerful, low cost alternative to the more firmly entrenched Macintosh. With so many booths demonstrating music software, the Centrum was in a state of near constant cacophony.

Hybrid Arts was demonstrating its **ADAP** sampling system, a high-end (read, expensive) audio processing tool for serious musicians. Also on display were the **SMPTETrack, CZ-Android, DX-Android** and **Gen-Patch** programs. And of course, Hybrid's **MIDI Maze** game kept scores of fair-goers busy trying to blast each other into submission.

Dr. T's Music Software had some new products on display, including the **MPE** (Multi Program Environment). This powerful new program is similar to Switcher on the Macintosh; it lets you load any four Dr. T programs into memory and switch between them instantly with the click of a mouse button. According to Dr. T spokesman Al Hospers, MPE will also allow users to access features of their **KCS** sequencer *while using another program*. For example, you could adjust your synthesizer sounds with a patch editor as the music plays—a terrific idea. Another interesting new program is the **Programmable Variations Generator** (PVG), which creates variations on a prerecorded musical part. The PVG lets you specify a wide range of options such as variations in pitch, dynamics, rhythm, etc.

Users' groups represented included DASH, PACE, A-BUG, J-BUG and RHODE ISLAND ACE. Computer dealers/retailers at the show included Compuclub (who shared a booth with Berkeley Microsystems, makers of a build-it-yourself hard drive kit), Best Electronics, Bit Bucket, Computer Bug and Software Connection. Many dealers and exhibitors donated items to be given away as door prizes. One lucky fellow walked off with a Magic Sac cartridge. Not bad for a $5 admission price!

Many conferences were called in the two seminar rooms, with varied topics like "Word Processing for the Power User." After Atari discussed its marketing and price policies, a number of users got the feeling the only thing missing from the presentation were pom-pom girls chanting, "The Mega ST is *not* overpriced". Neil Harris spoke about the history of the Tramiels' takeover of Warner Atari, telling humorous anecdotes about warehouses full of unsellable software and computers.

Bill Teal of Avant-Garde (PC-Ditto) and Dan Moore of Data Pacific (Magic Sac) presided over a conference with the title "All-Star Software Hacking." They discussed issues such as copyright infringement (did you know that in some European countries, computer software cannot be copyrighted?) and programming philosophy.

Unfortunately, not everything went smoothly. On Sunday morning, we entered the show an hour before the general public (one of the privileges of badge-holder status), and immediately noticed that a tall table at the ANTIC booth was tilted at a sickening angle, leaning on the lower table next to it. The Mega ST4 and color monitor had been spared a lengthy free-fall to the concrete, caught against the second table. However, the Mega's keyboard lay on the floor, and atop it, upside down, were two hard disk drives. The sight gave new meaning to the phrase "hard disk crash," and as a result of the accident, many of the large animations could not be run that day.

But all in all, it seemed that most of the attendees had a good time. The three of us writing this article had never before met in person. Our previous encounters were limited to a few phone calls, and conferences on Delphi. It's strange to travel to a place you've never been before, to meet people you've never seen face-to-face, and yet feel like old friends.

For many of us, the hectic pace of the show seemed to carry on into the evenings. We'll not forget the confused look on a waitress's face when, confronted by a mob of computer people, she was given the number in our party by (PaperClip programmer) Dan Moore. . . in hexadecimal! ("Waitress, we reserved a *hex* table for 8:30. . . party of 1C.")

Nor will we soon forget the way the term "blittered" came into being, or the way conversation would leap from "horizontal blank interrupts," to how the word *kludge* is pronounced, to the most important thing to do during a major earthquake (park your hard drive heads—in case anyone asks you) to "Clayton Walnum And The World's Worst Banana Cream Pie." Justified comments were made to the effect that most of us acted like we were "just out of high school."

And last, but not least, we had the most fun meeting some of *you*, the readers, in person, instead of via printed page or terminal screen. That, in and of itself, was the perfect capper to a great show. ///

Andy Eddy, Charles F. Johnson and Maurice Molyneaux are all long-time users of Atari computers, and regular contributors to **ANALOG** *and* **ST-Log**. *Their collective experience covers cable television, programming, hang gliding, music, game design, animation and graphics, in addition to various other sundry activities we won't mention here. This article is the first combination effort for the trio. Their next project will probably have something to do with meeting to engage in studies of the long-term effects of getting pleasantly "blittered."*

Data Manager ST

TIMEWORKS, INC.
444 Lake Cook Road
Deerfield, IL 60015
(312) 948-9208 (800) 323-9755
Medium resolution $79.95

by Scott Wasser

Ever feel cheated by a software purchase that didn't live up to the claims on the packaging? Well, rest assured; you won't have to worry about that problem if you choose **Data Manager ST** to handle your database needs. Not only does it live up to its advance billing, it goes one step further.

Buyers of **Data Manager ST**—and all other Timeworks products, for that matter —have access to a toll-free technical support help line. Don't underestimate the importance of this service, especially when dealing with a productivity package. Even the most well written and detailed instruction manual is no guarantee that you won't have some questions about a program's operation.

It's comforting to know a software publisher is willing and able to answer those questions for you, particularly when you have just plunked down a sizeable chunk of money for their product. The folks at Timeworks seem well aware of this. Their technical help line adds new meaning to the term *user support*.

I called that help line with a few questions about **Data Manager ST** during the course of evaluating the program for this review. The technical staff person I spoke with seemed to be quite familiar with the program and answered my questions satisfactorily. But, when we began discussing a problem I was having with the setup of a somewhat complicated database, she offered to put me in touch with someone who had even more experience with the program.

When that person called me back, she had **Data Manager ST** booted up right in front of her. This made it possible for her to work out a solution to my problem while we spoke. Within minutes, she suggested a format for my database that was efficient and would do everything I wanted it to do.

That pretty much describes the program itself. **Data Manager ST**—which can be used by itself or interfaced with Timeworks' word processor and spreadsheet for the ST—is a simple to use, yet powerful information storage and retrieval system. It's capable of generating labels, customized reports and a variety of graphs from the information in your database. The program comes on one single-sided disk and can be used in a two-drive setup.

Data Manager ST falls into the *hierarchical*, rather than *relational*, category of databases. Hierarchical databases are easier to set up and use than relational ones, but they require fairly rigid and inflexible formats for data storage. These rigid structures limit a hierarchical database user's ability to create interfacing relationships between different fields or tables of information.

Data Manager ST, however, offers a feature that overcomes this shortcoming to an extent. You can create special calculated fields within your record, which take information contained in other fields and generate data based on them. These calculations can be simple mathematical formulas or logical statements. The whole process works in a fashion similar to a spreadsheet and can be very handy, depending on the type of database you need.

Another **Data Manager ST** feature that's usually found only in relational databases is the ability to display records in column form. With one keystroke or click of your ST's mouse, you can go from index card style to columnar display. The index card style display shows just one record at a time, while the alternative shows all records within a particular file (limited, of course, to screen size).

The third way in which **Data Manager ST** overcomes the traditional limitations of a hierarchical database is in the way it allows users to manipulate record formats. Fields and columns (the categories in which information is stored on a record) can easily be relocated or resized using the ST's mouse. They can also be selectively "hidden" or deleted if they contain information you don't want to appear in a printout, or that doesn't need to be seen every time a record is called up.

It is **Data Manager ST**'s design that allows this flexibility. Unlike most other file managers, which constantly access the host computer's floppy or hard disk, **Data Manager ST** operates completely in RAM. This design is what makes it possible to manipulate records and forms so freely. Because it takes advantage of the ST's inherent speed, this database is faster than most when performing important database functions such as searches and sorts.

There are, however, some disadvantages to the design of **Data Manager ST**. For one thing, any database created with it can be no larger than the available RAM of the computer you're using. This means that someone using a 520ST won't be able to create a database as large as someone using a 1040ST. It also means that 520ST owners with double-sided drives won't be able to take full advantage of a disk's storage capacity. Also, since a database must be contained on one disk—even a double-sided disk is capable of storing only 720,000 bytes of information—1040ST owners won't be able to take full advantage of their machine's available RAM.

Data Manager ST's RAM-dependent design poses a couple of other potential problems. The larger a database, the longer it will take to load and save. This wasn't a real hardship with the relatively small sample databases I created for test purposes, but could prove to be a larger headache for someone wanting to store great quantities of information.

Data Manager ST users would be well advised to remember that the information in their database is also very volatile while it's in RAM. An hour's worth of editing could easily be wiped out by a microsecond power glitch. For this reason, it makes sense for **Data Manager ST** users to get in the habit of regularly saving their data to disk while working with the program. This is a simple task and should be only a minor inconvenience during most work sessions.

The creators of **Data Manager ST** did their best to make sure that those work sessions are pleasant ones. The program's full GEM interfacing and logical command structure make it simple and fun to use, even for serious business.

The 161-page manual, which features a tutorial that guides the user step-by-step through the creation of a sample database, is written in clear, concise and easy to understand English. It contains an extensive index and a quick reference section. The manual is excellent despite one shortcoming: it does skimp a bit on technical aspects of the program, in an effort to keep the reading simple and easy to follow.

The bottom line, however, is that it succeeds at its primary task of teaching one how to use **Data Manager ST**. Even a newcomer to computers—or the ST—should not need more than an hour or two to begin designing, creating and using a personal database.

Those with considerably more experience on an ST will probably be able to start using the program immediately. **Data Manager ST**'s extensive use of prompting plays a major role in making it easy for new users to jump headfirst into the program. Operators are guided through most major tasks by screens that ask for specific responses.

And just in case you *still* manage to get stuck, you can bail yourself out by accessing one of the many detailed and easy-to-reach help screens that are always just a click of the mouse away.

Whether you're the type who'll scrutinize the manual before trying **Data Manager ST**, or who'll boot it up as soon as you open it, you'll soon discover a program that will deliver just about everything you need in a database.

The program will perform virtually all the tasks that make databases so valuable. It's capable of sorting alphabetically, numerically or chronologically, in increasing or decreasing order. You can search for information using any of the categories on a given record, and search by range or specification—while matching or not matching specific search criteria. I found all of these features easy to use and totally dependable.

Once your database is established, you will be able to generate customized reports and/or mailing labels and tags. You can even create pie charts, bar charts and other types of graphs, using **Data Manager ST**'s built-in graphics capability. Again, these features performed exactly as the program's documentation indicated they would. I was particularly impressed with the black-and-white charts (the program won't print color charts) I generated with **Data Manager ST** and an Okimate 20 printer.

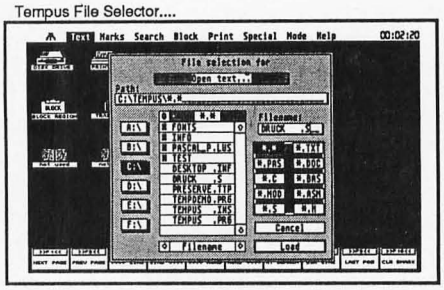
In fact, I was very impressed with **Data Manager ST**—which isn't copy protected—as a whole. Using it, I created a database comprised of rock'n'roll albums. Each record in my database contains—among other things—the album's name, the artist and one of the songs. By entering every song on every album, I've got a database that's perfect for a music nut. Using **Data Manager ST**'s search functions, I can quickly locate the name of the album on which a certain song is contained, or I can find all of the albums made by a particular artist.

Granted, your database needs may not be as demanding or esoteric as mine, but **Data Manager ST** is so easy to use it's practical for even the simplest databases, yet powerful enough to handle more difficult tasks. When you factor in the reasonable price (it's regularly advertised for $55 or less) and the help that's available from a top-notch technical staff, you'll find yourself looking at a program that's just plain tough to beat. ///

Scott Wasser has been a daily newspaper reporter and editor for the past eleven years, and has been interfacing with computers for the past three. He has written columns and feature stories about computer hardware, software and home electronics, and takes pride in writing fair and thorough evaluations of the software he tests for ***ST-Log****.*

SOFTWARE GEMS

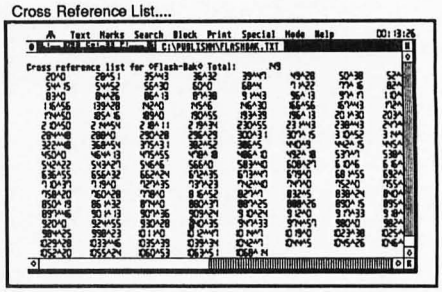
Tempus File Selector....



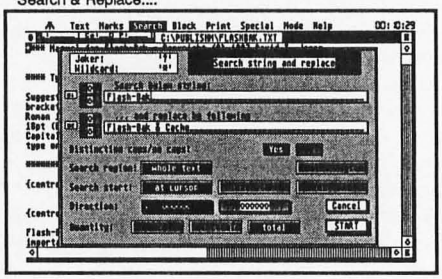
TEMPUS

Only $49.95

Cross Reference List....



Search & Replace....



Tempus is a text editor with a difference! Most text editors are left waiting for GEM to update the screen, while Tempus zooms ahead thanks to specially written, speed optimized routines. For instance, Tempus can search and replace some 4000+ characters in about 7 seconds and scroll a 50K document from top to bottom in around 15 seconds! Tempus works in a standard GEM environment but at around 300-400% faster! Tempus is fully compatible with most common language shells and is a must for ANY serious programmer!

Listed here are just some of Tempus' outstanding features:

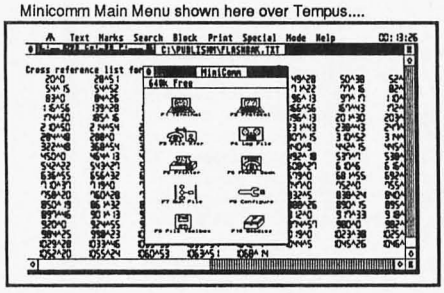
• Work on up to 4 documents at once • High-speed scrolling and searching • create indexed cross-reference lists • Programmers RPN calculator • On-line help • Fast block move/copy/delete operations • Enhanced File selection boxes • All mouse commands available from keyboard • Auto backup when saving • Configurable to most printers • Load GEOS/Degas™ fonts • Auto indentation • Visible clock • 20 programmable function keys •

MINICOMM

DeskTop Communications

Only $39.95

Minicomm Main Menu shown here over Tempus....

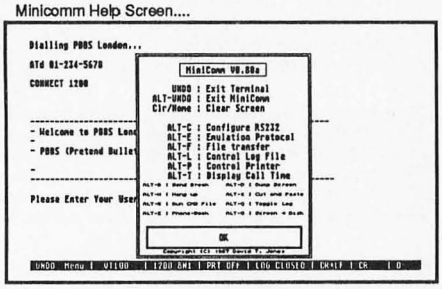


Minicomm is a complete communications package in a single Desk Accessory, making it available at almost any time during your normal day-to-day work. Minicomm features advanced auto-dialling and background file transfer.

Minicomm features:

• Runs as a Desk Accessory so is available over any GEM™ program • Background File Transfer • Background Re-dialler • 80 entry phone book • Spool data to file or printer • Script commands to control Minicomm •

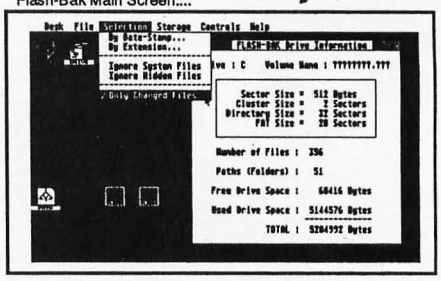
Minicomm Help Screen....



FLASH-BAK & CACHE

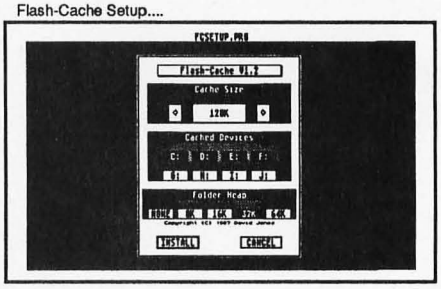
Only $79.95

Flash-Bak Main Screen....



The Flash-Bak and Flash-Cache package of Hard Disk Utilities makes working with any ST hard disk that little bit easier. Flash-Cache sets aside computer memory for storing hard disk information to speed up access times.

Flash-Cache Setup....



Flash-Bak is a complete hard disk backup/restore program which features backup/restore of selected file types, by date stamp or files used since last backup. Data encryption with a key is possible for security. Flash-Bak is also fast - up to 1Meg/Minute!

FOR ORDERS & CATALOG REQUESTS
1-800-992-9198
BIX:CONF:EIDERSOFT
GENIE:EIDERSOFT

EIDERSOFT
EIDERSOFT USA, INC. · PO BOX 288
BURGETTSTOWN · PA 15021

DEALERS/DISTRIBUTORS
Please call us now on
412-947-3739

CIRCLE #123 ON READER SERVICE CARD

ChkDsk

A SPECIAL INCLUSION

Try DOS-level repair for those trouble spots.

by Dan Moore and David Small

ChkDsk brings you a program that's a powerful tool for exploring and repairing floppy/hard disks at the DOS level. It takes most of the work out of getting to know FATs and directories.

This article is a brief summary on the program's use. The program itself was too long for inclusion in these pages; it can be found on this issue's disk and on the ANALOG Publishing Atari SIG on Delphi.

How it works.

You run the program CHK.TTP by double clicking on it. You'll then be asked for options. You must *at least* tell it which drive to check over. For example: at the prompt, enter *c:* and hit RETURN to test drive C.

You can specify several options with *-option* (a hypen and the option). Only one option at a time, please. Example: at the prompt, enter: *c: -v* and press RETURN.

Note: don't be worried by the techo-talk in this summary. All these terms are explained later.

The options, which are also explained in detail later in this article, are summarized below. (You can get a summary when running the program, by pressing RETURN at the prompt.)

-f — fix disk. Does a normal check, then converts lost cluster chains to files, which you may then delete.

-m — map disk. Does a normal check, then produces a map of the disk clusters, used/empty.

-s — statistics. Just gives you the basic statistics of the disk (size, and so forth). Does *not* do a check of the disk.

-t — talkative. Gives the drive's statistics and *does* do a normal check.

-v — verbose. Does a normal check and displays each file's cluster chain.

-(number) — Does a normal check and tells the name of any file using that cluster. Note: the "number" is hexadecimal, and must have a *0* in front of it if it begins with a letter from *a* through *f* (since those letters are options).

Introducing *ChkDsk*.

Most people have no problems formatting, copying or deleting files, once shown how. But let's face it. . .They really don't like learning the Innermost Dark Secrets of the disk: the File Allocation Table (FAT), the directory, and, dread of dreads, the boot block.

Why? Most people find FATs and directories pretty intimidating.

And for good reason. They are complex, contain lots of hexadecimal entries, and are meant for machines to use, not people. They don't make good reading material.

What's needed is a translator: a program that reads all this machine-coded information and gives us a nice, English display of just what's going on in the disk.

I'll bet you've already guessed what **ChkDsk** is about—a nice, easy way to see what's really happening on your disk, be it floppy or hard.

But the CHK program is more. It is a partial disk repair tool. You're going to want to team it with a good sector editor (of which there are many, some public domain, some commercial). If you run into bad disk troubles, CHK can set your disk right without the usual drop-dead fix, the dreaded reformat.

And, finally, CHK gives you confidence in your data from one day to the next. It discovers hidden problems when they happen. Run it once per day, and you'll have a lot more reassurance that your system is working right and your data is intact.

Let's dive in and learn how to use it.

The directory and FAT.

How does the Disk Operating System (DOS) figure out

where a file is on the disk? With two tables, called the directory and FAT.

They really aren't that complex; let's find out a little about them. By the way, if you want to learn every last bit about the directory, any good IBM PC reference (such as Peter Norton's) will tell all. This is meant more as an overview, to get you oriented.

Each disk contains 80 tracks, with 9 sectors per track, and 512 bytes per sector. Hence, 360K per disk side (720K double sided). Now, let's organize those 720/1440 sectors into "clusters." A "cluster" is just a collection of sectors, grouped together for convenience; in fact, as of now, just forget about the "sectors" part of it. On the ST, each cluster is 2 sectors long, or 1K. (On other systems, notably IBM, this varies.) We have either 360 or 720 clusters per disk, each one composed of 1,024 bytes.

Now, remove a few for the directory and FAT, and we have the rest dedicated to storage. And yes, remove one cluster DRI forgot about. (True!)

Now, how do we "chain" a bunch of these 1K clusters for, let's say, a 46K file? Good question.

Those of you familiar with 8-bit Ataris will remember that the last 3 bytes of each sector were used to point to the next sector. For instance, if the first sector was number 33, then the last 3 bytes of sector 33 would point to the next sector to be used. Typically, they're in order, so we'd see: *33 → 34 → 35*, and so on.

The ST's DOS uses the same idea, but doesn't store the actual cluster number in the sector data. Rather, it gathers all these numbers together into a one-dimensional array, called a FAT.

Each FAT entry corresponds to one cluster and is 12 bits long. There are 5 sectors dedicated to the FAT, and they pack all the FAT entries together as tightly as possible into this small space. Result: lots of bit shifting, as *12* is not an "even" number to the ST.

Anyway, don't reach for your hexadecimal calculator yet. CHK does all the dull bit-fiddling for you, and returns the FAT to what it really should be—a one-dimensional array.

If we were in BASIC, we'd make a FAT like this (assume we have 360 clusters):

```
DIM F(360)
```

Tough, right?

Now, let's assume we have a file named DAVE. DAVE is 40 clusters long (40K). How does the operating system find the start of DAVE, then trace through all 40 clusters? Like this:

First, we look up DAVE in the "directory" (just a set of filenames on your disk—and, just as importantly, the first cluster or FAT entry in that file). So, let's say that the directory entry looks like this:

```
DAVE        First Cluster: #23.
```

Okay, so we go read cluster 23, for 1,024 bytes of data. Where's the next 1,024?

For this, we go to the FAT. We look at the FAT entry for cluster number 23, or *FAT(23)*; it says *24*. This is the next cluster in the chain.

We go read cluster 24 and, to find the next cluster, we look at FAT(24). And so on. Finally, we'll reach a FAT entry that says *end of file*. (For you hex types, it's $FF7 through FFF—12 bits, remember.) DOS, at this point, knows the file has come to an end and quits reading.

Hence, the FAT is a collection of "pointers" to next clusters; a given cluster's FAT entry just points to the next cluster in the chain.

Thus, the term *FAT* is a little misleading. *NEXTCLUSTER* would have been a better name, but where would we computer professionals be without weird acronyms? However, a FAT *does* end up being a kind of disk map, and it's the only one we've got.

As you can probably guess, when you delete a file, DOS goes through the cluster chain and carefully marks each cluster as "empty" once again, so that new files can use them.

Finally, there is a special "FAT" entry for sectors that are physically bad. Most hard disks today typically have a few bad sectors; it isn't any big deal, as long as the number doesn't grow. At format time, DOS marks those clusters as bad (hex freaks: $FF0-$FF6). Then, from that point on, DOS thinks of them as "used," and never tries to access them or use them in a file.

Now, let's take a typical problem: a system crash during a disk directory access. Let's assume we create a new file, using up a bunch of clusters (and FAT table entries), thus marking them as "non-empty"—but no directory entry ever gets written to point to the first member of the chain. What happens? Those sectors are marked as "used" (nonzero), but, since there's no file associated with them, they can never be marked "unused," or returned to the free sector pool. These are called "lost clusters."

Or, worse: DOS becomes confused (seems to happen a lot) and "cross links" two files together. This is where the same cluster shows up in two files' cluster chains. Now, only one of them is right—and one of the files has bad data in the middle of it.

Though this shouldn't happen, it does. DOS crashes for a variety of reasons at very bad places—the 40-folder limit is a good example. Cross-linked files can happen quite easily.

Well, the first thing you'll see when you've got a cross-linked file is something like a TOS error 35. But the damage may be more subtle, turning up in a piece of code that is only rarely executed, or in a data table you haven't accessed in a while. Who knows, maybe that fourteenth file in your fifteenth subdirectory is cross linked with your DEGAS program file. Without **ChkDsk**, you'll never know about the damage until you try to access that data.

There's an even more horrible possibility (are we having fun yet?), called a "circular link." Here's a typical example:

```
Cluster 45 → 46 → 47 → 45 → 46 → 47. . ., etc.
```

When DOS tries to delete such a file, it's got a problem. I'll leave the results to your imagination.

CHK to the rescue.

So, let me introduce you to CHK. CHK gives your disk a thorough test, looking for things that have gone wrong at the DOS level. If it finds anything, it will let you know; you can then use many of the CHK options to fix the prob-

lems. At least you'll know something is wrong—and the file that's damaged. That beats the alternative!

When you run CHK, it gives you the standard desktop *TOS Takes Parameters* prompt. You type in, at a minimum, the disk drive you want to check over. If you just press RETURN, CHK will give you a short summary of how you run it, to jog your memory (a nice feature).

When you specify that drive number, CHK gets to work.

First, it creates a new temporary "FAT" in memory, with all clusters empty. It traces each file's cluster chain on your disk, and makes sure none of the clusters are cross linked (used in another file). It also checks for circular chains. Then it marks those clusters as used in the temporary FAT.

Next, it compares the temporary FAT, a kind of real-world FAT, with the actual FAT on disk. Let's say the real FAT has cluster 23 marked as used, but that sector shows up as free in the temporary FAT. Because no file uses it, it's an "orphaned" cluster. DOS won't use it because DOS has it marked as being in use, even though the cluster is doing nothing but taking up space. CHK will also see if 23 points to another cluster—a chain of orphaned clusters!

CHK then produces a summary of your disk drive as it really is. It tells you the total capacity of the drive, the number of folders you have, the number of bytes taken up by those folders (overhead), the number of bytes taken up by programs, how much of your disk is physically bad, and, finally, how much room you have left on the disk.

Go ahead and run CHK a few times with no options. It won't change anything, or do anything except observe (no writing), unless you explicitly tell it to, so there's nothing to be afraid of.

Special note for you hard disk users: the 1986/1987 version of the ROMs has a bug known as the "40 folder" bug. Basically, if you "touch" more than 40-folders in any session (RESET to RESET), you crash. "Show Info" touches every folder in any drive it does a Show Info on, so if you have more than 40 folders, you've got a problem.

Anyway, this is pretty well known, and I mention it only because I wanted to tell you that CHK does not aggravate the 40-folder problem. In fact, if you have a disk with more than 40 subdirectories, it may be the only safe way to get a Show Info of that disk.

Okay, let's assume we found some orphan clusters. How do we go about telling DOS we want them back?

CHK -F: (Fix) Orphan Fix.

If any orphaned clusters are found, CHK will produce a warning message. But you've still got no way to recover them (to mark them as empty so another file can use them). What you need to do *now* is run CHK with the parameters *(drive): -F* (where *(drive)* is the drive identifier). The *-F* option will create some temporary files. Each temporary file (named: FILE1.CHK, FILE2.CHK, and so on) points to one lost cluster chain.

What you should do at that point is look carefully at these FILEs. They may be important data that has been accidentally lost. For instance, a system crash that orphans a chain may still leave you with good data on the disk—you can edit and copy the FILE files.

If, as is common, you find only trash in the FILE files, just delete them. DOS will then do what it should have done in the first place—return them to the "free sector pool," or mark them as empty.

CHK -V (Verbose).

CHK will, if you ask, produce a listing of each and every cluster in the sector chain. It's something you'll want to do once, to see how a disk is really laid out. Be forewarned: this listing can be gruesomely long on a big disk drive, like a hard disk.

This option can also be helpful in determining exactly which clusters are used by a file, such as for direct sector editing. The formula for converting clusters to sectors, however, is not trivial.

At the end of *-V,* you get a map of your disk, with each cluster marked as either used or empty.

CHK -T (Talkative) -S (Stats).

Not being content with writing a merely useful program, we decided to give you the "boot block" information, as well. This is the data out on the very first sector of the disk, that tells the operating system critical values on said disk, like: how many clusters are there, what the "name" of the disk is, how many sectors are there per track, how many sides, and so on.

If you run CHK with the parameter *-S,* you'll be treated to all this information, and nothing else. If you run CHK with the parameter *-T,* you'll get all this information *plus* the usual CHK of your FAT and directory.

CHK -M (Map).

There are times when having a map of the free areas on a disk can be quite useful; say, when using hard disks. Hard disks suffer from an extremely slow empty-FAT search. When writing to a hard disk, the farther away from the start of the disk you get, the longer it takes. We're talking up to 30 seconds longer on a hard disk!

Hence, if you concentrate your present files toward the end of the hard disk and leave the first clusters free for your work, you'll get very fast write response from your hard disk.

CHK with the parameter *-M* gives you a map of each and every cluster on the hard disk. Each entry, if empty, is marked *EM.* If it points to another cluster, it's marked with the number. If it's a bad sector or end-of-file mark, it's appropriately marked.

What you want on a hard disk is mostly EM clusters toward the start of the disk, where your temporary files will be, and then used sectors toward the end.

For more information, and a technique to get back your hard disk's performance when it begins to slow down (as a result of this phenomenon), see "Restoring Your Hard Disk's Performance" in next month's **ST-Log**.

The cross-link blues (CHK - number).

Go ahead. Walk up to any IBM programmer, and say "Cross-linked file." Be prepared to catch them as they faint. Smelling salts are helpful.

I don't know how it happens. Maybe cosmic rays, maybe some Macintosh enthusiast with a voodoo doll projecting bad karma my way. But files can accidentally manage to use the same cluster. And, since a cluster can only point

to one other cluster as the "next in the chain," your files are basically the same chain from that point on.

Then you've got a problem. You've got to get rid of all the files using that cross link (because, odds are, they're all screwed up). But, when you do the first delete that goes into the bad area, it'll delete all those clusters—and then, the next delete you do (on the second file linked in there), wham—DOS will become unhappy with you. *That cluster is already deleted!*

Plus. . .you may have multiple files, all ending up in that same chain. You need to know each and every file that's butchered, so you can at least know where to begin damage control.

For these, we humbly (well, all right, not all that humbly) offer *-(number)*. Number is a hex (not decimal!) cluster number. You will be told all files that access that cluster. Normally, there will be none or one; that's okay. Two or more is bad news; best go check through and delete all those files.

When you get the dreaded cross link message, run CHK with the parameter *-###* on the sector involved (like: *C: -20A*), and find out every file listed as owning that sector. Then go delete them all.

Well, there you have it, **ChkDsk**, another Dan Moore utility for ST users.

You'll find yourself using **ChkDsk** almost daily to keep tabs on your system, especially if you have a hard disk. There, it is invaluable; there's no other tool like it available. It's a great feeling to run CHK on my 15-meg drive, with some 100 folders, and find out that everything's fine in the directory department; it also gives me confidence that the backups I do won't turn out to be corrupted. When I run CHK daily, I know exactly when any file problems occur, and can fix them.

If GEM and DOS were perfect, we wouldn't need CHK. Since they are not, CHK is the utility to recover any damage they cause. ///

The author of the CHK program is Dan Moore. Dan is more or less the Tom Scholz of computer programming, especially in C. (Who's Tom Scholz? He runs the band Boston, which just took seven years to produce one album; he's known as a perfectionist.)

Anyway, Dan wrote CHK in C, and it compiles and runs perfectly on the IBM PC. He then ported to the ST, and, since CHK was machine independent, had no problems with the port—it came right over.

Lots of people write C code on the ST; very few write C code that ports directly to the IBM.

Hence, CHK is not only interesting in that it reads tables, FATs and directories directly (ideas you can use for yourself), but in its machine-independent coding techniques. If you're interested in porting your C code from the ST to the IBM, CHK is a great place to look.

David Small wrote the article ***ChkDsk****. A long-time computer enthusiast, he has a computer science degree from Colorado State University. His achievements include books, articles and the Magic Sac.*

MEGA macrocosm

A close look at Atari's MEGA ST.

by Tom Hudson

Atari has been manufacturing its line of 16-bit ST computers for almost three years now. The earliest ST models, which appeared in mid-1985, contained 512 kilobytes of memory—quite a bit when compared to the 48K limit of the Atari 400 and 800 series 8-bit computers of just six years earlier. But there was a problem. Those STs loaded their operating system, TOS, from disk, using a large chunk of memory for system overhead. This seriously limited the size of programs that could be run—even the 300K or so left over was a restriction for graphics programs needing large amounts of memory for bit-mapped images.

The ST came equipped with sockets for ROM chips which could hold the operating system, so Atari tackled this issue by early 1986, when they released the TOS operating system on ROMs. Just about every ST owner upgraded to the ROMs as soon as they appeared, and everyone was happy—for a while. It was apparent that 512K, which earlier had seemed like all the memory a person could ever want, just wasn't enough for complex programs or some of the helpful desk accessories that were now available.

At this time, Atari was developing the 1040ST, the 1-megabyte version of the ST. This machine was to be essentially the same as the 520, but was more self-contained, with a built-in power supply and double-sided disk drive. Rumors circulated widely that the 1040 would contain a socket for a "blitter," a custom coprocessor which could speed up graphics operations immensely. When

Figure 1. — MEGA components.

the machine appeared, though, there was no blitter, no socket. The 1040 did have the TOS in ROM, which was a plus, but in reality it was nothing more than a 520ST with increased RAM. It was in a package which was more convenient to move around, and had a double-sided disk drive built in, but that was about it. As a result, many people (myself included) had their 520s upgraded to a megabyte and bought external double-sided drives, rather than buy a whole new machine.

Time passed, as time does, and Atari came out with the 520ST FM—basically, a 520 that had graduated to the 1040 case. It had a built-in power supply and a single-sided floppy, making the machine just as convenient to move around (no more power supply "bricks" to worry about!) This allowed Atari to reduce the tooling needed to manufacture the computers, with only one case to mold instead of two. It also got rid of an external power supply and its case. From an economic standpoint, it was a smart move.

During this time, there was a lot of talk about Atari's long-awaited blitter chip. The older 520s and 1040s didn't have a socket for this chip, but it could still be added through an add-on circuit board. People waited with bated breath. And waited. And waited.

While they waited for the blitter, they also wanted *more* memory! It seemed that, with the new desktop publishing, CAD and graphics programs, a megabyte of memory simply wasn't enough. Two- and three-megabyte memory expansions appeared on the market. The problem was that nobody knew which of the RAM expansions were of good quality, or if they were compatible with all software.

Atari was working on the answer to the problems, which they announced at the January Consumer Electronics Show: the MEGA ST.

MEGA power.

The MEGA ST is a radical departure from the 520 and 1040 systems in terms of its design. Unlike the one-piece 520 and 1040 units, which were literally tied to the desktop by cables, the MEGAs are two-piece units (Figure 1). The main component is the CPU box, which remains on the desk, while the second component is a separate keyboard connected to the CPU by a coiled cord. The keyboard may be placed in the lap, if desired.

The MEGA ST is planned to come in two configurations: 2 and 4 megabytes. Four megabytes should be enough to keep users happy for a while. . .or will it? This amount of memory will accommodate just about any existing application I can think of, with lots of extra room. More on that later.

Included with each MEGA ST is a language disk, which contains ST BASIC and the control panel and VT-52 terminal accessories. The ST BASIC Quick Reference Guide and a 133-page manual are included. The computer manual is very well written, with sections on the blitter chip.

Each MEGA has a built-in socket for the blitter chip, which is also included. This will speed up graphics operations considerably, resulting in faster, smoother program operation. New TOS ROMs, which fix some bugs of the earlier release and allow existing programs to use the blitter, are also included. The blitter may be enabled or disabled by a drop-down menu selection on the GEM desktop.

The MEGAs come with a double-sided floppy disk drive as standard equipment. This is built into the front of the CPU box (Figure 2).

There is a convenient addition to the MEGA that you won't find on either the 520 or 1040 systems: a standard battery-backed-up clock. This is a welcome addition for people like me, who don't want to have to set the system clock every time the system is started.

Figure 2. — CPU box.

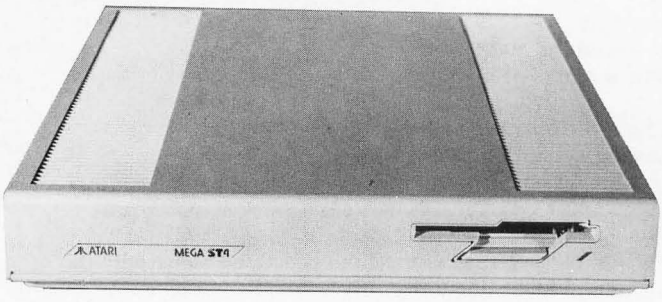
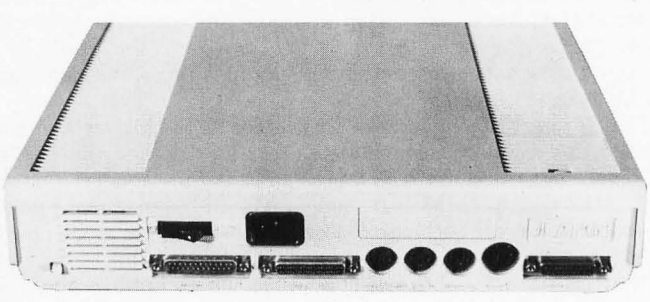


Figure 3. — Rear panel.



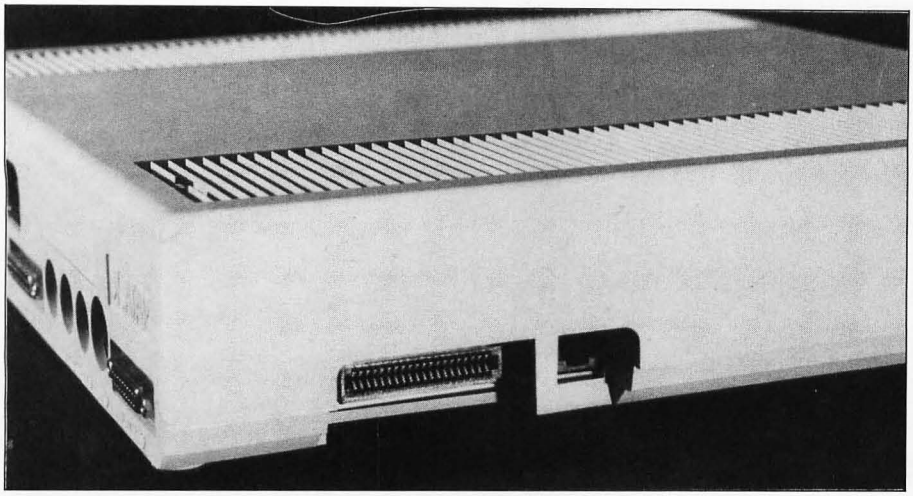


Figure 4. — Connections.

Externally and from a strict functionality standpoint, these are the only differences from the 520 and 1040 that many people will ever see. However, there are several features which make the MEGA worth a closer look.

Up close and personal.

Let's look at the exterior of the MEGA ST and see how it's laid out.

Figure 3 shows the rear panel of the MEGA ST. Essentially, this is the same as a 520 or 1040ST, with (from left to right) the reset button, power switch, modem connector, power connector, printer connector, MIDI out, MIDI in, monitor connector, floppy disk connector, and hard disk/DMA connector.

One item on the rear panel of the MEGA that you won't find on the other ST computers, between the reset switch and modem connector, is a small cooling fan. This fan is positioned next to the internal power supply and should provide welcome cooling to the rest of the system, for improved reliability. The fan is extremely quiet; if you have a hard disk, you won't even hear the MEGA's fan.

Another interesting detail on the rear section of the MEGA is the rectangular panel above the MIDI ports. This panel is removable, presumably for cabling needed for the Atari laser printer or other add-on equipment.

Access to all the rear connectors is easy, and I had no problems at all connecting my equipment to it.

Figure 4 shows the left of the MEGA's CPU box. This contains the standard Atari cartridge port (left) and the telephone-style connector for the keyboard's coiled cord. The cartridge port is made to the same dimensions as earlier ST models, so all cartridge-based software and peripherals should connect without difficulty. I have personally tested the following cartridge port add-ons, all of which connect without any problems: Hippovision video digitizer; Magic Sac Plus; Stereotek glasses; Kuma K-MAX; Hybrid Arts ADAP Soundtrack; and ComputerEyes video digitizer.

I had some initial worries that the keyboard cable would be a problem in fitting some of the units to the port, but the cord will flex out of the way, avoiding conflicts.

Figure 5 shows the battery compartment for the battery-backed-up clock. This compartment is located on the upper left rear of the MEGA's CPU box, and requires two AA batteries (not included).

A clock operated by batteries is a wonderful addition to the ST. It keeps the proper date and time (you must set it initially), so that files you create have a proper time/date "stamp" on them. This is particularly useful for compilers with "MAKE" utilities, which use the time and date information to determine which files are up-to-date.

The MEGA's keyboard unit, as mentioned earlier, is separate from the main CPU box, connected to it by a coiled telephone-style cord. The 18-inch coil can extend to over 36 inches, allowing you to sit back and type with the keyboard in your lap. If a longer cord is needed, a standard 6-conductor phone cord can probably be substituted.

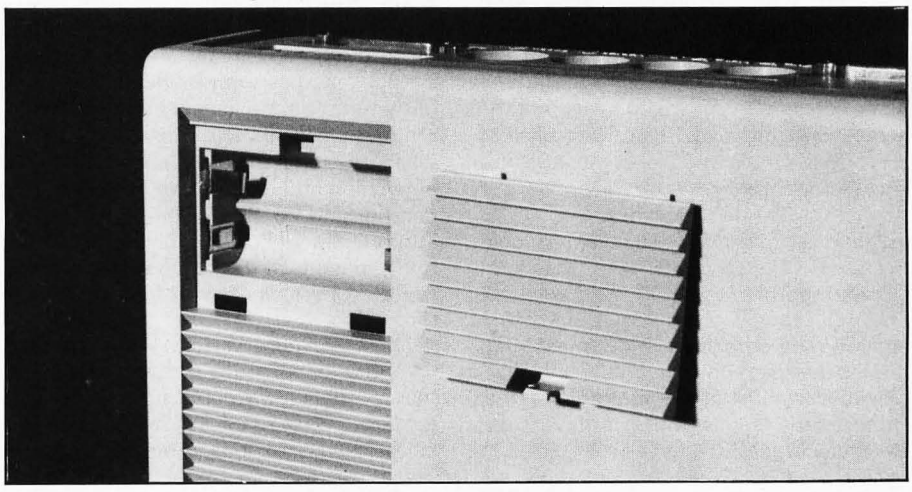
Figure 6 shows the underside of the MEGA ST keyboard. As you can see, the mouse plugs into the standard 9-pin connector in an inset area underneath the keyboard. A nice feature here is the channel which leads to the right from the inset. This is provided to hold the mouse cord in place, running to the right of the keyboard. A second inset area, on the left, contains another 9-pin connector for joysticks or other devices. Inserting the mouse and joystick plugs into the sockets is a little tricky, though not as bad as plugging them in on the 1040ST.

Inset into the bottom of the keyboard are two fold-out feet (the square units on either side). These pop out and snap into place with a nice, solid click, so that the keyboard can be positioned at a higher angle (Figure 7).

The keyboard itself has stiffer springs in the keys, helping prevent the problem many users experience on the 520 and 1040 machines, where a finger brushes against a nearby key and accidentally activates it. This probably happens mostly to non-touch typists, like me.

The keys themselves are placed and

Figure 5. — Clock's battery compartment.



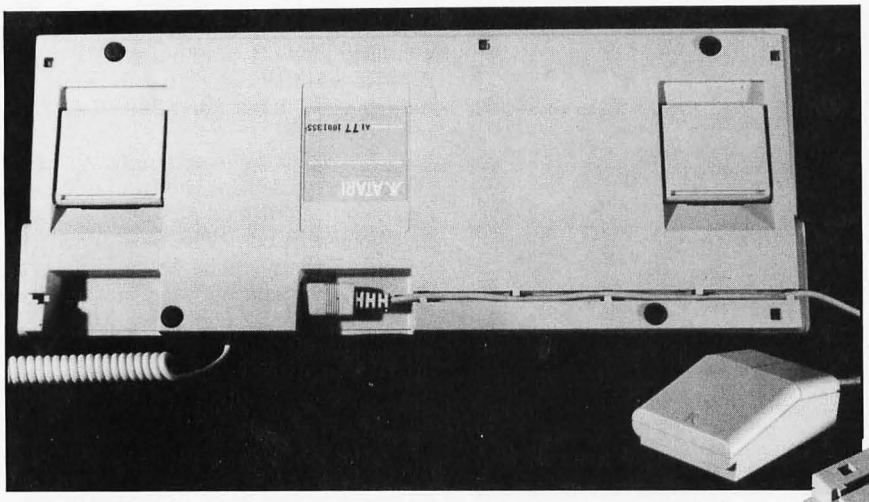


Figure 6. Under the keyboard.

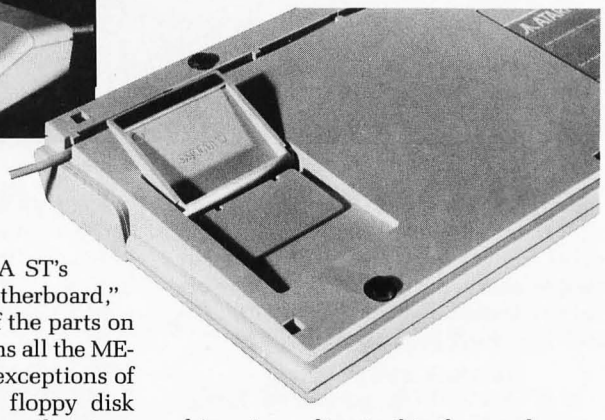


Figure 7. Keyboard feet help position the unit.

shaped identically to those of the 520 and 1040 keyboards, making it easy to move up to the MEGA from the older machines. The stiffer keys have received rave reviews from others who have used the MEGAs.

There are a couple of things I noticed about the new keyboard that may cause problems in use, however. The first is that when the keyboard feet are deployed to raise the keyboard's level, the back of the keyboard partially obstructs the floppy disk slot. The second is that since the mouse connector is located on the keyboard unit itself, if you want to put the keyboard in your lap, the mouse tends to want to come along. Removing the mouse cord from the "keeper" slot in the bottom of the keyboard can help avoid this problem somewhat, and the "joystick extender" cords sold by Radio Shack will probably help, as well.

The MEGA's design is, overall, quite good. The large CPU box is a perfect platform for a monitor, and Atari is planning a new hard disk drive the same size as the CPU box, which will allow the stacking of the components, to keep your desktop uncluttered. The unit is attractive and easy to set up. Cabling is restricted to the rear of the unit. Many people feel that the detached keyboard will improve ease of use, and the improved feel of the keys will boost productivity.

The inside story.

The real fun starts when you start looking at what the MEGA ST series has "under the hood."

Like all the ST computers, the MEGAs come equipped with a Motorola 68000 microprocessor chip, which runs at 8 megahertz. Most of the internal support chips are the same as those in the 520 and 1040 machines.

There are several differences in the MEGA ST that are worth looking at, however.

Figure 8 shows the MEGA ST's main circuit board, or "motherboard," and Figure 9 is a diagram of the parts on the board. This board contains all the MEGA's components, with the exceptions of the power supply and the floppy disk drive. The MEGA is put together in a "serviceman-friendly" way, using only nine screws for the main case, three for the floppy, and two for the power supply. Disassembly and reassembly is quick and easy, with quick-release connectors on all wiring harnesses.

As you can see in Figure 9, only the square chips, ROMs and two other chips are socketed. A reduced number of sockets cuts costs and increases reliability in the computer, but if a nonsocketed chip fails, the time and cost of repair climbs because the old chip must be carefully removed by painstaking desoldering.

You will notice that the square MMU and DMA control chips are both held in place by special metal clips placed diagonally across their sockets. This is a fairly recent improvement by Atari, which reduces problems with the chips coming loose due to vibration in shipping or the expansion and contraction caused by the heating and cooling of being turned on and off. Since the clips have been added to ST machines, the number of problems reported by users has dropped substantially.

A related observation: When the 520 machines appeared, a technician told me that computers with built-in keyboards often have problems caused by the vibration of the keyboard loosening the socketed chips. According to this theory, the separate MEGA ST keyboard would reduce these failures.

The MEGA ST RAM is located in the lower right corner of the motherboard, directly below the floppy disk drive. Figure 8 shows the motherboard of a MEGA 4, which contains thirty-two 128K RAM chips, for a total of 4 million bytes. That's a *lot* of RAM.

Directly above the floppy drive ribbon cable (which partially obscures the RAM chips) are two small connectors. One of these is the power input connector, which receives power from the power supply board. The latter sits on top of the motherboard at the upper right.

The other connector is unused, suggesting that it's for other devices which may be installed in the MEGA ST in the future. My guess is that it would be primarily for the laser printer interface board. This may be one of the considerations that prompted Atari to add the fan, since, as a power supply provides more electrical power, it generates more heat. Incidentally, the power supply in the MEGA is rated at: +5V @ 3A; +12V @ 1A; and −12V @ 30mA.

Another welcome addition—one many ST users have wanted for some time—is the simple 64-pin connector that's directly above the 68000 chip. This connector al-

lows easy access to the ST's main bus, so that RAM expansions or other custom electronic add-ons may be simply plugged in, rather than soldered in. I can see a number of fantastic add-ons that could use this connector—it's available; all we need is someone to use it! According to Atari's Leonard Tramiel, additional RAM can be added via this connector.

Above the bus connector and to the left is a socket for the blitter. Final MEGAs will have the blitter included as a standard piece of equipment, though the early MEGA in this photo has no blitter chip. The blitter should speed up block image operations and text on the ST substantially, although I have no benchmark information at this time. Generally speaking, the larger the block image to be moved, the better the hardware blitter performance in relation to the software blit routines.

The final major change on the MEGA motherboard is the 24-pin connector in the upper left corner of the motherboard, directly under the hard disk connector. This is reportedly a special, internal DMA connector, which would conceivably allow such things as internal hard disk drives. It may also be used for the laser printer driver board, which will be installed in the MEGA ST's case when the Atari laser printer is purchased.

That's it for the major internal differences between the MEGA ST and the other ST models. The only other differences are in the physical layout of the motherboard and the battery-backed-up clock, whose connector runs from the battery compartment connector at the upper left to the point at the lower left.

Where's the beef?

Okay, we've seen what the MEGA ST is, inside and out. It's got several convenience features that the 520 and 1040 machines lack, and it's got *lots* of RAM. But is it really that different from the older machines?

In a word, the answer is no. From a software standpoint, the MEGA will run any software the 520 and 1040 machines will (at the time of this writing, some reports are surfacing that a number of titles don't work with the MEGA series, most likely due to incorrect programming techniques). It will hold more in memory at one time than the other machines will, but that's about it. The hardware is essentially the same, with the same graphics modes (320x200 with 16 colors, 640x200 with 4 colors, and 640x400 monochrome). It has room inside for add-on equipment and the connectors to allow the hookup. For the average user, is it worth getting a MEGA rather than a 520 or 1040? Let's examine the issues.

(1) Memory. The MEGAs contain 2 or 4 megabytes of memory, standard. The MEGA ST2 can probably be upgraded to 4 megabytes by simply adding RAM chips to the motherboard. A 520 or 1040 must be upgraded via a third-party RAM expansion board, and some of those can only go up to 3 megabytes. These boards are priced at upwards of $600, or $200 per megabyte. Extending this to 4 megabytes gives an approximate price of $800. Adding RAM to a 520 or 1040 may involve modifications that void the warranty, and some RAM expansion methods may be of questionable quality and reliability.

(2) Blitter. The MEGAs have a blitter as standard equipment. Adding one to a regular 520 or 1040 involves installing the blitter add-on board and a new set of TOS ROM chips. The hardware for this upgrade won't be free, and having a technician install it could add to this cost. A consideration here is that if a RAM expansion was added to the ST, adding a blitter may run into a problem with space inside the computer case or a basic compatibility problem with the RAM expansion.

(3) Battery backed-up clock. Third-party battery-operated clocks are currently available for around $40. This is a unit which fits underneath one of the ROM chips and is easily installed by anyone who can use a screwdriver, so adding one to a 520 or 1040 isn't a problem.

(4) Internal connectors. There isn't much you can do about this. The older STs don't have internal connectors for anything, so if companies come out with special add-on boards for the MEGA series, they won't be as easy to add to a modified 520 or 1040. There won't be a lot of space left inside a 1040 case after upgrading the RAM and blitter, either. Plan

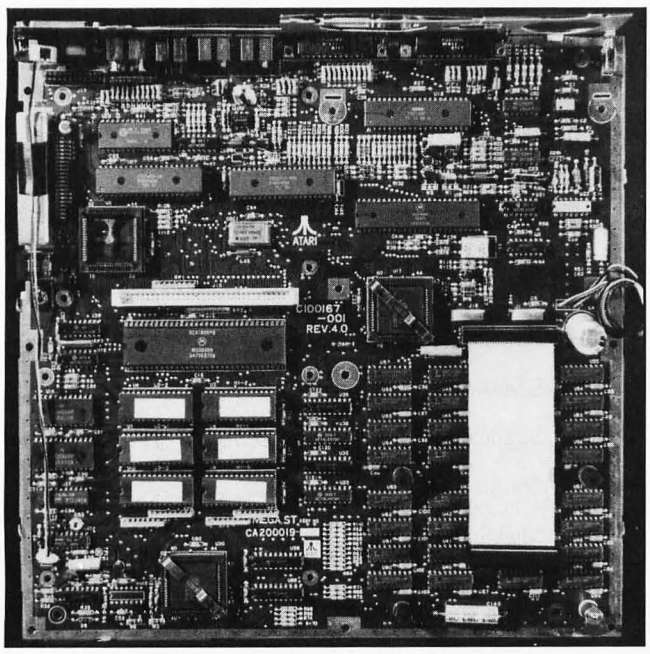


Figure 8.
The motherboard . . .

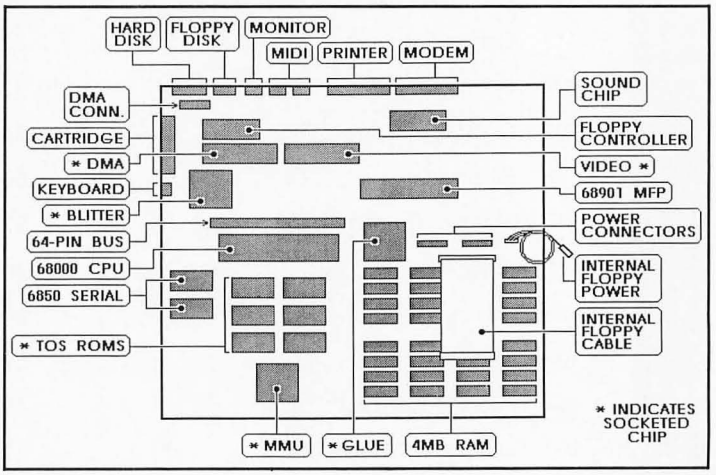


Figure 9.
. . . and its parts.

on doing a lot of soldering to add anything else in. This includes the Atari laser printer, something a lot of people will want.

(5) Keyboard. This is definitely improved, with the two exceptions mentioned earlier about the mouse cord and disk drive slot. I've heard of some people hacking their 520s and 1040s for a detachable keyboard, but this is a little extreme. The feel of the keyboard is definitely better on the MEGAs, something that would be tough to duplicate on the older STs.

These are the main points to consider when deciding whether to not to buy a MEGA ST. Based on the possible problems that could arise by tacking extra equipment onto a 1040ST to duplicate the MEGA's capabilities, and the fact that even an upgraded 1040 wouldn't have the expansion capabilities, a MEGA ST seems more logical if you plan to upgrade your system in the future.

Conclusion and ideas.

I can see a lot of applications for a MEGA ST (particularly the 4-megabyte system). On the one I have here, for example, I can set up a 1.5-megabyte RAMdisk, load up lots of desk accessories and a memory-hungry application, such as CAD-3D, which in its maximum memory configuration can use around 1.5 megabytes. The primary application has all the memory it wants, and a lot of room left over. The RAMdisk is a *fast* way to store temporary or often-used files.

Four megabytes opens many doors. You're no longer tied down to a small memory area in which to operate, and can divide up memory in many creative ways.

This leads me to a natural suggestion for the MEGA. There is a product called K-Switch for the 1040ST which divides up RAM into two parts, effectively creating two 520STs within one 1040. The user can switch from one application to another at any time. It isn't multitasking, because when you move to program 2, program 1 stops, and vice versa. But the idea behind it is a good one: Use your memory most efficiently by having more than one program in the computer at once.

As you can guess, 4 megabytes of memory could accommodate a RAMdisk and *several* application programs at one time! The MEGA ST4 needs a "Mega-Switcher" to allow users to partition memory into several user-defined parts.

Of course, Atari has more plans for the MEGA, including the eagerly-awaited laser printer and other items we can only speculate about.

With its convenience features, awesome memory capacity and expansion capability, the MEGA ST line should find an important part to play in the ST computer line for some time to come. ///

Special thanks for this article go to Larry Copenhaver of The Computer Room in Kansas City, Missouri.

Tom Hudson is a free-lance programmer who works primarily with the Atari ST series. His commercial products for the ST include DEGAS, CAD-3D and DEGAS Elite. Tom is a SYSOP in the Atari SIGs on the CompuServe telecommunications network, where his ID is 76703,4224.

INDEX TO ADVERTISERS

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

Inside the ST Xformer

A SPECIAL INCLUSION

Part 2 — the hardware.

by Darek Mihocka

Last time, we presented Part 1 of ***Inside the ST Xformer****. This, the conclusion, continues where the first installment left off. The program files described in this article were offered on the disk version, which you may order as a back issue—or you can find the listings in the* ***ANALOG Computing*** *Atari SIG, on Delphi.*

Hardware simulation.

Simulating hardware involves figuring out which hardware register is being accessed, and what to do about it. For example, if a memory location that appears on-screen is being written to, the screen display must be updated. All code for hardware simulation is in the file __XATARI.C.

This introduces a problem: how to trap the emulator when one of these special memory locations is being accessed, without introducing too much overhead. This is where the second 64K block of memory (pointed to by stat and REGSTAT) is used.

Each byte of the block pointed to by stat corresponds to a byte of the 64K main memory. The stat bytes are exactly that: status bytes which indicate the status of a byte in main memory. If the status of a memory location is a *0*, it can be read or written freely. If its status is nonzero, the access must be trapped and handled.

You may notice in the code that elements of mem are accessed through pointers, while elements of stat are accessed through an index. The reason I use an index in the second case is so that I can use the 16-bit offset mode of the 68000 to quickly check the status byte.

Some memory requires special handling for both read and write, and some memory only requires handling for write operations. For example, screen memory can be read freely without the need of any handling, while writing to the screen must be trapped so that the screen may be updated. To classify memory into one of these two categories, I use the high bit of the status byte to indicate the type of handling required. Here's a brief summary:

Status byte:

$00	Memory is regular RAM; no special handling.
$01-$7F	Memory can be read freely, but any write access must be trapped and handled.
$80-$FF	Both read and write operations must be trapped.

Note that there are no memory locations that can be written to freely but can't be read freely. Therefore, those types of status bytes do not exist.

The status numbers correspond to which handler routine must be used for that particular memory location. The array serv__hdwr is an array of 256 pointers, which point to up to 256 different handler routines.

For example, ROM locations have a status byte of 2. The array element serv__hdwr[2] contains a pointer to a routine which does nothing. In other words, an attempted write to a ROM location will result in no write at all. This is the simplest example out of the few dozen that are actually implemented.

Note that read and write operations are handled quite differently. Write operations return straight to the main loop though DISPATCH, while read operations have to return to the opcode routine to complete the read. Thus, one method exits through a JMP, one through an RTS. The variable isread must be used to keep track of this, to prevent major catastrophic stack overflows! I cannot emphasize enough the importance of setting isread.

Graphics handling.

Most hardware locations are fairly easy to emulate. For example, location 53770 will return a random number when read. The handler code at s__rnd is quite simple. The value in isread is checked by the macro TESTWRITE. If it is *0* (a write operation) a branch is made to nul, which does nothing and dispatches. If it is a read, a new randon number is generated, stuffed into memory location mem+53770, and an RTS returns back to the calling routine, usually doLDA, doLDY, or doLDX.

Note that the macros LOADREGS and SAVEREGS must be used anytime we go from assembler back into C. C uses registers D0-D7 and A0-A3 for its own storage, and so wipes out the 6502 variables.

Unfortunately, not all locations are this simple to emulate. The majority of the code in _XATARI.C is dedicated to screen handling. The Atari 800 has about seventeen different display modes and, through display lists, it can display them all at the same time, and scroll each one independently. This is simple on the 800, since the ANTIC and GTIA chips do all the work, at the same time that the 6502 is doing its stuff. It's a *nightmare* on the ST, since it must all be handled by the 68000, and as quickly as possible.

It became necessary to decide, for speed purposes, which features could be emulated in reasonable time and which couldn't. I decided that player-missile graphics, display list interrupts, and fine scrolling could not be supported efficiently on the emulator, given its already slow speed. What remains of my attempt at player-missile graphics is found at the end of _XATARI.C.

Once these features were axed, it was possible to write some very fast graphics routines, which work well with most 8-bit software. Of course, heavily graphics oriented demos still run much more slowly, but almost any display list combination possible is supported.

The graphics routines are called plot_2 through plot_F, and each one simulates one of the fourteen displayable ANTIC modes. Also, plot_F can simulate the three GTIA modes, for a total of seventeen modes.

Several problems still exist: all displayed bytes must have their status bytes modified, and, when a write to such a location is trapped, the memory address must quickly be converted into an X- and Y-location on-screen, and an ANTIC mode number—so that the appropriate plot_x routine can be called. Impossible you say?

The solution to this problem is a data structure I call DL, which is defined at the beginning of _XATARI.C; and an array of such structures called dlBlocks[200]. Each entry in dlBlocks is similar to an entry in the actual display list, except that scrolling and interrupt bits are ignored, and consecutive same bytes in the real display list are merged into one DL. The structure DL contains information such as the ANTIC mode for that portion of the display, the height in scan lines, the number of bytes per line, the number of consecutive bytes displayed, the starting scan line, and the location of the first byte to display.

The whole process is triggered by a write to memory location 559 ($22F). Remember the familiar POKE 559,0? Every time a GRAPHICS command is executed—or when the computer boots up—the operating system writes to that location to turn the screen off, and then on when the new display list is generated. This is trapped by the emulator, and sets off a long chain of events.

First, the flag dma is set to indicate that graphics are being displayed. Then a call is made to do_display(). This complex routine then traverses the display list and generates the DL structures. At the same time, the bytes making up the display list have their status bytes set to 64. Also, all displayed bytes are marked with a 65, and the plot_x routines are called. This loops until the end of the display list is reached, or the ST's screen is full. Remember, the ST can't display more than 200 scan lines. The 8-bit could display up to about 230, so some screens may get chopped off.

Once the screen is redrawn, any write to screen memory will get trapped because of the value 65 in the status byte. The routine do_byte() is called, which then quickly goes down the DL array until it finds a DL that corresponds to the screen byte being displayed. Note that this offers a significant speed increase over simply going down the display list, since a graphics 8 screen might have a 200-byte display list, but only a two-entry DL list. From the other information in the DL structure, we can then easily figure out the screen X- and Y-coordinates.

This almost completes the description of the graphics handling. When the display list is modified, or a POKE 559,0 is executed, the routine clear_stat() quickly goes through the DLs and clears the status bytes of all screen locations, so that a new call to do_display() can be made.

Player-missile graphics.

As mentioned in the previous section, player-missile graphics are not supported in this version of the emulator. Although it is not too difficult to simulate some of the memory locations required for PMG, it became obvious that the problem would be not with drawing sprites, but, instead, with erasing them. For example, when a sprite is moved on-screen, it requires a write to one memory location (on an 800). On the ST, which doesn't have real sprites, this requires undrawing the sprite by replacing the graphics that were below it, then redrawing it. Since a single sprite may occupy as much as one-quarter of the entire screen, this means that thousands of the ST's screen locations would have to be written to. I have included some code in _X ATARI.C which can be hacked on to make it draw sprites, but no code is included to erase them.

The solution will probably be the blitter chip.

This same problem also exists with display lists, but since display lists are much less likely to change drastically, it can be tolerated. However, any assistance the Blitter can provide will probably help a lot.

Joystick simulation.

Joysticks ports can't be fully implemented on the ST, due to its lack of support for paddles. Also, support for only two joysticks can be provided, and, since the ST's ports are not capable of output, plug-in peripherals cannot be used on the ST. But this is still adequate for most software, which simply reads joysticks.

To read the joysticks on the ST, one cannot simply PEEK a memory location, as on the 800. Instead, an interrupt is generated every time a joystick event occurs, whether it be the pressing of a button or a stick movement.

The interrupt routine is installed into a table of vectors known as kbdvecs. This table has nine vectors which point to handlers for such events as keyboard, joystick and MIDI input. The seventh vector is the pointer to the joystick handler. At entry into the joystick handler, A0 points to a 3-byte "packet." My routine Stick then reads two of the bytes which give the current status of the two joysticks.

The first byte, telling us which joystick generated the interrupt, is ignored, since it is faster to just read both bytes than to do extra processing to determine which byte should be read. The interrupt is switched on and off with the routines JoyOn and JoyOff, found in _XATARI.C.

Keyboard simulation.

Although the ST handles the keyboard in the same manner as the joysticks, it was almost unnecessary to write a keyboard handler, since I can easily call Bconstat() and Bconin() to get keys from the keyboard. The problem is that all the nonshift keys auto-repeat on the ST keyboard, and there's no way to tell if a key is still being pressed. We can only find out when it *gets* pressed. This presents a problem when trying to emulate the START, SELECT and OPTION keys, since they clearly do not and must not auto-repeat. Also, we have to know at any point in time, if any of those keys are still pressed down.

The keyboard handler has three routines: Install_Key(), Remove_Key(), and KeyPatch. Install_Key() installs KeyPatch as the ninth vector in the kbdvecs table. Remove_Key() un-installs it.

KeyPatch loads A0 with *$FFFFFC00*, which is the address of the hardware register where the keycode appears. It then reads the keycode and compares it against a list of keycodes for the keys F7, F8 and F9. Fortunately, an interrupt is generated both when a key is pressed and when a key is released, making it possible to monitor the state of the keys.

At address $FFFFC00 is a device known as an ACIA, which handles all joystick, mouse and keyboard events. It generates a code from $00 to $FF. What the ROM keyboard routine does is check the keycode; if this is $FE or $FF, it calls the joystick handler. If the code is $F6 to $FD, it calls the mouse handler and anything else is treated as a keycode. Note that this limits the maximum number of keys on the keyboard to $76 or 118. It also means that the joystick and keyboard routines could probably be merged into one routine. Any takers?

Vertical blanks.

Vertical blank interrupts were a bit tricky to implement. The problem was that every sixtieth of a second, the emulator had to somehow drop whatever opcode it was about to simulate and jump into a vertical blank routine, followed by a deferred vertical blank routine, and then go back to the original opcode.

The key lies in the DISPATCH macro. Note that, since pemul is already used to divert the dispatcher, it can be made to divert it straight into a vertical blank routine.

There are actually two vertical blank routines. The first, called VBI, does a few things the Atari 800's system VBI does, like incrementing the real-time clock and checking joysticks. This way, the real-time clock (locations 18, 19, 20) retains its accuracy, which keeps programs that depend on it up to speed. VBI then also changes the pemul vector to point, not to emul but, instead, to a routine called sysvbl. Then, when the current opcode being simulated finishes, the DISPATCH macro jumps to sysvbl, the second VBI routine. In that routine, we do things like update the color registers and check the keyboard.

Finally, we simulate a 6502 interrupt by pushing the A, PC, and P registers to the 6502 stack. Then we make an indirect jump through the deferred VBI vector ($224) to a routine which must end in an RTI.

This all results in vertical blank routines that run at real time. So things that most games have, like background music, will play at normal speed, even though the game itself plays at 20 percent of the speed. Some games which are totally VBI based, like **ANALOG Computing**'s Maze War (issue 36), will run in real time.

Operating system simulation (P: and D:).

One of the main problems with emulators is their slow speed. It's a lot easier to run the real thing than to try to translate the code on another machine, even a faster one. One way to increase speed is to take commonly used pieces of code and replace them with simulation routines. For example, the CIO call in the Atari 800XL operating system could be trapped, and a C language routine executed instead. This would actually increase the speed of the emulator so that, if the operating system is called a lot, it could run faster than the real computer. In fact, by replacing the whole operating system—and BASIC—with simulation routines, the emulated version could run many times faster than the real computer. Imagine BASIC XL running at ten times the speed.

Back to reality. Operating system call trapping could be implemented in the same way as hardware trapping, using the same serv_hdwr array. I haven't done any such trapping in this version of the emulator, because this method is slower than another simpler method: using the one-hundred or so unused opcodes to call emulator routines.

The principle is very simple. Suppose that one wanted to rewrite the output routine for the E: device. That's the well-known $F6A4 entry point. By putting an invalid opcode at that location, say opcode $FF, and making the appropriate entry into the vec_6502 array, anytime the program counter reaches $F6A4, it loads the opcode $FF and jumps to our new routine, instead of executing the original code in ROM.

This is exactly how the P: device is emulated. In the routine InitMachine() is code which places these unused opcodes at the six entry points to the P: handlers. Five of these are patched with the opcode $7F, which simply stuffs a value of 1 in the Y-registers and returns to CIO. The P: putchar handler is patched with $6F. That routine (op6F) then calls Bconout to print through the ST's printer port.

In a similar—but more complicated—manner, the D: device is emulated. The opcodes $0F, $1F, $2F and $3F are used to divert CIO to the routines that simulate OPEN, CLOSE, PUT and GET. Each of the routines makes the appropriate calls to GEMDOS and exits with the 6502 memory set as if a real DOS routine had just executed.

The patches for the D: handlers are actually made to the C: device handlers, and then C: is renamed to D: in the 6502 ROM. Best of all, we get the benefits of DOS without having Atari DOS loaded into memory, so most programs will enjoy about 5K more space. Similarly, the devices E:, S: and K: can be patched to call our own routines. We could even install new drivers, like R: for a modem.

What can be done to speed things up.

As I already mentioned, the code could be expanded for some speed increases. Also, many of the C routines could be rewritten entirely in 68000 code, but that will result in huge source code.

As also mentioned, by emulating the entire operating system, all calls to any device would be much faster. For example, the screen editor could run at real time, and even faster. Plotting and line drawing would be lightning fast, since Line A could be called. The floating-point routines could be rewritten in 68000 to execute at ten times their normal speed. One could even go as far as to rewrite Atari BASIC in 68000. Imagine BASIC running at ten times its normal speed!

Of course, all these improvements will take time and will still not solve one problem: any program that doesn't call the operating system (and many binary files don't) will not speed up at all, since all of its code will still be interpreted. Also, some BASIC programs will probably be unusable at ten times their normal speed.

But what about programs running on the current emulator? If they're in BASIC, they most likely get their timing from FOR NEXT loops. All that has to be done in most cases is to trim the loops by about a factor of 5. If the program gets its timing from the real-time clock, then there's no problem. For machine language routines, a similar reduction in loops can be done.

One could also simply use faster 6502 routines and emulate them. The file __FASTCHP.FPX contains the code for the Newell Industries Fastchip floating-point routines, which triple the speed of most floating-point operations. Simply rename it to __FASTCHP.FP and delete __ATARI.FP.

The BASIC XL runtime package works fine with the emulator, so it could be used in place of Atari BASIC to run BASIC files.

The file __NEWELL.OSX can be renamed to __NEWELL.OSB, and __ATARI.OSB deleted. The Newell operating system offers some enhanced and faster functionality, such as access to graphics modes 12 through 15.

Emulating other computers.

Anyone interested in modifying the **ST Xformer** to emulate other machines is free to do so. If it's to be a 6502-based machine, the file __X6502.C can be left untouched. Then __XFORMER.C has to be changed to simulate the DOS of the new machine, and __XATARI.C should be renamed and totally rewritten for that particular hardware.

If it isn't for a 6502-based machine, the __X6502.C should be renamed and the opcode handlers rewritten.

A word of warning: Other manufacturers won't be too pleased about emulators on the ST running their computers' software. I got an unfriendly response from Apple regarding my Apple II emulator. The same can be expected from other companies, since they're interested in selling their machines, not STs. Of course, by emulating the entire operating system of each particular machine, you can get around that, but then you suffer from the problem of lower compatibility. Look at the case of PC clones which will not run some *real* PC software. Another way around the problem is to take the path of the Magic Sac, but then you no longer have a software-only emulator—and costs are much higher.

Most other computers should be much more easily emulated. For example, the code for the Apple emulator is about 50K shorter and runs at about 40 percent of the speed of an Apple. This is due to the Apple's slower clock speed, simpler graphics modes and total lack of any interrupts.

A summary.

I hope that this explanation, along with a long printout of the program, will give you an insight into the process of emulating one computer on another. Some of you may even be encouraged to write your own versions of the emulator for other microprocessors, like the 6809 and Z80. If enough people work on this program, most of the major 8-bit machines will be emulated.

With the forthcoming 68020-based Atari TT, the speed of the emulator may increase five-fold, to the point where the emulated software will run at the same speed as the real thing. Thus, the Atari TT may become a completely universal machine, capable of running most available software on all machines.

Anyone having any further questions, code improvements, or a list of programs that work with the emulator, can contact me on Delphi (username DAREKM), CompuServe at 73657,2714 and on GEnie (also DAREKM). I would like to maintain one master copy of all improved versions, which would then be released periodically with an updated working program list.

If enough interest is generated, perhaps Delphi, CompuServe or GEnie might even set up a separate download section of software known to work with the emulator, as is currently done with Mac software for the Magic Sac. //

Darek Mihocka is a second-year co-op Computer Engineering student at the University of Waterloo (near Toronto). Between school terms, he works at Microsoft Corporation in Seattle. He is a licensed pilot and also enjoys heavy metal music.

Step 1
TEST DRIVE

Taking your printer out for a spin!

by Maurice Molyneaux

If you use a word processor, at one time or another you've probably experienced the frustration of trying to print something and not having it come out right. Let's say you tried to print bold, italic type and, instead, got Near Letter Quality (NLQ) out of your dot-matrix printer. You can blame the printer, you can blame the word processor, you can blame both—*or* you can blame the most likely source of the problem: the printer driver.

A "driver" is a special file which tells your word processor how to communicate with a printer. If you don't have a driver—or have the wrong one—you can end up with odd results when printing time rolls around. In this month's **Step 1**, I'll attempt to show you the basics of creating or customizing a printer driver.

Table of Babel.

You might wonder why all this fuss is necessary. After all, your computer knows how to get your printer to spew out regular text with no problem. Why should there be difficulty with text effects, layout and special characters?

Your ST and printer both understand regular text; the letters of the English alphabet and the most common punctuation symbols are represented by the same values on both. This is because both use ASCII (American Symbolic Code for Information Interchange) to define characters. With ASCII, the letters of the alphabet and most punctuation symbols are always represented by the same values, in computers ranging from Apples to Ataris, and printers from Epsons to Brothers. By standardizing values for such characters, ASCII permits various devices to share information. Unfortunately, the symbols beyond the normal alphabet (including punctuation marks) are not as standardized, so two printers using ASCII might well have completely different values for foreign-language and other unusual characters, if they feature special characters at all. This can present problems. For example, the ST's ASCII character set includes a copyright character, an infinity symbol, and the Hebrew alphabet! Chances are, your printer *won't* normally interpret these to be the same things the ST does.

Due to these ASCII inconsistencies, it's often necessary to fiddle with a printer driver, in order to get special characters to print . . . if your printer is capable of producing them.

To make a bad situation worse, there's really no standard for enabling certain effects with printers. The code for enabling boldface is almost surely different in a Star printer than it would be in a Panasonic. And, again, since various printers have differing abilities, some functions of your word processor just may not work. For example, a Star SG-10 can be toggled to print zeros *slashed*, while a Star Gemini 10X doesn't have a mode to do this. . . and they're both from the same company!

The specific examples I'm going to give refer to configuring drivers for 1st Word and ST Writer only, but most of what I cover here can be applied to almost any word processor's printer configuration. Before we begin, you'll need a few items: your printer's manual, your ST's *ST BASIC Sourcebook* (trust me), some blank paper, and the word processor of your choice (if you're using 1st Word and have the manual for it, dig that out and flip to Chapter 5).

HEXing decimals.

To configure both the 1st Word and ST Writer drivers, you'll have to understand a bit about the difference between decimal and hexadecimal math. Decimal math is the old base-10 math we've been using since we began counting our fingers and toes. Hexadecimal is a form of counting which you might assume evolved on another planet, because it's *base-16* math! In base 10, we count from 0 to 9 and, on passing nine, return to 0, adding a 1 to the left of the 0, steadily incrementing into tens, hundreds, thousands, etc. Believe

it or not, hexadecimal ("hex," from here on) works the same way. . . but with more numbers. To count from a one-digit number to a two-digit number in base 10 is as simple as: *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.* To do the same in hex, we would count: *0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10!* Not really any different from counting in base 10, except that we have sixteen numbers to deal with, instead of ten, before advancing to the tens, hundreds, etc. And, since hex is base 16, converting numbers can be confusing. For example, the decimal number *253* is represented in hex by *FD.* Decimal *12* is hex *C,* and decimal *20* is hex *14.*

Fortunately, you probably won't need to do any converting. If, by chance, you do (say your printer manual lists hex only, and you need decimal), there are several calculator-type accessories for the ST which feature decimal to hex conversion. Also, most printer manuals feature a table for such conversions. But, in the event you don't have either, that's where your *ST BASIC Sourcebook* comes in. In its Appendix E are two tables showing the *complete* ST character set. One table shows the usual 8x8-pixel characters used in low and medium resolution, and the second table shows the 8x16-pixel characters usually used in high resolution. The characters' values are the same on both tables, so it doesn't matter which you use. You can find the character corresponding to the hex value on the table, then merely check the decimal value (or vice versa). Even if you have an accessory or reference to convert values, keep the *Sourcebook* on hand, because we *will* be using it.

Now let's get to work. Most ST word processors have special text files with them, containing data for the printer configuration. The user loads this file into the word processor, edits the data (changing it to the codes for his/her printer), saves it, then runs a special program which reads the data from that file and writes the appropriate printer driver.

With 1st Word, you load one of the .HEX files (as in FX80.HEX) from the PRINTER folder (you can load ASCII.HEX if you want to start from scratch). Make sure WP (Word Processor) mode stays OFF when working with this file. For ST Writer, you'll load the file CONFIG.TXT and edit it. Other word processors may have a special program which prompts you for data. Refer to the manual for your program if you're unsure about this.

So, what are you waiting for? Boot up your word processor, get that configuration data on-screen, and let's get cracking!

Special effects.

The first thing we want to do is check all the major text and character effects, and make sure they're installed properly. With both the ST Writer and 1st Word configuration files, the codes for each function are separate and documented. What you need to do is open up your printer manual and find the table which lists *all* of the printer commands. Now, look at the file on your screen. Move down past all the header information until you see the following: (Note, the boldface copy is used in the following examples only to differentiate between 1st Word and ST Writer files, and will not actually appear in the files.)

```
1st Word:
* 0          *Character width
```

```
ST Writer:
*Turn underline mode on: <Esc>"-" 1
27
45
1
255
255
255
255
```

These represent the beginning of the main configuration data. In the case of 1st Word, you probably can ignore fields like "Character Width," "Linefeed WITH return" and "Horizontal," and "Vertical Initialisation" [*sic*], as they're probably correct. Let's try setting up "Draft bold" as a start. You should find a line like the following in your file (please note that these are generic examples, and the exact wording and codes in your file might be different—just read the comments to make sure you have the correct function):

```
1st Word:
* 6          *Draft bold on
ST Writer:
* bold on:
```

In your printer manual, find the command sequence for turning boldface type on. If you can't find a boldface command, try looking under "double-strike," etc. If your printer doesn't feature *anything* like this function, choose another (say, underlining). I'll quote from my Star SG-10 manual; using IBM mode, the control code I want is:

```
<ESC> "G" Double-strike print
```

Now, just *what* does <ESC> "G" stand for? Simple. To start double-strike/bold printing, my SG-10 must receive the ASCII values for Escape and uppercase G (in that order). What we need to do now is find out what these values are. Look at the character set table in your *ST BASIC Sourcebook* and find the character for Escape on the chart (represented by the symbol ←). Now, you have to calculate the value for the character.

If you're using ST Writer, you need the number in decimal, so go straight up from the Escape character to the top row of the chart ("decimal value"). The value should be *16.* Now, go to the *left* from Escape and find the number in the "decimal value" column along the side. It's *11,* right? Now, *16+11=27,* so the value for Escape is *27.* Do the same for G: *64+7=71.* So, for an SG-10 printer in IBM mode, the command sequence for double-strike/bold type would be decimal *27, 71.*

Now, enter these values into the data fields below the description, replacing the top number with *27* and the second number with *71* (and so on, if you have more than two numbers). There are eight values in the ST Writer configuration fields. If your command sequence is shorter than this, *don't* delete the extra numbers, just make sure they are *255,* the "null" character (for most printers). The "comment" line is begun with an asterisk (*), and you can type anything you wish on it, as it's ignored by the program which writes the printer driver (note that I've added the control codes here for future reference). The edited command sequence should look like this:

```
* bold on: <Esc> "G"
27
71
```

```
255
255
255
255
255
255
```

If you're using 1st Word, do the same as above, finding the Escape character on the chart. However, instead of reading the decimal values, we need the hexadecimal values. We find *1* in the hex column above Escape, and *B* in the column to the left. Now, we add these two up. If unfamiliar with hex, you're probably groaning at the idea of adding base-16 numbers. But in this case, it's *easy*.

Since the character table in the *ST BASIC Sourcebook* is a 16x16 grid, it is laid out perfectly for hex. The values across the tops are hex "tens," so *1+B=1B*. For G, the values are *4+7=47*. Easy, huh? Just as *D+9=D9* or *F+C=FC*. Simply put the value from the horizontal column in front of the one from the vertical column, and that's it. Now you have to enter the codes into your file. The data is entered sequentially, on a single line, each value followed by a comma and a blank space before the next value. To put the codes *1B, 47* into the file, we have to edit the line like so:

before: * 6 * Draft bold on

after: 6, 1B, 47 * Draft bold on

The asterisk at the beginning of a data line indicates that it is not in use. The number which follows the asterisk is the function number. You must delete the asterisk to make a given field functional, and put the function number (*6* in this example) in the first column, followed by a comma, a space, and then your printer codes. The above *6, 1B, 47* is read by 1st Word as "Function 6 - codes: Escape G." If you delete the function number, the first value in your codes will be seen as the function number, which can lead to real problems! The second asterisk in the first example, and the *only* asterisk in the second, tells the configuration program that what follows is a comment it should ignore. Don't delete the asterisk before the comment unless you enjoy problems.

The example we've just gone over is used throughout the configuration file. Do the same for functions like italic type, underlined text, etc. Look up the codes in your manual, use the chart in the *Sourcebook* to find the appropriate values, write them down (commented!) on a piece of paper, then punch them into the configuration file.

Be forewarned, however, that not every function your word processor has may be supported by your printer—and vice versa. Make *sure* you find any codes needed to turn functions *off* as well as *on* (if the configuration file requests such data). If you don't, your printer may not stop using a given mode until you turn the power off! Some printers do not have specific commands for shutting off things like italic or bold type. If yours is one of these, try using the command for normal print (or try "Pica," etc. if you don't have a "normal/draft print" command) in place of any missing "off" codes.

Parlez-vous ASCII?

Yes, but with an *accent*. As stated earlier, ASCII is used by both our computers and printers, but not all ASCII is interpreted the same way. You could say it's like a language. The language is common, but there are regional *dialects* which are somewhat different. In the case of our systems, the main alphabet is the common part of the "ASCII language," but the rest of it is usually a local dialect.

It's simple to see why printers don't have the same codes for various print functions, as those codes usually involve multiple values. However, you'd think that there would be *some* standardization with regard to things like carriage returns and accented (international) characters. Nope. Why, even Atari's 8-bit computers use a peculiar dialect of ASCII called ATASCII (ATari-ASCII), which doesn't even use the same character for a carriage return that the ST does. It gets no better with printers. Both 1st Word and ST Writer can print special characters not normally used on the (American) keyboard. You can put these on-screen, but getting them on paper is somewhat trickier. What you need in such trying situations is an ST-Printer ASCII Dictionary, better known as a "Character Translation Table."

Both 1st Word and ST Writer have facilities for character translation, but putting them to use isn't easy. If you feel you have your printer driver set up as you need, and don't have a use for accented and other special characters, you may wish to skip to "Test drive." If you're a daring soul, stay right here.

As always, some characters on your ST may *not* appear in your printer's character sets, and vice versa. Those which *are* common between the two machines are likely to have different ASCII values. What you need to do to use these is tell your word processor what value to send the printer to make it output a given special character. Keep the character set table of your *ST BASIC Sourcebook* on hand, and find the ASCII code conversion chart in your printer's manual.

This chart should list the decimal, hex (perhaps binary, too) and control character values for each and every character the printer can type. If your printer supports multiple character sets, make sure you keep your eye on the column for the characters of the mode you use. Scan down the list of printed characters until you find something unusual, say something like the character © (copyright). Now, the next question is: does your ST's character set feature this particular letter? Look at the table in your *Sourcebook*. Ah, there it is! Decimal *181*, a.k.a. hex *BD*. What is your printer's value for it? My SG-10, in Star mode, uses decimal *203*, or hex *CB*. Hmmm. Not the same.

1st Word users must search the Translation Table section of their configuration file for the description which matches that character (it's the same as the hex value in the ST character set, function number BD). Now, all we have to do is add the value for the character, just as we entered printer control codes earlier. Before edit, the line might look like this:

```
BD      * No copyright symbol
```

After edit, we might have. . .

```
BD, CB      * Copyright symbol
```

This tells 1st Word what value to send the printer in place of its own value for the character.

ST Writer users must take a slightly different approach. Go to the translation table of your CONFIG.TXT file (if you're using an ST Writer older than 1.50—1.05 or 1.07,

etc.—you will *not* be able to do this). This table consists of a series of hex numbers. The *0x* at the beginning of all the numbers tells CONFIG.TOS the value is in hex. The numbers proper are the second part. So *0x0C* is hex *C*.

These numbers might seem senseless, but they're not. To put the printer's value for the copyright symbol in place of the ST's, we simply have to find the hex number for the character in the translation table. The ST uses hex *BD*, so we find *0xbd* in the table. The value my printer uses is hex *CB*, so I replace the *0xbd* with *0xcb*. Now, ST Writer knows to send my printer *that* code instead of its usual one.

Do this as much as you like. Scan your printer's character set, find a character you'd like to print, then check your *Sourcebook* to see if it's available. If so, compare the hex values. If they're the same, go on to the next character. If not, make appropriate changes (as above) until you get all that you want.

Finally, your printer may have characters your ST doesn't. If this is the case, find a character in the ST character set that you don't use and replace *its* value with the one for the character you want to use. What you see on the screen won't be the character you want, but it should print out a-okay. Further, if your printer doesn't have certain characters available, like è, you may be able to get the equivalent of this character by giving *multiple* codes for the character. To print the è, you would enter the value for the letter *e*, the value for your printer's backspace command, and then the code for the accent mark. You can do this in 1st Word, but I'm not sure exactly *how* it's done with ST Writer (if it's even possible using its translation table).

Test drive.

Almost time to get this baby out of the hangar, but a few notes before you save your file. If you're using 1st Word, near the beginning of the configuration file is a parameter which is used to tell the program if your printer is a dot matrix or a daisy wheel.

Look under "Miscellaneous configurable variables." There are six parameters; of them, the first is the printer type. Below the descriptions are six zeros (*0, 0, 0, 0, 0, 0*), which hold the values specified by the six descriptions. You enter a *1* in place of the first *0* if your printer is a daisy wheel, otherwise you leave it *0*. Of the remaining five values, the second through fifth are used *only* if you set the first number to *1*. The sixth number is set to *1* or *0*, respectively, depending on whether or not you wish the program to pause between pages. Also, enter the name of your printer at the top of the file where asked, so when the 1st Print program is run by 1st Word, it will display the name of the printer, and you'll know it's using the correct configuration data.

If using ST Writer, check to make sure all the main print codes contain *eight* numbers (as described earlier). If any are short, add *255*s to fill the field to eight numbers.

Save your file when you're done, then exit your program. Now you have to create the actual printer driver. If using 1st Word, go to the PRINTER folder and run INSTALL.PRG. Select your configuration data file from the item selector and let the program do its work, creating a file called 1ST.DOT or 1ST.DSY (depending on your printer type). If using ST Writer, make *sure* your configuration file is named CONFIG.TXT, and make sure it's in the same directory (main or folder) as the CONFIG.TOS program. Run CONFIG.TOS, and it will create a file called XYZZX.DAT. When the configuration program finishes, copy the resulting driver to the directory containing your word processor.

The moment of truth. Run your word processor. Create a document and start typing, liberally turning on and off various effects and inserting special characters you set up in the translation table. When you've finished, save the file and turn loose your printer. If the gods have smiled on you, all will go well—but don't count on it. Most likely, one or more effects you tried simply will *not* work correctly yet. Carefully note which functions or characters didn't print correctly, reload your configuration data file, recheck the figures, then resave the file and try again. Sometimes it takes three or four passes to get everything kosher, so remember, he who perseveres shall reap greater rewards.

A few final notes. If you have something that consistently fails to work, *carefully* check the data and text corresponding to it. Is the asterisk which tells the configuration program to ignore a comment missing? Are necessary commas present between numbers? Are all fields properly filled? Watch for things like a capital *O* in place of a *0*.

Also, one common boo-boo concerns activating expanded print. Many people accidentally put in the codes for *one line* of expanded type, which is automatically shut off when the printer starts the next line. If you have this problem, make sure you have the code for expanded, not one-line expanded type.

Well, that wraps up **Step 1** for this time. I pray this hasn't been too confusing, and I certainly hope I've helped many of you with this small but important stumbling block in setting up a computer system. As always, if you have any questions or comments on this, or other **Step 1**s, please drop a line to **ST-Log**. Until next time, *au revoir!*

Addenda.

— In "Customizing the GEM Desktop" (ST-Log issue 14), when discussing how to install applications so that they'll run from a specific drive, I didn't make it absolutely clear that you *must* type a backslash (\) after the drive identifier, even if you're *not* using a folder name. If you don't, GEM gets confused and looks at drive A. Therefore, a DESKTOP.INF file would contain the following line, if you wanted to install Neo-Chrome to run from drive E:

```
#G 03 04    E:\NEO.PRG@ *.NEO@
```

Refer to **Step 1** in the May 1987 issue for more on this.

— I've previously stated that the ST cannot boot from any drive but drive A. Not true. I have a Supra 20-megabyte hard disk drive, and it has software with which the user can set the hard disk to auto-boot. To test the drive A boot claim I've heard so often, I set the hard disk to auto-boot, then turned off my two floppy drives (I have a 1-meg 520ST) and booted my system. After a few seconds, my system booted from the hard disk with no trouble. Everything went fine *without* the floppies. So scratch the contention that the ST can't boot from a hard disk. Clearly, it can. //

Building blocks.

by Ian Chadwick

The essential element of BASIC programming is the conditional statement—the building block on which most programs depend. GFA provides four conditionals: IF, EXIT IF, REPEAT/UNTIL and WHILE/WEND. Each of these works in a similar fashion, but there are subtle and important differences.

Most programmers are accustomed to IF commands. In GFA, the IF command begins a program block that must end with ENDIF. The simplest syntax is:

```
IF (condition is true) THEN
    (do something)
ENDIF
```

For example:

```
PRINT "Enter a number between 1 and 10 (99 to quit)."
Z = RANDOM(9)+1
REPEAT
INPUT A
    IF A = Z THEN
        PRINT "You guessed the correct number!"
        END
    ENDIF
UNTIL A = 99
```

THEN is optional, but worth using for the sake of clarity. If the condition in the IF line isn't true, then the program drops down to the line immediately following the next ENDIF, and everything in between is ignored.

Alternatively, you can add additional conditions using the ELSE statement:

```
IF (condition is true) THEN
    (do something)
ELSE
    (do something else)
ENDIF
```

For example:

```
PRINT "Enter a number between 1 and 10 (99 to quit)."
Z = RANDOM(9)+1
REPEAT
INPUT A
    IF A = Z THEN
        PRINT "Correct! You win!"
        END
    ELSE
        PRINT "You guessed wrong... try again."
    ENDIF
UNTIL A = 99
```

or, you can nest IF statements using ELSE:

```
IF (condition is true) THEN
    (do something)
ELSE
    IF (another condition is true) THEN
        (do something else)
    ELSE
        IF (another condition) THEN
            (do something)
        ENDIF
    ENDIF
ENDIF
```

For example:

```
PRINT "Enter a number between 1 and 10 (99 to quit)."
Z = RANDOM(9)+1
REPEAT
INPUT A
    IF A = Z THEN
        PRINT "You guessed the correct number!"
        END
    ELSE
        IF A = Z-1
            PRINT "You are one too low."
        ELSE
            IF A = Z+1
                PRINT "You are one too high."
            ELSE
                PRINT "Wrong... try again."
            ENDIF
        ENDIF
    ENDIF
UNTIL A = 99
```

Note that each IF condition must end with its own ENDIF. Any number of ELSE conditions can be nested within the original IF condition.

Logical operators and other functions can be used in the conditions. Conditions can be simple expressions or complex formulae. For example:

```
IF A >3
IF B=3 And C=4
IF A>1 Or B=6
IF ODD(A)
IF A=FALSE
IF PEEK(N)=22
IF OUT(0)=0
IF ((A/B)/C)>((D/A)*N)
```

EXIT IF is a single line conditional, requiring no ENDIF and accepting no ELSE conditions. Other than that, all the conditions that apply to IF apply to EXIT IF. EXIT IF is used to get out of loops such as FOR/NEXT and DO/LOOP:

```
FOR N = 1 to 1000
     PRINT N,
     EXIT IF N =234
NEXT N
```

or:

```
DO
     INPUT A
     EXIT IF A = 99
     B = B+A
LOOP
```

Dropping out of an unfinished FOR/NEXT loop in this fashion seems to have no adverse effect on the program. However, using an IF/ENDIF in a FOR/NEXT loop generates an error message, as in this example:

```
FOR N = 1 to 1000
     PRINT N,
     IF N = 234
          GOTO CODE2
     ENDIF
NEXT N
```

Using EXIT IF is the correct way to quit before completion—avoid IF alone to escape loops.

REPEAT/UNTIL and WHILE/WEND both allow a single conditional to determine iterations of a program block:

```
REPEAT
     (program code)
UNTIL (condition)
```

For example:

```
PRINT "Enter a number between 1 and 10 (inclusive)."
Z = RANDOM(9)+1
B = 0
REPEAT
     PRINT "Try again!"
     INPUT A
     ADD B,1
     PRINT "Guesses =";B
UNTIL A = Z
PRINT "Correct in ;"B;" guesses!"
```

and:

```
WHILE (condition)
     (program code)
WEND
```

For example:

```
PRINT "Enter a number between 1 and 10 (inclusive)."
Z = RANDOM (9)+1
B = 0
WHILE A <> Z
     PRINT "Try again!"
     INPUT A
     ADD B,1
     PRINT "Guesses =";B
WEND
PRINT "Correct in ;"B;" guesses!"
```

The difference between the two is where the condition is checked. In REPEAT/UNTIL blocks, it is checked at the end of the iteration, allowing other functions and commands to take place until the UNTIL line is reached (in the last example, B is incremented). The program block is therefore always executed at least once before the condition is checked. In WHILE/WEND blocks, since it is checked before the code is executed, the code will not be executed if the condition is true (B is not incremented in the last example).

Again, conditions that apply to the IF function apply to the conditions stated in UNTIL and WHILE.

Both REPEAT/UNTIL and WHILE/WEND loops can be nested within themselves and each other, not to mention within IF/ENDIF, DO/LOOP and FOR/NEXT loops. The danger, of course, is in the ability to create endless loops without exits—so use the EXIT IF command, at least, to provide a proper escape.

The other caution is to be careful when mixing variables within loops. Unlike procedures, there are no local variables for loops, only global variables, so it's easy to get confused.

Small plug.

MichTron runs a conference section on GEnie, the online database (this is separate from their own in-house BBS). Although not strictly limited to GFA BASIC, their database of download-able files includes many utilities, programming examples, games and demos. They also have a weekly online conference every Sunday about GFA, in which the MichTron technical staff participate along with any users—you get news, gossip, comments, questions, and so on. It's pretty free-form.

One of the more difficult (in terms of complexity) tasks in GFA is creating dialog boxes—there is no equivalent command as there is for alerts. However, there are dialog box creation programs for medium/high and low resolutions in their GEnie database (the archived files also include examples). I recommend you get them if you want to make the process a lot easier.

A lot of good information is to be had in both the conference and the database. The conferences are archived into files in the GEnie database after a week or so, if you missed one. I haven't seen these programs on any other service, although I have seen other programs on CompuServe and Delphi. ///

Ian Chadwick is a Toronto-based free-lance writer and editor who is currently working on a spy novel and a wargame, among other things.

The CinemaWare Story

Master Designer Software charts new directions in computer entertainment.

by Arnie Katz

"We had a unique opportunity to rethink what a computer game could be on the 16-bit computer," says Bob Jacob. And when the president of Master Designer Software (MDS) speaks these days, he commands the attention of the entire entertainment software community.

Success does that. In less than two years, MDS has become the hottest design house in home computing with its Mindscape distributed line of CinemaWare arcade adventures.

The softspoken Jacob is no newcomer to the field. Prior to starting Master Designer Software, he headed the Robert Jacob Agency, which represented many software authors. "I thought that I was a pretty knowledgeable guy about software," he recalls, "and I was growing disappointed with most of the games I was playing." Characteristically, he decided to do something besides complain. Aided by wife Phyllis, Jacob forsook agenting to launch Master Designer Software in January 1986.

Mindscape introduced the first three CinemaWare titles: SDI, Defender of the Crown and King of Chicago. Each game made its debut on a single system, but the company supports five computers: Atari ST, Amiga, Commodore 64, Apple II GS, and Macintosh. The process of extending all titles across the full range of machines has taken a year, because Jacob approaches conversions much differently than most other producers.

"We are unusual in that we've never done a straight port of any of our games," Bob boasts. When MDS tackles a new edition of an existing game, they approach the project as a fresh opportunity—not just the expressway to a quick profit. "Too many publishers turn out assembly line conversions," notes Jacob. "We completely reprogram for each machine." This greatly benefits gamers, because CinemaWare products almost invariably improve with each version.

Defender of the Crown offers a perfect illustration. "We learn something from each version we do," asserts John Cutter, vice president of product development. A successful author and producer, Cutter joined MDS in time to help finish the first wave of games, and has guided the development of all subsequent projects.

Although players generally liked Defender of the Crown on the Amiga, there were a few criticisms of the solitaire adventure. The main one was that the strategic part of the game seemed a little thin. Accordingly, lead designer Kellyn Beeck beefed up this aspect of the program when Defender moved to the Commodore 64, and more recently, to the ST. For instance, Beeck underscored the difference between the Saxon and Norman lords by allowing the player to get "safe conduct" through the lands of fellow Saxons.

The ST action sequences, which are the heart of any CinemaWare epic, are also much better than all previous versions. If the player decides to lay siege to a rival castle, the catapult now throws a choice of boulders, Greek fire or disease bombs. It is now possible to parry as well as strike in the swordfighting sequence, and strength bars now monitor how fast attackers and defenders expend energy. Even jousting got a facelift. Now, the player moves the mouse to align the mounted knight's lance and pushes the button at the precise instant of contact to thrust at the target.

According to Jacob, MDS's desire to make maximum use of the ST's potential dictated other changes. "The size of the game was one thing we addressed," he explains. "Data compression technology puts everything on two, one-sided disks for the ST. The Amiga original was two, double-sided disks.

"We wanted the best audio possible," says Jacob. The company turned to Musicon Design Limited of England to do all sound and music for Defender of the Crown.

The CinemaWare method.

It all starts with a concept, and most of the concepts start with Bob Jacob. "It took a while for the rest of us to get to the right mindset," says John Cutter, "but Bob had it right from the start.

"We believe that a concept is best if it

comes from one person," explains Cutter. "Bob has done a lot of this. Then we bring in many people to develop it."

What is the right mindset? "We look to the movies, not old software for our inspiration," Bob Jacob says. The establishing shots, jump cuts, long non-interactive animated sequences and pervasive theme music testify to the influence of film on CinemaWare games.

Sinbad, for instance is *Thief of Bagdad*, and a dozen other Middle Eastern romances rolled into one.

Stalking the mass market.

Some design houses cultivate an intensely loyal coterie of customers, but MDS unabashedly courts the broadest possible audience. Like a thrilling adventure movie, CinemaWare titles are readily accessible to virtually everyone. "We want to appeal to the casual gamer. That's why you can get right into one of our games without even reading the manual," says Jacob. That's why CinemaWare games avoid keyboard input. "We have tried to simplify the adventure game user interface. We want our games to make an emotional breakthrough with the player," continues Jacob.

"People can relate to our games," adds Cutter.

Like all entertainment moguls, Jacob avidly studies the demographics. "We instinctively felt that our customers would be older, and we designed accordingly. We weren't interested in doing games for kids," Jacob says. In fact, the typical buyer of CinemaWare's ST products is 32 years old and male.

The structure of their action adventures reflects the needs of this group. CinemaWare titles present the most dramatic moments as arcade contests, linked together by the overall plot and bridging animated sequences.

Jacob and crew don't look at arcade games like any other design group in the entertainment software industry. They don't try to make them outrageously challenging. That would only frustrate players who no longer have the lightning reflexes of their youth. Instead, their games require some timing and quick thinking. The goal is to let the player enjoy the experience, not set scoring records.

Romance is integral to CinemaWare games, as befits adventures aimed at adults. The sensitive love scene in Defender of the Crown is as appropriate for CinemaWare as it is unusual in a computer adventure.

Lights. . .camera. . .adventure!

Like a movie, a CinemaWare game blends the talents of many specialists. They pool their talents and abilities to create the finished work of art, each doing what he or she does best.

The higher memory computer systems like the Atari ST virtually demand such specialization. In earlier days, a competent programmer could whip up the few necessary beeps and boops which served as computer game audio. Only a skilled musician could write CinemaWare's stirring scores. Another example of the complexity of these programs: the graphics for the ST edition of Defender of the Crown took more than six months.

Once Bob Jacob approves a preliminary design proposal, he and John Cutter assemble the implementation team. While the main designer works with the project's producer to firm up the multi-pathed plotline, other members of the team draw sample illustrations and work on the non-interactive portions of the game.

Storyboarding keeps the many elements of each game in proper sync. "We probably do more storyboarding than any other software publisher," claims Cutter. Artists sketch each screen planned for the game, and the designer clearly establishes its relationship to every other screen and bridging segment.

Linear plotting is a major no-no at Master Designer Software, so the storyboarding is particularly useful to insure that no anomalies or paradoxes creep into the overall structure as fine-tuning proceeds. Many times, studying the storyboards prompts the team to change the plot, include extra animated interludes, or add more sound.

Once a project gathers a little momentum, the producer coordinates the activities of all the people working on the game. The producer, most often John Cutter, orders adjustments and revisions as he fits the pieces together. Other members of MDS's resident staff—like Kellyn Beeck, and, of course, Bob Jacob—review work in progress.

Throw a hot game concept into the MDS hopper, and a beautifully polished piece of entertainment software rolls out the other end in approximately one year.

CinemaWare 1990.

Some software publishers, after setting new standards, have lapsed into a static period. They ride the success as far as it can go and leave innovation to newer, leaner competitors. MDS doesn't plan to fall into that trap. The company expects to improve upon its first batch of games and explore fresh territory.

"It has taken us a while to get into the groove," says John Cutter. "We started with four different design teams. Our next group of games will have the best features from each team.

"Most of our games feature characters who are larger than life," Cutter states. "We need to develop characters with more human dimension." In that way, the player can identify more intimately with the protagonist of the game and enjoy a more intense vicarious experience.

Bob Jacob sums up the Master Designer Software philosophy in a few incisive words: "We want to make emotional breakthroughs." That's quite an ambitious goal, but don't bet that Master Designer Software won't make its president a true prophet. //

Ninja Mission

MASTERTRONIC
7311B Grove Road
Frederick, MD 21701
All resolutions $19.95

by Matthew J.W. Ratcliff

Ninja Mission places you in Akuma's fortress of death, the Shijo. You must battle your way through fifteen rooms of the Shijo, against "Ninjas, Thugs and Karatekas," in search of the seven jade idols of the Tambo Machi tribe.

This game is designed to be fast action, with a quick joystick response. The Thugs are judo experts with strong punches, best killed from kicking distance. The Karatekas are experts at hand-to-hand combat, with deadly kicks. They are best killed with the sword and rapid kicks.

Finally, the evil Ninjas have skills that equal yours. They too are capable of sword fighting, and can throw stars and knives, as you can. If a knife or star hits the floor, you may retrieve it and throw again—as can the evil Ninjas.

They're not easily defeated, and are best combatted with throwing weapons. You can duck and jump over the blades hurled at you by the Ninjas. Avoid close fighting with them, and use the sword when all other blades have been used. If a knife or star is on the ground, pick it up quickly—before the opposing Ninja can retrieve and use it on you.

In between the battles, you can also retrieve the jade idols, whose mystical powers can restore your fading energy (represented as a bar graph at the bottom left of the display). I found it odd that the idols are bright red, since all the jade I've seen has been a dark green. But this is the least of the "quirks" I found in **Ninja Mission**.

There are sixteen different moves you can make with the joystick, eight for each direction with the button up or down. The movement controls are much like those of World Karate Championship from Epyx. One frustrating factor I found in **Ninja Mission** is that, while you can jump safely over an opponent, you cannot move too far "behind" him to pick up a weapon or idol. Once you get behind an opponent, the software completely ignores your "forward" commands and forces you to back up to the opponent you've engaged. If you aren't fully prepared to do battle at this point, your only alternative is to back off the opposite edge of the screen to the previous room.

When you throw a star or knife, you can do a little or a lot of damage to your opponent. You can time the knife throws so that, as it spins through the air, the point hits your enemy. Sometimes you can make a kill with a single throw. It is always important to pick up your weapons as soon as possible after throwing them, because you'll have to accumulate a lot of them near the final levels of this game—where you may face five guards at once.

You should never leave a room where knives and stars are lying about, because they have a nasty habit of disappearing for no apparent reason. Upon reentering a room, you will find your *dead* opponents move. There's no apparent reason for this; maybe they're the ones stealing your knives!

The greatest drawback of **Ninja Mission** stems from two factors. First, whenever you get close to either edge of the screen, you're automatically sent to the adjacent one. This is denoted by a separate title screen and a *long* delay (over 6 seconds). It's terribly frustrating, and interrupts the rhythm of your battles. When you're engaging an opponent, it's easy to accidentally bump the edge of the display and go to the next screen. The rule here is: never back up when fighting. It costs time and weapons (since they can disappear anytime you leave a screen).

Second, I've put the game into pause mode, come back to disengage it, and discovered the program locked up (even though the music continued to play). At one time, I left a dead man on the screen, came back later and found his arms separated from his body.

The music and graphics are well executed in this program. The play is fast, overall. The manual, five printed pages, is better than any documentation I've seen from Mastertronic for their 8-bit games. It is concise and seems to be complete.

Ninja Mission is a good game, but needs some minor refinements. Once you engage an enemy, you should be "locked" onto that screen until the outcome is decided. The "edges" should be nearer the edge of the monitor, since it's easy to bump onto the next screen when simply attempting to pick up an item near the limits of the current display. The jade idols should be green. Dead people should not move. Dropped knives and stars should never move, unless touched by yourself or an opposing Ninja. The Ninja warrior you control should be able to turn around and attack an opponent from behind, and should be able to jump over an opponent to pick up something behind him. All these updates are minor, but games are not generally supported after release, the way applications are.

I like **Ninja Mission**, and plan to stick with it until I can win it at least once. Its game play is much faster than World Karate Championship, in exchange for less detailed graphics in the player movements. The joystick response is acceptably fast, also. For the low price, I think **Ninja Mission** is a good value. I just hope that Mastertronic will support the product by addressing at least some of the complaints I had above. It's a good game, in need of a little spit and polish. //

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.

Hardball!

ST translation by Distinctive Software
(Original design by Bob Whitehead)
ACCOLADE
20833 Stevens Creek Blvd.
Cupertino, CA 95014
Low resolution $39.95

by Bill Kunkel

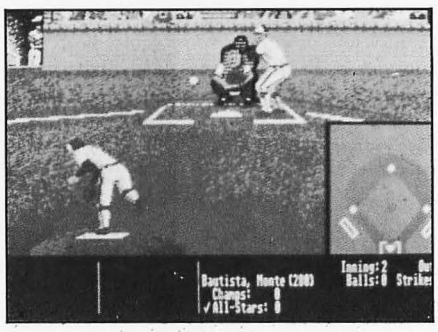
The ST version of **Hardball!** is a game with strong virtues *and* weaknesses. The graphics are mostly superb, the game interface is user-friendly and the basic pitcher vs. batter *mano a mano* is the best ever computer recreation of that elemental sports confrontation.

Visuals are displayed via three graphic screens, augmented by a small, stationary overhead view in the lower right or lefthand corner (depending on which side of the plate the batter swings from). There is also a stat/scoreboard screen that appears periodically. The pitcher/batter screen represents the now familiar TV camera view from left centerfield (i.e., the perspective of a runner taking a short lead off second). The fielding screens present the standard behind-the-plate view, but with the stadium split up the middle into two screens. Depending on where the ball is hit, either the left or right views are displayed.

These fielding screens are a problem. It was clearly not possible to squeeze in the entire field depths, yet here they are anyway. The centerfield seats appear to begin about fifteen feet behind second base, while left, right *and* center fields share the same dimensions. This inaccuracy creates massive distortions in the play. Once a ball is hit, the program "assigns" the human manager control (via joystick or mouse) over the player best able to field the ball. But as the horsehide comes rocketing off an opponent's bat to, say, the left side of the field, the user has no way of knowing whether he will be controlling the shortstop, third baseman or left fielder!

Line drives travel like wire-guided missiles about 16 inches off the ground, while the occasional high fly can only be tracked by its shadow. The short outfield causes quite a few of those ridiculous centerfield to first base putouts on ground balls (there are even occasional 8-4-3 relay putouts at first base). But, for some reason, throws from right field to third base, or left field to first, seem to take hours.

On the offense, players select strategic



Hardball!

moves (steal a base, bunt, swing away) and bat position (swing high, low, inside, outside, etc.) from a simple menu. When on the field, they select the type of pitch (fastball, curve, slider, etc.) and location. Choosing the center location always results in a strike, but more marginally positioned pitches can also cross the strike zone.

Unfortunately, pitch selection is seemingly assigned at random, rather than on the basis of what a real pitcher might actually throw. Available pitches include: fastball, curve, screwball, change-up, sinker, off-speed and slider. Each pitcher has a menu of four pitches, but rather than the normal configurations—fastball/slider, curve/change-up, etc.—players sometimes wind up throwing a change-up *and* an off-speed pitch, but no fastball! What's the point of *two* off-speed pitches and no fastball off which to vary them?

In an attempt to inject some strategic viability into the program, each nonexistent player is assigned a batting average. Batting averages are meaningless enough in a stat-oriented milieu (slugging percentages are superior power indicators, and on-base averages better gauge one's ability to reach base), but this is an action game where the player is not only swinging the bat, but positioning it as well. It would have made far more sense to indicate hitting *tendencies*, (i.e., line-drive hitter, pull hitter, power hitter). As it is, the numbers do add some interest, but only from a defensive perspective in one-player contests (when pitching to computer-controlled batters).

The program allows for substitutions, but provides no statistical or ratings basis upon which to make decisions. Players are not rated for speed or fielding, which makes stealing bases a coin toss. (How does one deduce a player's base-stealing ability from a batting average?)

The computer manager is pretty awful. With runners on first and third, and two outs, a ground ball was hit to the computer team's second baseman and *he threw home*, allowing a run to score. This sort of tactical malfeasance is commonplace.

Another sore point is the lack of a box score. No tally is presented to list hits, walks or strikeouts, other than the basic line score (runs, hits, errors). Moreover, there's no allowance for saving games, and, considering how long a 9-inning game takes, this feature would be exceedingly welcome.

What *does* work on **Hardball!** is pitching and hitting. In fact, what this program *really* reminds me of is stickball, the two-player game where a target is chalked on a wall, with one player pitching and the other hitting (using a broomstick handle for a bat), with the strike zone painted on the backstop. Pitching is especially enjoyable, providing the user with a fascinating insight into the pitcher's art. The skills involved in pitch selection and the importance of location have never been better simulated.

In any case, **Hardball!**'s virtues—excellent visuals and a brilliant simulation of the pitcher-batter showdown—are more considerable than its inadequacies, and it *is* the only action baseball game on the ST. So, until something better comes along, this one is recommended. //

Touching the databases

A tutorial on relational databases and the ST.

by Frank Cohen

The relational database was created in the early 1970s as an answer to the old hierarchical solution to database design. Since then, the relational model has become the recognized standard database system in the mini- and mainframe computer world. With the power of the Atari ST, the relational database has been brought into the microcomputer industry. This article describes the fundamental properties of the relational database model and the way it works on the ST.

Introduction.

When I began writing Regent Base more than a year ago, I was confronted with a challenge. Regent Software's management team—myself included—needed a lot of information about the company's productivity, sales figures and general profit and loss. At first, I thought I would write several little programs to individually manipulate and report on the information needed. Then the idea of writing a general, all-purpose program to handle all of Regent's information processing needs rolled in—like a locomotive. The result was Regent Base, a program capable of balancing a checkbook just as quickly as it can print a list of receivables.

Relational databases.

If you were to browse through the pages of any computer magazine, you'd probably run across a bunch of advertisements for database products calling themselves relational. Most people think that a relational database is any software product that has the capability to compare information stored in two separate files and report on the similarities and differences. This is true, but certainly not a good description of the real meaning behind the relational database standard.

The word *database* means an information storage and retrieval system. A database is software for organizing the computer equivalent of 3x5 cards on a disk, where each card has several pieces of information—such as numbers, sentences, times or dates—stored on it.

The word *relational* means, in part, that a database can do more than simply store and retrieve information. The information stored in one relational database can be logically linked to information in another. For example, let's say you keep a list of people who owe you money in one database. In another, you keep a list of people who work with you. With a relational database, you could logically pair the two databases together, creating a list of people with whom you work who owe you money.

The important word in the last example is *logical*. Database systems that are not relational link information held in databases with physical pointers, to keep the computer informed as to the whereabouts of each piece of information (see Figure 1). Nonrelational databases are called "hierarchical databases," because of the physical link needed to store information in a database. Because the structure is so rigid, a lot of design and thought must go into a hierarchical database before the actual database is built. Relational databases don't have this limitation.

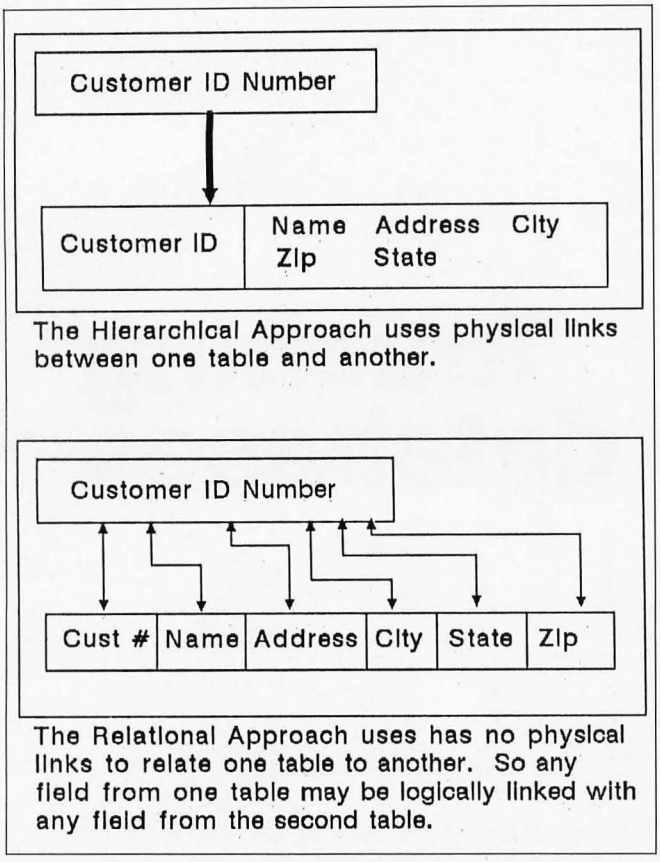


Figure 1.

The origins of a relational database.

The term *relational database* was originally coined by a man named Dr. E.F. Codd about fifteen years ago. Dr. Codd wrote a standard for a theoretical system called a relational database. At the time, the new standard was developed for large-scale mainframe computers.

Mainframe computers are great for handling large information processing needs; they normally have huge amounts of memory and disk space. Mainframes are still prevalent in large business and industry, but, with the advent of microcomputers, like the Atari ST, small business and even home users can have the utility and power of a mainframe at a reasonable price.

When Dr. Codd wrote the definition of the relational database, twelve rules were defined as a standard to determine if a program was truly relational. Dr. Codd was not a programmer, so when he wrote the definition it was referred to as a model after which programmers should pattern their software. In this article, the term *relational model* will be used to describe the application of the model. The rules developed by Dr. Codd are very specific in determining ease of use and flexibility of a database program.

The underlying message of the twelve rules is that a database should be able to separate the end user from the actual techniques of handling the contents of the database. When information is stored in a relational database, it's available by means of a common language. The end user enters requests in this common language, and the relational database internally figures out the best possible way of carrying out the user's command.

The first rule of a relational database determines how information is logically stored in a table. A table is made up of records. Records, in turn, are made up of a group of fields. This ordering of priorities makes a table look much like a spreadsheet, with fields running left to right as columns, and records running up and down as rows (Figure 2).

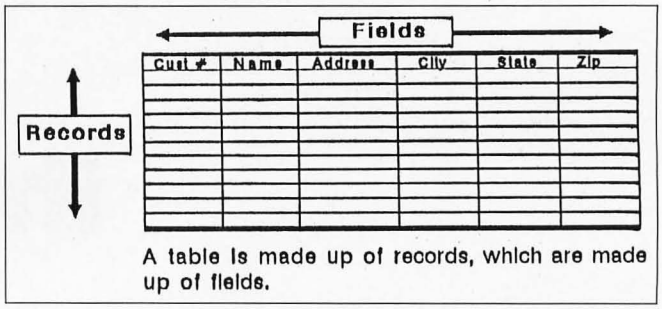


Figure 2.

The guaranteed access rule states that all information inserted into any table may be retrieved, manipulated or deleted without restrictions. So, if you create a table and store information in it, any information stored may later be accessed, modified or deleted under all circumstances. The hierarchical database links its information with physical pointers, while the guaranteed access rule makes it possible to link information in a logical manner. More on this later.

The SQL language.

The comprehensive data language rule describes the use of a set of common commands that are necessary to manipulate information and control the database. The data language rule shows that the language has to follow some sort of consistent behavior. IBM was one of the first companies to develop a working relational database product. IBM developed the SQL, or Sequel, language. Later, the SQL language was adopted as the industry standard relational database language by the American National Standards Institution (ANSI).

There are two main divisions of the SQL language: query commands and procedural commands. Query commands are those that cause manipulation of the structure of a table or the data stored in a table. Procedural commands control a database program and usually resemble the control commands of BASIC, Pascal or some other high-level language.

Query commands allow you to do two different things. First, you can create, modify or drop a table, thus allowing you to establish the structure of a table or change the structure of an existing table. Second, you can manipulate the information stored in an existing table.

There are four basic query commands to manipulate table information. With them you can:

— Retrieve information already stored in a table.
— Insert new information into a table.
— Update or modify information already stored in a table.

— Delete information stored in a table.

These are the four basic instructions that give you full control over all the information stored in a relational database.

The procedural commands affect the program flow and control of the system. Most of the procedural commands can be found in other high-level languages (such as IF. . . THEN, GOTO and LET.

Data types.

The guaranteed access rule of the relational model states that you can store and manipulate a variety of different types of information. There are many kinds of information. For example:

Types	Examples
Integer numbers	1, 100, 99
Decimal numbers	10.01
Character strings	"A person's name"
Normal dates	2/1/86
Short dates	FEB 1, 86
Long dates	February 1, 1986
European dates	1/2/86

The other relational data type is the logical data structure. The logical field contains one of three values: true, false or null. Null indicates that the field doesn't contain a value.

When a table is created, the fields that will comprise every record within the new table are defined. When a field is defined, three facts are stated about the new field: the field's name; the type of field; and how big the field is going to be.

What is the field's name?

A record is made up of fields. When a record is to be added to a table, the predefined fields are assembled into a new record. During this assembly period, each field is referred to by its field name. For example, let's say we're talking about a table that will hold a company's customer address. A typical entry would look something like this:

John Rayston
1313 Harbor Lane
St. Petersberg
Florida
31733

Each line of this address would be entered into a different field. So, the first line is the name field, next would be the address field, and so on. . .

Field name	Contents of field
NAME	John Rayston
ADDRESS	1313 Harbor Lane
CITY	St. Petersberg
STATE	Florida
ZIPCODE	31733

The column to the left contains the field names, while the right column contains the field's contents. If you were to retrieve the zipcode field, it would show *31733*.

What is the type of the field?

The relational model can handle many types of fields, like numbers, characters, dates, etc. When new information is put into a table, the field type becomes quite important. There's a big difference between how a *date* is stored and how a *number* is stored.

In the example above, two field types were used to store the customer address.

Field name	Field type	Contents
NAME	Character	John Rayston
ADDRESS	Character	1313 Harbor Lane
CITY	Character	St. Petersberg
STATE	Character	Florida
ZIPCODE	Integer	31733

The first four fields are character fields, since both numbers and letters may be used within them. The last field is an integer field, meaning only numbers can be stored. Regent Base support seven field types:

Character . . . . . Any numbers or letters may be used
Integer . . . . . Any whole numbers;i.e., 99, 100, 1, -1
Decimal . . . . . Any numbers with a decimal point, i.e., 100.00, 35.99, .25
Date . . . . . A date in this form: 1/1/86
Sdate. . . A Short Date, shown in this form: JAN 1, 86
Ldate . . . . . A Long Date: January 1, 1986
Edate . . . . . A European Date: 1/1/86 (month/date reversed)

How big is the field going to be?

Every field has a maximum size. If you want to store a person's name within a field, it must fit within the set maximum. For example, if you wanted to store the name *JOHN RAYSTON* in a name field thirty characters wide, the name field would contain:

```
John Rayston
123456789012345678901234567890
          1         2         3
```

But, if the name field had a maximum of only eight characters, it would look like this:

```
John Ray
12345678
```

Only the first eight characters of the field will be stored. You must define fields that are large enough to handle the data to be stored in them.

Of the seven types of fields, only three must have defined sizes: character, integer and decimal. Character fields can be from 1 to 32,000 characters wide and integer fields can be defined up to 10 digits wide. Decimal fields may also be defined up to 10 digits wide, with up to 10 digits to the right of the decimal point. Date fields are of a predetermined size.

Simple sample example.

Here's a typical example of the field definitions for a table.

Field name	Type	Size
Name	Character	20
Phone	Character	20
Age	Integer	2
		42 Bytes

This table will hold several people's names, phone numbers and ages. The first is the name field and is a character field, twenty characters wide. Next is the phone field, which is also a character type, twenty characters wide. The age field is an integer field that's two digits wide.

When new information is added to the table, the individual fields are assembled into one record. Let's say we're going to add the following information:

NAME	John Rayston
PHONE	(305) 555-1616
AGE	35

The record might look like this when in the table:

Name Field	Phone	Age
John Rayston	(305) 555-1616	35

If another person's information is to be added, the new record will be placed at the end of the table.

Applying SQL.

In the last section, we described an imaginary table which would hold three fields of information for every stored record. The query command to create that table would look something like this:

```
Create Table Phonelist Name Char(20),Phone Char(20),Age Int(2);
```

This command will create a new table, Phonelist. In it are three fields for every record: name field (up to twenty characters); phone field (also up to twenty characters); and age field (any number, *-99* to *99*).

Now that a table has been defined, let's see how the SQL query commands work. There are four basic query commands which manipulate information within tables: INSERT, SELECT, UPDATE and DELETE.

Inserting.

The INSERT command adds a new record into an existing table. When a table is created, it initially contains no records. This command adds records, while the other commands retrieve, modify and delete records.

Let's try adding a new record into the Phonelist table. The following command will do this:

```
INSERT INTO PHONELIST NAME="John Rayston",
PHONE="(305) 555-1616", AGE=35;
```

The INSERT command syntax is typical of the English-based format of the SQL language. Once this command has been processed, the Phonelist table will contain one record. Before going on to the SELECT command, let's add another record:

```
INSERT INTO PHONELIST NAME="Martha Windum",
PHONE="(818) 555-2646", AGE=22;
```

Thus, the Phonelist table will contain two records for use in the next section.

Selecting.

Now that we have two records in the Phonelist table, we can retrieve the records using the SELECT command. Each record contains three fields: the name field, phone field and age field.

The SELECT command gathers data from certain selected fields in one or more tables. The retrieved information may be displayed on the screen, printed, sent to an output port (i.e., RS232), or used in another SQL command.

The SELECT command has a number of different formats, but all start with one command:

```
SELECT * FROM PHONELIST;
```

This is the most basic form of the SELECT command. It will retrieve all (*) fields from the Phonelist table. When this SQL command is processed, the following information will be retrieved:

Name field	Phone	Age
John Rayston	(305) 555-1616	35
Martha Windum	(818) 555-2646	22

Selecting specific field.

When we entered the SELECT command, the * retrieved all of the fields in the Phonelist table. If we wanted to retrieve only the name field, we could substitute the field name for the * character. For example:

```
SELECT NAME FROM PHONELIST;
```

This command, when processed, will retrieve only the name field from the phonelist table:

Name
John Rayston
Martha Windum

So far, the SELECT command examples have retrieved all of the records in the table. If we want to be more specific about which records to select, we can add a WHERE clause to a SELECT command.

When processed, the following SELECT command first checks to see if each record meets the WHERE criteria; if a record tests true, it's displayed; if not, it's ignored.

```
SELECT * FROM PHONELIST
WHERE NAME="John Rayston";
```

When you process this SQL command, only the John Rayston record will appear.

In addition to a simple check for equivalency (as we just tried in the above example), there are a number of other expressions we can use in the WHERE clause. For example:

~Contains checks if a phrase or word exists in a field
^ Like checks a field using wildcard characters: * ?
! = Not equals
> Greater than
< Less than
>= Greater than or equal
<= Less than of equal

These expressions can also be used in the other SQL data commands.

You may also use the AND and OR operators within a WHERE clause. For example, suppose we want to retrieve only those records in which a person's age falls between 25 and 40. We could use the following command to perform this function:

```
SELECT *
FROM PHONELIST WHERE AGE <=25 AND AGE>=40;
```

Processing this command would retrieve the following record:

Name Field	Phone	Age
John Rayston	(305) 555-1616	35

Arithmetic express in select.

SQL also supports math functions when processing information. Suppose we wish to how old John Rayston will be in 10 years.

```
SELECT    name, age+10
FROM      phonelist
WHERE     name5"John Rayston"
```

Processing this command would retrieve the following record:

Name Field	Age
John Rayston	45

SQL can also retrieve the youngest person in the table by using the MINIMUM function. This function finds the smallest value for a numeric field.

```
SELECT
FROM        phonelist
WHERE       age=MIN(AGE)
```

Processing this command would retrieve the following record:

Name Field	Phone	Age
Martha Windum	(818) 555-2646	22

Updating.

Once a record has been inserted into a table, it can be modified using the UPDATE command. This command allows you to specify the fields and records to modify.

We previously added two records to the Phonelist table. Using UPDATE, we can change either of these two records. For example, let's say we want to change both phone numbers to 555-2244. The following command will do just that:

```
UPDATE PHONELIST SET PHONE="555-2244";
```

When you process this SQL command, the contents of the phone field in all records of the Phonelist table will be changed to the new phone number in the UPDATE command.

To see if the Phonelist table really was changed, enter the following SELECT command:

```
SELECT * FROM PHONELIST;
```

This will retrieve the two records in the Phonelist table as shown below. Notice that both phone fields are now equal to 555-2244.

Name Field	Phone	Age
John Rayston	555-2244	35
Martha Windum	555-2244	22

When we processed the last UPDATE, every record in the Phonelist table was modified. Like the SELECT command, UPDATE may also use the WHERE clause.

Using the WHERE gives us a lot of flexibility. If we wanted to change the Phone field of Martha's record, we could use an UPDATE WHERE command like this:

```
UPDATE PHONELIST SET PHONE="555-9512"
WHERE NAME="Martha Windum";
```

When this is processed, the contents of the phone field in the Martha Windum record will be changed to the new phone number.

Deleting.

Now that we know how to INSERT, SELECT and UPDATE, the last function to accommodate the guaranteed access rule is the DELETE command, to remove a record from a table.

There are two forms of the DELETE command: DELETE and DELETE WHERE. The former allows us to delete everything from a specified table, and the latter allows us to delete specific records from a specified table.

If you were to process the following command, every record in the Phonelist table would be deleted:

```
DEELETE PHONELIST;
```

Using DELETE WHERE, we can delete the Martha Windum record from the Phonelist table:

```
DELETE PHONELIST WHERE NAME="Martha Windum";
```

The real power.

So far, we've described the most primitive functions of a relational database using the SQL language. The real power of a relational database is in its flexibility. The SELECT command allows us to treat the information stored in a table as individual objects. The logical pairing of these objects of information is where the relational database shows its greatest power.

For example, let's say we have two tables which contain information about certain customers of a company. The first table, Custinfo, has two fields: Customer, which holds the customer number; and Name, which holds the customer's name.

Table: CUSTINFO

CUSTOMER	NAME
1010	John Sinkley
1020	Fred Barnes
1030	Mary Hartner

The second table, Accounts, has two fields: Customer, which holds the customer number; and Amount, which holds the amount owed to the company.

Table: Accounts

CUSTOMER	AMOUNT
1020	$ 30.00
1030	$132.00
1010	$ 5.49

The SELECT command is able to produce a report showing each customer's number and the amount owed. For example:

This command finds every record in both tables where the Customer fields in both tables hold the same name. The results of this command would be:

1010	John Sinkley	$ 5.49
1020	Fred Barnes	$ 30.00
1030	Mary Hartner	$132.00

This form of SELECT performs an algebraic function called "intersection." The intersection of the two tables is the common Customer field. The relational database logically produces the intersection of the two tables, based on the conditions of the WHERE clause.

Conclusion.

This article is by no means a complete description of all the power and functionality built into the relational database model; but it's a good jumping-off point.

The relational database model was previously restricted to large-scale mainframe computer systems. The relational approach to databases, though, is gathering momentum in the microcomputer world. Companies such as Ashton Tate, which produces dBase II and dBase III Plus, have announced relational products. Other companies already have a relational database on the IBM PC, like Microrim's RBASE System V. //

Database Delphi

by Matthew J.W. Ratcliff

Can you create a *fast* arcade game with C? Which is the preferred C development package for the ST? See what the experts have to say.

To C or not to C...

From: JUANSTXE (Juan Bravo)
To: MATRAT (Matthew J. W. Ratcliff)

Do you think C is powerful enough to produce a fast arcadelike game with over thirty sprites at the same time—without using machine language subroutines? I'm working on a "home-brew" version of the popular arcade game Gauntlet, and trying to capture the look and "feel" of the arcade original.

From: MATRAT
To: JUANSTXE

If your code is still on paper, you have a *long* way to go before you get something working on the ST—unless you're talking about C on the 8-bit. No high-level language is practical for the 6502 machine when it comes to something as huge and complicated as a full-blown video game.

On the ST, the 68000 microprocessor was designed to suit the UNIX/C philosophy. Yes, you can write a *fast* game in C on the ST, but you can write a faster one in assembly. How much faster? It depends on your programming talents.

A great C programmer might be able to write programs that run ten times faster than a sloppy assembly language programmer. I subscribe to the common belief that 90 percent of the time your program is running 10 percent of the total program (the low-level screen I/O drivers, for example).

What you do is write it all in C, get it debugged and working flawlessly, then sit back and *play* it. Let your friends play it. Get feedback as to what's too slow and figure out how to speed it up. You then rewrite those sections of code in assembly, about 10 percent rewrite, not 100 percent assembly—and you end up with the same result with a lot less effort. Megamax C has an ASM directive, allowing you to drop into 68000 assembly any time you like, then immediately switch back to C. The benefits can be phenomenal.

C compilers.

Are you still unsure about which C compiler is best for you? So was ATARIFLASH (Stephen G. Roquemore), until he started asking around Delphi. Below are the pros and cons of the top three C compilers for the ST.

From: ATARIFLASH
To: CFJ (Charles F. Johnson)

I'm seeking advice on which C compiler to buy. I have pretty much narrowed my choices to Megamax and Mark Williams C, with Lattice a distant third. Can you offer any advice?

From: CFJ
To: ATARIFLASH

I have the Megamax C package, and when I program in C (I still do most of my work in assembly), I like it a lot. Megamax is very fast, and the included shell makes it relatively painless to get through the edit, compile and link procedure. I personally haven't tried Mark Williams C, but the impression I have is that it's a very nice package, perhaps not as friendly as Megamax, however.

From: MATRAT
To: ATARIFLASH

CFJ is an assembly genius when it comes to programming the ST, but uses Megamax when he *does* do C work. I have Megamax and just *love it*. The development environment is excellent, the compiler is *fast*. The editor stinks, but you can always use MicroEMACS or any word processor (which can output your files as ASCII only). When I have a lot of "surgery" to do on some code, or cutting and pasting of routines between programs, I'll often use Word Writer for those tasks.

Mark Williams C has a much better and faster floating point than does Megamax; that's why Tom Hudson switched over to it for his latest version of CAD 3D, 2.0. But it comes on four (count them *four*) disks. Tom tells me that it just isn't practical to use it without a hard drive under you. Megamax fits on a single floppy.

If you plan to do a lot of *serious* floating-point software development, MWC may be the better choice. But, I think that Megamax is the best choice otherwise.

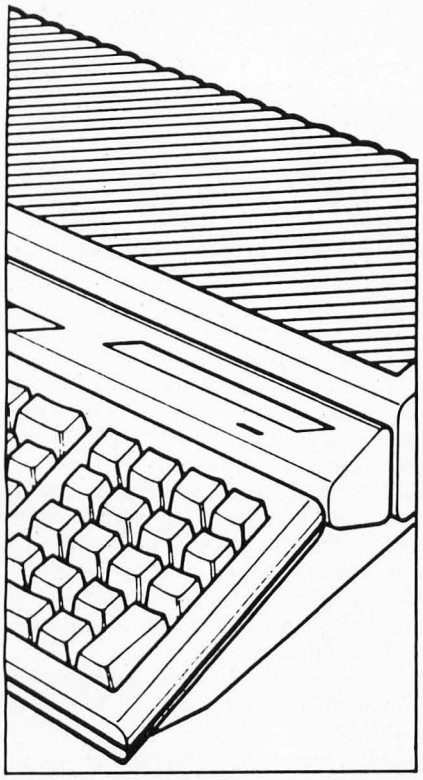
From: DLM (Daniel L. Moore)
To: ATARIFLASH

If you don't have a hard drive get Megamax. Mark Williams C is huge and really slow without a hard drive and a large RAMdisk. Megamax runs well on a DS/DD floppy, although that doesn't leave much room for source code. If you don't have a second floppy, a RAMdisk—even a small one—fills the bill.

Both are very good compilers and both have some big problems. I own both (plus Alcyon and Lattice), and use both Megamax and Mark Williams C on a regular basis, depending on what I'm writing. MWC has a debugger, albeit a lousy one, but it's better than nothing. Megamax has a resource editor. MWC is designed to run from a UNIX-style command shell, but I prefer Beckemeyer's shell. Megamax has a GEM operating shell, which isn't bad.

MWC comes with a lousy version of EMACS for editing. Megamax comes with a clone of Edit (a popular editor on the Mac), which is a real pain to use and very buggy. Whichever you get, I'd recommend another editor, such as the version of EMACS by Russ Wetmore, here on Delphi, or possibly Tempus from Eidersoft.

Megamax gives you the minimum you need (compiler, linker, librarian) and ignores the fancier tools (egrep, diff, etc.) that MWC has. MWC compiles slower and produces slower running programs, except in floating-point intensive applications. Megamax compiles *very* fast and produces faster code by 20 or 30 percent. Megamax also provides a resource editor for creating menus and dialog boxes, which is currently not available for the MWC package. ///



ST user

THIS MONTH:
A look into the future of CD-ROMs, and pinball plaudits.

by Arthur Leyenberger

I've seen the future and the future is almost here, depending upon which computer you own. You see, I've just returned from the first CD-ROM Expo, held in New York City.

To refresh *your* memory, "CD-ROM" means Compact Disc-Read Only Memory. Read Only Memory (ROM) is a concept familiar to most of us. The Atari ST contains six ROMs which hold the operating system, about 192,000 bytes. The information in the ROMs—GEM desktop, Input/Output routines, etc.—is permanent and can only be read by the computer. Another example of ROMs: cartridges used in Atari and other video game machines.

The CD used for the storage medium is similar to the Digital Audio Disc (DAD) that holds up to 73 minutes of music and is played on a CD player. The difference, of course, is that CD-ROMs contain data to be read and interpreted by computers. This data can take many forms: programs, databases, dictionaries, encyclopedias, or any other collection of information. The amount of information that can be stored on a CD-ROM is staggering—550 megabytes, or the equivalent of 1500 IBM PC floppies, 700 double-sided ST disks or 6000 Atari 8-bit floppy disks.

DADs and CD-ROMs both use players in which a laser shines a beam under the disc and detects the presence—or absence—of microscopic pits on the disc surface. A pit is a *0* and no pit is a *1*; thus, the binary code used by all computers.

The audio players use error correction procedures and rely on the human ear to mask brief gaps in the music, which can be due to temporary losses in reading the data from the disc. A few bytes missing from Bruce Springsteen's *Born To Run* is unlikely to be noticed. On the other hand, a computer cannot interpret absent data, so the CD-ROM drive must have greater precision and cause fewer errors when reading the data.

Atari's blast from the past.

My primary interest in attending this show—which was billed as "The first ever user-oriented CD-ROM conference and exposition"—was to find out when this technology was going to become inexpensive enough for home use. You may recall that, at the 1985 Summer CES, Atari showed a CD-ROM player and Grolier's Encyclopedia running on the ST. The CD-ROM drive was connected to the DMA port of the ST, and Activenture's (now called Knowledge Set Corp.) GEM-based Knowledge Retrieval System (KRS) was used to access the encyclopedia. At the time, Atari announced that the CD-ROM player would be available by the following fall, for "under $500."

Two years later, Atari has yet to introduce a CD-ROM drive, claiming that the prices have not fallen enough for a $500 retail price. This is true. Drive prices *have* fallen under the $1000 level, but, by the time you add even one application package (such as an encyclopedia), the cost climbs back up to the $1000 mark—a price that's hardly affordable by users whose computers cost half that much.

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

CompuServe — 71266,46
Delphi — NJANALOG

But progress is progress, albeit slow. The consumer side of the technology has shown that, as CD player prices dropped near the price of a decent cassette deck (about $200), the demand for audio CDs skyrocketed. Perhaps when the CD-ROM player price drops to the level of a decent peripheral—such as a printer, color monitor or hard disk—the computer side of the technology will also take off.

The show.

Regardless of the current affordability of drives (or lack thereof), there's good news to report in several areas. As mentioned above, CD-ROM drive prices are falling, and several people I talked to agreed that, within the year, an under-$500 drive was reasonable.

Second, the thorny tangle of data storage standards has been sorted out. Until recently, there were no set standards for how data was represented on the disc, the use of volume and file structures, and how directories and paths should be organized. Although every CD-ROM disc could be played on all drives, the software used by the computers to read the disc wasn't the same and caused tremendous compatibility problems. The lack of standards in the past has inhibited the growth of CD-ROMs, to be sure, but that hurdle has been overcome, since almost all CD-ROMs now use the so-called "high-Sierra" format.

Most significant were the introductions and demonstrations of new and existing CD-ROM application packages that one might actually want to use at home.

Of the thirty-odd vendors at the Expo, Microsoft had the most exciting product: Microsoft Bookshelf. If you do any kind of writing, whether it be term papers, reports, or computer articles—anything—the Bookshelf's collection of ten of the most useful reference works will certainly make your writing easier—and better, as well.

On one compact disc, the Bookshelf contains a dictionary (*American Heritage Dictionary* with over 200,000 definitions), a couple of style manuals (*Chicago Manual of Style*, Houghton Mifflin's *Usage Alert*), *Bartlett's Familiar Quotations* in its entirety, an almanac (*World Almanac* and *Book of Facts*), a thesaurus (*Roget's II: Electronic Thesaurus* with 500,000 synonyms), a spelling checker (Houghton Mifflin's phonetic *Spelling Verifier and Corrector*), a collection of over 100 form letters, a guide to business information sources, and the complete U.S. zip code directory.

Microsoft Bookshelf is a memory resident program that works with any IBM PC word processor. It's always ready to help in the background, making it easy to gather facts, find errors, or come up with just the right word. You can use a word in your document to trigger a search in any of the references, or you can enter your search criteria directly in a dialog box. Once the program has found the information for you, it can be copied from the reference library to your document automatically.

Microsoft Bookshelf retails for $295 and requires: an IBM PC or compatible with at least 512K bytes of memory; two disk drives or one disk drive and a hard disk; MS-DOS 3.1 or higher; a CD-ROM drive; and the Microsoft MS-DOS CD-ROM extensions. At this time, Microsoft has no plans to support the Atari ST, although certain company representatives (who asked to remain nameless) agreed that the ST would make an excellent engine for CD-ROM data retrieval. No doubt, once the CD-ROM drive prices do down, the ST will make a logical machine on which to run CD-ROM applications.

One of the best deals currently available to prospective CD-ROM users is the Laserdrive-1 CD-ROM system from Amdek. The Laserdrive-1 system contains the Microsoft Bookshelf, the MS-DOS CD-ROM extensions, a Hitachi CD-ROM disc drive and an interface card for the IBM PC or compatible. Interestingly, the Hitachi drive can also play audio CD—the only drive now available that's able to do so.

Audio CDs can be listened to with headphones plugged into the front headphone jack, or by using the stereo line output jacks to connect the drive to a stereo amplifier and speakers. Not only is this a thoughtful feature, but it helps bring the price of the drive down, by not requiring you to purchase a separate audio CD player. If you want to check out this system, look for a Sears Business Systems Center near you. They carry the Amdek CD-ROM drive and Microsoft Bookshelf system.

Another CD-ROM vendor at the Expo was Grolier. They were demonstrating their $300 American Academic Encyclopedia, that's very similar to their on-line electronic encyclopedia available through CompuServe Information Service. The Grolier's Encyclopedia was the first CD-ROM application for the general public. When you consider that the price of the 20-volume printed version is about $850, the price for this high-tech version—and a CD-ROM player—isn't all that high. Further, once you own the drive and have purchased the encyclopedia, yearly updates which are, in reality, entire new versions of the encyclopedia, cost under $200.

The electronic version of the encyclopedia is not only cheaper than the book form, but results in faster and often superior information searches. The advantages come from using key words for the search criteria. Following the program's prompted commands, you type in one or more words to search. The retrieval software quickly finds every occurrence of those words in the more than 30,000 articles, and displays the list of articles containing the words. You can then select an article and display the text with all occurrences of the search words highlighted. You can also print the article, or save it to disk.

Grolier was demonstrating their encyclopedia using an IBM PC. When asked about support for the Atari ST computer, John Cole, Director of CD-ROM products, replied that they're eager to support the ST—once the hardware is available for the consumer. He said, although Grolier is not a hardware company, he agrees that the ST would make an excellent vehicle for their product, and they'll be ready when Atari (or someone else) introduces a CD-ROM drive for the ST.

There were other interesting CD-ROM applications being demonstrated at the Expo, as well. Bowker was showing their Books in Print product that will be of most interest and use to libraries and bookstores. Geovision was demonstrating their U.S. Atlas GEOdisc, a CD-ROM geographic database. The U.S. Atlas contains a number of separate data layers for the entire country, including roads, waterways, political boundaries, railroads, federal lands, plus a place and landmark file of more than one million locations that can be accurately situated on the geographic displays.

Of course, there were all kinds of other vendors at the show, too. One company, Discovery Systems, was offering a special CD-ROM manufacturing deal. For $2000, they'll take your 550 megabytes of data stored on IBM PC floppy disks, and produce 200 CD-ROM discs for you. It's a good price if you need this type of thing—and many companies probably will do it.

Think for a minute; if you had a CD-ROM player attached to your ST right now, wouldn't you pay $10, $25 or $50 for the

entire collection of **ANALOG** and **ST-Log** magazines? I certainly would. But alas, the technology is still too young to make this a reality.

The recent CD-ROM Expo in New York gave me a taste of what the future will bring. Before we know it, we *will* have CD-ROM peripherals hanging off our STs. Without even leaving the house, we'll be able to boot up with the *Readers Guide to Periodical Literature*, do a little searching for articles on Atari computing, then insert the Collected Works of ANALOG, or the Complete Byte Disc into the player. We'll access the specific articles of interest, spool them to the printer, take the output to our favorite easy chair and relax while we read the articles.

Albert Einstein once said, "The whole of science is nothing more than a refinement of everyday thinking." I think I'm going to like the everydays of this future.

Game of the month.

The nice UPS man (before moving, I had a nice UPS woman) was just here to deliver a package from Accolade. Let's open it up and see what we have. Pinball Wizard. All right. I've been a pinball cuckoo since I was knee-high to a grasshopper. I *love* pinball games.

Pinball Wizard is an arcade-style pinball simulation and construction set. Up to four players can play sequentially, using either the mouse or keyboard. There are four different pinball simulations, each offering a variety of obstacles. Each game consists of a pinball table complete with backboard, flippers, tilt sensors, table shake and more. The level of difficulty can be adjusted for each table by altering parameters like point score, slope, number of balls and elasticity.

The Pinball Wizard building set features a parts menu containing targets, flippers, bumpers, discs and spinners that can be placed anywhere on the table. In addition to the obstacles, you can control the sensitivity of the tilt sensors, the speed of the ball, the slope of the table and the elasticity of rebounds. All of these settings are made by simply moving sliders with the mouse. A paint menu allows you to illustrate the table from a full palette of sixteen colors. If you really make a mess, there's an "oops" icon that deletes the last item painted. After a minute or so of inactivity while in "building mode," a demo screen begins showing the construction and illustrating of one of the default tables.

After the table has been constructed, you assign point and bonus values to the targets and obstacles, as well as special combinations, such as double bonus, extra ball or additional game. The tables you create with the building set can be saved to disk for future play. Pinball Wizard retails for $35.

I've played a lot of pinball games on the 8-bit and ST computers. Accolade's Pinball Wizard is one of the best I've tried. It takes full advantage of the color and sound capabilities of the ST, and features detailed graphics, lights and sounds that mimic a real game. If you enjoy computerized pinball simulations, or are looking for a new game for your ST, I highly recommend Pinball Wizard. //

Play Ball!

Create an exciting baseball league *and* World Series.

by Daniel A. Silvestri

The bases are loaded. One out, bottom of the ninth. The game is tied. You need this win to put your team in the play-offs. Here's the stretch, the pitch, the swing. It's a deep fly ball to center field. Back, back, pulled in on the warning track! Here comes the runner, racing from third to tag up. Here's the throw, then a headfirst dive. He's. . .SAFE! You win and advance your team to the play-offs and a chance at the World Series!

The season for the Great American Pastime is almost upon us. So let's get in some practice before opening day. We'll put together a package that will let you design a baseball team *you* can manage, providing full details for running a computer baseball league and drafting players, creating World Series rules, tracking ERAs (Earned Run Averages) and more.

Forming the baseball league.

As in real baseball, organization is important, and some money will have to change hands before the first ball of the season is thrown. This means purchasing some software. The purpose of the league we're about to form is to create a schedule of computer baseball games to culminate in a World Series competition.

To form the league, you'll need managers. So organizing personnel is the first task at hand. Each manager you recruit will represent one team in the league, and will lead that team to either the World Series Championship, or a less lofty ranking. So go out and sign up some managers who want to enter a team in your league.

A baseball package for great fun.

To begin, you'll need to purchase MicroLeague Baseball. You can find it at most good software stores, or you can call 1-800-Playbal to order directly from the publisher. You can also buy it from mail-order houses. Its retail price is $39.95 for MLB-I and $59.95 for MLB-II. If you like the thought of managing a baseball team, this game is worth every penny. The disk comes with full rosters of over twenty-five of the all-time great teams.

This game is not a hand-eye coordination game, but rather a strategy game that lets you *manage* a team. That means choosing a starting pitcher, a lineup, batting order, sending a pitcher to the showers, selecting a pinch-hitter or runner, and so on. You can call for a bunt, steal, aggressive or safe base running, or virtually anything else a manager does in a game.

In fact, the new ST version, MLB-II, takes into account "stadium dimensions, injuries, arguments with the ump, rain delays and other acts of God," according to the publisher, MicroLeague Sports Association. This version also contains a box score and statistics compiler so you can track your team's performance game by game. You can even use the real statistics of actual play, or update the starting stats on an ongoing basis. This adds multifaceted dimension to the game, as you can now track a batter's actual performance against a certain pitcher automatically.

You'll also want to buy the General Manager/Owners disk, which gives you the power to trade players, construct a team to include Babe Ruth and Bill Buckner, and even "create" players not on the original game disk. (I'll tell you more about this later.) The addition of this disk to your league will add a tremendous amount of flexibility and excitement to the game. Both the game disk and the General Manager/Owners disk should be available by the time you read this.

If you go with the MLB-I game (which doesn't include the box score and stats compiler), and you already own a spreadsheet program, you have the option of tracking your pitcher's Earned Run Average against certain opponents. In our Chicago League, we listed our starting and relief pitchers in the left column under "Names," then created three additional column headings for "Innings Pitched," "Earned Runs Allowed" (or runs this pitcher gave up) and one for the ERA. The formula you need for the ERA column is: Earned Runs Allowed, divided by Innings Pitched, times 9. This will compute ERAs, and give you a clear picture of who's doing the best job. The lower the ERA the better, of course.

If you want to get fancy, you can track

each pitcher against each opponent, to really know who should pitch against who. Just jot down the number of innings your pitcher played, and the runs he allowed to score for input after the game.

For additional fun, if you own a graphics program such as Print Master, you can create posters and banners announcing big games (like the first game of the World Series), to add atmosphere to your "ballpark," and enhance the mood and excitement of the games. You can even create letterhead for the appointed "Commissioner of MicroLeague Baseball" to use when settling any rule discrepancies, or create a calendar for each team with their schedules.

Hurry, the Rules Committee is about to meet!

Rules/Setup.

The above items will start you on the right track, while the following section will save you valuable hours in creating the rules and providing a setup to ensure fair play. Then get ready to put your management savvy to the test.

Player skill levels.

Because players have different skill levels, you want to create teams that will be balanced and of equal caliber. Therefore, it's necessary to create player guidelines to assure uniformity.

"A" caliber players will be those players who appear on either an All-Star Team or an American or National League Greats Team. This is true whether they're pitchers or fielders. These teams are on the game disk.

"C" caliber players are defined as any player with a batting average of .240 or below *and* 15 or fewer home runs. A "C" pitcher is defined as a pitcher (whether starter or relief) with a losing record *or* an ERA of 4.50 or higher.

"B" caliber players are all players (pitchers or fielders) who don't meet the criteria for either "A" or "C" players.

If you manually input player statistics from *The Baseball Encyclopedia*, for example, then any player who has appeared on an All-Star Team is classified as A. The same B and C rules apply.

Each team that you or your fellow managers create can have five A players, fifteen B players and five C players. A full roster is fifteen fielders and ten pitchers, and the program will require this. As manager, you can acquire any combination of A, B or C fielders or pitchers. One manager may elect to have ten C pitchers and load up the bench with hitters, for instance.

Rule 1: This balance of A, B and C players must always be maintained on your playing team.

The draft.

Yes, there will be a draft. So your managerial skills will be put to the test before you even see a bat or baseball diamond. You must decide on the best strategy: will you go for the A pitchers, or A hitters? For balanced players or those who can smack the most home runs? For lefties or righties?

Your roster of twenty-five players will consist of ten pitchers and fifteen fielders. The draft will be divided into twenty-five selection rounds, and in each round each manager will select one player for his or her team. Depending on the number of managers and teams in the league, you, the manager, should prepare a draft selection sheet, taking into account that you *may not* always be able to select your first choice. You should have several backups for each position and each class of player. So, if a competitive manager selects your first round draft choice for shortstop, you'll be ready to select another player without bogging down the system.

The manager must announce four things when choosing a player: (1) the player's name; (2) the team and year; and (3) his playing position and class (A, B or C). So a selection might go like this: "I take Johnny Bench, from the 1975 Reds. He's a catcher and an A class player." All managers should keep track of who's been drafted. Each manager selects in rotation (you can draw numbers for this) in each round.

Rule 2: A player can *never* appear on more than one team; once drafted, a player is ineligible to be drafted by another. (See "Trading Players" below.)

After twenty-five rounds, the reserve team roster draft is begun. This requires the same preparation by each manager, and proceeds in the same fashion as the regular draft. The only difference is that the roster is limited to nine players: three A, three B and three C class players, as defined under "Player skill levels." (See also "Farm club" below.)

Trading players.

After the dust clears and the noise subsides from the drafting process, managers can strike deals with other managers. For example, Manager One might know that Manager Two wanted Ted Williams, who Manager One has drafted. Williams, an A player, might be traded for Willie Mays, another A player. All kinds of deals can be struck to recreate the excitement in putting together a team of your own.

Rule 3: You must maintain the balance of five As, fifteen Bs and five Cs for your final team and three of each class on your reserve roster.

Farm club.

By having the General Manager/Owners disk, you can also have a reserve team, or farm club. As stated above, the reserve team is limited to nine players (fielders or pitchers): three As, three Bs and three Cs. Reserve team players are confined to a total of four trips up to the regular team or back down to the reserves. The manager must always exchange a like-caliber player for a like-caliber player (an A for an A, etc.). After the fourth trip up to the regular team or down to the reserves, the player becomes a free agent, and any other manager can pick him up for *free.*

Pitching rules.

To maintain the spirit and flavor of the game—and increase a manager's responsibilities—a starting pitcher must be off at least two consecutive games. Likewise, if for some reason a starter is used as a relief pitcher, he cannot make an appearance as a starter in the next two games. A pitcher who doesn't have at least one complete game to his record can't start. (Stats are plainly visible on-screen).

Rule 4: Pitching rotation must be maintained, so selecting your rotation is very important. Plan ahead!

Schedule of games.

Set up your game schedule by making a list that includes columns for the teams playing (home and away), which pitchers started and the final score. This should be made available to all managers, so they can review their team's performance, anticipate opponent's pitchers, and so on. You should alternate home and away games so that every other game a team plays is a home game. A schedule of about thirty games is ideal.

Play-off rules.

When the season's over, and the diamond dust has cleared, play-off times are here!

The play-off rules will vary depending on how many teams are entered in the league. You should have at least three teams, but can have as many as you want. Of course, the more teams, the more time it will take to complete the season. One good Saturday session, however, could see a fair number of games come and go—and some dreams too.

If you have only three or four teams in the league, then the top two teams play in the World Series. If there's a two-way tie for second place, a three-game play-off series is needed to see which of those two teams will advance to the World Series. First place automatically qualifies.

If you have more than four teams, the top four teams must have a play-off. The first place team plays the third place team in a three-game series, while the second place team plays the fourth place team. The play-off winners then advance to the World Series. The teams with the most runs during the regular season have home field advantage in the first and third games.

Remember: pitcher rotation counts all the way through the World Series, so plan accordingly.

The World Series.

The Micro World Series Champion will win the best of seven games. The home field advantage goes to the team with the most runs during the regular season—not counting play-off games. Home field advantage will be enjoyed the first, third, fifth and seventh games—if the Series goes that far.

A special series rule: no more rotations can take place from reserve team to regular team, or vice versa, *after* the play-offs. Therefore, whatever roster you have during the last game of the play-offs—or the last game of the regular season if your team isn't in the play-off games—is the team you bring into the Series. Any players who've become free agents can't be picked up after play-off games.

General tips for play.

As a manager, you'll be very busy throughout the game. Remember, the statistics for each player are very important. They outline his batting average, number of hits, doubles, triples, home runs, strike outs, bases on balls, stolen bases and more. Each player is also rated for general fielding ability, so knowing your top defensive players can help you late in a game, because these guys—just like their real-life counterparts—make errors too! You may want to do some substituting.

Selecting pinch-hitters, pinch-runners, and knowing when to pull your pitcher, are all important issues. If you can, practice against the computer and remember who's doing the hitting. To be fair, let other managers practice too.

A conservative game plan usually works best, but then again, you may want to throw your opponents off by stealing bases a lot or using the hit-and-run. Be creative, and always think.

One last tip: each player has a listed batting average, but the number of hits, plus the number of bases on balls (walks) he received, divided by his At-Bats will give you the statistical average for that player, or how often he gets on base (no matter how). Know your players.

Conclusion.

Well, if you have any desire to be a baseball manager, this forum is about as good as it gets. And your heart will literally pound with excitement! The software is superb in recreating a true feel for managing the game, and the rules outlined above will provide a quick and headache-free environment for getting your league teams on the playing field as soon as the Commissioner yells, "Play ball!"

Daniel A. Silvestri has been working with computers for three years now, and is employed by a major publisher of business software for MS-DOS machines.

His interests in computers lie in business applications, financial planning, adventure gaming and strategic battle simulations.

Refresh Your Memory

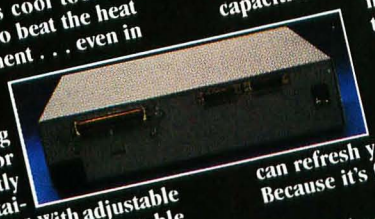
And Keep Your Cool.

Introducing the ST Hard Drive System from ICD that refreshes your memory better than any other ST hard drive around. *No problem.*

It's the drive that not only looks cool, but stays cool too. All because of a built-in fan that knows exactly how to beat the heat and maintain a calm, cool and collected environment . . . even in *your* most heated situations. *No sweat.*

And, it's the hard drive that takes a refreshing approach to aesthetic case design as well. See for yourself. It's easy on the space, fitting perfectly under the monitor. And it's easy on the eyes, tailored to look great in the company of your Atari ST. With adjustable front legs, your monitor gets the lift it needs for comfortable viewing. *No strain.*

Despite a sleek and compact exterior, the ICD ST Hard Drive System is packed full of overwhelming enhancements. Like an internal clock that tags each file with up-to-the-minute time and date information. Not to mention expansion capabilities that welcome the connection of up to six SCSI devices and daisy-chaining Atari's DMA Bus (ACSI). It's available in more memory capacities than you can imagine. With storage ranging from 20 megabyte systems up to 280 megabytes. And, there's dual drives too, that double your protection and double your confidence. *No stress.*



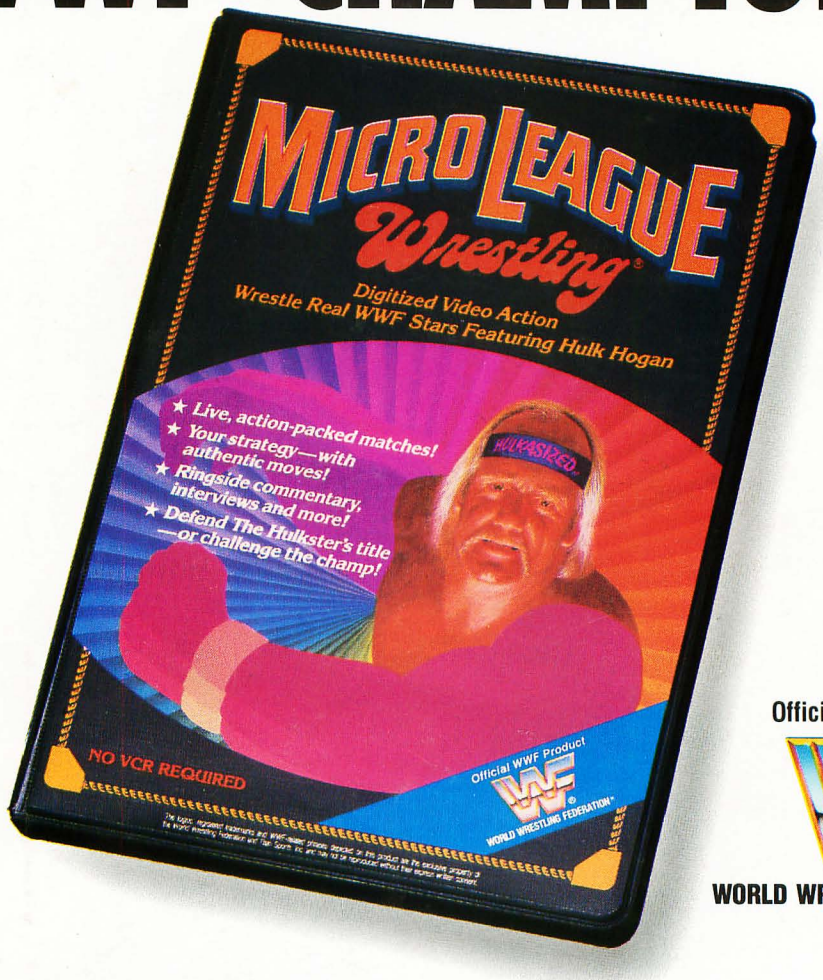
So, the next time you think about a hard drive for your Atari ST, think about the countless ways we can refresh your memory. It's the only drive worth remembering. Because it's from ICD. *No wonder.*

For further product information, please call or write for our catalog today.

ICD

1220 Rock Street
Rockford, IL 61101-1437
(815)968-2228
MODEM: (815)968-2229
FAX: (815)968-6888

INTRODUCING ANOTHER WWF® CHAMPION...



Official WWF Product

WORLD WRESTLING FEDERATION®

FEATURING EXCLUSIVE DIGITIZED VIDEO ACTION.

Computer game players...bored with "pretend" animation and "pretend" wrestlers? Your head and wrist limp from joystick coordination? Then take the challenge of the *only* strategy simulation good enough to be sanctioned by the World Wrestling Federation! MicroLeague Wrestling®.

Defend Hulk Hogan's™ claim to the championship! Or you can be Randy "Macho Man" Savage or "Mr. Wonderful" Paul Orndorff and try to capture the belt. See the real WWF Superstars perform the moves *you* want.

MicroLeague Wrestling has all the action and drama of live WWF matches including interviews by Mean Gene Okerlund and ringside commentary by Vince McMahon, Jesse "The Body" Ventura, not to mention the lengendary, Bruno Sammartino. And who knows what lurks in the mind of Bobby "The Brain" Heenan or in the heart of the lovely Miss Elizabeth?

So check it out, dude! Orndorff's devastating pile driver, Savage's awesome elbow drop–and everything else that's in the book or whatever you can get away with! It's MicroLeague Wrestling. The *ultimate* in computer sports simulation!

If your computer/software store is out of stock, no need to retaliate with the Hulkster's Atomic Drop. Just call us at (302)368-9990. And for VISA and MC orders phone us at 1-800-752-9225. Or drop us a (clothes) line to **MicroLeague Sports, 2201 Drummond Plaza, Newark, DE 19711.**

AVAILABLE FOR ATARI ST AND COMMODORE 64/128 COLOR TV/MONITOR REQUIRED. 1–2 PLAYERS.

CIRCLE #101 ON READER SERVICE CARD