BUILD A CARTRIDGE-PORT INTERFACE FOR YOUR ST!

ST LOG

THE ATARI ST MONTHLY MAGAZINE

FDC 50076

JANUARY 1989 ISSUE 27

USA $3.95
CANADA $4.75

TRANSWARP
SPEED UP YOUR FLOPPIES BY 100%

OMNILIFE
STEREO AND 3-D GRAPHICS

DRAMA-CIDE
A GAME OF MYSTERY

...WS:
...CADD
...ssible Mission II
...op Publisher ST

SupraDrive
Atari ST Hard Disk Systems

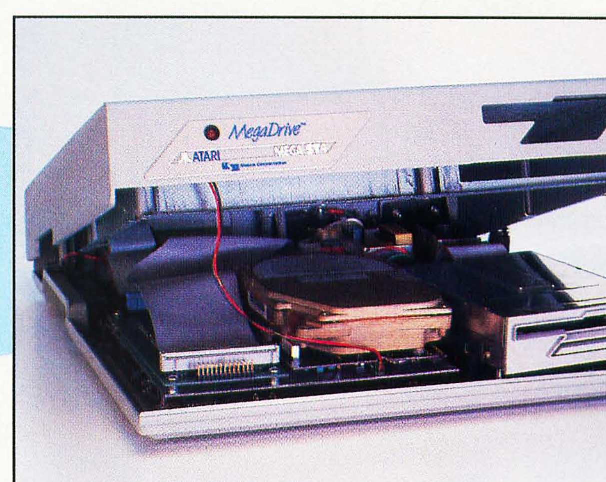
- High performance hard disk drives for Atari ST and Mega ST computers
- DMA and SCSI connectors for daisy-chaining additional devices
- 20, 30, 45, 60, 120, and 198MB capacities
- Includes these useful utilities:
  - Formatting software that allows dividing the drive into as many as 12 partitions
  - Autobooting directly from the hard disk
- Battery backed-up real-time clock
- Built-in fan

NEW!
SupraDrive FD-10
10MB Removable Floppy

- Uses 10MB removable floppy disks for unlimited storage
- Fast transfer speed and disk access (80ms average access time)
- Reads IBM 360K and 1.2MB floppy disks
- Connects to Atari ST DMA port
- Built-in SCSI and DMA connectors
- Battery backed-up real-time clock
- Useful as primary storage and for hard disk backups and archival data storage

NEW!
MegaDrive
Internal Hard Disk System

- 40MB capacity
- Mounts inside the Atari Mega ST case
- High speed DMA data transfer
- Fast RLL drive — 28ms access time
- Leaves DMA and 68000 port free for other peripherals



Available at your local dealer, or call:

Supra Corporation

1-800-727-8772 (Orders Only) or 503-967-9075
1133 Commercial Way, Albany, OR 97321 USA
Fax: 503-926-9370 Telex: 5106005236 (Supra Corp)

CIRCLE #101 ON READER SERVICE CARD.

EDITORIAL

BY ANDY EDDY
ASSOCIATE EDITOR

Change.

That word rears its head at us constantly, day to day, month to month. Whether we care for its effects or not, we're forced to face it in our own manner. We have to. Life goes on.

Change.

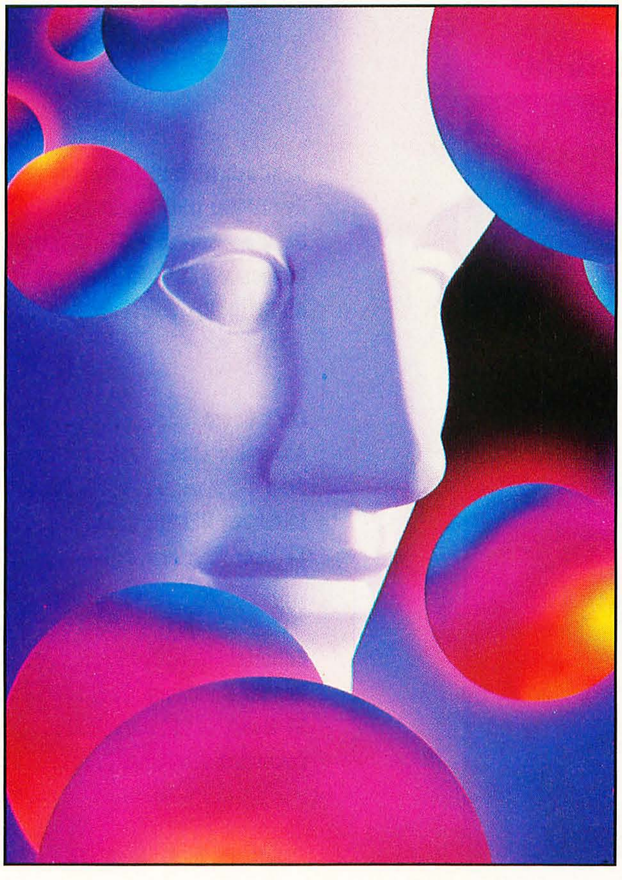
Thanks to the lead time involved in creating a magazine, I'm writing this opening to the January issue in late-September. In my present tense, the Space Shuttle began a new start this morning, following the layoff brought on by the Challenger disaster. Images of the fireball and smoke trails from nearly three years ago still loom ominously in the minds of everyone; so most of America watched this launch of Discovery with gritted teeth and tense anticipation. With all our advanced technology, these things normally turn out well. But the reworking of the shuttle program gave everyone a chance to shake the *ho hum* feeling out of their minds, making the space program new again.

Change.

This past week—again, in my current events—Neil Harris, long considered the public voice of Atari in the United States, called it quits. He leaves behind the harsh reception a public relations position brings, especially with a company as active in the Vaporware Hit Parade as Atari has been. Harris was one of the remaining crew from the Tramiel/Commodore days, brought over to Atari in an attempt to again bring the company to the forefront of computer innovation. Now Harris takes his marketing abilities back East, putting them to work for the GEnie telecommunications service. We wish him luck in his new venture.

Change.

By the time you read this, Atari will have most likely made public its plans for the continuing of its computer division. In the last year or so, for example, more and more ST owners have expressed their disdain with the way Atari has handled their favorite computer. Some yelled over the lack of IBM compatibility, claiming that this



would be the final step to more public recognition and respect. Others point to the lack of continued development from inside Atari, bringing a reliance on third-party contributions (such as the enhanced GDOS replacement, Codehead Software's *G+Plus*).

Primarily, most are upset with Atari's marketing of the ST in the U.S.: namely, no major advertising. Atari cries that a shortage of computers makes it impossible to advertise, but the lack of public visibility doesn't make it any easier to sell what few it has. Besides, IBM, Apple and Commodore don't seem to have major problems keeping up with the demand for their computers. Strong rumors abounded that the COMDEX show in November will have changed that—that Atari had rented a large booth and would be introducing some spectacular products. The word drifting through the Atari community is that COMDEX will spell make-or-break for Atari and its computer development aspirations. (As you read this, all of that will be in the past.)

Change.

Lastly, I write this piece from a laptop computer in the heart of Southern California. The temperature is expected to hit 100 today, and a local disc jockey warned us that the air would be "brown and chunky." Around here they joke about the smog, like calling it "filtered sunshine."

I'm here to take on a new job, one that will bring me closer to the operation of ST-Log and its sister publication, ANALOG; but mostly I'll be involved in our new magazine VIDEOGAMES & COMPUTER ENTERTAINMENT. It's a wonderful opportunity.

Change.

Lives in transition. Nothing we can do except replace the calendar on the wall with one that says 1989. Take down the Christmas tree and put away the ornaments. Let's sing a few bars of "Auld Lang Syne" with a toast, then try to deal with all this *change* as best as we can.

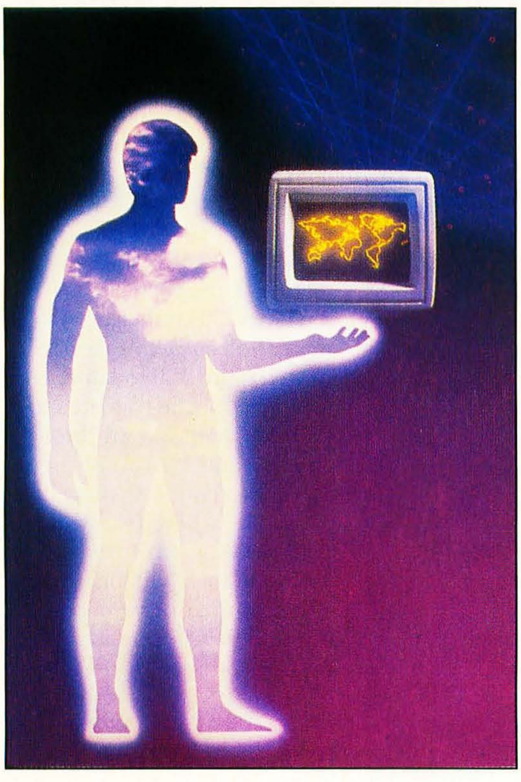
Happy New Year from ST-Log.

IN THIS ISSUE

FEATURES

DRAMA-CIDE 10

SOFTWARE ENGINEERING 50



REVIEWS

COLUMNS

DEPARTMENTS

PROGRAM · LISTING · GUIDE

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.

STAFF

Publisher: Lee H. Pappas. **Executive Editor:** Clayton Walnum. **Associate Editor:** Andy Eddy. **Art Director:** Andy Dean. **Managing Editor:** Dean Brierly. **East Coast Editor:** Arthur Leyenberger. **West Coast Editor:** Charles F. Johnson. **Contributing Editors:** Ian Chadwick, Frank Cohen, Maurice Molyneaux, Steve Panak, TG, Douglas Weir. **Copy Chief:** Katrina Veit. **Copy Editors:** Sarah Bellum, Anne Denbok, Pat Romero, Kim Turner. **Chief Typographer:** Klarissa Curtis. **Typographers:** Judy Villanueva, David Buchanan. **Editorial Assistant:** Norma Edwards. **Contributors:** Blake Arnold, A. Baggetta, Randy Constan, Tom Hudson, Dan Moore, Kevin Pehr, Matthew J.W. Ratcliff, David Small. **Vice President, Production:** Donna Hahner. **Production Assistant:** Steve Hopkins. **National Advertising Director:** Jay Eisenberg. **Corporate Advertising Director:** Paula S. Thornton. **Advertising Production Director:** Janice Rosenblum. **Subscriptions Director:** Irene Gradstein. **Vice President, Sales:** James Gustafson.

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

ST-Log Magazine (L.F.P., Inc.) is in no way affiliated with Atari. Atari is a trademark of Atari Corp.

ADVERTISING SALES

Correspondence, letters and press releases should be sent to: Editor, **ST-Log**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

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 (see Authors below), 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.

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

Address all advertising materials to: Paula Thornton — Advertising Director, **ST-Log**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

PERMISSIONS

No portions of this magazine may be reproduced in any form without written permission from the publisher. Many of the programs printed herein are copyrighted and not public domain.

Due, however, to numerous requests from Atari club libraries and bulletin-board systems, our policy does allow club libraries or individually run BBSs to make certain programs from **ST-Log** available during the month printed on that issue's cover. For example, software from the January issue can be made available January 1.

This does not apply to programs which specifically state that they are *not* public domain and, thus, are not for public distribution.

In addition, any programs used must state that they are taken from **ST-Log** Magazine. For further information, contact **ST-Log** at (203) 645-6236.

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 with your manuscript to **ST-Log**, P.O. Box 1413-MO, Manchester, CT 06040-1413.

ST LOG THE ATARI ST MONTHLY MAGAZINE

JANUARY 1989
ISSUE 27

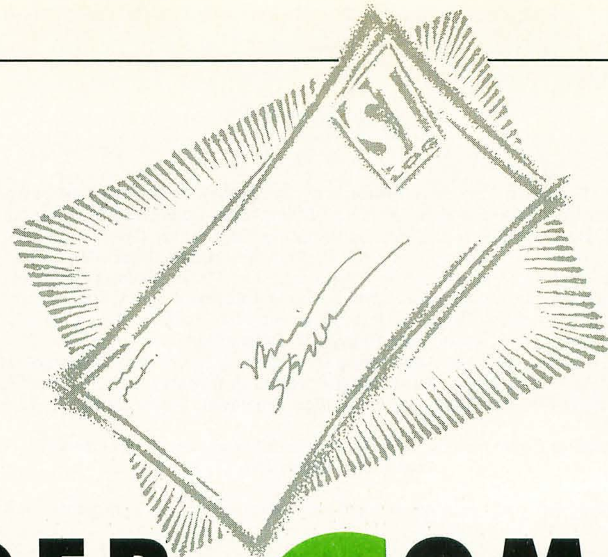
COVER ILLUSTRATION:
BY ALAN HUNTER

INTERIOR PHOTOGRAPHY:
STEVEN HUNT
DON LANDWEHRLE
MICHEL TCHEREVKOFF
MICHAEL ROCHIPP

STATEMENT OF OWNERSHIP

Statement of Ownership, Management and Circulation (Required by 39 U.S.C. 3685). 1A. Title of Publication: ST-Log. 1B. Publication No.: 08909601. Date of Filing: September 26, 1988. 3. Frequency of Issue: Monthly. 3A. Number of Issues Published Annually: 12. 3B. Annual Subscription Price: $28.00. 4. Complete Mailing Address of Known Office of Publication: 9171 Wilshire Blvd. Third Florr. Beverly Hills, CA 90210. 5. Complete Mailing Address of the Headquarters of General Business Offices of the Publisher: 9171 Wilshire Blvd. Third Floor, Beverly Hills, CA 90210. 6. Full Names and Complete Mailing Addresses of Publisher, Editor and Managing Editor: Publisher: Lee Pappas 9171 Wilshire Blvd. Third Floor, Beverly Hills, CA 90210. Editor: Clayton Walnum, 9171 Wilshire Blvd., Third Floor, Beverly Hills, CA 90210. Managing Editor: Dean Brierly, 9171 Wilshire Blvd., Third Floor, Beverly Hills, CA 90210. 7. Owner: LFP Inc., 9171 Wilshire Blvd., Third Floor, Beverly Hills, CA 90210. Names and Addresses of Stockholders Owning or Holding 1% or More of Total Amount of Stock: Larry Flynt Publications Inc., 9171 Wilshire Blvd., Third Floor, Beverly Hills, CA 90210. 8. Known Bondholders, Mortgages and Other Security Holders Owning or Holding 1% or More of Total Amount of Bonds, Mortgages or Other Securities: Larry Flynt, 100%. 9. Does not apply. 10. Extent and Nature of Circulation: Average number of copies of each issue during preceding 12 months: A. Total number of copies (net press run): 23,439. B. Paid and/or requested circulation. (1) Sales through dealers and carriers, street vendors and counter sales: 9,567. (2) Mail subscriptions paid and/or requested): 5,021. C. Total paid and/or requested circulation: 14,588. D. Free distribution by mail, carrier or other means, samples, complimentary and other free copies: 1,063. E. Total distribution: 15,651. F. Copies not distributed. (1) Office use, left over, unaccounted, spoiled after printing 2,008. (2) Return from news agents: 5,780. G. Total: 23,439. Actual number of copies of single issue published nearest to filing date: A. Total number of copies (net press run): 27,472. B. Paid and/or requested circulation. (1) Sales through dealers and carriers, street vendors and counter sales: 13,589. (2) Mail subscriptions (paid and/or requested): 5,166. C. Total paid and/or requested circulation: 18,755. D. Free distribution by mail, carrier or other means, samples, complimentary and other free copies: 710. E. Total distribution: 19,465. F. Copies not distributed. (1) Office use, leftover, unaccounted, spoiled after printing, 2,064. (2) Returns from news agents: 5,403. G. Total: 27,472. I certify that the statements made by me above are correct and complete.

—Vincent M. Del Monte
V.P., Client Relations



READER COMMENT

The Mail-Order Dilemma

Last month, I received in the mail my first issue of ST-Log. I had been reading a friend's past issues for months and decided to start my own subscription. You may find this strange being that I do not own an Atari computer.

Here is my dilemma: I am in the market for (and am very anxious to get) an Atari ST computer. I have sent written requests for information about ST system packages to many of the advertisers in your magazine. From one of the companies, whom I won't mention by name, I have received a very confusing and distressing response. Right off the bat, the company tells me, "Atari no longer allows mail order to sell their products."

This almost sent me into a state of shock! I am in the U.S. Army and am stationed overseas. For myself and countless others, mail order is a vital method of obtaining certain merchandise that is not readily available as it is in the U.S. I would think that orders by mail would constitute quite a large portion of Atari's total sales.

How can Atari do this? Could you please enlighten me with any information that you may have as to why this new policy has been initiated, if in fact it is a policy at all? —Andrew M. Brown
APO, New York

Yes, it's true that Atari now prohibits the sale of the ST computer systems by mail order. Yes, it is also true that mail order used to account for a goodly number of Atari sales, and unfortunately, you are not alone in your concerns. Believe it or not, though, Atari had no choice but to incorporate this drastic strategy, even though it will cost them some sales.

You see, a good service network is one of the most important things a computer company must establish if it is to gain the trust of the buying public. Whenever a computer is bought via mail order, the purchaser is gambling that the computer will not require service in the near future. Because mail-order companies do not provide service facilities, if a computer bought through them stops functioning, the purchaser must either return it for replacement (assuming, of course, that the machine is less than a couple of weeks old) or send it to Atari for repairs.

All official Atari dealers are now required to have service facilities. That way the purchaser knows that, should his machine need fixing, there is someone handy who can do the job quickly and cheaply (free if the machine is still under warranty). In order to give the ST the credibility it needs to be a success, Atari had no choice but to eliminate all mail order.

The no-mail-order rule applies only to the machines themselves though. Once you've purchased your computer, you will have no difficulty obtaining through mail order the software you need. So the situation isn't as bad as you might think. And once Atari gets its new dealer network installed, the machines themselves will be more readily available.

I would suggest that you give Atari a call (408-745-2000) and tell them you would like to purchase a computer. I'm sure they will find a way to satisfy your needs. ■

All official Atari dealers are now required to have service facilities. That way the purchaser knows that, should his machine need fixing, there is someone handy who can do the job quickly and cheaply.

MAIL YOUR COMMENTS TO:
ST-LOG
P.O. BOX 1413-MO
MANCHESTER, CT
06040-1413

Articles such as *Software Engineering* provide information and motivate readers to appreciate more fully, the fine art of programming.

More Applications

I want to compliment you on the "new" ST-Log. After the recent delay for several months, I was hesitant to renew my subscription. But the recent issues have renewed my faith in the magazine.

I want to especially commend Frank Cohen for his article on relational databases (April '88) and Karl Wiegers for his series on software engineering. These types of articles provide information and motivate readers to appreciate more fully, the fine art of programming.

I was also impressed with the *Opus* spreadsheet in the September '88 issue. First, how do I order that month's program disk? Second, I would like to see more application programs published. Fancy graphics and games are nice, but solid application programs allow users, both novice and experienced, to enjoy the abilities of these programs without having to spend big bucks on the "name" programs. Just one *Lotus 1-2-3* purchase saved would pay for several years of ST-Log. If the reader becomes a "power user" after several months or years with your application program, they can always upgrade, but many users will be satisfied with a spreadsheet like Opus forever! Keep up the good work!

—James W. Maki
Indianapolis, Indiana

Thanks for all the compliments, Jim. I'm sure the authors of the articles you mentioned will also appreciate your kind words. To answer your question, if you flip through this issue, you'll find a full-page advertisement that'll give you all the details you need in order to get your copy of an* ST-Log *disk. Keep in mind that only the last six months are represented in the list; other disks may also be available.

***We agree with you that the applications programs published in our pages save* ST-Log*'s readers a lot of money, as well as provide useful, easy-to-acquire software. We have many other interesting programs forthcoming, such as an outline generator and the long-awaited ST version of MicroCheck. The editors of* ST-Log *encourage software authors to send their programs in to us for evaluation. Programs chosen for publication in* ST-Log *receive the highest author payments available for magazine publication.* ■**

Where's Microsoft Write?

I recently received and read the September '88 issue, and I wish to say I am quite impressed with the overall improvement in the magazine's layout and content. I do wish you would soon improve the cover to the quality of the old ST-Log. It was really quite outstanding every issue.

More specifically, I would like to comment on Mr. Plotkin's survey of ST word-processing programs. I was impressed by the article in general, but, as a user of *Microsoft Write*, I feel that the omission of this program from the article was a glaring error. Many people will make a purchasing decision based upon this article, and it is indeed worthy of this. However, these people will not be aware of Microsoft Write without consulting other sources. I feel that this program, though containing flaws, is very useful. I use it with GDOS and find the print quality on my FX-85 to be absolutely superb! Would it be possible to ask Mr. Plotkin to review it as an addendum to this article? I think his comments comparing it with the others in his article would be very useful to those seeking a word processor. Please keep up the excellent work! ST-Log is the *best* ST magazine I have seen.

—Stephen G. Roquemore
Sonoma, California

Well, I guess this is confession time. The omission of Microsoft Write from the word processor article was not so much an error, but a necessity due to the time constraints under which the article was written. Poor David Plotkin was given only two weeks to complete his article, and because he was not, at that time, familiar with Microsoft Write, he warned us that he would not be able to include it. I am sure, though, that Dave would be willing to fill in the gap. What do you say, Dave?

***And thanks, Steve, for the compliments.* ■**

Attention Owners of Enhanced ST BASIC!

It has come to our attention that, due to ST BASIC's recent enhancement, the keyword DEF SEG is no longer supported. This means that people with enhanced ST BASIC cannot run the "BASIC loader" programs, such as the one used to create "The DEGAS Fast Loader" from the September '88 issue of ST-Log.

Owners of enhanced ST BASIC should make the following changes to any of the BASIC loader programs that they wish to run:

First, delete lines 100 through 200 from the BASIC loader program you wish to modify (such as Listing 1 of the DEGAS Fast Loader), and then add the following code:

```
90 'REPLACE "FILENAME.EXT"
 IN LINE 100 WITH THE PROP
ER FILENAME
100 OPEN"R",#1,"A:FILENAME
.EXT",16:FIELD#1,16 AS B$
110 A$="":FOR I=1 TO 16:RE
AD V$:IF V$="*" THEN 140
120 A=VAL("&H"+V$):PRINT "
*";:A$=A$+CHR$(A):NEXT
130 LSET B$=A$:R=R+1:PUT 1
,R:GOTO 110
140 CLOSE 1:PRINT:PRINT "A
LL DONE!"
```

Now, go to the next-to-the-last line in the BASIC Loader program (on the DEGAS Fast Loader that would be Line 3090) and add enough 00's to the data to make the line 16 bytes long. For example, for the DEGAS Fast Loader, you would modify Line 3090 so that it looks like this:

```
3090 data 06,0E,04,04,04,0
4,04,06,00,00,00,00,00,00,
00,00
```

The program should now run properly with enhanced ST BASIC.

Use ST Check to validate your typing *before* making the above changes. The modified program will not yield the same checksums. ■

ST NEWS

The New GDOS is *G+Plus*

Charles Johnson and John Eidsvoog have produced their own version of GDOS, the missing part of your Atari ST's GEM operating system that allows programs to display and print graphic fonts and styles. Johnson and Eidsvoog began the project as curious hackers; they were interested in the inner workings of GDOS and the font system being used. What they found was unexpected.

GDOS was originally developed in C by Digital Research, the creators of the GEM operating system. During the development of the ST, Atari and DRI decided not to include GDOS in the ST's ROM operating system because of physical size limitations of the ROM and problems interfacing new device drivers to the ST version of GEM. Atari fell from grace with DRI shortly after the ST was released, which made hopes of GDOS being made available for the ST a distant chance. Through some delicate maneuverings, Atari eventually bought the rights to produce an ST version of GDOS. First tests of GDOS were embarrassing. The ST Desktop slowed down to a crawl, and most ST programs bombed when run.

Johnson and Eidsvoog originally intended to seek Atari's official approval of the new GDOS, before they realized the delay in releasing the product to the market such a move would cause. Instead, Johnson and Eidsvoog created a partnership, Codehead Software, which is marketing *G+Plus* directly to ST users.

G+Plus is a more technologically superior software product than GDOS. When using GEM compatible fonts, GDOS forces your ST to load all of the fonts into memory when you power up your computer. This causes long boot-up periods and severely reduces available memory for your applications—try running a spreadsheet while GDOS is resident, you will sometimes find less than half the worksheet size available. G+Plus solves this problem by allowing you to define which fonts will be loaded when an application is opened. When running your word processor, many fonts might be loaded. While running a graphics program, a different set of fonts could be loaded.

G+Plus offers many other advantages, which will be covered later in a full ST-Log review. The package comes with a large instruction manual, and has a list price of only $34.95 (add $2 for shipping and handling).

Codehead Software
P.O. Box 4336
North Hollywood, CA 91607

CIRCLE #130 ON READER SERVICE CARD.

Give Your ST Math Smarts

A math coprocessor board for your Mega ST has been developed by Malcolm Campbell, a hardware designer in Fullerton, California. The special board plugs into the expansion slot of a Mega ST, giving your Mega ST a 16-mhz math coprocessor that is ready to tackle scientific or engineering problems. The coprocessor is memory mapped according to Atari's recently released guidelines for memory addressing on the Mega.

Campbell has been running mostly raw speed trials in the final testing of the new hardware accessory. Motorola provides floating-point multiplication *and division* software routines that can process 30,000 iterations with software conversion in five minutes. Using the math coprocessor, the same iterations take less than 17 seconds. To the average end-user, this means that *CAD-3D* could run up to 20 times faster with the coprocessor. It also means the Mega ST can be sold into the scientific community as a powerful numbers-cruncher.

Malcolm Campbell
2380 West Orangethorp #13
Fullerton, CA 92633
CompuServe: 70157,3363
DELPHI: Oscartalker

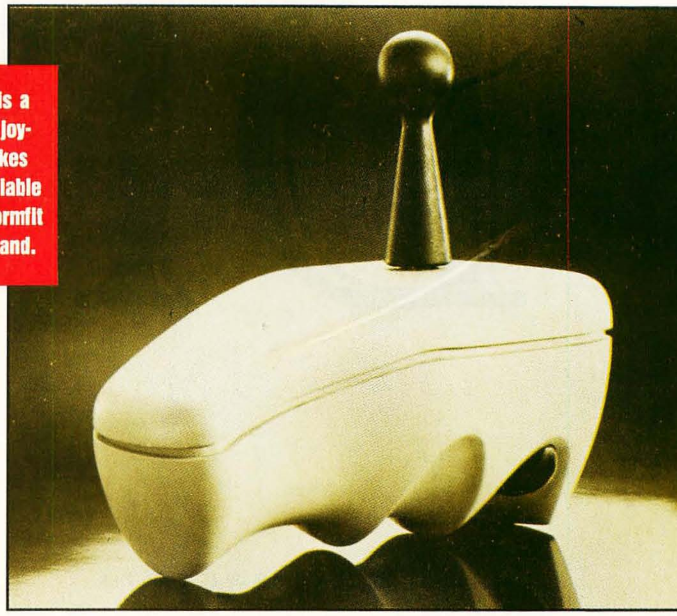
CIRCLE #131 ON READER SERVICE CARD.

Stop Suffering From *Pac-Man* Blisters

After a hearty game of *Pac-Man* played with a standard $9.95 Atari joystick, your index finger can look significantly older than the rest of your body. The Atari joystick is functional, but hardly developed for the human hand. Wico has introduced a new joystick which solves *Starglider* wrist and *Dungeon Master* elbow.

The *Ergostick* is a new design in joysticks that makes use of a soft, pliable plastic that is formfit to the human hand. The top portion of the joystick uses a stiff plastic to give structure to the stick that extends upwards from the center of the joystick. The Ergostick works well for left- or right-handed game players, as the trigger, or fire button, is conveniently located to the left and right of the index finger position.

The Wico Ergostick has a moderate $19.95 list price and is now available at your local toy store and computer outlet.



The *Ergostick* is a new design in joysticks that makes use of a soft, pliable plastic that is formfit to the human hand.

Wico Corporation
6400 W. Gross Point Road
Niles, IL 60648
(312) 647-7500

CIRCLE #132 ON READER SERVICE CARD.

Command the World

Global Commander is a new game from Datasoft that pits your wits against the entire world economy. The game play takes you to the near future, where you have assumed authority over the world's natural resources, food supplies, technology and economies. Although all of the usual nationalities still exist, the Global Commander is the leader that determines the destination of the world's supplies.

The game play is limited to making tactical and influential decisions according to the requests received from the participating countries. Global Commander might not be for you if you are used to an action-packed arcade game. However, if intrigue and political maneuverings motivate your game-playing spirit, Global Commander may be just right.

Datasoft Inc.
19808 Nordhoff Place
Chatsworth, CA 91311
(818) 886-5922

CIRCLE #133 ON READER SERVICE CARD.

Megafile 20

Atari's new 20-megabyte hard-disk drive, the *Megafile 20*, is now available. The Megafile 20 is compatible with all ST and Mega computers, but offers a footprint identical to the Mega CPU. Hard-disk users can place the new drive under the Mega 2 or Mega 4. The front panel shows power and reads status lights.

The Megafile 20 replaces the SH204 hard-disk drive, which has been discontinued, and carries a list price of $680.00 (and also goes by the name SH205).

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086

CIRCLE #134 ON READER SERVICE CARD.

Computer Tutors

The computer revolution that made microcomputers a reality in the 1980s has spawned a cottage industry attempting to change the way people think about using computers. Five years ago, the mainline solution to word processors was a dedicated minicomputer that did one thing: word processing. The present has created new market leaders that offer higher throughput, better cost and more features than the outdated, dedicated word-processing machines.

The result is a community of computer users that have not invested in microcomputers because of the huge number of unknowns involved in jumping off into a new system. The solution is to teach computer operators and supervisors to learn the new language and programs. The demand for computer education has spawned a cottage industry of firms that offer instruction, hand-holding and training to end-users.

Techlancers Inc. is a firm that has pushed traditional education approaches into new areas through its program of *conversion*. Their training method provides a modular curriculum in which clients with specific needs can have their training tailored accordingly. On-site tutors and troubleshooters provide expert training and hand-holding during the early days of a software conversion. Although they mostly focus on conversion to the *WordPerfect* word processor, Techlancers also supports other programs.

Techlancers Inc.
19 E. 41st St.
New York, NY 10017
(212) 370-1960

CIRCLE #135 ON READER SERVICE CARD.

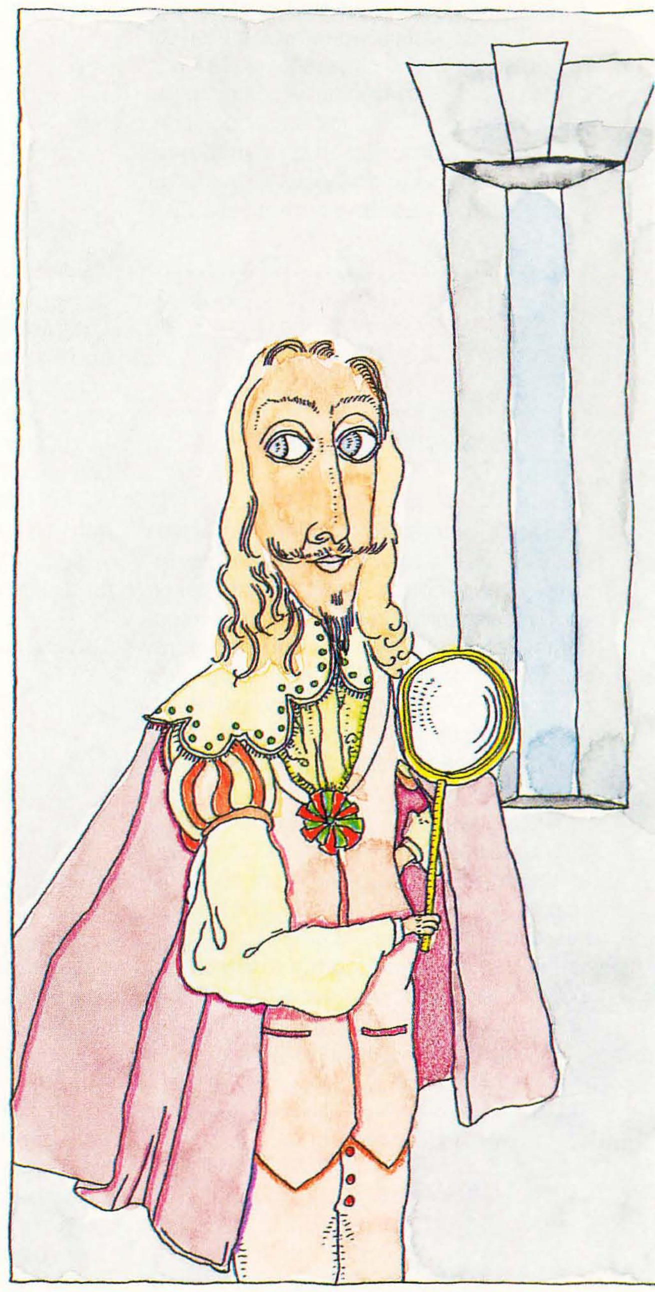
People on the Move

Frank Foster, a founder of Hybrid Arts, is replacing Larry Samuels at Atari Computers in Sunnyvale. Foster could often be found representing Hybrid at the trade shows, Atarifests and other Atari related events.

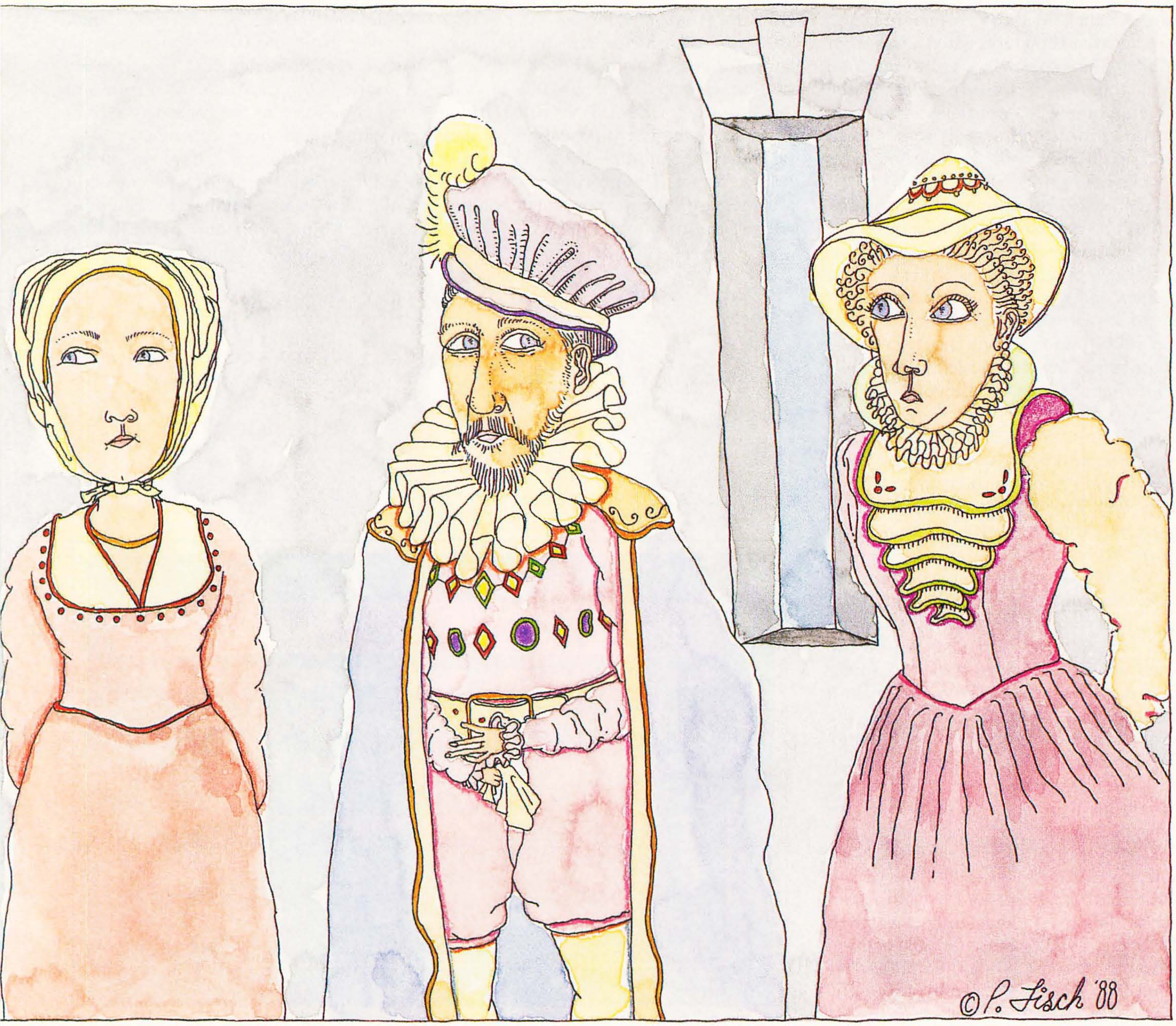
DRAM

by A. BAGETTA

Picture yourself as a highly trained, underpaid detective on assignment in the past, rubbing elbows with some of William Shakespeare's most prominent characters. (You know, those strange creatures that once inhabited your readings in Sophomore English class.) Imagine yourself lurking in the damp, stone corridors of a musty castle, trying to avoid trapdoors and fiends who try to stifle you. Envision yourself working against time, aggravated by the world around you, hoping to solve some of the most mysterious cases you have ever encountered. Ah, yes, the romantic life of a public investigator. If you have the imagination (and an Atari ST computer with color monitor) there is a world of fantasy awaiting you in the game of *Drama-cide.*



A-CIDE



First of all, you will have to dig out and dust off that little disk that came with your computer, plug it into your disk drive and boot-up the ST BASIC language. When you get the "Ok" prompt, type in (checking your work with ST-Check, found elsewhere in this issue), save and then run the program that accompanies this article; or if you already have it on disk, just type LOAD DRAMA, hit Return and wait for the "Ok" prompt; then type RUN, and hit Return.

You will see a colorful title and be introduced to the ominous success tune. After a few seconds you will be informed of the game's scenario: You are a medieval detective (if there ever was such a thing) called in to solve a crime of murder at the king's castle. (Turn on that imagination.) It seems that the king invited a group of William Shakespeare's most distinguished characters to a little soiree. During the evening, however, a murder was committed and all of the guests have vowed to clam up unless you, the investigator, guess what is going on.

When the investigation actually begins, you will be able to guess at four elements: the murderer (remember it could also be suicide), the victim, the weapon used in the murder, and the room in which the murder took place. If you press any key here, you will be presented with the layout of the King's small but comfortable castle. Each labeled room will contain the following elements: two characters (each is a potential murderer and/or victim) and a weapon.

After examining the castle's contents, press any key and you will be taken to the main hall. This will become your headquarters. From this foyer you may enter all of the rooms.

Your display at this time will show the following information. At the top of the screen there is a title bar which displays your current score for the case you are working on, the number of the case (there are ten cases numbered zero to nine) and the title of the game. The middle of the screen will inform you of your present location and give a list of the five rooms in the castle (Study, Dungeon, Library, Tower and Bedroom), each numbered for easy access.

The bottom of the screen shows your notebook. Being an efficient detective you will want to keep notes of your discoveries. As you unveil each element related to the crime, it will be automatically entered in your notebook. When the notebook is filled, you have successfully solved the crime, and you will be duly rewarded for your services as explained below.

Let's go to one of the rooms to see how the game is played. Press a number key (one through five) to go to one of the rooms (number six allows you to abort the game); you will not have to press Return. After a second or two of screen set up, you will find yourself in the respective room. The number of the room is displayed in the lower right corner of the screen. The screen also indicates which room you have entered, who the suspects/victims are, and which weapon is present in the room. Notice that every time you enter the room some of the characteristics of the room might have changed. The weapon, for example, might be in a new location or gone entirely from the room.

Because of some unexplainable force, you will not be able to know if either of the characters in the room is alive or dead at this time, whether this was the murder room, and whether the weapon present was indeed the murder weapon. You must inquire to get this information.

Underneath the weapon statement, you will see the prompt, "QUERY?". This is where you inquire. To find out if any of the room elements is involved in the crime, just type in the suspected name, room or weapon and press Return. If you are right, the ominous tune will play, and you will be told of your accomplishment.

You will not be allowed to guess any element not in the room, unless it was there before; and you will not be allowed to offer your guess in lowercase letters, so keep these two caveats in mind while playing the game.

The maximum number of guesses per room on each visit will be four, but you might not get four guesses because of various interruptions. For example, you might be called away for help or an important call might come in from the head office. Sometimes you are forcibly or accidentally taken from the room. Often you will even be called for lunch. After all, even a detective has to eat. You may, of course, always come back to the room to continue your investigation if necessary.

If you find that you have no further need to be in a room, just type one of the room numbers (you'll have to remember these on your own) and press Return.

(to page 24)

After examining the castle's contents, press any key and you will be taken to the main hall. This will become your headquarters. From this foyer you may enter all of the rooms.

ATARI®

AUTHORIZED SERVICE CENTER FOR ALL ATARI PRODUCTS

MICROTYME

A DIVISION OF MICRO PERIPHERALS, INC.

P.O. BOX 369 • KETTERING, OHIO 45409

VISA® MasterCard DISCOVER

ATARI®

HARDWARE
ST'S. . .IN STOCK!!!

Color Monitors	CALL
Mono Monitors	CALL
GTS 100 Drive	CALL
SF 314 Drive	CALL
IB 5¼ Drive	199
Navarone Scanner	CALL

PRINTER'S DEVIL HI-RES GDOS FONT & CLIP ART PACKS IN STOCK! (Great for Desktop Publishing!!)

MODEMS

SX-212 300/1200 bps	CALL
Avatex 1200E	79
Supra 2400	139

ATARI ST SCANNERS, SOUND & VIDEO DIGITIZERS In Stock!

!!!UNBELIEVEABLE!!!
HAYES COMPATIBLE 2400
Baud Modem - RS232
$125
$ $ $ SAVE $ $ $

HARD DISK DRIVES FOR ST'S

SUPRA 20 MB HARD DISK $569

SUPRA 30 MB HARD DISK $659

SUPRA 60 MB	CALL
ATARI SH 204	CALL

PRINTERS

PANASONIC	call for latest
1080i	CALL
1091i . . . . 180 cps	CALL
KX-P110 Ribbon (Blk)	9.95
KX-P Color Ribbons	10.95
STAR	Call for latest
NX-1000 . . . . NEW!	CALL
NX-1000 Color	CALL
1000 Ribbon (Blk)	6
1000 Ribbon (Color)	8
OLYMPIA	simply, the best!
NLQ modes use 18 x 24 matrix!	
NP-30 . . . . . 130 CPS	199
NP-80s . . . 240 CPS . . . changeable font cards	389
NP-136 . . . . . 15 inch	529

ACCESSORIES

ST Dust Covers	from 8
Mouse Mat	9
Power Strip w/Surge	15
Deluxe Power Strip w/Surge	24
TERMINATOR Joystick . . WOW!	19
EPYX 500 XJ Joystick	17
WICO Ergo Stick Joystick	17
Printer Stand-Heavy Duty	13
Mail Labels 3.5x15/16-500 pk	4
1000 pk	6
PAPER-1000 Shts-Microperf	14
Compuserve Starter Kit	24
On-Line Encyclopedia Kit	36
Printer Cable 6′	19
Modem Cable	17
Supra 64k Printer Buffer	69

MIDI

Midi Cables 5′	6
Software (Hybrid Arts etc.)	CALL

★ ST SOFTWARE ★

10th Frame Bowling	26
221 B Baker Street	28
3D Breakthru	26
3D Helicopter Simulator	34
AB Zoo	21
Advanced OCP Art Studio	31
Air Ball	26
Air Ball Construction Set	17
Algebra 1, 2, 3	ea 14
Aliants	19
All About America	41
Alt	21
Alternate Reality-The City	32
Alternate Reality-The Dungeon	32
America Cooks Series	ea 9
Architectural Design	25
Arctic Fox	26
Art Gallery 1, 2, 3	ea 19
Assem Pro	39
Autoduel	34
Award Maker	27
Balance of Power	34
Bally Hoo	27
Barbarian	26
Bards Tale 1 or 2	ea 34
Base Two	45
Basketball (Two on Two)	26
Battle Droidz	25
Battlezone	19
Beyond Zork	34
Biology 1, 2, 3 or 4	ea 14
Bismarck	28
Black Lamp	17
Blockbuster	27
Boulderdash Construction Kit	17
Brataccas	15
Breach	27
Bridge 5.0	24
Bubble Ghost	24
Bureaucracy	11
Business Tools	26
Cad 3D	65
Captain Blood	33
Carrier Command	33
Certificate Maker	33
Championship Baseball	27
Championship Wrestling	26
Chartpak	34
Chess (Psion)	38
Chessmaster 2000	29
Circuit Maker	54
Clip Art 1, 2, 3, 4, 5, 6	ea 13
Club Backgammon	23
Colonial Conquest	27
Color Computer Eyes	179
Colorburst 3000	25
Compubridge	20
Copyist (DR T)	165
Cosmic Relief	26
Cracked	21
Crazy Cars	25
Cross Town Crazy 8	13
Cyber Control	45
Cyber Paint	58
Cyber VCR	49
Dark Castle	27
Data Manager ST	49
Datatrieve	33
DB Man	159
Death Sword	13
Deep Space	31
Defender of the Crown	33
Degas Elite	39
Desk Cart	69
Diamond Mike	13
Digi Drum	27
Dive Bomber	26
Dr. Drums (DR T)	19
Dr. Keys (DT T)	19
Drafix	129
Dungeon Master	26
Dyna Cadd	449
Easy Draw (Regular)	66
Easy Draw W/Supercharger	99
Easy Tools	33
Empire	38
Expert Opinion	72
EZ Score Plus	99
EZ Track Plus	43
F15 Strike Eagle	26
Fast Basic	67
Fast Basic M Compiler	39
Fire and Forget	26
First Cadd	33
First Letters & Words	34
First Math	27
First Shapes	29
First Word Plus	63
Flash	23
Flash Cache	54
Flight Simulator 2	35
Scenery Disks	ea 18
Font Disks (Pub Part)	ea 20
Fonts and Borders	24
Fontz ST	23
Foundations Waste	26
Fracton Action	26
Frostbyte	17
Gateway	31
Gato	34
Gauntlet	33
Genesis (Molecular Modeler)	59
GFA Basic	39
GFA Basic Book	27

LARGEST SELECTION IN THE U.S.

GFA Companion	34
GFA Compiler	39
GFA Draft Plus	89
GFA Quick Reference Manual	12
Ghosttown	13
Global Commander	28
Goldrunner	26
Goldrunner 2	27
Goldrunner 2 Scenery Disks	ea 7
Gone Fishin'	28
Great Chefs Vol. 1, 2, & 3 Set	39
Gridiron (Football)	19
Guild of Thieves	29
Gunship	26
Hard Disk Backup	23
Hardball	26
Harrier Combat Simulator	34
High Roller	27
Hippo Concept	45
Hollywood Hijinx	19
Home Accountant	34
Human Design Disk	25
Hunt for Red October	34
IB Copy	23
Impossible Mission 2	27
Indiana Jones Temple of Doom	33
Interlink ST	26
International Soccer	26
Into The Eagles Nest	27
Inventory Manager	52
Jet	36
Jinxter	27
Joust	19
Juggler	34
K Resource	36
Karate Kid 2	27
Karateka	23
KCS Level 2	215
KCS-Keyboard Control (DR T)	165
Kid Progs	27
Kids Stuff	27
Kinderama	27
Kings Quest 1, 2 or 3	ea 32
Knickerbockers	12
Label Master Elite	29
Lattice C	109
Leaderboard Dual Pack	17
Leatherneck	27
Leisure Suit Larry	26
Leviathan	11
Liberator	14
Lock On	26
Lords of Conquest	15
Lurking Horror	21
Macro Mouse	25
Magic Sac Plus	115
Magic Sac Roms	CALL
Major Motion	26
Make It Move	47
Marble Madness	27
Mark Williams C	124
CSD Source Debug	46
Master Cad	132
Match Point	25
Mavis Beacon Teaches Typing	28
Megamax C (Laser C)	119
Mercenary	27
Metro Cross	16
Micro Kitchen Companion	26
Microleague Baseball	39
Midi Maze	26
Midi Recording Studio (DR T)	27
Missile Command	19
Mixed Up Mother Goose	21
Modula 2 (Developer's Kit)	99
Moebius	41
Mouse Trap	14
Music Construction Set	35
Music Studio	34
N Vision	29
Neo Desk	20
New Tech Coloring Book	15

★ ST SOFTWARE ★

Ninja	14
Obliterator	27
Ogre	27
Oids	24
Omnires	23
Orbiter	26
Paint Pro	33
Paintworks	14
Paperboy	26
Partner Fonts	21
Partner ST	46
Pawn, The	29
PC Ditto	65
Perfect Match	27
Personal Pascal	66
Phantassie 1, 2 or 3	ea 26
Phasar	59
Pinball Wizard	24
Pirates of the Barbary Coast	17
Planetarium	26
Plutos	21
Police Quest	33
Power Plan	52
Prime Time	27
Print Master Plus	26
Pro Copy	28
Publisher ST	79
Publishing Partner Pro	CALL
Q Ball	21
Quantum Paint Box	31
Quink	11
Read & Rhyme	27
Renegade	14
Road Runner	26
Roadwars	22
Rockford	22
Santa Paravia	19
Scan Art	33
Scruples	29
SDI	34
Shadow	22
Shadowgate	34
Shard of Spring	27
Shuffleboard	19
Silent Service	27
Sinbad	33
Sky Fox	14
Slagon	27
Soko Ban	23
Space Quest 1 or 2	ea 33
Spectrum 512	49
Speed Buggy	29
Speller Bee	29
Spiderman	7
Sprite Factory	26
Spy vs Spy 3 (Arctic Antics)	19
ST Disk Drives Inside & Out	18
ST Gem Programmers Ref Man	15
ST Internals Book	15
ST Intro to Midi Book	15
ST Machine Language Book	15
ST Peeks & Pokes Book	14
ST Pool	21
ST Talk	5
Star Fleet 1	37
Star Raiders	19
Starglider 2	26
Stellar Crusade	36
Stock Market - The Game	18
Strip Poker 2	27
Sub Battle Simulator	26
Sundog	27
Super Base Professional	199
Super Cycle	14
Super Star Ice Hockey	33
Swift Calc St	49
Tanglewood	27
Tau Ceti: Lost Star Colony	11
Temple of Apshai Trilogy	13
Terror Pods	27
Test Drive	27
Three Stooges	34
Thunder	19
Time Bandit	24
Top Gun	11
Trailblazer	33
True Basic	52
Tune Up	34
Turbo ST	36
Typhoon Thompson	23
Ultima 2, 3 or 4	ea 39
Uninvited	34
Universal Item Selector	14
Universal Military Sim.	31
Universe 2	46
Vampires Empire	20
Vegas Craps	24
Vegas Gambler	23
Video Titleing	22
Vip Professional	149
War Ship	39
Wargame Construction Set	24
Winnie The Pooh	16
Winter Challenge	11
Wiz Ball	11
Wizards Crown	26
Word Perfect	239
Word Up	49
Word Writer ST	49
World Games	26
World Karate Championship	19
WWF Microleague Wrestling	33
Xevious	19
Zork Trilogy	46

HOURS: M-F 9 a.m.-9 p.m. EST
SAT 9 a.m.-5 p.m.

ALL 50 STATES CALL TOLL FREE
1-800-255-5835

For Order Status or Tech. Info, Call (513) 294-6236

TERMS AND CONDITIONS

• NO EXTRA CHARGES FOR CREDIT CARDS! • We do not bill until we ship • Minimum order $15 • C.O.D. - $3.50 • SHIPPING: Hardware, minimum $4; Software and most accessories, minimum $3 • Next day shipment available at extra charge • We ship to Alaska, Hawaii, Puerto Rico (UPS Blue Label Only), APO and FPO • Canadian orders, actual shipping plus 5%, minimum $5 • Ohio residents add 6% sales tax • Please allow 3 weeks for personal or company checks to clear • All defective products require a return authorization number to be accepted for repair or replacement • No free trials or credit • Returns subject to 15% re-stocking charge • Due to changing market conditions, call toll free for latest price and availability of product. FOR YOUR PROTECTION, WE CHECK ALL CREDIT CARD ORDERS FOR FRAUD.

CIRCLE #102 ON READER SERVICE CARD.

THE TRANS-WARP DRIVE

by
DAVID SMALL
and
DAN MOORE

This program lets you run all of your floppy disks at up to twice their present speed. Copy it into your Auto folder; that's all you need to do. You'll notice the difference immediately the next time you go to floppy.

To make your copy of *Trans-Warp*, type in the ST BASIC program shown in Listing 1. When you've finished, check your typing with ST Check (ST-Log, September '88), then run the program by loading it into ST BASIC and typing "RUN." The Trans-Warp program will be written to the disk in drive A.

Introduction

Dan and I have long been obsessed with speeding up and improving the ST computer. It might seem like overkill; after all, the ST is already one fast machine. However, there are certain parts of the ST which can stand some improvement. (Besides, it's always fun to soup up a computer.)

There have been several programs written recently that correct a subtle bug in Atari's seek-with-verify code. Using disks that are formatted by such programs, you can get data to and from the floppy disk at twice the normal floppy speed.

Now, wouldn't it be nice if *all* your floppies ran at high speed? Most of us already have a considerable library of floppies, and copying/reformatting them to a new format takes a lot of work.

However, destiny waits for no one, not even us. One evening Dan and I were sitting in his office, grimly swigging Boodles and Strohs after crashing yet another hard disk. I was staring at his *Bill the Cat* doll. It's a cute thing, what with its bug eyes, lolling tongue, and "*Ack!*" vocabulary. (Ever seen Teddy Ruxpin? Imagine a Teddy that just says, "*Ack!*", and you'll get the idea.)

I was staring at Bill—and—and—Bill began to speak to me. A strange sight indeed, but probably no stranger than a Shirley MacLaine novel.

It said, "Dave! Ack! Dave, what you need to do is re-code RWABS to handle sector latency! Do *two* reads instead of one per track, skew them, and all floppies will then run at double speed! *Oop Ack!*"

In the throes of inspiration, I turned to Dan, trying to speak. All I could manage to choke out was, "*Ack!*" Dan looked intensely interested, stared at me in concern, and said, "If you're going to be sick, go outside."

A few evening's hacking sessions gave us what we cheerfully call the Trans-Warp drive. Only the interest in a high-tech sounding name, and Berke Breathed's copyright, prevent us from calling it the Trans-Bill drive.

The Problem

A brief review of the problem that we're correcting will help explain why Trans-Warp works.

Atari diskettes have nine sectors, labeled one through nine in a circular track. An "index pulse" marks the beginning and end of a track. From the point of view of the disk head, the sectors spin by underneath, like this:

(index pulse) 1-2-3-4-5-6-7-8-9

(index pulse) 1-2-3-4-5-6-7-8-9

and so forth. Floppy disks rotate at five revolutions per second (300 RPM). When you read a large amount of data (each track is about 6K), the process goes like this:

—Read Sectors 1-9 of current track
—Step head to next track
—Read Sectors 1-9 of that track
—Step head to next track

and so on.

Ideally, this should be done so that you "land" on the next track in time for Sector #1 to spin underneath the head. You then "catch" Sectors 1 through 9 in your next read. The floppy controller is pretty dumb, so if it doesn't catch Sector 1, it waits for Sector #1 to spin all the way around again, *then* reads 1 through 9. (There isn't any good way to tell the floppy controller to "read the next sector that spins by regardless of what it is," or there wouldn't be much of an article here.)

Alas, Atari made a small slip-up in their "Step head," or "seek," routine. They made it a seek-with-verify. In this variation of a regular seek, the head is moved to the new track, then the first sector "mark" encountered is read to verify that the head landed on the proper track.

Unfortunately, this screws up high-speed data transfer, as follows:

(1) Read Sectors 1 through 9 of current track. (The disk is now right near the end of the track.)

(2) Step to next track. (The disk has now spun to about the beginning of the next track.)

(3) Read sector mark to verify track number. (The disk now spins until we find Sector 1, which is used to verify the track number.)

(4) The seek now completes. We now try to read Sectors 1 through 9.

Alas, the disk is now coming up on Sector 2. So, we wait until Sectors 2 through 9 spin around (a complete revolution), and then find Sector 1, read Sectors 1 through 9, and continue. In short, every time we step the head, we cost ourselves a revolution—and one-fifth of a second.

Now, a one-fifth of a second delay per track may not seem like much. But there's 80 tracks on the disk; that delay adds up to 16-seconds per disk side (32 seconds on a double-sided floppy, such as 1040ST or SF 314 drive).

What can we do about this delay?

The first way around it is to lay out the sectors on the track differently, providing a few sectors to satisfy seek-with-verify without slowing us down very much. The second solution is to change the operating-system code.

Both of these have advantages and disadvantages. The format solution requires you to reformat all your disks. The operating-system solution requires a terminate-and-stay-resident routine that may not always be allowed. Hence, you can pick the one that works for you best.

How do we "steal away" floppy-disk requests so we can handle them? The solution is RWABS.

RWABS

The last place we have control of the disk is at the RWABS "vector" (Read/Write Absolute), a memory location that all disk accesses jump through. RWABS is provided as a convenient place to steal all disk accesses for devices such as hard disks and RAMdisks.

Whenever you do a disk access through RWABS, the system jumps to wherever the RWABS vector points to. For instance, if RWABS contains $12345678, then anytime you go to disk, the system jumps to address $12345678. Ordinarily, the RWABS vector points to somewhere in ROM, to the floppy disk RWABS routine (somewhere in the $FCxxxx area).

However, because this RWABS is a RAM location, we can redirect disk requests to other places. Typically we will also save the old RWABS value, so we can always go to where the original RWABS was pointing (e.g., the floppy-disk handler).

Let's assume you "steal" RWABS, by putting your routine's location in RWABS. Now, when a disk request is made, your routine gets jumped to. You get control of the system. You look at the various parameters which have been placed on the stack (device number, sector number to start at, number of sectors you want, and where to put them). If the device number is one you want to process, then you take control and never let the floppy RWABS routine handle the request. On the other hand, if it's a floppy disk request, you'd go to where the old RWABS used to point—to the floppy handler.

Let's say you have a two-floppy system (drives A and B) and a RAMdisk as drive C. (This is a very common setup.) The RAMdisk steals RWABS and directs all disk requests to the RAMdisk RWABS handler. Then the RAMdisk installation routine quits, typically with a terminate-and-stay-resident (TSR) call.

What's a TSR? It's a way for a program to return control to the operating system (GEM/TOS) leaving the program code itself in memory. In our case that's pretty important; we've just redirected all disk I/O requests to our program, so it had better hang around! (Desk accessories, incidentally, are another example of a TSR.)

Now, when the user calls the operating system with a disk request, our RWABS handler gets called. It says, "Is this a request for me (Drive C, The RAMdisk)?" If the answer is "yes," the RAMdisk handler takes care of the request and returns to the user. If it's a request for drives A or B, however, it just passes the request to the floppy RWABS and lets the floppy RWABS handle it.

Hard disks just add another level to this daisy-chaining; the hard-disk RWABS looks at the device number (drive letter), decides if it wants to handle the request, and if not, goes back to the original RWABS. Generally, the hard-disk handler is installed first after boot-up, then any further handlers (such as RAMdisks).

The Idea

Now, we don't *have* to give the operating system control if drive A or B is requested. In fact, we don't want to, because the OS handler is slow. So let's recode the system floppy RWABS handler and steal away all floppy-disk requests, forever. Our new handler will take care of floppy read/write requests in a special way which will mollify the seek-with-verify routine and thus not slow us down. Then, as long as our new RWABS routine is around, all floppy requests will run at double speed—we'll quit missing a revolution each time we read a track.

Basically, our installation routine will do what's described above in terms of stealing away RWABS and passing control to non-floppy devices. The only difference is we'll handle floppy requests ourselves.

Of course, our routine must stay resident, as all floppy disk requests will be going to it. If you want to write your own TSR utilities, this will probably be a good place to start.

We'll dig into exactly how we short out the system handler in a moment. First, some research on RWABS.

Inside of RWABS

RWABS does several things. Coming into RWABS, we have a request for a certain number of sectors, starting at a certain absolute sector number. This number of sectors can, and often does, span several tracks. Hence, RWABS must "break up" the request into multiple track requests. RWABS must also worry whether you've changed the disk—lest it accidentally write to a new disk—and whether or not the disk is double-sided.

RWABS calls a routine known as FLOPRW, which takes your disk request (which drive, starting sector, number of sectors, etc.) and breaks it up into track-sized requests. FLOPRW then goes and reads in a track at a time, using *floprd* and *flopwr*, the lowest-level sector requests.

Ordinarily, if FLOPRW wants an entire track, it just reads Sectors 1 through 9 on the current track. This is where we interfere. We break up an ordinary "read Sectors 1 through 9" request like this:

—If we're on Track 0, just go read Sectors 1 through 9.

—If we're on Track 1, read Sectors 3 through 9, then read 1 and 2.

—If we're on Track 2, read Sectors 5 through 9, then read 1 through 4.

—If we're on Track 3, read Sectors 7 through 9, then read 1 through 6.

—If we're on Track 4, just read 1 through 9.

The pattern repeats across all 80 tracks of the disk.

At first, this may seem confusing. However, let's draw a map of what sectors we're reading, on what track, and at what point in time. That'll help clear up what's

(to page 32)

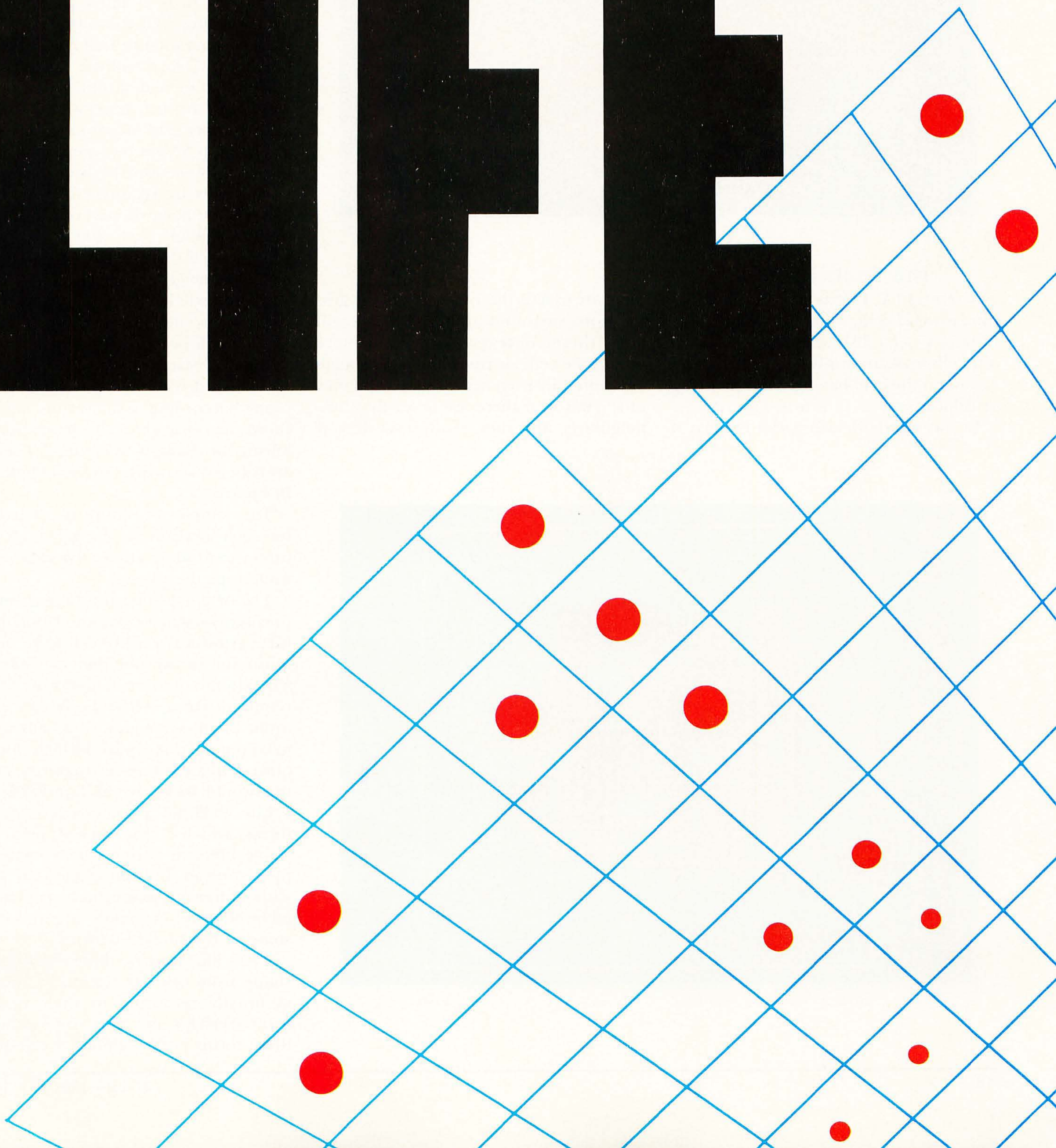
OMNI

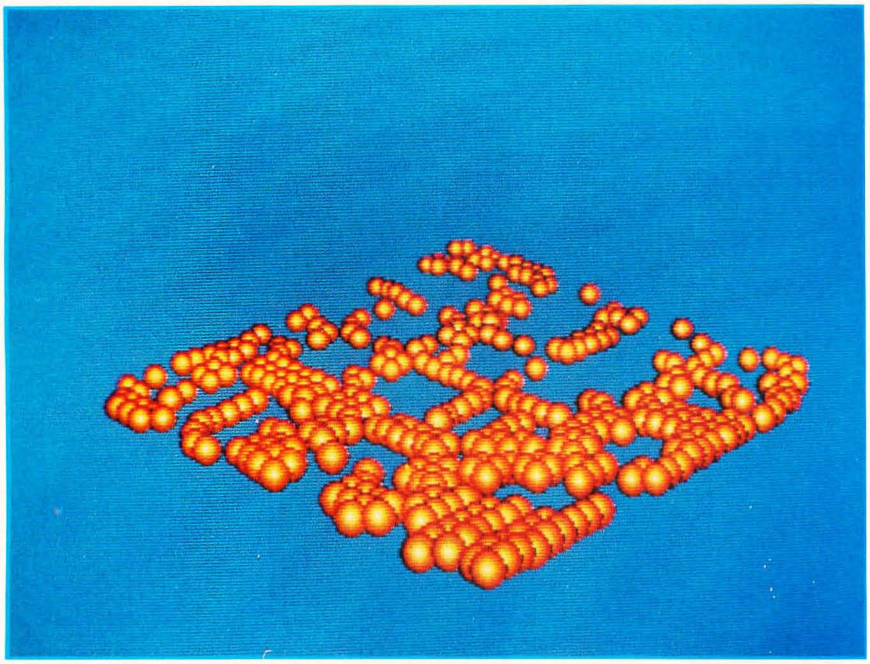
by Tom Hudson

One of the oldest computer games in existence today is a simple simulation called *Life*. Originated by John Conway in 1968, the game is often referred to as "Conway's Life." The game is a simulation of the life cycle of single-celled organisms, which is determined by simple rules, or parameters. The term "game" as used here doesn't mean it's a cellular-level shoot-em-up or adventure, but rather an exercise in logic. I prefer to think of Life as a simulation, but I'll leave the hair-splitting to you.

Due to the large size of this program, it is available only on this month's disk version or in the databases of the DELPHI *ST-Log* users' group.

LIFE

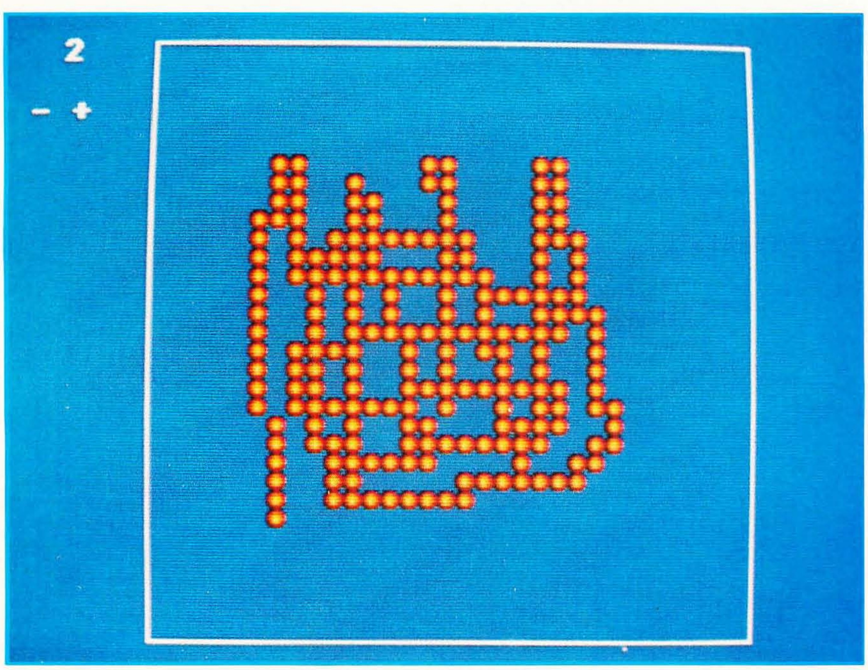




The Facts of Life

The original Life is played on a two-dimensional grid, which serves as a sort of binary petri dish where we can watch our cells grow, multiply and die. Each location on the grid has eight neighboring grid locations, as shown in Figure 1.

In the standard Life, a cell is born if there are exactly three live cells in the grid locations surrounding it. A live cell will die if there are fewer than two or more than three cells surrounding it. The analogy generally used here is that a cell dies of starvation if there are fewer than two neighbors, and dies of overcrowding if there are more than three neighbors.



With this in mind, look at Figure 2. There are five cells alive in this illustration, labeled A through E, and four unoccupied grid locations, labeled W through Z. Let's look at this setup and find out what will happen next.

Cell A only has one neighbor, cell C. It will die of starvation and will not be present in the next generation. Cell B has two neighbors, C and D, and will live to the next generation. Cell C had four neighbors, A, B, D and E. It will die of overcrowding. Cell D has two neighbors, B and C, so it will live. Cell E only has one neighbor, C, so it will die.

Now that we've determined which cells will live to the next generation, let's look at the empty grid locations W, X, Y and Z. Location W has four neighbors, A, B, C and D. Since exactly three live neighboring cells are required for a birth, no cell will be born there. Location X has three neighbors, A, C and E, and a cell will be born there. Likewise, location Y has three neighboring cells, C, D and E, and a cell will be born there. Location Z has only two neighboring cells, C and D. No cell will be born there.

After the status of all the grid locations have been determined, a new cell grid is plotted according to the results. Figure 3 shows the cell grid for the generation following the example in Figure 2. The cells are labeled according to the letters used in Figure 2.

This simple procedure is carried out for each location on the grid, and continues until there are no live cells or the user stops the process.

The original Life uses the parameters previously stated for the simulation of cellular growth: For a live cell to survive, a minimum of two neighboring cells and a maximum of three neighboring cells is required. For a cell to be born a minimum of three neighboring cells and a maximum of three neighboring cells is required. In a kind of shorthand notation, this would be known as Life 2333.

Life 2333 has a great variety of "lifeforms," which can be stable or cyclical. A stable lifeform is one which retains its form for an infinite period of time, without moving, as long as it is not affected by other cell "colonies." Figure 4 shows some of the stable lifeforms.

Cyclic life forms are those which change shape from one generation to another, eventually returning to their original form. Perhaps the most interesting of these forms is the "glider," a cell group

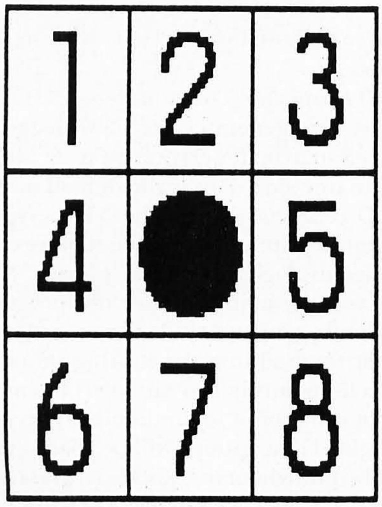


FIGURE 1

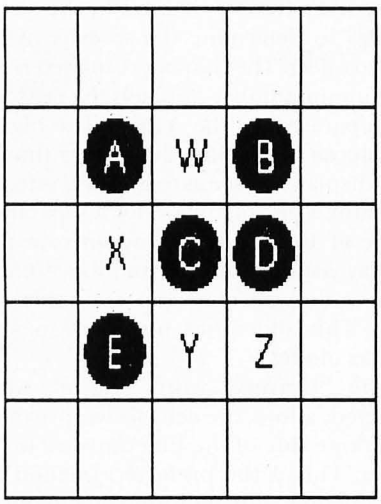


FIGURE 2

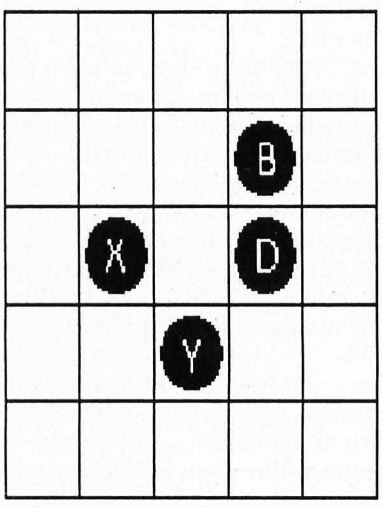


FIGURE 3

which goes through a four-generation sequence, and moves one grid diagonally in the process. As a result, it appears to "swim" across the screen like a tadpole, complete with a wiggling tail-like structure. Figure 5 shows some of the more common cyclic lifeforms. The glider is in the upper right corner.

There is nothing preventing us from running Life simulations with other parameters. Life 2333 is simply the classic standard version, and is a good starting point for new "Lifers." Some parameter sets may be more pro-life, with rapid overpopulation; others anti-life, with cells meeting their end quickly.

The Third Dimension

The original Life, as mentioned above, operates strictly in two dimensions, resulting in a simple plane of cells. Recently, in his *Scientific American* "Computer Recreations" column, A.K. Dewdney addressed the possibility of creating Life programs which operate in three dimensions, giving incredible possibilities for new lifeforms. Such a 3-D Life program requires a high-speed processor for reasonable speed in processing the cell matrix; it also requires a graphic display which can show a reasonable approximation of the 3-D group of cells. Fortunately for us, the ST computers have both.

Three-dimensional Life is an incredible extension of standard 2-D Life. It can display many of the same kinds of lifeforms as 2-D Life, but modified somewhat in order to operate in three dimensions. The primary difference between 2-D and 3-D Life is that in the 3-D version each cell has 26 neighboring cells: Nine in the plane above, nine in the plane below, and eight surrounding the cell in its plane. This forms a three cell by three cell cube with the center cell of the cube being analyzed. The layout of the cube is shown in Figure 6. I won't go into laborious detail here with an analysis of the Life process in three dimensions; it is exactly like the 2-D process except that there are 26 neighboring cells to check.

As you have probably already guessed, the parameters for 3-D Life are different from those of 2-D Life. Since there can be up to 26 neighbors rather than eight, the parameter count can range accordingly. According to Dewdney, the parameter set for 3-D Life which shows the most promise is 4555 (four-five neighbors to stay alive, and exactly five neighbors to be born). A secondary parameter set he mentions is 5766, which apparently mimics the 2-D Life 2333 more closely.

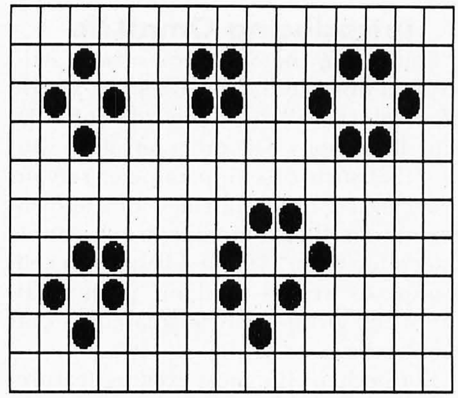


FIGURE 4

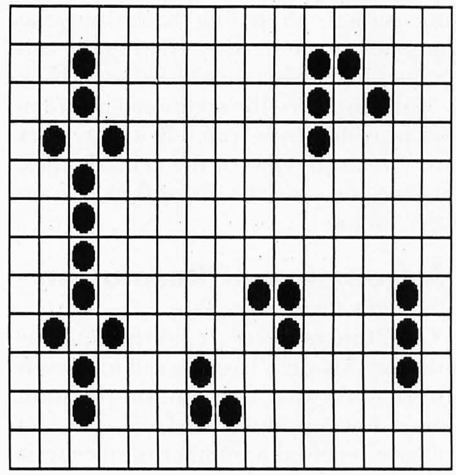


FIGURE 5

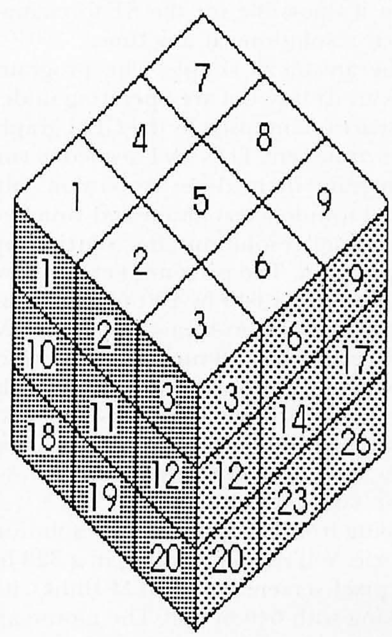


FIGURE 6

Introducing OmniLife

Spurred by the *Scientific American* article, I sat down and wrote *OmniLife*, a program which will run both 2-D and 3-D Life simulations. The program allows you to set all sorts of Life parameter sets in order to test different rules for lifeform generation. It also includes a cell editor screen for both 2-D and 3-D start-up cell groups, as well as random "primordial soup" cell groups, for the anarchists out there.

But perhaps the most exciting feature of OmniLife is the ability to show cell colonies in true stereo by using the LC Technologies StereoTek glasses. With these optional glasses, you can run 2-D or 3-D Life simulations which are actually three-dimensional, stretching back into the computer monitor. After viewing the cell colonies in true stereo, it's hard to go back to "flat" displays—the additional information provided by a true 3-D image gives a very realistic view of the cell colonies, showing the spatial relationship like no other method.

A Question of Resolution

A slight digression here.

OmniLife requires a color monitor, as it displays shaded, 16-color cell images. It also requires that you run the program from a low-resolution desktop.

I have received a number of questions from ST users as to why some programmers seem to be "lazy" and force you to start a program in certain resolution, when it's possible for the ST to change screen resolutions at any time.

The answer is simple. The programmers aren't lazy, but are operating under a restriction imposed by the GEM graphics environment. Let's say I needed to run a program in medium resolution, but started it in low resolution and switched to medium resolution after starting up the program. The programmer can now do things on a 640 by 200 pixel screen, but there will be problems. As far as GEM is concerned, it's still running in low resolution. The mouse arrow will only go halfway across the screen because GEM thinks the screen is only 320 pixels wide. Dialog boxes and drop-down menus won't be drawn correctly, either.

Going from medium to low resolution is worse. You're now working in a 320 by 200 pixel screen, while GEM thinks it's working with 640 by 200. The mouse arrow will go from the left side of the screen to the right, then wrap back around *again* to the left side and to the right before it stops! This is because the mouse handler is operating with a 640-pixel wide screen, which is twice as wide as the low-resolution 320-pixel screen.

If you have a dialog box which uses more than four colors in this case, only the first four colors will be displayed, because GEM thinks you're in medium resolution, with a four-color limit.

There are ways around all these problems, but working around them means abandoning the GEM user interface, with its common ground for all programs written using it. Programmers aren't lazy when they follow the GEM programming guidelines and force you to start a program in a given resolution. They're merely following the proper procedure which will allow their programs to run correctly when a new ST computer or set of ROMs is introduced. It's a limitation and an annoyance sometimes, but don't blame the programmers.

Back to Life

The program OMNILIFE.PRG on your ST-Log disk is the OmniLife program. When you run this program, be sure your computer is running in low-resolution. OmniLife uses the low-resolution screen for a colorful display and since it uses the mouse cursor for cell editing, it must be run in low resolution for proper mouse limits to be set.

When you start the program, the first thing you will see is the program credits display. Click on OK and the program options dialog will appear, as shown in Figure 7.

The program options dialog allows you to set the main operating mode of the program (2-D or 3-D, monoscopic or stereoscopic), as well as the number of groups of cells to randomly create if the random setup mode is used and the stereo separation amount if the stereo mode is used.

There are five operating modes. These are:

2-D Lookdown: The classic Life mode, looking down on a 32x32 plane of cells from above.

2-D Perspective: A view of the 2-D Life plane in perspective from above and to one side. This mode is sometimes referred to as 2 1/2-D.

2-D Perspective/Stereo The 2-D Perspective mode, with the added effect of stereo vision, requires the LC Technologies StereoTek glasses. If you are running OmniLife without the glasses and the cell display seems to flicker with a double-image, you have accidentally selected the stereo mode.

3-D Perspective: The non-stereo 3-D Life mode. This generates a 2 1/2-D image of the cell matrix in perspective from above and to one side. This is the default mode.

3-D Perspective/Stereo: The 3-D perspective mode with stereo vision, requires the LC Technologies StereoTek glasses.

If you are going to have the program randomly generate several groups of cells rather than editing the starting status of the cells manually, you can select the number of cell groups to randomize, from one to eight. These groups will be spaced randomly throughout the 32x32 cell grid (2-D) or the 32x32x32 cell matrix (3-D), providing a sort of "primordial soup" from which lifeforms may or may not grow.

The horizontal slider, labeled "Max Stereo Separation" is used in the stereo modes to determine the severity of the stereo effect. The number displayed is the maximum number of pixels the cells will be separated on the screen. The higher the stereo separation, the further the 3-D cell display will seem to go back into the monitor. I personally prefer a separation value of 40, but depending on your particular vision, distance from the monitor, and so on, you may prefer another setting. This slider has no effect in non-stereo modes.

The "Universe wrap" button, when selected, allows the cells to wrap-around from one side of the Life universe to the other. This is the preferred method for most people, as it allows moving lifeforms like gliders to roam the universe without stopping at the edges. If the Wraparound option is not selected, objects will stop when they hit the sides of the 2-D grid or 3-D matrix.

The "Show status" button, when selected, displays various information at the bottom of the screen in both non-stereo and stereo modes. The status line will tell you how many generations have been processed (up to 32,767) and the number of cells currently alive. In 2-D mode, with its 32x32 cell grid, the maximum number of live cells is 1,024. In 3-D mode, the maximum number of cells is 32x32x32, or 32,768.

Once you have set the parameters to the desired settings, click on OK or press Return to continue. You can also abort the program at this point by clicking on the "Quit" button, but with a program this fun, who wants to quit?

The next dialog you will see (assuming you didn't quit on the last dialog) is the

Master Life Controls dialog, shown in Figure 8. This dialog has four sliders in it, which set the Life parameters we discussed earlier. When you initially start the program, the default parameters will be 2333 for 2-D and 4555 for 3-D Life. You can change each of the controls to any number of cells from one to the maximum number of neighbors (eight in 2-D, 26 in 3-D).

The first two sliders control the minimum and maximum number of neighbors which allow a live cell to survive. The first slider, "Survive low," must be less than or equal to the second slider, "Survive high." If a live cell has fewer than "Survive low" neighbors or more than "Survive high" neighbors, it will die.

The second two sliders control the minimum and maximum number of neighbors which allow a dead cell to be born. Once again, the first slider of these two, "Birth low," must be less than or equal to the second, "Birth high." If a dead cell has at least "Birth low" neighbors and not more than "Birth high" neighbors, it will come to life.

When you have set the life parameter sliders, click on OK to continue. The program will check the slider controls to make sure you haven't entered improper values in your sliders, and will warn you if you have. Once again, if you want to quit at this point, you may do so via the "Quit" button. But why quit when there's so much fun ahead? Let's press on.

Pressing On

Since you're still reading, I'll assume you pressed OK. At this point, a small dialog will appear and ask you if you want to use a random initial cell setup or a user-defined setup. Clicking on "Random" will start the Life simulation using the number of cell groups specified in the Program Options dialog (Figure 7). The program will generate some cells in a random pattern and start the simulation.

If you want to try creating your own lifeforms or patterns for the initial setup, you can do so with the OmniLife cell editor, which is used if you click on the "User" button. The editor screen is very simple. When the editor is in use, the screen will clear and the mouse arrow cursor will appear. The cell box is in the center of the screen. Moving the mouse into the box and pressing the left button will create a cell in that location. You can "draw" a cell pattern quickly by holding the left mouse button down and moving the mouse. Clicking on a cell that is al-

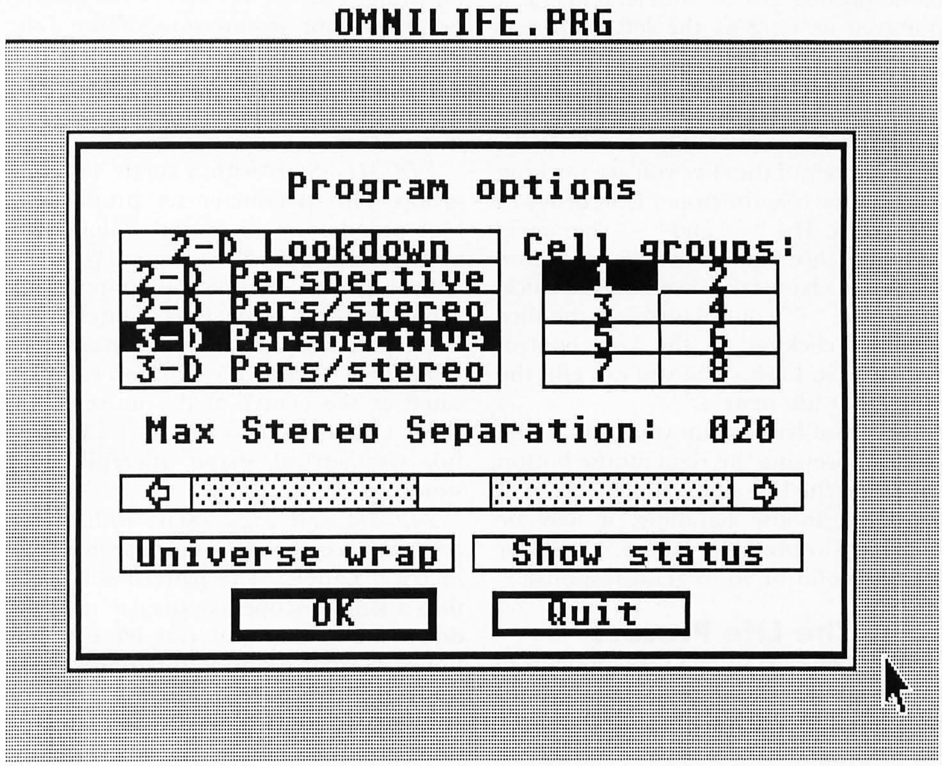


FIGURE 7

FIGURE 8

ready there will put the editor into erase mode, erasing that cell and all others you point to as long as the left button is pressed.

In 3-D mode, the editor allows you to edit a "slice" of the 3-D Life matrix. There are 32 slices, numbered from zero to 31. The number of the slice you are working with is shown in the upper left corner of the screen. The " + " and " – " characters below the slice number are controls allowing you to change the slice number. Clicking on the " + " control increases the slice number; clicking on the " – " control decreases it. In this way, you can edit the entire 3-D life matrix.

Once you have set up your initial Life pattern, pressing the right mouse button will begin the Life simulation. Because of the GEM mouse handling, it may be necessary to press the mouse button for a half-second or so to get a response.

The Life Process

Once you start up the Life simulation, the computer will process the cells until none are left alive or you stop the simulation. If all the cells die, a "NO LIFE" message will appear and the simulation will stop. The following keypresses are used by the program during the simulation mode:

Undo: Quits the program, taking you back to the GEM desktop. True Life fans will never use this key.

Help: Allows you to restart another simulation using the same parameters as the last one. You will be prompted for either a random or user-defined starting configuration.

F10: Returns you to the Program Options dialog to change the main operating mode of the program.

Space bar: Pauses the display. Press again to resume.

Lifers Anonymous

Life is a strangely addicting simulation. The first assembly language program I ever wrote for a microcomputer was a Life program, written on my old Compucolor II system in 8080 assembly language. It was crude, but I learned a lot and got kind of hooked on Life. When I saw the *Scientific American* article on 3-D Life, I couldn't resist. The added dimension of stereo vision was something I couldn't pass up, and if you try it, you'll see how important stereo vision can be on computers for giving realistic depth perception.

Following are some interesting parameter sets I have found by playing with OmniLife into the wee hours of the morning. You can use them as a starting point in your exploration of the Life phenomenon. I have a feeling you'll spend a lot of time watching lifelike patterns of cells crawl over the face of your monitor.

3-D/3311—Start with a single random group. This parameter set produces a high-population generation, followed by a large-scale reduction caused by overcrowding, then another high-population generation, and so on. Lots of interesting cubic patterns, very nice in stereo.

3-D/2344—Start with a 2x2-cell cube, located at the center of the universe, in slices 15 and 16. This will start a beautiful, symmetrical shape. Incredible in stereo.

2-D/3311 and *3322*—Start with a few cells at the center of the universe in a symmetrical pattern. The pattern will grow into a kaleidoscope-like display in look-down mode that you can let run for hours. Very nice, and it apparently does not repeat.

2-D/1233 and *1244*—Use several random groups. These parameters usually cause a quick general thinning of the population, leaving many groups of one, two or three cells in stable configurations. It also leaves some interesting cyclic lifeforms.

The Details

The OmniLife source code has been provided on the ST-Log disk for programmers to examine and modify. It is in two parts; a C source file and an assembly language source file, and an additional file for the stereo glasses driver software.

Whenever I write a program which requires high-performance operation, I like to set it up so that the parts which require a great deal of speed are written in assembly language. Compilers are notorious for producing inefficient code, and hand-coded assembly language results in a program that runs as quickly as possible.

The problem with assembly language is that it can become cumbersome in some cases, usually in those routines which won't benefit much from assembly language speed, such as dialog handling, GEM calls, and so on. For this reason, I use C for the user-interface details of a program. It is easy and quick to code, and since it is dealing with user input in most cases, it operates fast enough for most applications.

As you can guess, the OmniLife C source module takes care of all the user interface details, such as dialogs and the cell editor. If you look at the C source file, you will see that it is pretty straightforward.

The first third of the C source file deals with the processing of the dialog boxes and their associated slider controls. The sliders are simple constructs, built out of four objects with the GEM Resource Construction Set (RCS). All the objects in a slider are TOUCHEXIT objects, and when the user clicks the mouse on one of them, they are processed by the *do_hslider()* function. An in-depth discussion of the operation of the slider routines can be found in the article "Handy-Dandy Slider Subroutines," in ST-Log 10.

The next portion of the program deals with the actual setup and operation of the Life simulation. Most of the real work is performed by the assembly language module, which contains the *life2d()* and *life3d()* functions. As a result, this code is very simple to understand.

You will notice the use of the *stereo_switch()* function at several points in the C source. This is the function which turns the stereo glasses on or off, and is found in the STEREO.S assembly source file. A value of one in the first parameter of this function turns the stereo glasses on; a value of zero turns them off. The last two parameters are the addresses of the left-eye and right-eye view bitmaps, respectively. This function is only used in stereo mode.

The remainder of the C source file is mostly miscellaneous support functions for the program, such as dialog utility functions. One important function is the *editor()* function, which performs the cell editing operations.

The *editor()* function is pretty much self-explanatory. It watches the mouse, and if the left mouse button is pressed inside the cell-editing rectangle, the routine either draws or erases cells. The cell array used to store the cell-status values is the three-dimensional character array, *cell[][][]*. A one in any array location indicates a live cell; a zero indicates no cell. Cells are drawn by the *quikblit()* function, a custom bit-block drawing function which operates much faster than the ST's built-in routines. Since several thousand cells are plotted on some 3-D Life simulations, speed was important here. The Atari blitter chip may be faster than this software blit. Cells are erased via the *vr_recfl()* function.

Assembly Source

The assembly source code is rather lengthy, but contains only a few major routines. The main Life simulation functions for two-dimensional and three-dimensional Life, *life2d* and *life3d* are respectively. These are coded for speed rather than compactness, a necessity when processing the 32,768 individual cells in the 3-D version. Tests are done to see whether or not the cell being tested lies on the edge of the matrix. If it is on one or more edges and the wrap-around mode is enabled, the cell is handled by a special section of code which handles the matrix wrap-around.

As you can see in the source code, the cells are addressed by using the A5 address register as a pointer into the cell array, with the A4 register containing a pointer into a second cell array, which contains the cell status for the next generation. At the end of the processing of the current generation, the next generation cell array is copied back into the current generation cell array. Note that both cell arrays are byte arrays, but are word-aligned so that the copying of one array to the other can be done with long moves, which are faster than four single-byte moves.

Each cell in the matrix is checked to see how many neighbors it has. This number is compared to the master life parameters, and the new state of the cell is set in the *cell2* array. At this time, if the cell is alive, the cell is plotted on the screen by the *do_cell* subroutine (2-D/3-D perspective) or the *do_cell2* subroutine (2-D lookdown only).

The *do_cell* routine takes care of all the coordinate handling, perspective and stereo effects, if the stereo switch is enabled. To perform the rotation of the cells so that we look down on them from one corner, the routine rotates the coordinates 45° through a fixed-point math scheme. The scheme is a simple one I developed while writing a *Battlezone* program for the Atari 8-bit computers several years ago. Basically, it uses a long value to store a number with a pseudo-decimal format. The high-order word contains the whole number portion, and the low-order word contains the fraction. A value of 2.5, then, would be stored as:

(2*65536) + (.5*65536) = 163840

This can easily simulate a floating-point numbering system to a fair degree of accuracy, storing fractions down to 1/65536, or .000015.

The cell coordinates are rotated by 45° in this manner, and the resulting Z value is used to calculate a size for the sphere that is to represent the cell on the screen. There are ten sphere sizes, from zero to nine.

Using the Z value again, a simple perspective transformation is applied to the cells.

This transformation is:
ScreenX = (X/(Z + 600))/600
ScreenY = (Y/(Z + 600))/600

To get stereo effects, the stereo separation is added to the X value for the left eye and subtracted from the X value for the right eye. On non-stereo simulations, the stereo separation is set to zero in order to have no effect.

The cell location is calculated, then the *quikblit* subroutine is called to plot the cell.

The Quikblit Routine

During development of OmniLife, the plotting of the cell spheres seemed to take quite a bit of time, especially on stereo simulations, which obviously need twice the processing time in order to draw both the left- and right-eye views. I thought about it a little, and decided to code up a simple bit-block transfer (bitblt) routine for the ST's low-resolution mode. The fact that none of the spheres that represent the cells are greater than 16 pixels wide was a major part of my decision, since it would simplify the bitblt operation to operate on word-wide blocks of data. The cells are all designed to be 13 pixels high, even though the smaller cell images do not use the entire height.

The *quikblit* routine requires five parameters:

Bitmap address: The base address of the screen where the block is to be drawn.

Mask address: The address of a single-plane pixel mask for the image to be plotted. This mask is ANDed with all four destination bitplanes in order to "cut a hole" in the image where our cells can be plotted. Without this hole, the cells would show through each other.

Image address: The address of the four-plane image data.

X coordinate: The X coordinate on the screen of the center of the cell. The *quikblit* routine adjusts this value to the left. Seven pixels.

Y coordinate: The Y coordinate of the center of the cell. The *quikblit* routine adjusts this value up toward the top of the screen by five pixels.

The *quikblit* routine performs two operations. First, it uses the mask data to make a "hole" in the image on the screen, then it uses the image data to fill that hole. The mask data is ANDed with the screen to make the hole, and the image data is ORed with the screen to place the image.

Looking at the assembly source, you will notice several blocks of data for the cell sphere images. There are two sets of cell image data, one for the left-eye and one for the right-eye view. The spheres for the right-eye view have the yellowish highlight shifted to the left somewhat in order to simulate the difference in the highlight on a spherical object as seen by the two eyes. The images were created with DEGAS Elite and converted to DC.W data lines.

Both the mask and image data are shifted into the proper alignment with the screen by using the 32-bit data registers to hold the 16-bit mask or image data. The two halves of the shifted value are processed, first the low-order 16 bits, then after a SWAP operation (where the high-order 16 bits are moved to the low-order 16 bits) the high-order bits are transferred to the screen.

The *quikblit* routine is made faster by the fact that it does not do any horizontal bounds-checking. The data being transferred cannot overlap the sides of the screen, or improper images will result. The same goes for the top of the screen. The bottom of the screen is properly handled, however, since some of the cells plotted by OmniLife may run off the bottom. Cells are clipped to the proper limits there.

Life Goes On

I hope you find OmniLife to be as enjoyable as I do. It is an interesting exercise, and I often find myself mesmerized by the patterns of cells on the screen. As soon as one simulation ends, you may find yourself starting up another with a slightly different cell pattern. There are an almost infinite combination of cell patterns and parameters, so get to it!

On another note, I hope some of the techniques and routines in the program source are helpful to programmers out there. The *quikblit* routine may be one of the more useful ones, but those people interested in 3-D and stereo should find the 3-D and stereo transforms of the assembly code particularly illuminating.

Tom Hudson is a freelance programmer who has to his credit such classic ST software packages as DEGAS *and* DEGAS Elite, *as well as the* CAD-3D *family of graphic tools.*

(from page 12)

This will immediately take you to that room. If you enter H, you will go to the hall. And so it goes until you have correctly guessed all of the crime elements—or until disaster strikes!

Since this is an adventure game of sorts, there has to be some kind of conflict involving you. So be aware that there is imminent danger built into the game. At times, somewhere in the recesses of the castle, someone (maybe the murderer) sets a ghastly "Bear Grease bomb." There is nothing more devastating than this weapon. When it explodes, it not only kills you, but it destroys the entire castle and all its contents. Needless to say, you will have lost the case.

The bomb won't come as any surprise, though. When you find a ticking message, it means that the bomb has been set (you will be reminded of this when in the hall), and you must work feverishly to solve the crime. From this point on, you are in the hands of fate. With some luck, however, you might enter the room which contains the bomb. At this point, you will automatically disarm it and again breathe easily—until the next one is set.

Drama-cide is actually one big game of chance. It is a dramatic game of roulette, geared to blow you away if Lady Luck is not on your side. Each element of the crime is chosen at random. In fact, there are 13 random statements controlling the establishment of each case. In spite of this, you are in control of your own destiny; it is your decisions that will determine if you turn out to be a hero or a dead detective.

The good detective will notice some clues that will help speed up the investigation. When you enter a room, read the information on the screen carefully and you will discover certain patterns and expressions that will lead you to some of the involved elements. They are not always reliable but under certain conditions they will reveal important clues which can show the way to success. If a character is identified as sleeping, for example, it might mean he or she is dead. Could this be the victim or is the character just a heavy sleeper? With the bomb ticking away and your life hanging by a thread, a clue of this sort could be very important.

Scoring is important as a reward for the weary detective. Generally, points are given or subtracted as follows: +10 points for each element discovered (the murderer/victim, weapon, room); +20 points for disarming the bomb; −2 points for a wrong guess; and −2 points for interruptions.

The good detective will notice some clues that will help speed up the investigation. When you enter a room, read the information on the screen carefully, and you will discover certain patterns and expressions that will lead you to some of the involved elements.

Bonus Points: Sometimes, on examination of the castle plan, fate will deal you a hand and tell you which room the murder was committed in. You will be given ten points then, and when you formally guess at the room, you will be granted another ten points. Wow! Talk about generous.

Should you solve the case, you will be taken to the Modus Operandi screen, which reveals the who, when, why, how and where of the murder in context. There are five of these plots randomly placed here. Each is imbued with differences in the elements of the crime. Four of these motives are designed for the murderer and victims as separate characters. The fifth plot will appear if the crime is one of suicide. This motive screen will also display a verbal reward or reprimand, current score, and the option of continuing with another case or leaving the game.

As each case is solved you will be presented with a case-scoring screen. Here you will be able to compare your score with the previous cases (played by you or someone else), and you will see the total score for all games played up to that point—detectives having bad luck might end up with a minus score.

Drama-cide is only set up for ten cases at a time. If you manage to investigate all ten in one sitting, you will be given the total score and the chance to start all over again.

It was interesting working on this program because I learned a lot about ST BASIC, especially its hidden powers to access the VDI and AES. It also gave me a chance to play with some simple artificial intelligence ideas. Because of the random changes in the text, the machine seems to be the detective's antagonist. Also notice how the program takes care of matching pronouns with their antecedents and gender.

For the programmer I have added a little secret code, a little trick that can add a whole new dimension to the game if discovered—a sort of mystery in the mystery. So all you Sherlocks put your caps on, and see if you can solve the mysteries of Drama-cide. ■

Albert Baggetta is a teacher of English and a professional guitarist who has an avid interest in programming the Atari ST. He resides in Agawam, Massachusetts with his wife, Beverly, and his two children, Angela and Michael. He can be found wandering the ST-Log and ANALOG Atari SIGs on DELPHI.

DRAMA-CIDE
LISTING 1
ST-BASIC

```
10 REM ****************************
20 REM *        DRAMA-CIDE        *
30 REM *        A. BAGGETTA       *
40 REM *        ST BASIC          *
50 REM *        (c)1988           *
60 REM ****************************
70 FULLW 2:CLEARW 2:CLEAR:ROUND=0
80 IF PEEK(SYSTAB)=4 THEN 160
90 A#=GB:GINTIN=PEEK(A#+8):GINTOUT=PEE
K(A#+12)
100 ADR#=PEEK(A#+16):DEFBUT=1:POKE GIN
TIN,DEFBUT
110 TEXT$="[3][|SORRY! LOW REZ ONLY|]"
120 TEXT$=TEXT$+"[BYE]"+CHR$(0)+CHR$(0
)
130 POKE ADR#,VARPTR(TEXT$)
140 GEMSYS(52)
150 END
160 GOSUB 2490:POKE SYSTAB+24,1:RESTOR
E 170
170 DATA 0,0,0,0,1,0,0,0,2,1200,00,00,
3,0,0,0
180 DATA 4,0,0,1200,5,1900,1900,1900,6
,0,0,0,7,1100,800,1100,8,700,700,700
190 DATA 9,0,0,0,10,1000,400,100,11,0,
900,0
200 DATA 12,0,0,1000,13,0,800,900
210 DATA 14,1100,800,200,15,0,0,0
220 FOR I=0 TO 15
230 READ CI,RED,BLUE,GREEN
240 GOSUB 2600:NEXT I:COLOR 2,14,14,4,
2
250 SH=1:X=50:Y=5:GOSUB 3220:FILL 28,1
0
260 COLOR 2,8,8,4,2:X=250:Y=100:GOSUB
3220:FILL 228,110
270 HT=9:GOSUB  2550:GOTOXY 9,7:? "D R
 A M A - C I D E"
280 TYPE=0:GOSUB 2520:HT=6:GOSUB 2550:
COLOR 4
290 GOTOXY 11,10:? "By A. Baggetta"
300 COLOR 2,14,14,2,2:FOR CLM=120 TO 2
75 STEP 50:CIRCLE CLM,25,25:NEXT CLM
310 COLOR 2,2,8:FOR CLM=30 TO 185 STEP
 50:CIRCLE CLM,135,25:NEXT CLM
320 COLOR 8,2,14:GOSUB 2940:FOR DEL=1
TO 1000:NEXT DEL
330 TYPE=5:GOSUB 2520:HT=5:GOSUB 2550:
CLEARW 2:GOTOXY 10,1
340 COLOR 11:? "THE SCENARIO":COLOR 8:
GOTOXY 1,3
350 ? " SOME OF SHAKESPEARE'S MOST PRO
MINENT"
360 ? "CHARACTERS WERE INVITED TO THE
KING'S"
370 ? "CASTLE FOR A WEEKEND OF CELEBRA
TION."
380 ? "ALAS...THIS WAS NOT TO BE FOR D
EATH"
390 ? "WAS AFOOT.  YOUR JOB IS TO DISC
OVER"
400 ? "THE MURDERER (OR SUICIDE), VICT
IM,"
410 ? "WEAPON, AND MURDER ROOM.  BE CA
REFUL"
420 ? "THOUGH, BECAUSE SOMEONE HAS SET
 A"
430 ? "DEADLY TRAP...";
440 COLOR 4:? "FOR YOU!":?:?:COLOR 2
450 ? " Press any key to see who is in
 "
460 ? "each room and to begin your inv
est-":? "gation.";
470 NN=INP(2):UNLK=NN:COLOR 4:? "  **
GOOD LUCK **":GOSUB 3010
480 DIM MV$(10),RM$(5),WP$(5),GAM$(5,2
0)
490 DIM C(10),N(10),N2(5):RANDOMIZE 0:
GOTO 520
500 FOR SPIN=1 TO 3:V=INT(RND(1)*G)+1:
NEXT SPIN:RETURN
510 REM CLEAR VARIABLE AND ARRAYS
520 FOR I=0 TO 10:RD(I)=0:NEXT I:REM C
LEAR SCORES
530 TYPE=0:GOSUB 2520:HT=6:GOSUB 2550:
COLOR 2,2,14,4,2:TALLY=0:CLOZ=0
540 POKE SYSTAB+24,1:QWIT=0:SC=0:MMM$=
"":VVV$="":WWW$="":RRR$="":SCAPE$=""
550 J=0:J1=0:J2=0:J3=0:J4=0:BMSG=0:BOM
B=0:START=0:TAG=0:SP$=SPACE$(40)
560 FOR Z=1 TO 5:FOR ZZ=1 TO 4:GAM$(Z,
ZZ)="":NEXT ZZ:NEXT Z
570 FOR B=1 TO 10:C(B)=0:N(B)=0:NEXT B
580 FOR B=1 TO 5:N2(B)=0:NEXT B
590 RESTORE 670:FOR I=1 TO 10:READ MV$
(I):NEXT I
600 RESTORE 690:FOR I=1 TO 5:READ RM$(
I):NEXT I
610 RESTORE 700:FOR I=1 TO 5:READ WP$(
I):NEXT I
620 REM SELECT MURDERER/VICTIM/ROOM/WE
APON
630 RESTORE 670:G=10:GOSUB 500:FOR I=1
 TO V:READ V$:NEXT I:MUR$=V$
640 RESTORE 670:GOSUB 500:FOR I=1 TO V
:READ V$:NEXT I:VIC$=V$
650 RESTORE 690:G=5:GOSUB 500:FOR I=1
TO V:READ V$:NEXT I:LOC$=V$
660 RESTORE 700:GOSUB 500:FOR I=1 TO V
:READ V$:NEXT I:WEP$=V$
670 DATA HAMLET,MACBETH,OPHELIA,OTHELL
O,CASSIUS
680 DATA ROMEO,JULIET,HENRY V,FALSTAFF
,IAGO
690 DATA STUDY,DUNGEON,LIBRARY,TOWER,B
EDROOM
700 DATA DAGGER,SPEAR,MACE,ROPE,AXE
710 RESTORE 690:Q=1:Q2=1
720 FOR I=1 TO 5
730 J=J+1:GAM$(I,J)=RM$(I)
740 FOR T=1 TO 2
750 J=J+1
760 B=INT(RND(1)*10)+1
770 FOR K=1 TO Q
780 IF B=N(K) THEN 760
790 NEXT K
800 N(Q)=B:Q=Q+1
810 IF C(B)=0 THEN GAM$(I,J)=MV$(B):C(
B)=1
820 IF T=1 THEN GOSUB 850
830 NEXT T:J=0:NEXT I
840 GOTO 920
850 B2=INT(RND(1)*5)+1
860 FOR K2=1 TO Q2
870 IF B2=N2(K2) THEN 850
880 NEXT K2
890 N2(Q2)=B2:Q2=Q2+1
900 GAM$(I,4)=WP$(B2):RETURN
910 REM START MAIN PROGRAM GAME
920 GOSUB 4240:POKE SYSTAB+24,1
930 IF TAG=0 THEN BMSG=INT(RND(1)*10)+
1
940 IF BMSG=9 AND TAG=0 THEN GOSUB 473
0:TAG=1
950 CLEARW 2:COLOR 8:TYPE=1:GOSUB 2520
:LIMIT=0:HT=9:GOSUB 2550
960 GOTOXY 1,2:? "SCORE:  ";SC
970 GOTOXY 23,2:? "DRAMA-CIDE":HT=6:GO
SUB 2550
980 COLOR 11:GOTOXY 13,2:? "CASE ";ROU
ND
990 LINEF 5,10,299,10:LINEF 5,10,5,30:
LINEF 299,10,299,30
1000 LINEF 5,30,299,30:FILL 9,12
1010 TYPE=1:GOSUB 2520:GOTOXY 10,0:?:?
:?:?
1020 COLOR 10:? " You are in the HALL
-- CHOOSE ROOM #":COLOR 8:?
1030 GOTOXY 3,6:? "(1)";
1040 COLOR 11:? "STUDY";:COLOR 8:GOTOX
Y 11,6:? "(2)";:COLOR 11:? "DUNGEON";
1050 COLOR 8:GOTOXY 21,6:? "(3)";:COLO
R 11:? "LIBRARY":COLOR 8:?
1060 GOTOXY 3,8:? "(4)";
1070 COLOR 11:? "TOWER";:COLOR 8:GOTOX
Y 11,8:? "(5)";:COLOR 11:? "BEDROOM";
1080 COLOR 8:GOTOXY 21,8:? "(6)";:COLO
R 11:? "RETIRE":COLOR 8
1090 LINEF 5,100,299,100:LINEF 5,100,5
,160
1100 LINEF 299,100,299,160:LINEF 5,160
,299,160
1110 TYPE=4:GOSUB 2520:COLOR 2:GOTOXY
12,11:? "~ NOTEBOOK ~":TYPE=1:GOSUB 25
20
1120 COLOR 8:TYPE=0:GOSUB 2520:HT=4:GO
```

DRAMA-CIDE

```
SUB 2550
1130 GOTOXY 2,13:? "Murderer: ";:COLOR
 4:? MMM$:COLOR 8
1140 GOTOXY 2,15:? "Victim  : ";:COLOR
 4:? VVV$:COLOR 8
1150 GOTOXY 18,13:? "Weapon: ";:COLOR
4:? WWW$:COLOR 8
1160 GOTOXY 18,15:? "Room  : ";:COLOR
4:? RRR$:COLOR 8
1170 IF BMSG=9 THEN GOTOXY 14,10:? "BO
MB SET"
1180 TYPE=1:GOSUB 2520:HT=6:GOSUB 2550
:POKE SYSTAB+24,0
1190 IF QWIT=4 THEN GOTO 2260
1200 NN=INP(2):GOSUB 3010
1210 IF NN=49 THEN COLOR 5:GOTOXY 3,6:
? "(1)"
1220 IF NN=50 THEN COLOR 5:GOTOXY 11,6
:? "(2)"
1230 IF NN=51 THEN COLOR 5:GOTOXY 21,6
:? "(3)"
1240 IF NN=52 THEN COLOR 5:GOTOXY 3,8:
? "(4)"
1250 IF NN=53 THEN COLOR 5:GOTOXY 11,8
:? "(5)"
1260 IF NN=54 THEN COLOR 5:GOTOXY 21,8
:? "(6)"
1270 IF NN<49 OR NN>54 THEN 1200
1280 IF NN=54 THEN CLOZ=1:GOTO 5010
1290 NN=NN-48
1300 WLOC=INT(RND(1)*5)+1:RMCON=INT(RN
D(1)*5)+1
1310 POKE SYSTAB+24,1:SUS=INT(RND(1)*2
)+2:TIM=INT(RND(1)*4)+1
1320 REM CHECK FOR ACTION OF VICTIM/CL
UE FOR MURDERER
1330 ACT=INT(RND(1)*3)+1:REM CLUE FOR
PLAYER
1340 MCLUE=INT(RND(1)*2)+1
1350 IF (GAM$(NN,2)=VIC$ OR GAM$(NN,3)
=VIC$) THEN ACT=3
1360 IF (GAM$(NN,2)=MUR$ OR GAM$(NN,3)
=MUR$) THEN MCLUE=2
1370 CLEARW 2:COLOR 14
1380 HT=9:GOSUB 2550:GOTOXY 1,15:? "Dr
ama-cide                    Room";NN
1390 HT=6:GOSUB 2550:COLOR 12
1400 GOTOXY 1,1:? "~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~"
1410 GOTOXY 1,16:? "~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~"
1420 COLOR 8:GOTOXY 1,3
1430 IF MCLUE=2 THEN ? "Blood leads in
 the ";:GOTO 1450
1440 ? "You enter the ";
1450 ON RMCON GOTO 1460,1470,1480,1490
,1500
1460 ? "well lit ";:GOTO 1510
1470 ? "dimly lit ";:GOTO 1510
1480 ? "quiet ";:GOTO 1510
1490 ? "stuffy ";:GOTO 1510
1500 ? "huge ";
1510 COLOR 2:? GAM$(NN,1);:COLOR 8:? "
."
1520 GOTOXY 1,5:? "In this room you im
mediately notice"
1530 COLOR 2:GOTOXY 1,6:? GAM$(NN,2);
1540 COLOR 8:? " and ";:COLOR 2:? GAM$
(NN,3);:COLOR 8:? ",";
1550 ON ACT GOTO 1560,1570,1580
1560 ? " reading.":GOTO 1590
1570 ? " arguing.":GOTO 1590
1580 ? " sleeping."
1590 GOTOXY 1,8:ON WLOC GOTO 1600,1620
,1640,1660,1680
1600 COLOR 2:? GAM$(NN,SUS);:COLOR 8:?
 " is holding a ";
1610 COLOR 2:? GAM$(NN,4);:COLOR 8:? "
 in hand.":GOTO 1690
1620 ? "The ";:COLOR 2:?GAM$(NN,4);:CO
LOR 8:? " hangs from the ceiling."
1630 GOTO 1690
1640 ? "The ";:COLOR 2:?GAM$(NN,4);:CO
LOR 8:? " leans against the wall."
1650 GOTO 1690
1660 ? "The ";:COLOR 2:?GAM$(NN,4);:CO
LOR 8:? " is missing from here."
1670 GOTO 1690
1680 ? "The ";:COLOR 2:?GAM$(NN,4);:CO
LOR 8:? " hangs over the doorway."
1690 IF QWIT=4 THEN CLEARW 2:POKE SYST
AB+24,0:GOTO 2260
1700 IF LIMIT=TIM AND QWIT<>4 THEN GOS
UB 2890:GOSUB 2700:GOSUB 2580:GOTO 930
1710 POKE SYSTAB+24,0: COLOR 10:GOSUB
2680
1720 G$="":INPUT "QUERY: ";G$:COLOR 8:
IF G$="" THEN 1710
1730 IF VAL(G$)>0 AND VAL(G$)<6 THEN L
IMIT=0:NN=VAL(G$):GOTO 1300
1740 IF G$="H" THEN GOSUB 3010:GOTO 93
0
1750 IF LEFT$(G$,1)<"W" THEN 1780
1760 GOSUB 2680:? "** HIT CAPS LOCK **
":GOSUB 2990:GOTO 1690
1770 REM IF THE FOLLOWING DISARM BOMB
1780 FIND=INT(RND(1)*4)+1
1790 IF TAG=1 AND GAM$(FIND,FIND)=G$ T
HEN GOSUB 4970
1800 IF BMSG=9 THEN BOMB=INT(RND(1)*10
)+1:IF BOMB=9 THEN GOSUB 4880
1810 POKE SYSTAB+24,1
1820 LIMIT=LIMIT+1:FOR UU=1 TO 4
1830 IF GAM$(NN,UU)=G$ THEN 1880
1840 NEXT UU
1850 GOSUB 2680:GOTOXY 1,10:? "Careles
s move!!!"
1860 FOR DEC=15 TO 0 STEP -1:SOUND 1,D
EC,1,1,1:NEXT DEC
1870 GOSUB 2660:GOTO 1690
1880 FOR I=1 TO 10
1890 IF G$=MV$(I) AND (G$=MUR$ AND G$=
VIC$) THEN GOTO 1990
1900 IF G$=MV$(I) AND G$=MUR$ THEN GOT
O 2030
1910 IF G$=MV$(I) AND G$=VIC$ THEN GOT
O 2120
1920 NEXT I
1930 FOR I=1 TO 5
1940 IF G$=RM$(I) AND G$=LOC$ THEN GOT
O 2180
1950 IF G$=WP$(I) AND G$=WEP$ THEN GOT
O 2220
1960 NEXT I:? CHR$(7)
1970 GOSUB 2680:? "!! TRY AGAIN !!":GO
SUB 2580:GOSUB 2660:GOTO 1690
1980 REM SUICIDE CASE VICTIM
1990 IF J1=1 OR J2=1 THEN GOSUB 2680:?
 "ALREADY KNOWN":GOSUB 2580:GOTO 1690
2000 GOSUB 2680:? G$;" HAS COMMITTED S
UICIDE":GOSUB 2580:GOSUB 2640
2010 VVV$=VIC$:MMM$=MUR$:QWIT=QWIT+2:J
1=1:J2=1:GOTO 1690
2020 REM GOT MURDERER
2030 GOSUB 2680
2040 IF J1=1 THEN GOSUB 2680:? "YOU AL
READY HAVE MURDERER!":GOSUB 2580:GOTO
1690
2050 IF G$="OPHELIA" OR G$="JULIET" TH
EN MSEX$="MURDERESS":GOTO 2070
2060 MSEX$="MURDERER"
2070 ? "YES, ";G$;" IS THE ";MSEX$;"."
:GOSUB 2580:GOSUB 2640
2080 MMM$=MUR$
2090 IF MUR$=VIC$ THEN QWIT=QWIT+1:J2=
1:VVV$=VIC$
2100 QWIT=QWIT+1:J1=1:GOSUB 2940:GOTO
1690
2110 REM GOT VICTIM
2120 IF J2=1 THEN GOSUB 2680:? "YOU AL
READY HAVE VICTIM!":GOSUB 2580:GOTO 16
90
2130 GOSUB 2680:? "YES, ";G$;" IS THE
VICTIM.":GOSUB 2640
2140 VVV$=VIC$
2150 IF VIC$=MUR$ THEN QWIT=QWIT+1:J1=
1:MMM$=MUR$
2160 QWIT=QWIT+1:J2=1:GOSUB 2940:GOTO
1690
2170 REM GOT THE ROOM
2180 IF J4=1 THEN GOSUB 2680:? "YOU AL
READY HAVE ROOM!":GOSUB 2580:GOTO 1690
2190 GOSUB 2680:? "YES, ";G$;" IS THE
ROOM.":GOSUB 2640
2200 RRR$=LOC$:QWIT=QWIT+1:J4=1:GOSUB
2940:GOTO 1690
2210 REM GOT THE WEAPON
2220 IF J3=1 THEN GOSUB 2680:? "YOU AL
READY HAVE WEAPON!":GOSUB 2580:GOSUB 1
690
2230 GOSUB 2680:? "YES, ";G$;" IS THE
WEAPON.":GOSUB 2640
```

DRAMA-CIDE

```
2240 WWW$=WEP$:QWIT=QWIT+1:J3=1:GOSUB
2940:GOTO 1690
2250 REM END GAME ON CORRECT GUESS
2260 SOUND 1,0,0,0,0
2270 IF MUR$=VIC$ THEN GOSUB 3580:GOTO
 2300
2280 CHOICE=INT(RND(1)*4)+1
2290 ON CHOICE GOSUB 3410,3740,3890,40
60
2300 GOSUB 2680:GOTOXY 1,13:? "Score:
";SC;"  ~~~~~~  Play Again  Y/N":COLOR
 11
2310 IF SC<=0 THEN ?:? "      YOUR LICE
NCE HAS BEEN REVOKED"
2320 IF SC>0 AND SC<=10 THEN ?:? "
  NOT A BAD DETECTIVE"
2330 IF SC>10 THEN ?:? "  YOU GET A PR
OMOTION AND A RAISE."
2340 NN=INP(2):GOSUB 3010:POKE SYSTAB+
24,1
2350 IF NN<>78 AND NN<>110 THEN 5010
2360 CLOZ=1:GOTO 5010
2370 CLEARW 2:RESTORE 4390:COLOR 2,4,1
1
2380 FOR RL=1 TO 8
2390 READ LX1,LY1,LX2,LY2
2400 LINEF LX1,LY1,LX2,LY2
2410 NEXT RL:FILL 6,6
2420 GOTOXY 8,9:? "GOODBYE...":SH=1:X=
200:Y=55:COLOR 2,3,4:GOSUB 3220
2430 HT=9:GOSUB 2550:GOTOXY 8,12:? "Dr
ama-cide"
2440 HT=6:GOSUB 2550:POKE SYSTAB+24,0:
NN=INP(2):GOSUB 3010
2450 CLEARW 2:GOSUB 2470:END
2460 REM CURSOR ROUTINES
2470 POKE CONTRL,122:POKE CONTRL+2,0:P
OKE CONTRL+6,1
2480 POKE INTIN,1:VDISYS(0):RETURN:REM
 SHOW
2490 POKE CONTRL,123:POKE CONTRL+2,0:P
OKE CONTRL+6,0
2500 VDISYS(0):RETURN:REM HIDE
2510 REM TEXT TYPE
2520 POKE CONTRL,106:POKE CONTRL+2,0:P
OKE CONTRL+4,1
2530 POKE CONTRL+6,1:POKE INTIN,TYPE:V
DISYS(0):RETURN
2540 REM CHARACTER SIZE
2550 POKE CONTRL,12:POKE CONTRL+2,1:PO
KE CONTRL+6,0
2560 POKE PTSIN,0:POKE PTSIN+2,HT:VDIS
YS(0):RETURN
2570 REM DELAY LOOP
2580 FOR DEL=1 TO 2500:NEXT DEL:RETURN
2590 REM SCREEN COLORS
2600 POKE CONTRL,14:POKE CONTRL+2,0:PO
KE CONTRL+6,4
2610 POKE INTIN,CI:POKE INTIN+2,RED:PO
KE INTIN+4,BLUE
2620 POKE INTIN+6,GREEN:VDISYS(0):RETU
RN
2630 REM + SCORE
2640 SC=SC+10:RETURN
2650 REM - SCORE
2660 SC=SC-2:RETURN
2670 REM MESSAGE SCREEN LOCATION
2680 GOTOXY 1,10:? SP$:GOTOXY 1,10:RET
URN
2690 REM INTERRUPTIONS TO INVESTIGATIO
N
2700 RUP=INT(RND(1)*8)+1
2710 ON RUP GOSUB 2730,2750,2770,2780,
2800,2820,2840,2860
2720 RETURN
2730 GOSUB 2680:? "A phone call from t
he office"
2740 ? " interrupts your investigation
.":RETURN
2750 GOSUB 2680:? "You leave to check
out"
2760 ? " ";GAM$(NN,2);"'s credentials.
":RETURN
2770 GOSUB 2680:? "!! LUNCH IS READY !
!":RETURN
2780 GOSUB 2680:? "Somebody slugs you
from behind."
2790 ? " When you come to...":RETURN
2800 GOSUB 2680:? "You leave to invest
igate"
2810 ? " a strange noise in the hall."
:RETURN
2820 GOSUB 2680:? "You hear a scream f
or"
2830 ? " help out in the hall...":RETU
RN
2840 GOSUB 2680:? "You have to leave t
o"
2850 ? " check on a burning smell outs
ide.":RETURN
2860 GOSUB 2680:? "Ayyyy!! You fell th
rough"
2870 ? " a trap door and find...":RETU
RN
2880 REM INTERRUPT SOUND
2890 FOR SD=1 TO 3
2900 SOUND 1,10,1,6,1
2910 SOUND 1,10,8,6,1
2920 NEXT SD:SOUND 1,0,0,0:SC=SC-2:RET
URN
2930 REM - SONG
2940 WAVE 3:SOUND 1,10,10,2,20:SOUND 1
,10,12,2,5
2950 SOUND 1,10,1,3,20:SOUND 1,10,10,2
,20
2955 FOR DECAY=12 TO 0 STEP -1:DEC=DEC
AY-4:IF DEC<0 THEN DEC=0
2960 SOUND 1,DECAY,4,3:SOUND 2,DEC,6,6
,10
2970 NEXT DECAY:RETURN
2980 REM CAPS LOCK SOUND
2990 FOR SND=1 TO 12 STEP 2:SOUND 1,10
,SND,5,1:NEXT SND:SOUND 1,0,0,0,0:RETU
RN
3000 REM KEY ALERT SOUND
3010 SOUND 1,10,12,5,1:SOUND 1,10,1,5,
1:SOUND 1,0,0,0:RETURN
3020 REM MUR$ -> SHE/HE PRONOUN
3030 IF MUR$=N$(1) OR MUR$=N$(2) THEN
GEN$="she" ELSE GEN$="he"
3040 RETURN
3050 REM VIC$ -> SHE/HE PRONOUN
3060 IF VIC$=N$(1) OR VIC$=N$(2) THEN
GEN$="she" ELSE GEN$="he"
3070 RETURN
3080 REM VIC$ -> HER/HIS PRONOUN
3090 IF VIC$=N$(1) OR VIC$=N$(2) THEN
GEN$="her" ELSE GEN$="his"
3100 RETURN
3110 REM VIC$ -> HERSELF/HIMSELF PRONO
UN
3120 IF VIC$=N$(1) OR VIC$=N$(2) THEN
GEN$="herself" ELSE GEN$="himself"
3130 RETURN
3140 REM VIC$ -> HER/HIM PRONOUN
3150 IF VIC$=N$(1) OR VIC$=N$(2) THEN
GEN$="her" ELSE GEN$="him"
3160 RETURN
3170 REM MODUS HEADER
3180 N$(1)="JULIET":N$(2)="OPHELIA"
3190 HT=9:GOSUB 2550:GOTOXY 7,1:? "**
MODUS OPERANDI **"
3200 TYPE=0:GOSUB 2520:HT=6:GOSUB 2550
:RETURN
3210 REM MASKS
3220 RESTORE 3280
3230 READ XO1
3240 IF XO1=999 THEN RETURN
3250 READ YO1,XO2,YO2
3260 LINEF X+(XO1/SH),Y+(YO1/SH),X+(XO
2/SH),Y+(YO2/SH)
3270 GOTO 3230
3280 DATA -25,0,-21,3,-21,3,-16,4,-16,
4,-6,5
3290 DATA -6,5,6,5,6,5,5,4,5,4,20,3,20
,3,25,0
3300 DATA 25,0,27,21,27,21,25,41,25,41
,21,51
3310 DATA 21,51,18,56,18,56,8,63,8,63,
3,64
3320 DATA 3,64,-3,64,-3,64,-8,63,-8,63
,-19,56
3330 DATA -19,56,-22,51,-22,51,-26,41,
-26,41,-27,21
3340 DATA -27,21,-25,0
3350 DATA -12,15,-4,24,-4,24,-20,24,-2
0,24,-12,15
3360 DATA 13,15,21,24,21,24,5,24,5,24,
13,15
3370 DATA 1,24,10,35,10,35,-9,35,-9,35
,0,24
3380 DATA -17,49,-11,44,-11,44,9,44,9,
```

DRAMA-CIDE

```
44,15,49
3390 DATA 15,49,7,48,7,48,-9,48,-9,48,
-17,49,999
3400 REM MODUS #1
3410 GOSUB 3180:REM HEADER
3420 ?:? " It seems that ";:COLOR 2:?
VIC$;:COLOR 8:?" found out that"
3430 COLOR 2:? MUR$;:COLOR 8:? " was s
tealing gold from "
3440 ? "the King's coffers.  When ";:C
OLOR 2:? MUR$:COLOR 8:? "realized it,
";
3450 GOSUB 3030:REM HE/SHE PRONOUN
3460 ? GEN$;" killed ";:COLOR 2:? VIC$
;:COLOR 8:? " with "
3470 ? "the ";:COLOR 2:? WEP$;:COLOR 8
:? ", in the ";
3480 COLOR 2:? LOC$;:COLOR 8:? ". Then
 ";GEN$
3490 FOR I=1 TO 5
3500 IF GAM$(I,1)<>LOC$ THEN 3540
3510 IF GAM$(I,2)=MUR$ OR GAM$(I,3)=MU
R$ THEN SCAPE$="THE SERVANT":GOTO 3550
3520 IF GAM$(I,2)=VIC$ THEN SCAPE$=GAM
$(I,3):GOTO 3550
3530 SCAPE$=GAM$(I,2)
3540 NEXT I
3550 ? "tried to pin the crime on ";:C
OLOR 2:? SCAPE$;:COLOR 8:? ","
3560 ? "who was in the deadly room.":?
:RETURN
3570 REM MODUS #2 SUICIDE
3580 GOSUB 3180:REM HEADER
3590 ?:? " It is sad to say, but ";:CO
LOR 2:? MUR$
3600 COLOR 8:? "really had no friends.
  He was"
3610 ? "rebuffed by ";:COLOR 2:GIRL=IN
T(RND(1)*2)+1
3620 ? N$(GIRL);:COLOR 8:? ". Realizin
g ";
3630 GOSUB 3090:REM HER/HIS PRONOUN
3640 ? GEN$:? "pathetic state, ";
3650 GOSUB 3060:REM HE/SHE PRONOUN
3660 ? GEN$;" decided to ";
3670 IF WEP$="DAGGER" OR WEP$="SPEAR"
THEN WND$="stab"
3680 IF WEP$="ROPE" THEN WND$="hang"
3690 IF WEP$="AXE" OR WEP$="MACE" THEN
 WND$="bludgeoned"
3700 ? WND$;" "
3710 GOSUB 3120:REM HERSELF/HIMSELF PR
ONOUN
3720 ? GEN$;:? " yesterday, in the ";:
COLOR 2:? LOC$;".":COLOR 8:?:RETURN
3730 REM MODUS #3
3740 GOSUB 3180:REM HEADER
3750 COLOR 2:?:? MUR$;:COLOR 8:? " is
a sick drinker and suffers"
3760 ? "from delerium tremins.  Two da
ys ago,":? "it seems, ";
3770 GOSUB 3030:REM HE/SHE PRONOUN
3780 ? GEN$;" had an alcoholic seizure
"
3790 ? "while drinking it up with ";:C
OLOR 2:? VIC$:COLOR 8
3800 ? "in the ";:COLOR 2:? LOC$;:COLO
R 8:? ".  Things got a little"
3810 ? "out of hand and ";:COLOR 2:? M
UR$;:COLOR 8:? " accidentally"
3820 IF WEP$="DAGGER" OR WEP$="SPEAR"
THEN WND$="stabbed"
3830 IF WEP$="ROPE" THEN WND$="hanged"
3840 IF WEP$="AXE" OR WEP$="MACE" THEN
 WND$="bludgeoned"
3850 ? WND$;" ";
3860 GOSUB 3150:REM HER/HIM PRONOUN
3870 ? GEN$;" with the ";:COLOR 2:? WE
P$;".":COLOR 8:?:RETURN
3880 REM MODUS #4
3890 GOSUB 3180:REM HEADER
3900 ?:COLOR 2:? MUR$;:COLOR 8
3910 ? " and ";:COLOR 2:? VIC$;:COLOR
8:? " were cousins by"
3920 ? "blood, and both were heir appa
rent to"
3930 ? "the royal throne.  But ";:COLO
R 2:? VIC$;:COLOR 8:? " was"
3940 ? "older; hence, next in line.  I
t was"
3950 ? "too much for the greedy ";:COL
OR 2:? MUR$;:COLOR 8:? ",":? "so ";
3960 GOSUB 3030:REM HE/SHE PRONOUN
3970 ? GEN$;" secretly lured ";:COLOR
2:? VIC$;:COLOR 8:? " to the"
3980 COLOR 2:? LOC$;:COLOR 8:? " and "
;
3990 IF WEP$="DAGGER" OR WEP$="SPEAR"
THEN WND$="stabbed"
4000 IF WEP$="ROPE" THEN WND$="strangl
ed"
4010 IF WEP$="AXE" OR WEP$="MACE" THEN
 WND$="bludgeoned"
4020 ? WND$;" ";
4030 GOSUB 3150:REM HER/HIM PRONOUN
4040 ? GEN$;" to death.":RETURN
4050 REM MODUS #5
4060 GOSUB 3180:REM HEADER
4070 COLOR 2:?:? VIC$;:COLOR 8:? " pro
mised ";
4080 COLOR 2:? MUR$;:COLOR 8:? " a top
 place"
4090 ? "in the kingdom, if ";:COLOR 2:
? MUR$;:COLOR 8:? " would pay "
4100 GOSUB 3150:REM HER/HIM PRONOUN
4110 ? GEN$;:GOLD=INT(RND(1)*10)+1:GOL
D=GOLD*1000
4120 ? GOLD;"gold pieces.";
4130 ? " After ";:COLOR 2:? VIC$:COLOR
 8:? "got the gold ";
4140 GOSUB 3030:REM HE/SHE PRONOUN
4150 ? GEN$;" feigned ignorance, so"
4160 COLOR 2:? MUR$;:COLOR 8:? " kille
d ";:COLOR 2:? VIC$;:COLOR 8
4170 ? " with a ";:COLOR 2:? WEP$:COLO
R 8:? "in the ";
4180 OWN=INT(RND(1)*3)+1
4190 IF OWN=1 THEN OWN$="Queen's":GOTO
 4220
4200 IF OWN=2 THEN OWN$="King's" :GOTO
 4220
4210 OWN$="servant's"
4220 ? OWN$;" ";:COLOR 2:? LOC$;:COLOR
 8:? ".":?:RETURN
4230 REM ROOM OCCUPANCY CHART
4240 CLEARW 2:COLOR 4:GOTOXY 11,1:? "R
OOM OCCUPANCY":RESTORE 4310
4250 FOR RL=1 TO 16
4260 READ LX1,LY1,LX2,LY2
4270 LINEF LX1,LY1,LX2,LY2
4280 NEXT RL
4290 REM GIVE ROOM -- MAYBE!
4300 GAMB=INT(RND(1)*5)+1:IF GAMB=5 TH
EN SC=10
4310 DATA 10,35,110,35,10,35,10,90
4320 DATA 110,35,110,160,10,90,110,90
4330 DATA 200,35,300,35,200,35,200,160
4340 DATA 300,35,300,90,200,90,300,90
4350 DATA 10,105,110,105,10,105,10,160
4360 DATA 10,160,110,160,200,195,300,1
05
4370 DATA 300,105,300,160,200,160,300,
160
4380 DATA 110,60,200,60,110,120,200,12
0
4390 DATA 5,5,300,5,300,5,300,165,300,
165,5,165
4400 DATA 5,165,5,5,14,14,290,14
4410 DATA 290,14,290,155,290,155,14,15
5
4420 DATA 14,155,14,14
4430 CCT=0:RESTORE 4650:FOR I=1 TO 20:
COLOR 8
4440 READ V1,V2,V3,V4
4450 IF UNLK<>218 THEN 4510
4460 IF GAM$(V3,V4)=LOC$ THEN COLOR 7
4470 IF GAM$(V3,V4)=MUR$ THEN COLOR 2
4480 IF GAM$(V3,V4)=VIC$ THEN COLOR 7
4490 IF GAM$(V3,V4)=WEP$ THEN COLOR 7
4500 GOTO 4540
4510 CCT=CCT+1
4520 IF CCT=1 THEN COLOR 2
4530 IF CCT=1 AND GAM$(V3,V4)=LOC$ AND
 GAMB=5 THEN COLOR 7
4540 GOTOXY V1,V2:? GAM$(V3,V4):IF CCT
=4 THEN CCT=0
4550 NEXT I
4560 REM ON GAMB
4570 IF GAMB<>5 THEN 4700
4580 HT=7:GOSUB 2550:TYPE=4:GOSUB 2520
:COLOR 7
4590 GOTOXY 14,3:? "MURDER":GOTOXY 14,
```

DRAMA-CIDE

PROGRAM LISTINGS

```
4:? "ROOM"
4600 GOTOXY 14,5:? "BONUS":TALLY=TALLY
+1:IF TALLY=10 THEN 4700
4610 SOUND 1,10,8,7,1
4620 GOTOXY 14,3:? "      ":GOTOXY 14,
4:? "     "
4630 GOTOXY 14,5:? "      ":SOUND 1,10,
1,7,1
4640 GOTO 4590
4650 DATA 2,4,1,1,2,6,1,2,2,7,1,3,2,8,
1,4
4660 DATA 23,4,2,1,23,6,2,2,23,7,2,3,2
3,8,2,4
4670 DATA 2,12,3,1,2,14,3,2,2,15,3,3,2
,16,3,4
4680 DATA 23,12,4,1,23,14,4,2,23,15,4,
3,23,16,4,4
4690 DATA 13,7,5,1,13,9,5,2,13,10,5,3,
13,11,5,4
4700 SOUND 1,0,0,0,0:TYPE=16:GOSUB 252
0:COLOR 11:GOTOXY 13,15:? "Press Key"
4710 POKE SYSTAB+24,0:NN=INP(2):GOSUB
3010:RETURN
4720 REM BOMB WARNING
4730 CLEARW 2:RESTORE 4390
4740 FOR RL=1 TO 8
4750 READ LX1,LY1,LX2,LY2
4760 LINEF LX1,LY1,LX2,LY2
4770 NEXT RL
4780 TYPE=0:GOSUB 2520:HT=8:GOSUB 2550
4790 COLOR 12:GOTOXY 9,5:? "<><> WARNI
NG <><>"
4800 COLOR 13,2,2:HT=6:GOSUB 2550
4810 GOTOXY 3,7:? "Someone has set a g
hastly BEAR"
4820 GOTOXY 3,8:? "GREASE bomb. It wil
l incinerate"
4830 GOTOXY 3,9:? "you and engulf the
whole castle"
4840 GOTOXY 3,10:?"in flames, if deton
ated."
4850 SH=2:X=150:Y=117:GOSUB 3220:WAVE
6,1,14,1000
4860 COLOR 8,2,14:FOR D=1 TO 8000:NEXT
 D:WAVE 6,0,0,0:RETURN
4870 REM END ON BOMB
4880 CLEARW 2:SOUND 1,10,1,1,100:SOUND
 1,0,0,0,0
4890 COLOR 4:GOTOXY  3,5: ? "YOU WERE
WARNED ABOUT THE BOMB"
4900 GOTOXY 3,7:? "***  AND NOW YOU AR
E DEAD  ***"
4910 FOR D=1 TO 3000:NEXT D
4920 GOTOXY 3,9:? "        SOME DETECT
IVE!      "
4930 GOTOXY 3,15:? " PRESS ANY KEY TO
START OVER  "
4940 WAVE 6,1,14,1000:NN=INP(2):POKE S
YSTAB+24,0
4950 GOTO 5010
4960 REM DISARM BOMB MESSAGE
4970 GOSUB 2680:? "YOU FIND BOMB!!
             "
4980 ? " YOU DISARM IT SAFELY.":SP$=SP
ACE$(100):SC=SC+20
4990 GOSUB 2580:GOSUB 2680:SP$=SPACE$(
40):TAG=0:BMSG=0:RETURN
5000 REM SCORE BOARD
5010 CLEARW 2:COLOR 8:ROUND=ROUND+1:RD
(ROUND)=SC
5020 HT=9:GOSUB 2550:GOTOXY 10,1:? "**
 SCORES **"
5030 HT=6:GOSUB 2550:?:FOR LST=0 TO 9
5040 IF LST+1<>ROUND THEN 5080
5050 COLOR 2
5060 ? "   CASE ";LST;"...............
......";RD(LST+1):COLOR 8
5070 GOTO 5090
5080 ? "   CASE ";LST;"...............
......";RD(LST+1)
5090 TOT=TOT+RD(LST+1)
5100 NEXT LST
5110 COLOR 5:GOTOXY 1,14:? "          TO
TAL SCORES  > ";TOT:TOT=0:COLOR 8
5120 GOTOXY 1,15:? "        ~~  PRESS A
 KEY  ~~"
5130 IF CLOZ=1 THEN 5180
5140 IF ROUND<10 THEN 5170
5150 GOTOXY 4,15:? "PRESS A KEY TO STA
RT GAME OVER"
5160 POKE SYSTAB+24,0:NN=INP(2):GOSUB
3010:POKE SYSTAB+24,1:GOTO 70
5170 POKE SYSTAB+24,0:NN=INP(2):GOSUB
3010:POKE SYSTAB+24,1:GOTO 530
5180 POKE SYSTAB+24,0:NN=INP(2):GOTO 2
370
```

DRAMA-CIDE LISTING 1 CHECKSUM DATA

```
10 data 180,484,558,273,849,190,467,66
1,748,15,4425
110 data 364,771,465,702,790,364,381,3
79,264,324,4804
210 data 314,976,494,508,636,241,284,6
10,560,736,5359
310 data 312,436,401,98,945,883,569,42
5,507,351,4927
410 data 503,225,659,334,412,498,407,1
56,238,927,4359
510 data 249,161,543,89,646,385,471,70
7,881,900,5032
610 data 896,916,566,783,83,773,544,95
3,244,663,6421
710 data 16,838,950,849,990,433,911,42
3,306,881,6597
810 data 591,971,668,421,453,94,823,45
5,380,355,5211
910 data 18,800,779,515,972,551,670,94
3,300,494,6042
1010 data 202,317,667,947,729,675,937,
322,491,940,6227
1110 data 874,657,355,130,976,666,441,
846,452,948,6345
1210 data 190,228,237,190,252,261,506,
163,447,657,3131
1310 data 412,32,435,103,352,727,665,8
50,578,143,4297
1410 data 182,809,753,515,613,695,881,
214,501,699,5862
1510 data 527,218,353,46,393,650,696,7
07,145,291,4026
1610 data 262,99,586,134,588,742,590,2
50,870,506,4627
1710 data 4,490,711,452,525,965,888,84
9,307,216,5407
1810 data 520,24,646,478,460,562,930,5
2,589,102,4363
1910 data 87,358,14,65,115,421,740,672
,142,233,2847
2010 data 212,702,652,876,431,651,658,
700,223,179,5284
2110 data 442,563,648,738,154,187,616,
337,397,497,4579
2210 data 951,741,649,577,429,135,173,
114,483,734,4986
2310 data 930,713,35,144,79,433,786,48
,11,208,3387
2410 data 552,364,307,148,418,271,627,
769,630,888,4974
2510 data 281,627,86,57,439,648,465,92
9,844,442,4818
2610 data 984,363,844,526,850,402,341,
335,408,755,5808
2710 data 130,462,263,139,985,222,581,
984,65,617,4448
2810 data 871,940,68,795,331,638,905,2
87,37,217,5089
2910 data 239,264,729,368,434,710,788,
929,47,704,5212
3000 data 82,863,878,723,445,848,651,4
48,78,837,5853
3100 data 444,207,447,447,71,830,450,7
30,33,634,4293
3200 data 724,807,43,483,342,525,472,5
72,310,445,4723
3300 data 493,244,580,45,229,786,273,3
56,620,922,4548
3400 data 108,193,571,358,143,259,485,
559,600,9,3285
3500 data 673,300,295,261,352,160,53,2
06,203,857,3360
3600 data 293,518,502,555,771,274,461,
799,520,633,5326
3700 data 474,670,894,126,205,600,790,
270,257,292,4578
3800 data 270,911,400,890,634,600,548,
833,137,213,5436
3900 data 606,205,558,154,26,996,275,2
31,742,410,4203
4000 data 547,609,575,523,3,115,188,91
8,79,698,4255
4100 data 523,736,872,126,251,472,355,
771,739,522,5367
4200 data 348,116,543,906,346,147,9,21
3,452,336,3416
4300 data 361,363,689,835,704,809,982,
131,856,751,6481
4400 data 300,951,244,398,549,70,591,6
40,591,624,4958
4500 data 584,670,64,928,285,355,962,8
81,804,994,6527
4600 data 766,236,3,630,601,92,468,465
,855,706,4822
4700 data 387,180,660,289,60,23,227,46
6,744,588,3624
4800 data 826,751,606,181,799,637,142,
514,446,420,5322
4900 data 803,131,324,248,630,593,634,
76,748,647,4834
5000 data 624,417,666,580,628,337,601,
586,581,762,5782
5100 data 600,427,629,724,975,256,295,
444,25,4375
```

DRAMA-CIDE
END

PROGRAM LISTINGS

STPlus•STPlus•STPlus•STPlus

P.O. 1197, Berkeley, Ca. 94701 • *add 3% credit card for hardware*

Front line NEWS: GENLOCK for the ST, $400, preorder.

Spectre 128 (run Mac SE programs) $179 • PCDitto, DOS, & Drive $399

BUSINESS

DBMan 4.0	175.00
Datamanager	39.95
Superbase	104.95
Trimbase	69.95
Phasar 3.0	63.95
Zoomracks 2	84.95
Base 2	42.95
The Informer	69.95
Wordperfect	189.95
1st Word Plus	69.95
Word Up!	64.95
Best Accounting	279.95
Equal Plus	139.95
Inventory Mgr.	69.95
Rolobase Plus	63.95
Logistix Spread	104.95
Microlawyer	49.95
Payroll Master	69.95
Construction EST.	35.00
Microsoft Write	94.95
Datatrieve	35.00
STOneWrite	48.95
VIP GEM	104.95
DacEasy Payroll	48.00
DacEasy Acctg	52.00
WordWriter ST	56.00
SwiftCalc	39.95
EZ Calc by Royal	48.95
Analyze Spread	25.95
Final Word	99.95
PublishingPartner	140.00
T-works Publisher	89.95
EZData Base	48.95
Chart Pak	35.00
Compute Roots	27.95
Thunder NEW!	28.95
Habawriter 2	48.95
Text Pro	35.00
Becker Text	62.95
Expert Opinion AI	59.95
Time Link	35.00
Partner ST	48.95
Labelmaster Elite	35.00
ST Accounts	149.00
The Juggler 2.0	35.00
Max Pack	35.00
Stuff	27.95
Flash 1.5	21.00
Omni Res	27.95
Turbo ST(~blitter)	35.00
Signum *technical word processor*	249.95
SBT DLedger	195.00
SBT DPayables	195.00
SBT DInvoices	195.00
SBT DMenu	49.95
Neo Desk	27.95
Sales Pro	69.95
Mail Manager	39.95
Mighty Mail	35.00
First Word 1.6	14.00

GRAPHICS

Degas Elite	41.95
CAD 3D 2.0	63.95
Cyber Paint	49.00
Quantum 4096	27.95
Adv Art Studio	26.00
Spectrum 512	49.00
EzDraw&Superch	104.95
Canon Scanner	1040.0
GFA Artist 1000cl	55.95
Drafix 1	139.95
General Symbols	105.00
Elec, or Arch, Sym	105.00
Athena 2	69.95
Circuit Maker	55.95

CLIPART

Warriors(720)	14.95
Outdoor(720)	14.95
Buildings(720)	14.95
Victorian(720)	14.95
Etchings(720)	14.95
People (720k)	14.95
Politics(720k)	14.95
Religion(720)	14.95
Holidays(720)	14.95
Ad Art (720k)	14.95
Vehicles(720)	14.95
Boats (720k)	14.95
Planes (720k	14.95
All (9.2 meg)	99.95

GAMES

Gunship	35.00
Shadowgate	35.00
Uninvited	35.00
Mouse Quest	14.00
Slaygon	27.95
Barbarian	27.95
Obliterator	27.95
Gauntlet	35.00
Dark Castle	27.95
F-15 Strike Eagle	27.95
Star Trek-Rebel U	27.95
Questron II	35.00
Lock-On	27.95
Carrier Command	32.50

MUSIC

Passport	280.00
Master Tracks	104.95
MasterTracks Jr.	69.95
Midisoft Studio	
Hybrid Arts	
Smpte Track	499.95
Sync Track	299.95
EZ Track Plus	48.95
Midiscore	call
EZ Score Plus	104.95
DX-Android	139.95
CZ-Android	69.95
Gen-Patch	104.95
Dr. T's	
KCSequencer	199.95
KCS 1,6 w/PVG	289.95
MIDI rec studio	56.00
Copyist level 1	75.95
Copyist level 2	185.95
Copy3-Postscript	299.95

HARDWARE

10 Meg Supra flppy	875.00
20 Meg w/clock	599.95
30 Meg Supra	749.95
60 meg Supra	1249.95
33 Meg Tulin	699.95
51 Meg Tulin	839.95
80 Meg BMS RRL	1249.95
20 Meg SH205	639.95
250 Meg	3250.0
10 Meg Floppy	849.95
AST PS LASER	3350.00
Canon Scanner	1040.00
IMG Scanner	90.00
ComputereyesMon	120.00
Supra 2400 modem	159.95
Atari SX212 modem	79.95

Remember: Every one thousandth purchaser gets a hundred dollars credit, and the ten thousandth purchaser will get a 10 meg Supra floppy. RULES: *Have your customer number or credit card ready. Purchase as often as you like. We're your computer supermarket and we're ready to fulfill your dreams but we're not an information service. We ship right away and its your part to know what you want. Don't be cheap, at these prices you can splurge!*

SPECIAL-SPECIAL-SPECIAL

MICROSOFT WRITE FOR ONLY $50

while supplies last.

PROGRAMMING

GFA Basic 3.0	56.00
GFA Book	35.00
GFA Compiler	56.00
Mark Williams "C"	125.00
Laser "C"	159.95
Cambridge Lisp	139.95
RAID	27.95
Fast Editor	35.00
Alice Pascal	69.95
OSS Pascal	59.95
Fortran 77 GEM	139.95
BCPL	104.95
Modula 2 dev. kit	104.95
Assempro	48.95
Fast Basic	56.95
True Basic	69.95

EDUCATIONAL

Arakis Series	14.00
Unicorn Series	27-35
True Basic Stuff	69.95

Autumn 520 STFM Special

$599 Mono

$799 Color

Your Art Scanned

$10 per page

75 to 300 dpi

GAMES

Tanglewood	27.95
Test Drive	35.00
Chessmastr2000	32.95
Starglider II *NEW*	35.00
Hunt for Red Oct	35.00
TyphoonThompson	27.95
Aliants	24.95
Alien Fire	35.00
Santa Paravia	21.00
Lurking Horror	27.95
Star Fleet 1	39.95
Empire	39.95
Liesure Suit Larry	27.95
Gridiron	35.00
Dungeon Master	27.95
Flight Simulator	35.00
Trailblazer	27.95
····SPECIALS····	
Jewel of Darknss	19.95
Silicon Dreams	19.95
Cardiac Arrest	48.95

The Best European Games

I Ball (neat,fast)	19.95
RanaRama(d&d)	29.95
Warlock'sQuest	29.95
The Flintstones	29.95
Trivial Pursuit	29.95
The Enforcer	19.95
Seconds Out	29.95
Scruples(board)	29.95
Livingstone	29.95
Battle Ships	29.95
Outrun(fast cars)	29.95
Crazy Cars	29.95
Tetris (from USSR)	29.95
Screaming Wings	29.95
Spitfire	29.95
Blue War	29.95
Star Quake	29.95
Enduro Racer	29.95
BMX Simulator	29.95
Arkanoids	29.95
Better Dead n Alien	29.95

Are you a gamer? How about joining our game of the month club? $10 gets you the hottest new title at an extra 5% off & you can return it for 75% credit. Plus you'll be eligible each purchase to win as #1000 or #10,000. Call us and be first to play the new ones. Start now with Typhoon Thomson for $25.95.

ONE NEW NUMBER **800-759-1110** Prices subject to change without notice.

We ship ANYWHERE! $4.00 min S&H. No 1040's or Megas mail order. Hand delivery only, List plus $100.

CIRCLE #109 ON READER SERVICE CARD.

(from page 15)

going on. We'll assume a long sequential read, which is what we're trying to speed up.

Also remember that any time we place a new read request, we need to give one or two sectors up to the seek-with-verify routine to complete the seek. Remember, at least one sector is going to have to spin by before the seek will finish up.

The logistics of breaking up one track request into two sequential reads aren't that difficult. We take the modulus/4 of the track number (the remainder after dividing by four), then use it to look up in two arrays where to begin reading on the first read and where to end reading on the second read. In our code, we call this the "primary read" and "secondary read." On "even" tracks, there is no secondary read.

You've got to look at a track read as occurring after a read from the previous track, a read which we don't know about anymore. We assume that the head is at a certain location on the track we are currently on, and begin reading where the head is (or will be quite shortly). After that, we go grab the beginning of the track.

What happens if we "miss," or if this is the start of a read? The worst is that we miss nearly one spin, which is what Atari's code does at best. Not bad at all.

The Revenge of the Seek-With-Verify

We thought we had it conquered. We wrote the code, got the syntax errors out, (I hate to say, "debugged" it), and fired it up. Yes, the beer and Boodles were flowing freely; we were celebrating.

Alas, the celebration soon turned to mourning.

There was no speed increase at all!

Atari's seek-with-verify wasn't finished with us yet, regrettably. If you like, refer to your Atari BIOS listing—every time you do a floppy-read request, you do a seek-with-verify to whatever track you need to read, even if it is the track you're on. This means every time you issue a read request, you're probably going to lose a spin. Seek-with-verify doesn't care that it already knows what track you're on; it's too dumb. It'll go ahead and re-verify the current track, using up a sector mark.

You can confirm this for yourself (and it's something to keep in mind when you're writing software!). Do a floppy read (*floprd*) of all nine sectors on a track. You'll find the read will complete in about two and three-fifths seconds: one spin to mollify the seek, one spin to do the read, and some extra depending on where the head was.

Now, go do nine individual reads of one sector: 1,2,3,4,5,6,7,8,9. You'll find it takes *nine spins* to do it, or nine times as long as the track-sized read! It's because every time you go do the read, the seek-with-verify grabs the next sector to verify the track—and that next sector is usually the one you want. For the software designer, this means *always read a track at a time* unless you want to get some really bad floppy-disk response.

I know, your hair is standing on end. But it gets worse.

Nightmare on Track Street

If you go to the second side of a double-sided disk, guess what? You get another seek-with-verify. This is why we had to "double twist" the double-sided version of our disk formatter, adding an extra two sectors to any second-side request. (At the time, we didn't know why it was needed, only that it was needed.)

That's sad, because ordinarily double-sided disks are quite fast; switching heads is instantaneous, and you don't have to do a seek, which is the slowest part of a disk operation. This *one* seek-with-verify operation manages to slow up three different aspects of floppy operation.

Our number one request for the new ROMS: Get rid of the seek-with-verify! Bribes available upon request. Write for details. (Note: Atari has now already done this. Hooray!)

Back to Our Story

At this point, the supply of Boodles and Stroh's was getting thin, and the program still wasn't working. So Dan and I decided to commit the unpardonable sin and go straight to the ROMS, bypassing the seek-with-verify, to get the routine working. There just wasn't any other way to do it.

Well, when the ROMS change, preferably sometime before the next Ice Age, we'll be happy to update the code—but in the meantime, that's no reason not to enhance the old ones. We even put a check in to make sure you're running with the right ROM set.

If you'll look at our listing (available on this month's disk version), you'll see two new routines, which handle *floppyread* and *floppywrite.* They bypass the seek-with-verify routine part of floppy read/write, then jump to the "real" ROM handlers. We use these new routines anytime we're content that we're on the proper track (e.g., we *know* we have seeked to the right track, for instance, on double-sided requests).

So, we put in the new routines, fired the system up, opened our last beer and tonic water, and zoom!—it worked. Ordinary mortal floppies began to read and write at Twister speed. Thank heavens.

Of course, the read-after-write-verify must be switched off. If this offends you, remove the MOVE to location $444 and reassemble. We also make all Resets into real Resets, which prevents lots of problems; if you don't like that, get rid of the "memvalid" clear.

Does It Work on Ten-Sector Disks?

Ever since floppy owners have figured out that you can put ten sectors on a track, and get 400K per disk side instead of 360K, the ten-sector format has been popular. Proper credit should go to James Eli, whose public domain program FORMAT turned us on to ten sectors per disk.

It's a little more work for us, because there's little or no room left at the end of the track in a ten-sector format. This room gives lots of "slop" for seek time. However, we make up for it by skipping two sectors per spin; the seek ends right around the first sector's start, leaving the second sector to verify the seek.

Trans-Warp uses GEM's variable, passed to RWABS, which tells how many sectors there are per track. (It's straight from the boot block data that *protobt* writes.)

Conclusion

Well, there you have it: maximum floppy speed for ordinary floppy disks. Put Trans-Warp (TWARP.PRG) in your AUTO folder, and you'll really hear the floppies accelerate. (Just listen to the tick-pause-tick-pause of the floppy seek changing to tick-tick-tick). We hope you enjoy Trans-Warp, and hope that it finds its way to many of your AUTO folders. Certainly it's a permanent resident in ours.

You install Trans-Warp by either placing it in your AUTO folder, then starting up the computer, or just double-clicking on TWARP.PRG directly. Either way, you'll get a friendly sign-in message, then return to the regular desktop/command line. Your floppies are now set up for double speed.

People doing GEM-based DISKCOPY should be warned—DISKCOPY does so much screen drawing between each track that it fouls up our careful sector skewing. You'll see this as a slow-down to normal speed during DISKCOPY. We decided not to slow down all floppy requests to handle this one special case.

IAN'S QUEST

by Ian Chadwick

A while back I wrote a somewhat critical review of *Universal Military Simulator* (UMS). That review caused the publisher some consternation, in part, because it had some errors in it. I'm going to address those errors now and eat a few words.

1) The company's name is Rainbird, not Firebird. The latter is their name in the U.K., not the U.S.A. A rose by any other name. . . .

2) UMS is *not* rehashed C64 software. I apologize for suggesting that this was the case. In the original press releases, a C64 version was mentioned. It never came about. I mistakenly assumed that, given the graphics, UMS was an upward port from the C64. According to Rainbird, it was developed on the Atari ST.

3) And according to Rainbird, UMS allows 60 units per side, not 24 as I originally stated. It does not say this in the manual. It *does* say: "An army may contain up to six wildcard or user-defined units in addition to the 18 predefined units." However, it seems they meant "unit types," not units. There can be more than one unit of a given type, something not well explained in the manual, and I only discovered how to do this later by accident.

The review was the center of a minor storm of controversy on DELPHI's ST-Log Forum; not so much for what I said (some agreed, others didn't), but whether or not a reviewer should review only products he likes. In other words, if you haven't got anything good to say, say nothing at all.

I don't agree with that approach and, I was pleased to note, neither did anyone else on the Forum, even those who thought my conclusions about the game were wrong. More importantly, I received the support of the ST-Log staff, who have a commitment to integrity in publishing and will publish a critical review—as long as the facts are right.

Okay, enough of that.

SubLOGIC was kind enough to send me its latest products, including the new Western Europe and Japan scenery disks, plus *Jet*, its F16/F18 flight simulator.

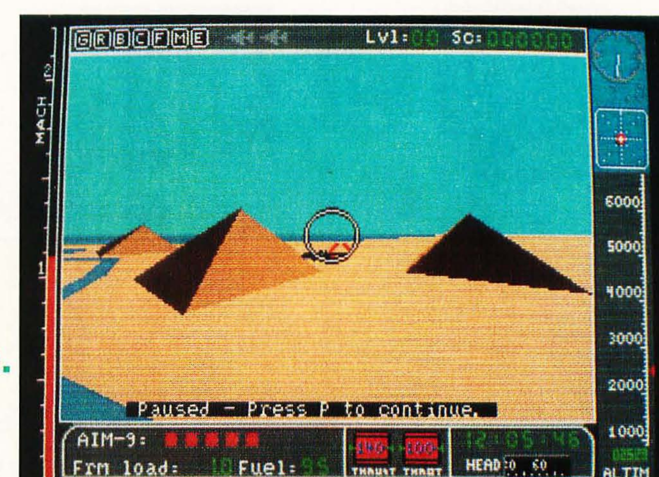
For those of you who have wondered, the ST scenery disks are all "enhanced." It used to call them "Star scenery disks," but we don't have to worry about that sort of distinction now. We get the best of the lot automatically.

Now you should know from previous columns that I like *FS2* a lot. I've spent many pleasant hours flying over the binary countryside, exploring, soaring and, of course, crashing. Still, despite the occasional aerial acrobatics that plant me in the ground, I really do enjoy the program and find it very relaxing, as well as entertaining.

My biggest complaint has always been the lack of Canadian scenery. Yes, I know that the Detroit-area scenery disk included a whole chunk of Canada, even Toronto (alas, my house is missing), but it was blasé, colorless (that's colourless, in Canada), and unconvincing. I wanted more.

Now, contrary to popular belief, Canada is not entirely made up of featureless tundra or windswept, treeless plains (it only looks that way from the air). So I wrote to SubLOGIC and asked for its file format. The company didn't send it to me. I wanted to create my own scenery files and fill in the data to make a detailed flying environment.

Well, soon I'll be able to do just that.



JET, SubLOGIC's F16/F18 FLIGHT SIMULATOR.

SubLOGIC is going to release a scenery-creation utility. Although it couldn't tell me for sure, it may even permit adding detail to existing scenery files. Hooray! For me, that's a tinkerer's delight. As soon as I get it, I'll start to work. Watch for my custom-designed scenery files on DELPHI.

Oh, yes, Jet. Well, I haven't had much time with it; so it's not fair to review it in detail yet, but I like what I see so far. It's *very* different from FS2. For one, the Jet controls are super sensitive, and my first dozen or so efforts got me up but tumbling. I crashed a lot. However, once I managed to stabilize, I enjoyed the ride. Boy, you sure go up in a hurry—and down!

Jet's main attraction, as I see it, is the ability to actually dogfight with another ST user through a modem or null-modem link. This is great! I only wish they had a version of FS2 that allowed dogfights, not merely mutual flight. I'm a tad more comfortable in the slower prop planes than a missile-laden F16. However, the latter is preferable over gridlocked highways at rush hour.

So far, I haven't managed to land in one piece, a minor drawback (what's $30 million worth of aircraft anyways). And forget the carrier scenarios. I have to practice a lot more before I can land on one of those rolling, shifting demons!

And if you were wondering, Jet also works with the scenery disks, so you can *really* have dogfights over Broadway!

Have I ever mentioned my problem with the ST keyboard?

My AT and laptop both have keys that measure 12 cm wide by 14 cm high, with a 6-cm gap between keys (almost 7 cm on the AT). These are typical measurements for the PC/MS-DOS keyboards, based on the ergonomics of the IBM Selectric typewriter. My Smith Corona electric has keys 13 cm square, with a 7-cm gap.

All of these keyboards are easy to use, with keys well-spaced and small enough so that big fingers don't bang several keys at once.

The ST keys measure 15 cm square, with only a 3-cm gap between them. That's a key gap 25% larger with 50% less space between it and the next key. *Bump, slam.* My fingers *always* hit more than one key on it. It's a touch typists' hell.

I make so many typos on the ST that I stopped using it for writing. My messages on DELPHI are riddled with typos, because I still use my ST for telecommunications. But I don't make a tenth of the mistakes on any other keyboard that I make on the ST. And I'm not alone. I've seen this thread on CompuServe and know others share the same problem.

And then there's the tactile response—the springiness of the keys. The ST keyboard is mushy, and although the Mega keys seem much better (I haven't measured the Mega key size), it's still sloppy. I don't care for the mechanical click some PC keyboards have—they make sounds like rats running on a linoleum floor. But I do like the solid resistance they offer.

Why doesn't anyone offer a new ST keyboard? I've seen ads for a revamp kit that adds the spring, but no one offers redesigned keys in the proper shape. I think the keyboard alone is a significant stumbling block to getting the ST accepted in the business or professional world.

Okay, let's talk about business. Do I hear some groans? Come on, this is important stuff.

It doesn't mean just the Fortune 1000 as one reader suggested. Business means any commercial enterprise or professional use, be it a dentist's office, a retail store, an accountant, stockbroker or a writer.

Hands up anyone who can tell me why the PC/MS-DOS systems are so successful in the business marketplace. What do they have that the ST doesn't? Yes, I know, software, but I've ridden roughshod over that path before. What else? Okay, you in the back.

Right, they share a common bus architecture. With some exceptions and caveats, you can buy a card for a PC/clone and move it upwards into an XT, AT and 386 machine. Sometimes you have to buy for your machine type, but even so, there are a lot of cards from which to choose. And the machines have lots of slots. Expansion is a breeze.

The Mega ST has *one* slot. And that slot isn't anything even akin to anyone else's standard. Has anyone seen the tech specs for the slot? I haven't.

Right now, I can buy a PC/MS-DOS machine, get a super high-res card and color monitor and work in at least 1,280 by 800-pixel resolution, maybe go up to 1,600 by 1,200, 1,200 by 1,664 or higher. Even with the Mega, I'm limited to 640 by 400, monochrome—that's not even as high as the Hercules Graphics standard on the PC.

My own AT sports a 2-mb RAM card, a multi-graphics card (which can push my multi-sync monitor to about 800 by 500 resolution) and a mouse card. I'm looking into getting a fax card, another 2 mb of RAM, an optical scanner and a hardcard (a hard disk on a card), all available at any run-of-the-mill computer store. I can also get network links, printer buffers, telephone answering machines, additional drive controllers, bar code readers, tape backups, modems, laser-printer drivers (one that even gives HP Laserjets a postscript capability!) and other cards.

What cards can I add to a Mega? Time's up. Have you seen any?

Cards aren't merely the method for expansion, they're also the means to customize your system to suit your own needs. That's what open architecture is all about. One slot doesn't make it as far as "open" is concerned.

Of course, there's Apple selling its slotless Macintoshes in their various configurations.

One of the reasons the Mac has only achieved a lukewarm success (compared to the PC system) in the business world is that most of the variations have no expansion or customizing capability, while the PC has oodles of it. Only the SE sports a single slot. *Wow.* However, Apple has pounded the pavement long enough and got its foot in enough doors that it's managed to sell lots of Macs to people easily convinced they'll never want to expand or change or customize anything. Hardware off the cookie cutter.

Apple did that "big brother" ad on TV, remember? You can fool some of the people all of the time. . . .

(There's also UNIX, which despite the fervor of its proselytes, isn't going anywhere fast. Let's forget about them for a while, shall we? Say perhaps for the rest of the Age of Mammals.)

Where's the ST in all of this? A game-machine? Fie on you! Stand in the back of the class with your nose to the wall.

Not that there's anything wrong with games or entertainment. I think the computer's potential for entertainment is highly underrated by the pundits of the computer world. It's no worse than flaking out in front of TV sitcoms and can be a lot more intellectually stimulating. I myself often take a load off the grey matter by spending a few hours playing games on my ST.

What's wrong with the game-machine image? Well, for one, it won't sell many systems. And that's what the bottom line is all about.

And guess what? Computers sell better in the business market than in the home market. Around 1984, that was a hard lesson for many manufacturers to learn. I wonder if Atari has learned it yet? ■

Ian Chadwick is a Toronto-based freelance writer and editor who, by the time you read this, should be with his wife Susan, soaking up some rays on the beaches of Mexico, drinking cold Coronas and eating spicy tacos.

A 16-BIT CARTRIDGE PORT INTERFACE

BY RANDY CONSTAN

On the side of your Atari ST lies one of its most powerful hidden features: the cartridge port! Through this subtle interface lies the potential for an incredible array of untapped resources. Consider 15 address lines, 16 data lines and four control lines are brought out. Any device connected to the cartridge port can be directly controlled by the mighty 68000 processor with no "middle man." Theoretically, over 256K bytes/second could easily be transferred. With the proper circuitry, over one million discrete processes could be individually monitored and controlled through this port. And, with the proper software added, an entire robot-controlled assembly line could be run entirely with the ST!

Consider our problem: One of my current projects involves the design of a new electronic musical instrument. The sounds produced by the instrument, however, won't be new at all; they will be "sampled" sound. Sampling is a technique in which a sound wave is converted into numeric computer data, which can later be reconstructed into sound. If you've heard a CD recording through a quality stereo system recently, you know what the state of the art in this technology sounds like. Making a CD-quality recording, however, is no easy task. It involves doing over 40,000 analog-to-digital conversions per second, each to an accuracy of 16 bits. Several articles could easily be written on this subject alone, but whatever circuitry is used, the questions of speed, memory and control immediately come to mind. How can we precisely control the timing of these critical devices and instantly save the hundreds of thousands of data words that are pouring in at a rate of 44,000 per second?

Warning: *The following article is for ST owners with a large amount of electronics experience. If you lack this experience, do not attempt to build this project for use with your ST. You may damage your computer.*

ST to the Rescue

A general-purpose interface device for the ST was in order. The 68000 processor easily surpassed the speed and memory requirements, and the cartridge port provides neat and eloquent access to this power. In this article, we'll discuss the design and construction of a versatile cartridge-port interface, which will provide 16-bit, latched data, read-and-write capability, complete with status and handshaking control. Before you go running for your soldering iron, however, a word of warning. Building this interface does require a lot of tight wiring on a relatively small circuit board. Wiring errors, aside from causing a great deal of frustration, could damage your ST. I have found that the cartridge port will tolerate some momentary shorting of address, data and control busses, and even an occasional power-supply short, with seldom more than a system crash as a result. *But don't count on it!* Be very careful wiring and rechecking. Let's just say that if you're contemplating this as your first electronic project, skip over this article!

Cartridge Basics

All 16 data lines and address lines, A1 through A15, are brought out to the ST cartridge port. Since the 68000 can address two bytes (16 bits) of data at a time, there is no address bit zero. Instead, two control signals, UDS (upper data strobe) and LDS (lower data strobe), are present. When the processor wants a word (16 bits) of data, the proper address is asserted on the address lines and both UDS and LDS are simultaneously brought low (Logic 0). If only a single byte (eight bits) is desired, the same process occurs except that only one of these signals will be strobed: UDS for even bytes, and LDS for odd. So, it is possible to create an A0 address line using UDS and LDS. Moreover, it is possible to have several different operations based on the various combinations of these signals.

With 16 address lines, we could address up to 64K in our circuits. The 68000 actually has 24 address lines however, and can address up to 16 megabytes! So the ST cartridge port has two more control signals available, and our circuit will know when the port, and not some other 64K bank of memory is being addressed. These are ROM3 and ROM4. Rather than bring out all 24 address lines, two possible 64K cartridge-bank address ranges are sensed by the ST's "glue" chip. When any address in the range of $FB0000 through $FBFFFF is asserted, ROM3 is strobed to logic zero. The other cartridge bank occupies the range of $FA0000 through $FAFFFF, and any address in this range will cause the ROM4 control line to be strobed. Note that numerically, it would seem as if this were an error. However, the text is indeed correct: ROM4 is for the lower $FAxxxx range and ROM3 for $FBxxxx. So, by using these control lines, we can easily prevent our interface circuit from interfering with any other memory operation that does not involve the port.

Now for another "zinger." Since the cartridge port was originally intended to (you guessed it!) read cartridges which are normally just ROMs, Atari in its wisdom saved a few bucks by making the data lines (D0 through D15) unidirectional. In other words, data can be read in, but not written out on these lines. Since we want to be able to transfer data in both directions through our interface, we need to overcome this problem. Notice though, that the address lines have the exact opposite property: Data ("address" data, that is) can only be written out of the port. So if, for example, we wanted to write the word $E812 through the port, we could make our circuitry respond to the ROM4 address range, and simply do:

```
def seg = &hfa0000
a = peek (&hE812)
```

in BASIC, or:

```
move.b $FAE812,d0
```

in assembly.

What we've done here is fool the system. The ST thinks we're reading a byte from location $FAE812, but what's actually happening is that the lower portion of the address data is being written to the port. (Note that the data returned is irrelevant.) So, since we have two memory-bank control lines available (ROM3 and ROM4), I decided to simply make two independent 16-bit interfaces on the one board: one for reading data, the other for writing. Also, the UDS and LDS control lines are used to provide a "get status" function, in addition to read and write.

The Interface Circuit

Take a look at the schematic in Figure 2. The two upper 74LS374 chips each contain an 8-bit latch. Whatever data is present on the latch inputs will be saved when Pin 11 goes high, and will remain available for output until Pin 11 cycles low and high again. At this point the new data will replace the old. All 15 available address pins (A1 through A15) are connected to these inputs, and since one more bit is desired (A0), UDS is connected as well.

The control pins, UDS, LDS, ROM4 and ROM3 are all active low. This means that they normally sit high (+5 volts) until activated, when they momentarily become low (0 volts). The ROM4 line is inverted and passed to Pin 11 of both chips, so that addressing the $FAxxxx range will cause the lower 16 bits of address to be latched. As it turns out, the ROM control lines change last, after the address, data and UDS/LDS, making them perfect for this critical timing.

In our previous example, the bit pattern corresponding to $E812 would now be available at the output of our interface and would remain available indefinitely. The inverted ROM4 signal is also used to set a flip-flop, which can be read by the ST or an external device as an output data pending (ODP) signal. This flip-flop can also be reset by an external device by pulling the acknowledge (ACK) line low. The point here is to accomplish "handshaking." An external device such as a D/A converter can be set up to wait for the ODP signal before accepting new data, and after the data has been transferred, can cause the flip-flop to reset. In the meantime, a software loop can poll this flip-flop, until reset has occurred, then send new data to the port.

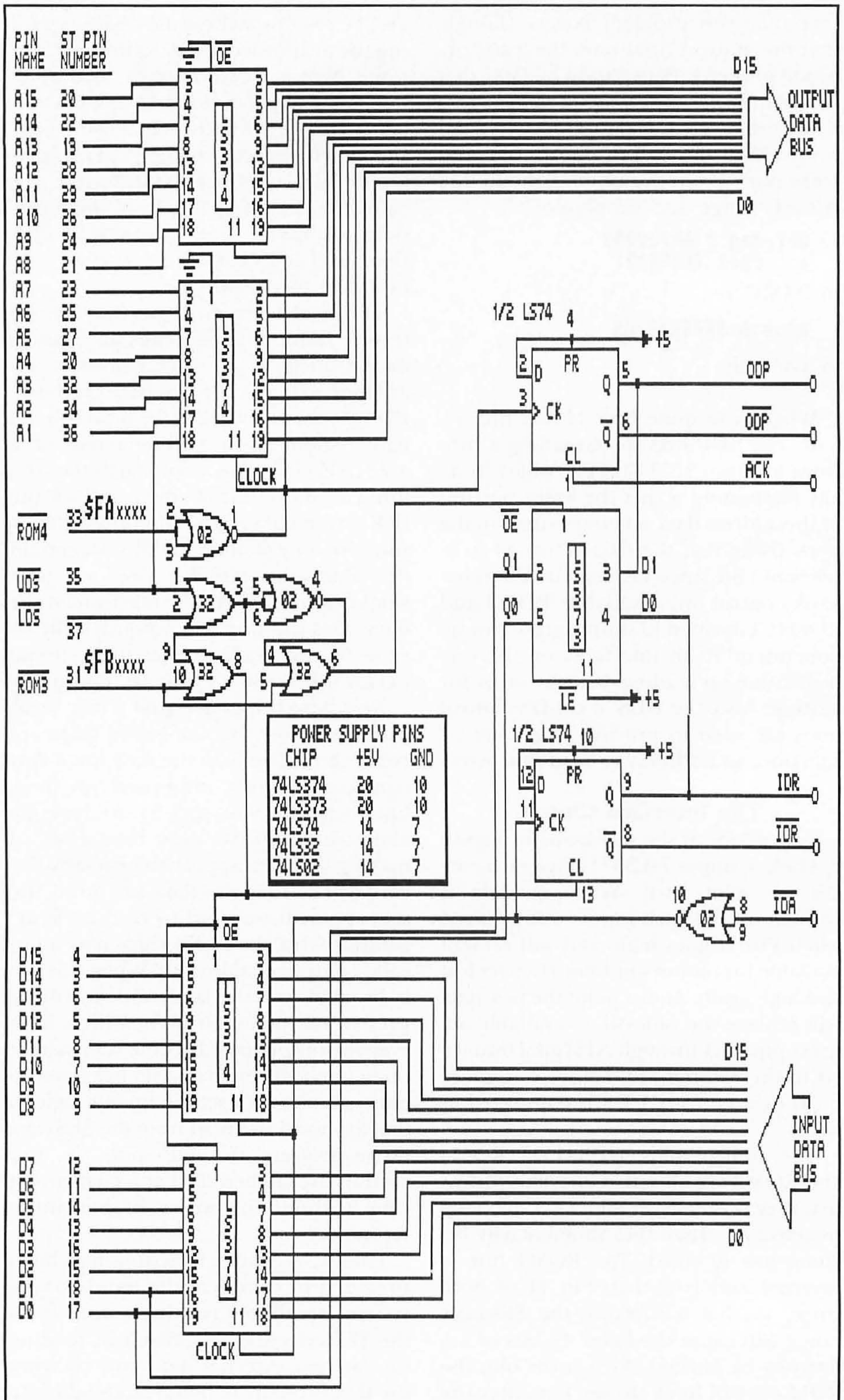
The read logic works in a somewhat reverse fashion. In this case, an external device such as an A/D converter (or another ST!) would place data on the 16 INPUTs provided. Then, the device would have to momentarily pull input data available (IDA) low. This would latch the data into two more LS374 chips, and set the IDR (input data ready) flip-flop. Again, a software loop polling the flip-flop would determine that IDR has been set, so it would "know" when it's time to read the data. This flip-flop will automatically be reset by the read, so that the external device will know it's time to send more.

Now here things get a bit tricky (bear with me!). Note that the LS374 chips are connected directly to the data lines. Any data continuously impressed on these lines would cause the ST to lock up almost instantly. We must have a way of making this data appear to be totally disconnected from the data bus until the very moment we need to read it. To accomplish this, the LS374 chips have a pin called output enable (OE). When this pin is brought low, the latched data will be present on the output. When high, however, the output pins become transparent (high impedance), so that no bus interference can occur. By setting up our logic so that any word size read from the $FBxxxx range triggers the OE pin, we can read in the latched data at a convenient time without the worry of bus interference.

Finally, a Status function has been provided to complete the handshaking system. Recall that reading a word from the $FBxxxx area will result in reading the latched external data and clearing the IDA flip-flop. If, however, an odd byte is read from the $FBxxxx range, the Q outputs of both flip-flops will be read into D0 and D1, without clearing either one. This is accomplished by activating the OE pin of the LS373 during a "byte" operation.

Construction

So, enough talk—let's get busy! The first thing we'll need is an electronic pro-



With 16 address lines, we could address up to 64K in our circuits. The 68000 actually has 24 address lines however, and can address up to 16 megabytes!

totype board, with pin spacing to accommodate the 40-pin port.

Until recently, you would have to build and etch this board from scratch. Now however, there are at least a few vendors for this item. I obtained some very nicely constructed boards for only $10 each from Douglas Electronics Inc. in California. (See Figure 1 and the address in the parts list.)

Second, you'll need the eight ICs shown in the parts list, together with three 14-pin and five 20-pin wire-wrap sockets. A 40-pin termination header is also required on both the internal and external side of the interface board to facilitate connections and testing. Finally, you'll also need at least three 0.1 microfarad, ceramic disk capacitors to be connected across the power supply rails at various points on the board.

The IC sockets, termination headers and capacitors can all be obtained at your local Radio Shack store. For the ICs however, you may have to shop around at an electronic specialty shop. Also if you've never done wire wrapping before, now is a good time to start. While I did quite a bit of point-to-point soldering on my prototype, I found the process to be quite troublesome. The connections are so close to each other that hairline solder bridges between soldered points are almost inevitable. If I had it to do over, I would have wire wrapped the entire project.

Figure 2 shows both sides of my completed board. The plastic cover on the "wired" side helps protect the unit, and helps it to stay nearly level when plugged into the port. The board layout is by no means critical, so feel free to use your own judgment here. I do suggest that you obtain at least three colors of wrap wire, so that output data, input data and steering-logic wiring can be more easily distinguished.

The Douglas board already has several supply and ground conductors etched on the board, so that power-supply wiring to each chip requires a minimum of effort. The supply lines to your wire-wrap IC sockets, the three capacitors, and the pin headers will have to be carefully soldered in. The internal (ST) side header must be soldered with the pins facing down, since that's where all the wiring will be done.

The rest of the project can then be completed exclusively with wire wrapping. With the prototype board face up (the side with the manufacturer's name), Pin 1 is the at the top rear of the board as shown in Figure 1. You can refer to

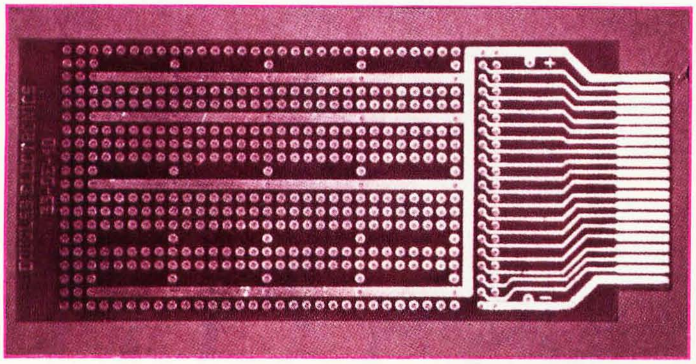


FIGURE 1

The Douglas prototype board. Pin 1 is marked for reference.

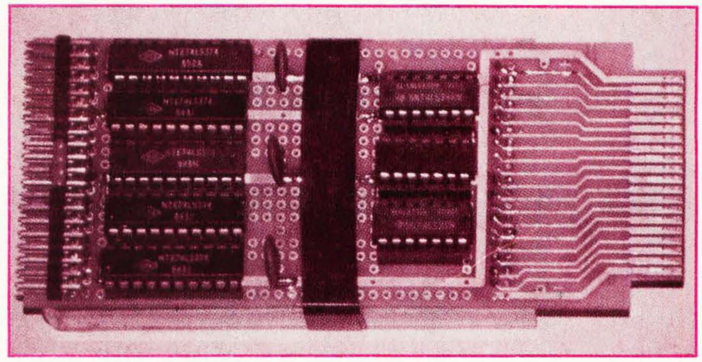


FIGURE 2

The completed cartridge port interface. Note that component layout is not critical.

your ST owner's manual (page 77) for identification of all the cartridge port pins. These are also listed on the left side of the schematic.

Carefully solder in the IC sockets (supply pins only), along with the headers and capacitors. The small chart in the center of the schematic shows the power pin locations for all the ICs used. Check the wiring with a multimeter until you're sure no shorts or wiring errors are present. Power supply errors are among the most unforgiving, so be forewarned!

When this step is completed, proceed with the rest of the wiring. Again, no special layout is required, but remember that neatness goes a long way in avoiding headaches when the time comes for correcting errors or modifying the circuit. It would also be advisable to draw a map showing how you intend to wire the external side pin header. As a final step, connect two eight-inch lengths of insulated, flexible wire to a convenient ground (zero volts) on the board, and strip off no more than ⅛ inch of insulation at the unconnected ends. These will be used for testing the interface.

Testing the Unit

The first test of the unit should be done with none of the ICs plugged in! Carefully plug the board into the cartridge port, and power up the ST with a formatted, but blank, disk. If you don't see the desktop come up within a few seconds, turn off the power immediately and re-check your wiring. Again, an inexpensive multimeter is helpful in determining whether address/data lines have been shorted. Since no chips have been inserted, infinite resistance should be present across any pair of these pins. If the desktop comes up normally, check for power on all the IC sockets.

Next, power down the ST, remove the interface and insert the ICs into their proper locations. Repeat the above "smoke test" to make sure the desktop appears with the complete interface inserted. If all goes well, it's time to test the unit for proper operation under software control.

Connect one of the temporary ground leads to the negative test probe of a multimeter and set the meter to measure five volts. Then, power up the computer and boot-up good ol' ST BASIC. All right, I know, I shouldn't have used the word "good," but for simplicity's sake, it would be nice if we could use the port from BASIC. We've already seen how we can test the Write operation. In direct mode (command mode), type:

```
def seg = &HFA0000
```

This does two things. First, it sets the base memory location for future peeks and pokes. Second, when DEF SEG is equal to anything above zero, it causes all peeks and pokes to return a byte value.

Now type:

```
a = peek(0)
```

Next, using the positive probe of the voltmeter, check that all 16 data outputs on the interface are at near zero volts. Then, type:

```
a = peek(65535)
```

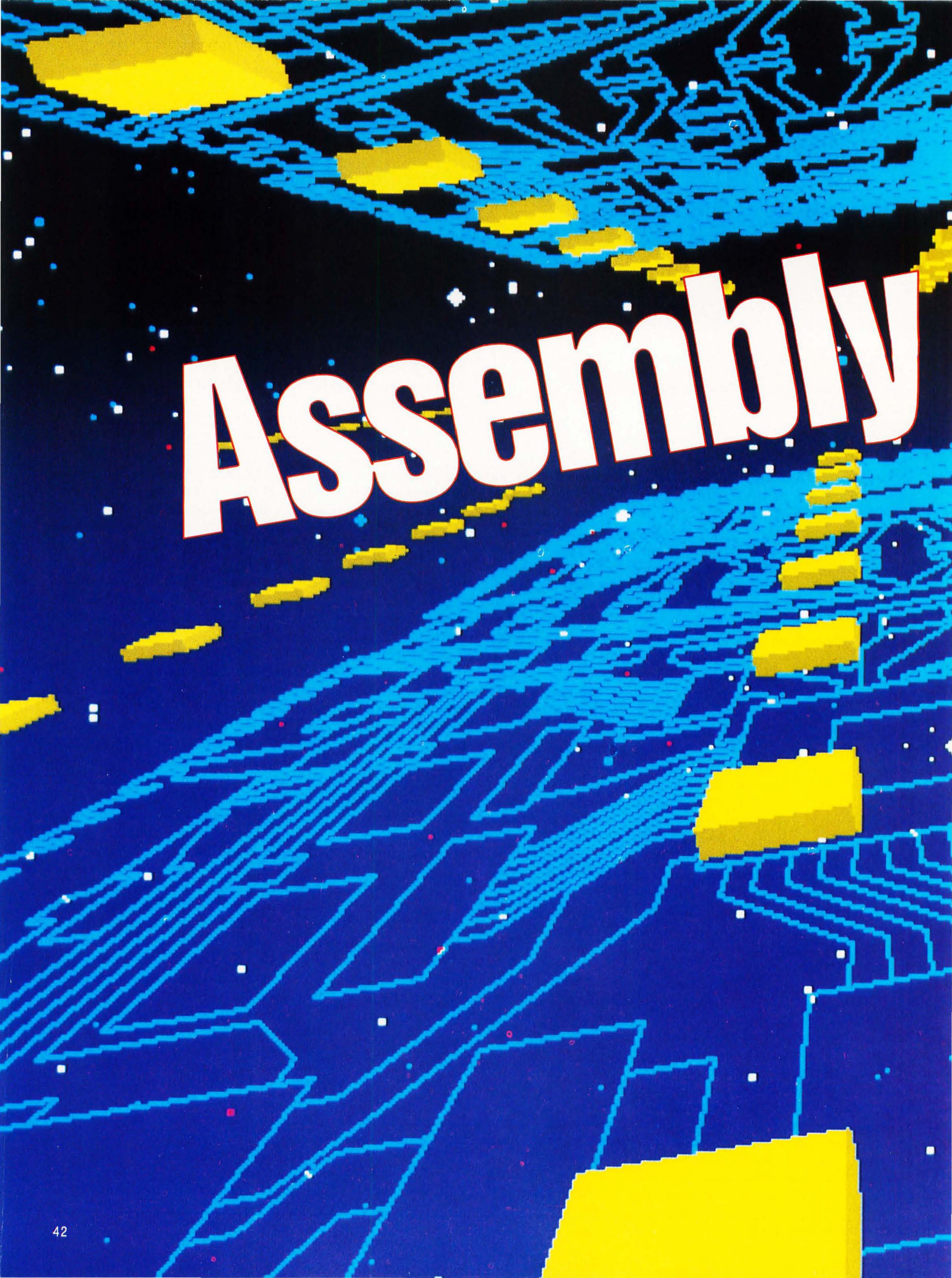
Using the same procedure, you should now find that all the data output terminals are at approximately five volts. If this does not work as expected, you still have some wire checking to do! By peeking the number 65535, we have effectively written a binary 1111111111111111 on the address bus, and so all outputs should be high. Now recall that the second ROM bank (ROM3) in the cartridge address space is used for reading data in our interface design. But, here there is a serious problem! Unfortunately, when BASIC performs a PEEK (or POKE), only one byte at a time is addressed. This is true even though a BASIC PEEK command will return a two-byte number if DEF SEG = 0. BASIC simply peeks two bytes, one at a time.

There's no easy way to test the Read command of our interface, since the byte-oriented BASIC will never cause UDS and LDS to simultaneously strobe. Arrrgghh! Then I discovered the Call command which allows BASIC to access a machine-language program. (Special thanks to Charlie Bachand for showing me that the Call command is much simpler to use than the manual would have us believe!)

So if you've gotten this far, boot-up BASIC, type in Listing 1 and save a copy on disk. This short program uses a machine-language patch that allows the Write, Read and Status functions to be tested and will make our experimentation a whole lot easier. For those interested, the source code for the patch is given in Listing 2, which you need not type in.

When the program is run, a short menu is displayed with the options Write,

(to page 76)



Assembly

Line

File handling, Part 1

by Douglas Weir

File operations are the lifeblood of "real" computing. Files are where a program's input often comes from, and the precious output of most programs is usually saved in files. Not to mention the fact that the program itself exists as a disk file before it is loaded by the operating system and run (unless perhaps it's written in an interpreted language like BASIC).

So far we haven't done anything with files, but that will change with this month's episode. After we're finished with this miniseries, we'll know how to do all the usual file operations in assembly language: opening, closing, reading and writing files, as well as using disk directories.

GEMDOS treats disk I/O, as far as it can, as just another kind of "plain" I/O. For example, the function we'll be using to read from a disk file can also be used to read from the keyboard. As a result, the basic disk operations are pretty simple from the programmer's point of view. Programming for disk input can even be easier than keyboard I/O, since there are some things that can happen at the keyboard in an "interactive" environment that can be ignored when reading from a disk.

The one great danger with disk I/O is that, if you do something wrong, you can accidentally erase or overwrite valuable files—including the source code for the program you're working on! So it's sensible to take some extra precautions. If you have two disk drives, test your I/O on a disk in the drive that *doesn't* contain your program disk. If you have only one drive, make sure the program disk contains *only* the program and the necessary data files, *and* keep a constantly updated backup of the program file(s) on a separate disk. Finally, if you have a hard disk, I strongly recommend that you do your I/O testing on a disk in the floppy drive, *not* on the hard disk. It goes without saying that, within your program, you should be extra careful to make sure that the I/O drive is (and remains) the one you want it to be, until you're sure all the basic things are working the way they should.

In any case, we shouldn't be able to get into too much trouble as long as we're not writing data to the disk. For that reason, we'll be only opening, reading and closing a disk file this time. Next time we'll begin writing data.

This month's program will, when run, open a file (any file), read it, display its contents on the screen in a special format, close the file and terminate. We'll learn a couple of new 68000 instructions as well as a handy feature of GEMDOS. First, the handy feature.

The Command Line

In previous programs we've used keyboard input to get necessary information from the user about various tasks we want to perform. There's nothing wrong with this, except that it can be a bit tedious to write a keyboard input routine just to get, say, a single vital piece of data (and no more) from the user. This month we find ourselves in just such a predicament. We need to know the name of the file the user wants to display, but that's all we need to know. Isn't there an easier way to pass simple parameters from the user of a program to the program itself?

As you've probably already guessed, there is. Instead of asking for a filename after the program has begun, we'll just require the user to type the filename after the name of the program (with at least one space between the two), on the same line, before hitting Return. For example:

```
dump a:glob.txt
```

Type that, hit Return, and *dump* will read and display the file *glob.txt* on drive A, if such a file exists on that drive. How does the program find out what was typed after its name when it was called? Read on.

Whenever GEMDOS loads a program into memory, it sets up a special area for that program called the "base page." The base page, according to an old piece of Digital Research documentation that I'm looking at "is a 256-byte data structure that defines a program's operating environment." In other words, it contains useful facts about the program such as, for example, how long its data area is, how long its program area is and so on. The second half of the base page contains what we're interested in right now: all the characters typed by the user after the program name and before Return, up to a total of 126 bytes. The entire string typed by the user, including the program name, is called the "command line"; anything typed after the program name (except for the Return) is called the "command tail."

A program running under GEMDOS is a lot like a subroutine. It was "called" by GEMDOS and, when it terminates, execution will return to GEMDOS. Its stack configuration is also similar to that of a subroutine. When it begins, the stack pointer is pointing to a return address within GEMDOS's command processor. Just above this return address is the address of the program's base page.

So the first thing our program does is read the base page address from the stack, just as any subroutine would read parameters that had been passed to it on the stack. The command tail string begins 128 bytes from the beginning of the base page, so we load that address into register *a2*. The command tail has a format somewhat like a BASIC string: Its first byte contains, not a character, but rather a number indicating the length of the string itself, which begins at the next byte. So we load this count into a data register, at the same time incrementing *a2* to point to the first character. Now we're ready to begin.

Open, Says Me

The command tail (which we are assuming contains only a valid filename) and its length are passed to the subroutine *open_file*. The first order of business now is to copy the string to a location within our own data area (labelled *filename* in the data segment). If we decide to fool around with the string's contents, we don't want to do this in the base page area, where all sorts of undefined havoc might occur if things went wrong.

There's one bit of fooling around you should probably always do: append a null to the end of the string. Depending on whose documentation you read, GEMDOS does or does not terminate the base-page copy with a null. I've never bothered to find out which is true (besides, things might change in a later version); I append it myself and that way I know it's there.

By the way, it would seem to be a good idea, when copying the command tail, to skip over any leading spaces in the string. However, it doesn't seem to make any difference to GEMDOS whether there are spaces in front of the filename string or not, so I skipped this step.

Now all we have to do is open the file. We could do some more error-checking on the filename string, but it isn't really necessary. If the filename for some reason isn't valid, then GEMDOS won't open the file, because it won't exist, and we'll find that out as soon as we try to open it. So let's.

The 3D Function

Of course, *all* the GEMDOS functions are, in a way, multi-dimensional. In this case, though, "3D" refers to the hex code for the GEMDOS file-open function.

The function takes three parameters, passed on the stack, as usual. First comes a number (word-size) from 0 to 2. This code tells GEMDOS whether you want to open the file to read only (0), write only (1), or to read and write (2). We pass a 0 to indicate that we want only to read from the file.

Next comes the filename string itself, which *must* be terminated by a null. We pass the address of our null-terminated copy here. Last comes the function code itself, $3d (61 decimal, if you like things that way).

GEMDOS will now try to open the file. If it's successful, a "file handle" will be returned in register *d0*; otherwise, an error code will be returned. The file handle is simply a number that GEMDOS uses to identify the file once it's been opened. Handles can be used for other I/O devices too: 0 and 1 refer to the keyboard and screen respectively (so, apparently, do 4 and 5), 2 refers to the RS-232 port, and 3 to the printer port.

Assuming that there is a formatted disk in the drive you want to access, an error return can only mean that the filename is invalid: Either there's something syntactically wrong with the name as originally typed by the user, or the name is correct but the file doesn't exist.

GEMDOS error codes are always negative numbers, so all we have to do is test *d0* and abort if it contains a negative value. Otherwise, we save the returned file handle (in an area labelled *handle*), and return to our caller. Note that our subroutine *open_file* returns to its caller in *d0* the same code (or handle) that was returned to it by the GEMDOS function. In order to do this, we must remember to *not* save *d0* at the beginning of the subroutine, even though we use it, or "restore" it at the end.

Meanwhile, Back At the Branch...

As usual, after we return to the instruction immediately after the "branch to subroutine" that called the subroutine, we adjust the stack to compensate for the extra parameters we pushed. Now we test *d0* to see if *open_file* did in fact open a file. If *d0* contains a positive number, then this must be a file handle and we can safely proceed. Otherwise, no file was opened and we must terminate the program.

Assuming everything went well, we now branch to the subroutine *display_file*, which does the real work of this program.

Translating Codes

"Dump" is a very simple program. It reads from its input file one byte at a time, and writes the value of each byte (one per line!) to the screen. Only ASCII codes (those between 32 and 127 inclusive) can be printed "raw," so some translation has to be done to the other values.

The values 0 through 31 are often called "control codes." Some of them are very familiar (such as 0, null, or 13, carriage return); others are quite obscure (my personal favorite is 21, "negative acknowledge"). Control codes were first used, as I understand it, on machines such as teletypes. They were adequate for managing the rudimentary formatting capabilities available at that time, as well as communicating "overhead" information about data transfers between machines (for example, end-of-text, start-of-message and so on). Nowadays host-terminal communications are much more complicated, and elaborate systems such as the ANSI escape sequences have been evolved to handle them—note, however, that "Escape" itself is a control code (27).

Most of the control codes are little used, but some of them are used all the time. Our program, whenever it reads a control code, translates it into its standard two or three-character abbreviation, which it then prints to the screen. The strings containing these abbreviations are found in the *ctrl* table in the data segment. There you can also find, if you're interested, what all those puzzling abbreviations actually *mean*.

Of course, like beauty, control codes exist only in the eyes of the beholder. A programmer can choose to use these values for something else and, as long as he or she is consistent about it (and the I/O routines don't interfere), there will be no problem. Or, consider the values you'd expect to find in a program file (i.e., a file containing "runnable" machine code). Values from 0 through 31 will be treated by our program as control codes, which they almost certainly won't be: they'll simply be machine instructions (or byte-sized parts of machine instructions) that happen to have values within this range. In general, control codes occur with regularity only in text files.

The values 128 through 255 can be used for all sorts of things, depending on the computer. Often "extensions" to a computer's ASCII set are implemented here: such things as "graphics characters," or math symbols, and so on. Our program simply translates all such values into two-digit hex numbers and prints them to the screen.

Reading a File

Now let's see how it's done. Most of *display_file* consists of a large loop (its top is at *d_go*). At each iteration of the loop one byte is read from the file selected, translated (if necessary), and displayed. This continues until the end of the file is reached. The user can stop the output at any time by pressing the keys Control-S (control codes again! "S" is the 19th letter of the alphabet, so this is actually code 19: "DC3"); press Control-Q and the display will resume. Pressing Control-C will abort the program.

GEMDOS function $3F is used to read from a file. It takes four parameters. First comes the address of a memory area (within your program) into which the data read is to be copied. This area should be large enough to hold the largest amount of data you plan to read at one time; otherwise, data after this location could be overwritten by data read from the file.

The second parameter is a byte count. Although we're only reading one byte at a time here, you can use the function to read much larger amounts of data on one call. Actually, our method isn't quite as inefficient as it looks: GEMDOS has an internal buffer which it always tries to fill on a read operation. Subsequent calls to the function simply return data from the buffer until it is exhausted, when another read is performed, and so on. The GEMDOS buffer seems to be about 512 bytes in size.

The third parameter (word-sized) is the file handle; the fourth (also word-sized) is the function code, $3F (63 decimal).

The GEMDOS Read function returns a value in register *d0*. If the value is negative, then some sort of error has occurred (we shouldn't have to worry about that right now). Otherwise, *d0* contains the number of bytes read. This should be zero (0) when end-of-file is reached, and that does seem to be the case when reading single bytes. However, the Abacus book *Atari ST Internals* warns that Read *never* detects end-of-file, and that the programmer should get the file's size from the directory in order to find out how far to read. Single-byte reads do seem to pick up end-of-file, so we'll keep our fingers crossed until next time, when directories is one of the things we'll learn more about.

I'll save the rest of the detailed explanation until next time, but I would like to mention the two new instructions used this time, as well as explain a new use for a third.

The *rol* (Rotate Left) instruction is something of a variation on the logical shift instruction we learned a couple of installments ago. The shift instructions, you'll remember, do just that: shift the bit-values in a data register the indicated number of bits left or right. Values shifted out of a register are lost. The rotate instructions are a bit different. The value in the specified register is shifted right or left as before; but the bit-values shifted out one end of a register are inserted back into the register at its other end—nothing is lost. You can specify byte-, word- or longword-size for a rotate. Suppose the low byte of register *d0* contained the following binary value:

```
10000001
```

After the instruction *rol.b #1, d0* is executed, *d0*'s low byte will contain the following value:

```
00000011
```

The leftmost 1 was rotated leftward out of the byte, and back into the right end of the byte. The number of bits to rotate is indicated the same as with the shift instructions: an immediate value can be used to specify up to eight rotates at one time, otherwise a data register (in the source operand field) contains a number indicating the number of rotates to perform. In our program, register *d3* (its entire 32 bits) is rotated left four bits.

The shifts and rotates between the labels *do_hex* and *do_ctrl* are used to get at the two low half-bytes (or "nybbles") in *d3* separately in order to translate each into a hex digit.

The instruction *pea* (Push Effective Address) works just the same as *lea* (Load Effective Address), which we learned about last time; the only difference is that, after generating the address value specified, *pea* pushes the result on the stack instead of loading it into a register.

Finally, note how the *lsl* instruction is used under *do_ctrl* to multiply the contents of a register by four. Just as, in decimal numbers, moving a digit one place to the left is equivalent to multiplying it by ten, moving a binary digit one place to the left is the same as multiplying it by two. Do this twice and you've multiplied by four. This little trick can be used whenever you want to multiply a positive integer by a power of two, and it can be much faster than using *mul*.

That's all for this time. Type in the program, assemble and run it. Next time I'll explain the rest of *display_file* in more detail, and we'll learn some more file operations. Until then, you might think about making some obvious improvements to the program as it now stands. For example, how would you go about printing, say, four bytes (rather than one) to a line? ■

Douglas Weir, who was once a technical editor for ST-LOG, *is now employed as a technical writer for Wang Computers in Boston. Besides programming, he enjoys classical music and good books.*

ASSEMBLY LINE
LISTING 1

ASSEMBLY

```
******************************************************************
*
* Assembly Line: File Handling part 1-- dump.s
* by Douglas Weir
*
* Copyright 1988 ST-Log
*
*
******************************************************************

start:
        movea.l 4(a7),a5        get base page address
        lea.l   $80(a5),a2      point to command line
        clr.l   d0              clear for byte value
        move.b  (a2)+,d0        get byte count

        move.l  a2,-(sp)        push filename string address
        move.w  d0,-(sp)        push length
        bsr     open_file       open file
        addq.l  #6,sp           pop args
        tst.w   d0              error?
        bmi.s   abort           if so

        bsr     display_file    read file

        move.w  d0,-(sp)        push file handle
        move.w  #$3e,-(sp)      code==close file
        trap    #1              do it
        addq.l  #4,sp           pop args
out:
        move.w  #0,-(sp)        code==terminate
        trap    #1              do it
abort:
        move.l  #abort_msg,-(sp)        error message string
        move.w  #$09,-(sp)      code==print string
        trap    #1              do it
        addq.l  #6,sp           pop args
        bra     out             and leave

*****
*
* open_file-- get filename, open file
*
* at entry:
*       (a6) + 8  -> length of filename string
*       (a6) + 10 -> address of filename string
*
* at exit:
*       d0.w contains file handle or neg number if error.
*       all other registers preserved.
*
*****
open_file:
        link    a6,#0           frame pointer
        movem.l a0-a2/d1-d2,-(sp)

        movea.l 10(a6),a0       address of string
        move.w  8(a6),d0        length of it
        movea.l #filename,a1    address of filename storage area
        bra     s_test          now start
scan:
        move.b  (a0)+,(a1)+     copy a byte
s_test: dbra    d0,scan         all of them
        move.b  #0,(a1)         null terminator

        move.w  #0,-(sp)        mode==read only
        move.l  #filename,-(sp) filename string
        move.w  #$3d,-(sp)      code==open file
        trap    #1              do it
        addq.l  #8,sp           pop args
        tst.w   d0              error?
        bmi.s   open_out        if so
        move.w  d0,handle       else save file handle

open_out:
        movem.l (sp)+,a0-a2/d1-d2
        unlk    a6
        rts
```

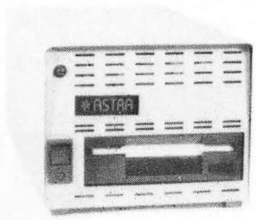
Assembly Line

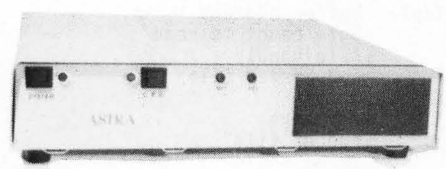
Astra News

2500-L So. Fairview, Santa Ana, CA. 92704 (714) 549-2141

NOW THREE SERIES OF HARD DRIVES !

System HD+ Series | Home / Office Series | Studio Series





THE "TANK"

The System HD+ has been recognized as extremely tough and reliable by hundreds of Atari ST users. It is built to exacting standards and scrupulously tested. Astra Systems is so confident of the quality of this unit we offer a limited one year warranty.

Originally offered as a 20 Megabyte hard drive with built-in 3 1/2" floppy, it now is also available in 30 and 40 Megabyte units with floppy.

Supplied with formatting, partitioning software, and backup program.

The floppy used in this unit is a precision drive with direct drive motor, and can be formatted with high density format programs.

THE EXPANDER

Internally expandable hard drives come either with or without precision 3 1/2" floppy drives.

Four AC outlets with full three line surge suppression are installed at the rear of the unit. One of these controls the CPU and the others are available for monitor, printer, etc. Two push button switches on the front control the CPU independently of other peripherals. EMI and RFI filtration is included.

Twenty, thirty, and forty megabyte units expandable to 120 megs.

All necessary hardware is already installed in original unit so addition of upgrade kits is fast and easy.

RM 60/120

The RM 60 rack mount hard drive for the MIDI musician fits both permanent and portable racks.

Expandable from 60 to 120 Megabytes internally with the addition of the +60 kit. Or purchase it complete in the model RM 60/120.

Astra hard drives for the Atari MIDI musician have become the standard for the industry, and are being used by top professional groups worldwide. Our power supply is equipped for 120 and 240 volt operation by merely moving one wire. This makes performing in UK and Europe easier and safer.

MAKE YOUR ATARI SING !

Astra BBS now on-line in PC Pursuit area ! (714) 546-5956

CALL FOR NEAREST DEALER

```
*****
*
* display_file-- display file contents in ASCII/symbolic/hex format
*
* at exit:
*       all registers preserved.
*
*****
display_file:
        movem.l a0-a4/d0-d5,-(sp)
        move.w  handle,d5          get file handle
        clr.l   d3                 clear for byte values
        clr.l   d4

d_go:
        move.l  #cr_lf,-(sp)       carriage return/line feed string
        move.w  #$09,-(sp)         code==print string
        trap    #1                 do it
        addq.l  #6,sp              pop args

        move.l  #buffer,-(sp)      buffer address
        move.l  #$1,-(sp)          byte count to read
        move.w  d5,-(sp)           file handle
        move.w  #$3f,-(sp)         code==read file
        trap    #1                 do it
        add.l   #12,sp             pop args
        tst.w   d0                 end of file?
        beq     d_out              if so
        bmi     d_out              if error

        movea.l #buffer,a2         point to buffer
        move.b  (a2),d3            get byte read
        cmpi.b  #32,d3             do control conversion?
        bcs     do_ctrl            if so
        cmpi.b  #127,d3            else: do ascii?
        bcs     do_ascii           if so
do_hex:
        movea.l #hexes,a1          base of hex table
        lsl.l   #8,d3              shift d3 left 12 bits
        lsl.l   #4,d3
        swap    d3                 get low nybble
        move.b  0(a1,d3.w),(a2)+           hex digit in buffer
        rol.l   #4,d3              restore complete low byte
        andi.l  #$0f,d3            mask out high nybble
        move.b  0(a1,d3.w),(a2)+           hex digit in buffer
        move.b  #0,(a2)            null terminator
        move.l  #buffer,-(sp)      push address of string
        bra     print_it

do_ctrl:
        movea.l #ctrl,a1           base of ctrl table
        lsl.l   #2,d3              multiply index by 4
        pea     0(a1,d3.w)         push address of string
print_it:
        move.w  #$9,-(sp)          code==print string
        trap    #1                 do it
        addq.l  #6,sp              pop args
        bra     d_go               and continue

do_ascii:
        move.w  d3,-(sp)           push char
        move.w  #$2,-(sp)          code==conout
        trap    #1                 do it
        addq.l  #4,sp              pop args
        bra     d_go               and continue

d_out:
        movem.l (sp)+,a0-a4/d0-d5
        rts

*****
*
* data segment
*
*****
        data
        even

handle          ds.w    1
abort_msg       dc.b    $0d,$0a,'oops-- this file does not exist',$0a,$0d,0
filename        ds.b    30

cr_lf           dc.b    $0d,$0a,0
buffer          ds.b    8
```

```
hexes   equu     *               table of hex digits
        dc.b    '0123456789ABCDEF'

ctrl    equ     *               table of control codes
        dc.b    'NUL',0          0  -- null
        dc.b    'SOH',0          1  -- start of header
        dc.b    'STX',0          2  -- start of text
        dc.b    'ETX',0          3  -- end of text
        dc.b    'EOT',0          4  -- end of transmission
        dc.b    'ENQ',0          5  -- enquiry
        dc.b    'ACK',0          6  -- acknowledge
        dc.b    'BEL',0          7  -- bell
        dc.b    'BS ',0          8  -- backspace
        dc.b    'HT ',0          9  -- horizontal tab
        dc.b    'LF ',0          10 -- line feed
        dc.b    'VT ',0          11 -- vertical tab
        dc.b    'FF ',0          12 -- form feed
        dc.b    'CR ',0          13 -- carriage return
        dc.b    'SO ',0          14 -- shift out
        dc.b    'SI ',0          15 -- shift in
        dc.b    'DLE',0          16 -- data link escape
        dc.b    'DC1',0          17 -- device control 1
        dc.b    'DC2',0          18 -- device control 2
        dc.b    'DC3',0          19 -- device control 3
        dc.b    'DC4',0          20 -- device control 4
        dc.b    'NAK',0          21 -- negative acknowledge
        dc.b    'SYN',0          22 -- synchronous idle
        dc.b    'ETB',0          23 -- end of transmission block
        dc.b    'CAN',0          24 -- cancel
        dc.b    'EM ',0          25 -- end of medium
        dc.b    'SUB',0          26 -- substitute
        dc.b    'ESC',0          27 -- escape
        dc.b    'FS ',0          28 -- file separator
        dc.b    'GS ',0          29 -- group separator
        dc.b    'RS ',0          30 -- record separator
        dc.b    'US ',0          31 -- unit separator
```

Assembly Line

END

PHOTO: MICHEL TCHEREVKOFF

SOFTWARE ENGINEERING

Module Madness

BY KARL E. WIEGERS

Can you believe it? After only three (count 'em) months of discussing how to analyze a computer-oriented problem and design a software solution for it, we're almost ready to begin writing some programs. Seems hard to believe that we're making so much progress, but it's true.

Forgive the slight sarcasm. I know some of you are probably frustrated that I've been holding you back from the keyboard for so long, but it's for your own good. I've been trying to impress upon you the benefits you can reap if you follow a software engineering approach in your own program development. While some small projects may not seem to warrant all this trouble, there's never anything wrong with making sure you know exactly what problem you're trying to solve before you start hacking away at it. This is just as true of computer programming as it is of woodworking.

But we really have made progress, and I hope you've begun using at least some of these software engineering concepts yourself. Like any other tool, they should be applied judiciously. The overall goal is to bring a measure of structure and discipline to your program development that may have been lacking in the past.

A Brief Rehash

In the first two installments in this series, we took a close look at the primary step of modern software development, the process of gathering requirements and writing a detailed specification. This document may be in the form of a so-called "structured specification," which uses data-flow diagrams to graphically depict the movement of data among processes, data stores and objects external to our system. A data dictionary also is built, in order to keep track of all the individual pieces of information associated with our system and their logical (and hierarchical) groupings.

Last month we took an initial look at how to translate a structured specification into an executable computer program. The process of structured system design results in a plan for satisfying the specifications revealed during the analysis phase. Overview design considers the more abstract facets of our plan, while the detail design step gets into the nitty-gritty aspects of what each piece of our system is going to do. A "piece" of the system is now a program module, which corresponds to a primitive process on one of our lowest-level data-flow diagrams.

The internal structure and function of each program module is described in a "process narrative," or minispec, which can be written using a variety of techniques. We looked at minispecs written for the same process using a flowchart, pseudocode and an action diagram. (Personally, I favor action diagrams.) What we didn't get to yet is how we fit all these modules together into a well-structured, hierarchical computer program. Read onward.

Program Building Blocks

I assume that you are already fluent in at least one computer language. It really doesn't matter much which ones, but I hope you have some familiarity with a modern structured language such as C, Pascal or one of the newer flavors of BASIC available for the Atari ST. Such languages share some common features, one of which is that they encourage you to subdivide your large programs into a bunch of smaller, self-contained pieces. These pieces are called procedures in Pascal and some BASICs, functions in C, and subroutines in FORTRAN and other BASICs. I will refer to all such entities as "modules." A module is basically a named, addressable (that is, callable) piece of computer code.

There are lots of good reasons for building a program from many small modules rather than as a single monster source file. We discussed some of these advantages last time. We also talked a little bit about some of the characteristics of good modules. These included being short, relatively simple, focused on a single purpose, independent from other modules, and having access to only that subset of the entire system's data that is actually required. We'll talk more about some of these ideas in a little while.

One thing we didn't get to last time is how you fit all of your little modules together to build the final program. Unlike construction toys, you can't just fit any old pair of pieces together. The data interface established for each module places some restrictions on how one can be connected to another. And it's also important to arrange your modules in some kind of hierarchy, so that it's clear which module can call another module when necessary.

> The main point to keep in mind is that each module should have access only to the information it requires for proper functioning: no more and no less.

Charting Your Course

A "structure chart" is often used to depict the hierarchical connections among the modules in a system. A sample structure chart appears in Figure 1. As usual, this figure refers to the educational chemistry game, *Reaction Time*, that I wrote for the 8-bit Atari computers. Bits and pieces of Reaction Time have been scattered throughout the preceding articles in this series.

Each box on a structure chart represents a separate program module.

The box labeled "Main" at the top of the chart represents the main program in the Reaction Time system. The main program is the big boss of modules, which is why it appears at the top of the chart (just like a corporate president). The primary function of the main is to invoke (or call) other modules whenever it needs to. Modules at a level below the main are "subordinate" modules; conversely, the main is the "superordinate" of those modules. This same sub/super relationship can be seen farther down the page in Figure 1, where some of the subordinate modules themselves call other, even lower-level modules.

Each of these modules should correspond to a bubble (process) on a data-flow diagram from your structured design. As an exception, the main procedure often is not shown explicitly on a DFD, since its principal function is to control which of the processes that really does the work gets switched on at any particular time. The main really might not do any of the inputs-to-outputs data transformations we've come to expect from the processes in a data flow diagram.

In fact, there is a more formal way to depict the behavior of a "control process" like this. Control processes often are used for real-time software applications, such as using the computer to control a piece of electronic equipment. Maybe we'll take a look at control processes one of these days.

The lines connecting the modules on a structure chart indicate which higher-level module can call which lower-level (subordinate) modules. More often than not, a particular high-level module can call several subordinates, each of which performs a different function. On the other hand, multiple lines going to the same subordinate from several superordinate modules indicates that the subordinate can be invoked in several different ways, from different higher-level modules. Figure 2 shows that Module A has three subordinates, but Module E has three superordinates.

Some other information can also be shown on the structure chart, using little arrows with either solid or open circles on the end. The open circles represent data that is passed from the module at the tail of the arrow into the module at the arrowhead. The arrows with solid circles indicate the movement of "control" information from one module to another. For the sake of clarity, I've shown just a few of these little arrows in Figure 1.

Control information can be things like status flags, return codes and so on. For example, a module whose function is to see if a particular disk file exists might return to its superordinate a control item indicating that the file exists and is open, exists and is closed, or doesn't exist. Of course, these "flags" are actually data items (variables), but their purpose is to control what happens next, rather than to actually move interesting information around from place to place. Hence, control flows like this are sometimes treated separately from ordinary data flows.

You will recall (I hope) from our earlier discussions that you really can't infer anything about the sequence in which different processes take place from a data-flow diagram. (The only exception is when a particular process requires a data flow from a data store; the process that places something into that store obviously must take place before the process that uses the data.) Structure charts give us more information about the sequence in which modules might be executed. For example, in Figure 2, Module A must be executed before Modules B, C or D, which in turn must be executed before Module E. However, to learn more about whether Module B, C or D goes first, you'd have to look at the process specification for Module A.

FIGURE 1 · Structure chart for Reaction Time.

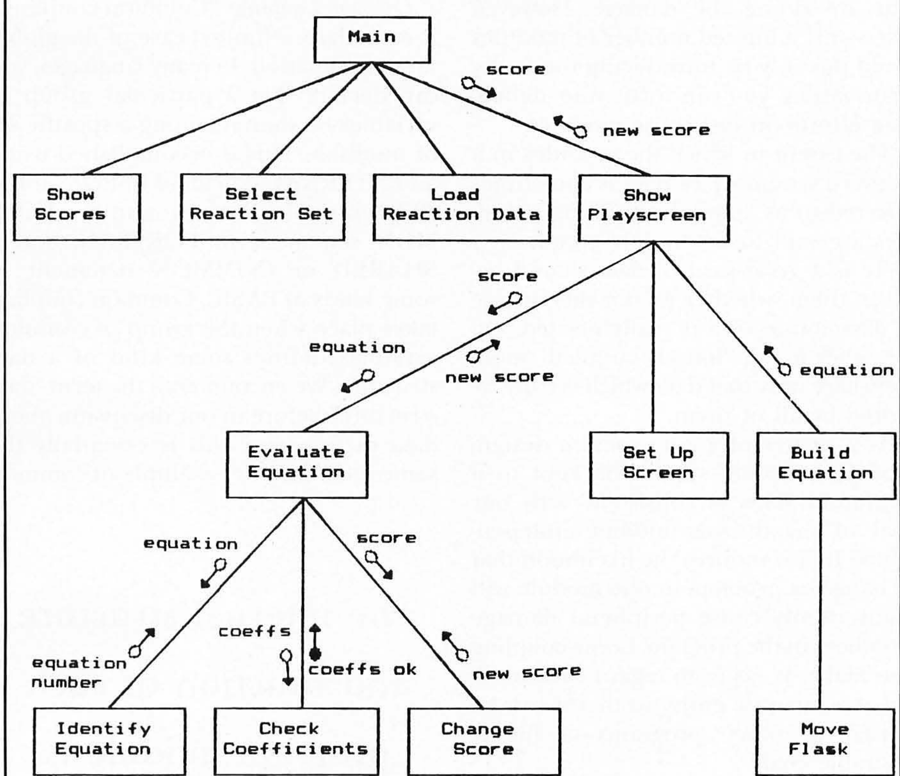
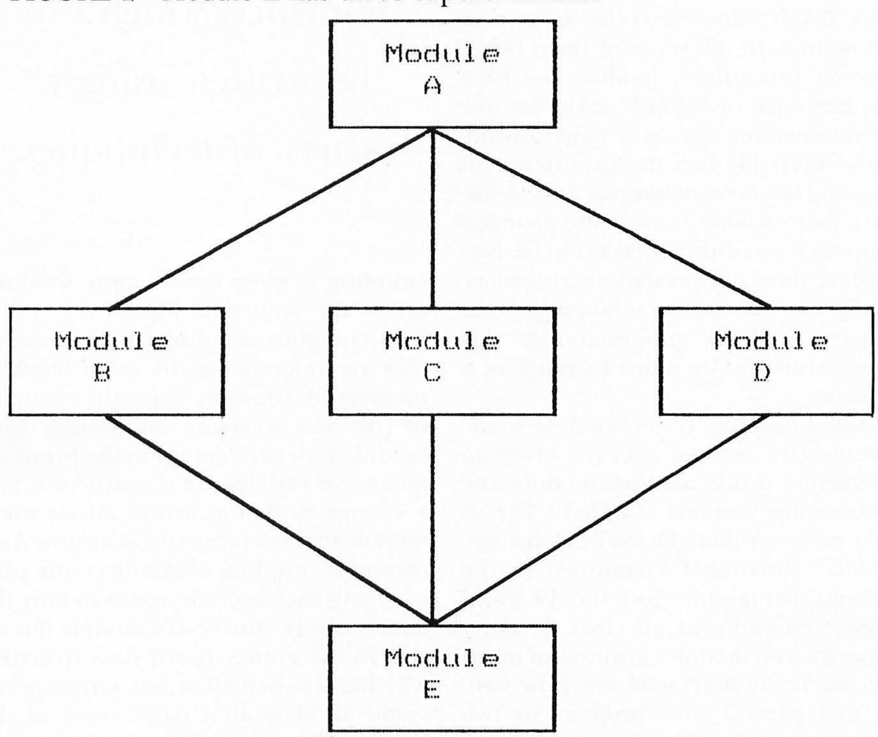


FIGURE 2 · Module E has three superordinates.



Module Coupling

In the past few issues, I've used the term "data interface" many times. Maybe I should explain this now, as we begin to delve more deeply into the inner structure of modules. A data interface is nothing more than a definition of the information that's shared between two modules. This definition can be on a strictly abstract level, as with the data stores that I stuck between every pair of processes on the data-flow diagram during structured analysis. Or, it can be at the most concrete level, as when you're defining the byte-by-byte contents of a disk file or the details of a complex record structure.

The main point to keep in mind is that each module should have access only to the information it requires for proper functioning: no more and no less. This is part of the software engineering precept of "information hiding." In plain terms, if a module doesn't have access to a particular chunk of data, it can't change the data. This becomes particularly valuable during program debugging. If all parts of the program have access to the data exhibiting the problem, it can be extremely difficult to track down the statements that are doing the damage. However, when only a limited number of modules could possibly be introducing the faulty information, you can focus your debugging efforts on just those modules.

The extent to which the modules in a software system share data is sometimes referred to as "module coupling." Modules are said to be "tightly coupled" if there is a great deal of data shared between them, whether or not this degree of data connection is really needed. On the other hand, "loosely coupled" modules share only that data which is truly required by all of them.

As a matter of good program design, module coupling should be kept to a minimum. This is consistent with our goal of maximizing module independence. It also reduces the likelihood that a change or problem in one module will inadvertently cause peripheral damage elsewhere in the program. Loose coupling also makes it easier to regard each module as a separate entity so that it can be reused in other programs—a highly desirable goal.

The very crudest form of data interface is, in fact, to have no data interface at all. This is the case in Atari BASIC programs, in which all variables can be used and changed anywhere within the source file. A GOSUB offers no protection, because Atari BASIC doesn't really have true subroutines, in the sense of them being separate procedures, modules or files. The existence of "global" variables like this is something you really want to avoid. You've probably had the experience of spending hours on debugging only to discover that variable *J* was being changed someplace you didn't expect it to be. Fortunately, since our present discussion assumes you are using a language that permits modular programming, this ghastly situation shouldn't be much of a problem.

Content Coupling. If one module somehow directly accesses data (or program statements) inside another module, the modules are "content coupled." This is really no better than the totally global, unmodular situation I mentioned in the previous paragraph. You should avoid content coupling at all costs. It intertwines the two modules involved so much that you really don't gain much by writing that part of your program as two separate modules. This is a case of very tight module coupling. With any luck at all, your favorite programming language won't even let you get away with this.

Common Coupling. "Common coupling" is essentially a limited case of the global variable situation. In many languages, you can declare that a particular group of variables be shared among a specific set of modules. This is accomplished using an EXTERNAL statement in PL/I, an EXTERN variable declaration in C, a COMMON statement in FORTRAN, and a SHARED or COMMON statement in some kinds of BASIC. Common coupling takes place when the group of common variables defines some kind of a data structure. We encountered the term "data structure" before in our discussion about data dictionaries; this is essentially the same idea. Another example of common coupling is when two or more modules access the same data file.

> The internal structure and function of each program module is described in a "process narrative," or minispec, which can be written using a variety of techniques.

In common coupling, all of the modules are referencing the same block of memory locations in which the elements in the data structure are stored. Each module is dependent upon the preset sequence of variables in the structure, and a change in that structure affects every module that references the structure. Also, common coupling contradicts our plan of giving each module access to only the data it needs, since every module has access to the entire shared data structure, whether it uses it all or not. One module could accidentally change some of the original data when it's not supposed to, which might make your debugging sessions feel like a life sentence. Avoid common coupling whenever you can.

External Coupling. This is a looser form of coupling than common coupling, in that only a single data element (as opposed to an entire data structure) is shared among several modules. External coupling is a little better than common coupling, since the sequence of items within the structure is no longer a factor, and since each module is accessing only that part of the structure that it needs.

The use of pointers in the C language is essentially a form of external coupling. Pointers give the called function access to specific variable storage locations, so the values of those variables can be both used and changed. This is called passing data "by reference," since the pointer is "referring" to the actual address at which the data item of interest is stored. You have to be extra careful when passing data by reference.

There are many times that external coupling is the most compact, efficient way to pass information from one module to another. So, sometimes it makes sense to use it. Just make sure you remember the ramifications, and be careful when you do get involved with external coupling. Sometimes module efficiency is less important than good overall software design.

Control Coupling. The next loosest type of module coupling is called "control coupling." This refers to the situation in which one module somehow controls the functioning of another.

A common instance is the passing of a control "flag" from one module to another; the called module uses the state of that flag variable to determine what it is supposed to do. While this is useful when used properly, it does run contrary to our plan of insulating the internal workings of each module from all other modules in the system. Obviously, the module passing a flag into another module must "know" something about what the second module does for a living, or the flag wouldn't mean much.

Stamp Coupling. I don't really understand why this term is used, but it's in the literature, so we'll use it too. Stamp coupling exists when several modules reference the same non-global data structure. For example, suppose module A reads a record from a file and passes the entire contents of that record into module B for processing. Modules A and B are stamp coupled. This isn't such a bad situation, but you have to remember that any change in the structure of the file (or, in general, of the passed data structure) will affect both modules.

Data Coupling. Now we're getting to the very loose end of the module coupling spectrum. Modules are data coupled if the only data communication between them is by passing individual data elements (not structures) in an explicit argument or parameter list. That way, only the data that really needs to be shared between the modules is exchanged, and any change in the physical grouping of individual data items really doesn't cause any problems.

Data elements (which could be single-valued variables or arrays) are typically passed in this way through a CALL statement, or through a function reference in C. The argument list in these cases defines the data interface between the modules. This is an example of passing data by value, as opposed to the case of passing data by reference that we mentioned earlier. Passing arguments by value is much safer, because the original values of the arguments can't be changed. The function or procedure being called simply assigns the values of the variables in the argument list to a parallel set of variables that are local within the subordinate module. This may seem wasteful of memory, but remember that the goal is to maintain the independence of each module. Data coupling is a good way to do it.

Of course, some modules have no direct coupling at all, if they don't operate on the same sets of data. I'm sure you can think of many examples from your own programming experience. And it's also likely that a given pair of modules may exhibit several of the kinds of coupling I described here. In that case, the "official" coupling level is the tightest, or highest level of coupling they exhibit.

The purpose of this discussion is not to make you memorize a bunch of arbitrary new terms. I just want you to be aware of the different kinds of interconnections that can crop up among the modules in a software system. Since our goal is to minimize module coupling wherever we can, it helps if you can keep in mind the various sorts of connections that might exist. The effectiveness of your modular, structured program design will be enhanced when you have the lowest possible coupling between the modules in every superordinate/subordinate pair.

Module Complexity

Throughout our software engineering discourse, we've concentrated on the idea of breaking a software problem down into elemental pieces that can be dealt with in an organized, manageable way. This is basically a divide-and-conquer strategy. The goal of the partitioning process during structured design was to wind up with primitive processes on the data-flow diagrams, each of which performs a single, well-defined function. Your success in partitioning will be revealed by the degree of complexity shown by each individual program module.

Basically, human beings aren't terribly bright. We have a limited ability to comprehend and deal with complex systems. In fact, the only sure-fire approach is to subdivide complex things into a bunch of less complex components. In the case of computer programming, the more complex a particular module is, the more likely you are to generate errors in the course of writing the program code. Since we want to avoid errors (right?), a good start is to minimize the complexity of each individual program building block.

Module "cohesion" is related to the complexity idea. A module's cohesion refers to how closely the module approaches the design goal of performing a single function. For example, a module that initializes some variables, opens a file, reads the file, does some calculations, writes to the file and closes the file is said to have a low "functional strength," in that many different functions are being performed by the module. The other extreme would be to write separate, tiny modules to carry out each of those processes. Each one would have high functional strength (or cohesiveness), but the system architecture (that is, the hierarchical arrangement of the modules) becomes very complex because there are so many pieces to fit together.

A whole spectrum of module cohesion can be identified, similar to the scale of module coupling. Mercifully, I'll spare you all the gory details. But I want you to strive for a high functional strength of each of your modules, by trying to make each one do a single task well. Don't throw diverse functions into the same module just because they fit. If you keep in mind the general goal of keeping modules small (down around 10-100 executable lines of code), you'll be able to think more clearly about how to segregate the various tasks your system must perform into individual program components.

Even if all of your modules do have good functional strength, they may still be quite complex. There are several ways to assess the complexity of a module. How many arithmetic operations does it carry out? How many decision-making statements does it have? How many possible paths are there through the code in the module? Do the best you can to keep the answers to all these questions small.

One Two Three, Go!

At this point, I'm going to skip over the large and very important topic of structured programming. I'm assuming that, as experienced programmers, you are already skilled in the art of writing good, clean, well-structured code. Your modules all have only one entry point, and only one exit point. You don't use GOTOs unless you absolutely have to. You indent your source code to make it readable, and you use plenty of comments. You declare all of your variables, whether the language insists upon it or not. You use structured constructs (where available) such as IF/ ELSE IF/END IF and SELECT/CASE statements. And, of course, you document your programs like crazy. Have I missed anything?

So now, since you've been champing at the bit for months, I hereby authorize you to go and write code! Take process narratives in hand, and write all the program modules your system requires, no more and no less. Test them and make sure they work, then put them all together. Test the result, and your project is complete.

Test them? Put them all together?? Test the result??? These are not trivial concepts. Entire books have been devoted to systems integration, program testing and software quality assurance. In our next installment, we'll talk about strategies and tactics for testing your modules, both individually and in concert with their buddies, as well as approaches for integrating your modules into the final software package. In the meantime, think about just what "software quality" means. Once we've tried to define it, we can strive to achieve it.■

After receiving a Ph.D. in organic chemistry, Karl Wiegers decided that it was more fun to practice programming without a license. He is now a software engineer in the Eastman Kodak Photographic Research Laboratories. He lives in Rochester, New York, with his wife, Chris, and the two cats required of all ST-Log authors.

STEP 1

NEW & IMPROVED

by Maurice Molyneaux

As I sit down to write this, it is late September 1988, and I have just recently received a disk-based Beta copy of what is promised to be the first major revision of the ST's Operating System (OS) in the three years since the 520ST was first released. If all goes well, this new version of the OS will be available to users by the time you read this (or shortly thereafter) and may even be included in the latest ST and Mega units to come out of Atari. In order to give you an idea of whether or not you'll want to upgrade to the new OS (provided Atari will even make it available as an upgrade), this month I'll be talking about the changes. I will not be listing bugs, as this is a Beta version, and it would be unfair to report any problems encountered that might be eliminated before the OS is released.

TOS

Backing up a bit, for those of you new to the ST, the Operating System is a series of programs and routines which are the core of the ST. All the disk, printer, video, keyboard and mouse-handling codes are part of the OS. The ST's OS is called "TOS" which stands for "The Operating System." It is this which, more than anything, makes your ST act like an ST. The hardware can act like other things if a different OS is plugged in. (Dave Small's *Spectre 128*, for example, makes your ST act just like a late model Macintosh.) In all STs manufactured after 1985, TOS is contained in ROM (Read Only Memory) chips within the computer (most pre-'86 520STs that did not have TOS in ROM have since been upgraded).

TOS is actually the sum of a number of smaller elements, of which some of the primary pieces are:

GEMDOS—(Graphics Environment Manager Disk Operating System) The hardware-independent part of TOS, which monitors and controls the input and output functions of the ST, including screen display, keyboard and mouse input, and disk and port I/O.

BIOS—(Basic Input Output System) The low-level software interface between the ST hardware and GEMDOS.

XBIOS—(extended BIOS) Special BIOS-type routines to support special features of the ST hardware.

VDI—(Virtual Device Interface) Graphics drivers that interface the hardware and software. Used to translate software graphics commands into hardware instructions.

AES—(Application Environment Services) Software tools used primarily to handle the graphics interface, such as windows, dialog boxes, menu bars and other GEM features.

As you can see, there are special parts of TOS for doing different jobs. Furthermore, you may notice how complex this whole thing is. GEMDOS receives a command, which is passed through the BIOS and/or XBIOS to the hardware, and may use VDI or AES calls (commands) to get the job done.

GEM

This is the user-interface of the ST, the "Graphics Environment Manager," which is what gives you all those icons, windows, drop-down menus, alert boxes, etc. Contrary to what some people think, GEM is not the ST's OS; it is a *program* run by TOS. Furthermore, some people think that GEM and the GEM Desktop are one and the same. Again, not true. The Desktop is a GEM application, run under GEM, which is in turn run under TOS!

GEM has its roots deep in GEMDOS, AES and VDI. However, there is a part of GEM which is not in TOS, and that is the "Graphics Device Operating System" or GDOS. GDOS's job is to load graphics device drivers (screen, printer, plotter, etc.), fonts, etc., and act as an interface between the application (program) being run and the output device being used. GDOS is usually found as an AUTO folder program packaged with some other software (such as *Easy Draw* or *DEGAS Elite*). It "hooks" into GEM when the machine is booted and stays in RAM until the machine is shut off.

The ST GDOS has numerous problems, however. Namely it is slow, and it loads device drivers and fonts only at boot-up. Thus, to change drivers and fonts, you have to reboot your ST. (There exists a replacement for GDOS, which corrects most of these shortcomings. Written by Charles F. Johnson and John Eidsvoog, it is called *G+Plus* and is marketed by CodeHead Software.)

Versions

I know some of you are asking, "What about the 'new' Mega ST ROMs?" Well, truth be told, those ROMs were just a "patched" version of the same old ones. The changes there were made primarily to allow for the blitter chip. Essentially, there are three ROM versions: the original ROM TOS, the Mega/Blitter ROMS and the new TOS.

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

SPECIALS XL/XE

#AAB822P 822 Printer Paper.....$2.99
#AA14746 T.V. Switch Box........2.99
#AA4010 Tic-Tac-Toe...........4.99
#AA4011 Star Raiders...........4.99
#AA4012 Missile Command.......4.99
#AA4013 Asteroids.............4.99
#AA4022 Pac Man...............4.99
#AA4025 Defender.............4.99
#AA4027 QIX..................4.99
#AA4102 Kingdom (Cass.)........1.99
#AA4112 States & Capitals (Cass.).1.99
#AA4121 Energy Czar...........1.99
#AA4123 Scram (Cass.)..........1.99
#AA4126 Speed Reading.........2.99
#AA4129 Juggle's Rainbow.......1.99
#AA415 File Manager............9.99
#AA4204 1020 Color Pens........1.99
#AA5047 Timewise (D)...........3.99
#AA5049 Visicalc (D)...........24.99
#AA5050 Mickey Outdoors........5.99
#AA5081 Music Painter (D)........9.99
#AA6006 Counseling Procedure....1.99
#AA7102 Arcade Champ (No J. Stk)6.99
#AA8030 E.T. Phone Home.......3.99
#AA8048 Millipede...............4.99

CLOSEOUTS XL/XE

ROM CARTS (XL/XE)
$3⁴⁹ ea or 5 for $14⁹⁹

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

Rocklyn
Gorf..........................2.99
Anti-Sub (Disk)................2.99
Journey to Planet..............2.99

Atari Program Exchange
10 Different Cassettes For......$11.99

SPECIALS XL/XE

Access
Leaderboard Golf..............13.99
Accolade
Hardball.....................19.99
Atari
Atariwriter Plus................35.99
Broderbund
Printshop....................26.99
Datasoft
Alternate Reality (City)..........23.99
221 Baker St...................20.99
Electronic Arts
Auto Duel....................23.99
Firebird
Silicon Dreams................19.99
Jewels of Darkness............19.99
Microprose
Top Gunner...................16.99
F-15 Eagle Strike..............22.99
Origin Systems
Ultima 4......................36.99
Strategic Simulations
Gemstone Warrior.............11.99
Sublogic
Flight Simulator...............34.99

ST SOFTWARE

Access
Leaderboard Golf...............22.99
Accolade
Pinball Wizard.................21.99
Activision
Hacker II/Music Studio (ea.)......28.99
Antic
CAD 3-D......................29.99
Avant Garde
PC Ditto......................59.99
Batteries Included
Degas Elite...................37.99

Data East
Speed Buggy **$25⁹⁹**

ST SOFTWARE

Epyx
Championship Wrestling........19.99
Dive Bomber.................29.99
Firebird
Jewels of Darkness...........19.99
The Sentry/Tracker (ea.)........12.99
FTL
Dungeonmaster...............29.99
Metacomco
ISO Pascal..................59.99
Michtron
Leatherneck 29.99
Microprose
Gunship.....................28.99
F-15 Strike/Silent Service (ea.)...24.99
Miles Software
ST Wars.....................24.99
Mindscape
Road Runner.................35.99
Mark of the Unicorn
PC Intercom.................79.99
Mark Williams
C.........................119.00
Paradox
Wanderer (3D)................24.99

ST SOFTWARE

Progressive Computer
Graphic Artist 1.5.............119.00
Psygnosis
Obliterator...................29.99
Soft Logik Corp.
Publishing Partner.............54.99
Strategic Simulation
Questron II...................35.99
Sublogic
Flight Simulator II.............33.99
Timeworks
Swiftcalc/Wordwriter.......(ea.) 45.99
Desktop Publisher.............79.99

Word Perfect **$179**

In U.S.A.

Call: 1-800-233-8950

In Canada call: 1-800-233-8949 All Other Areas call: 717-327-9575 Fax: 717-327-1217
Educational, Governmental and Corporate Organizations call toll-free 1-800-221-4283
CMO, 101 Reighard Ave., Dept. B7, Williamsport, PA 17701

MEMBER MMC MICROCOMPUTER MARKETING COUNCIL of the Direct Marketing Association Inc

OVER 350,000 SATISFIED CUSTOMERS • ALL MAJOR CREDIT CARDS ACCEPTED • CREDIT CARDS ARE NOT CHARGED UNTIL WE SHIP

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

B701

CIRCLE #108 ON READER SERVICE CARD.

Changes, Changes

Now that we've put to bed the issue of just what TOS is and what versions exist, let's take a look at the newest version, and what changes you will find. Please note that I was using Beta release TOS 1.4, and it's possible (though unlikely) that some odd or end may have been changed by the time the new TOS ROMs are released.

The Desktop looks the same. The only difference here being that the labels on the Drive and Trash icons are not as tall.

The drop-down menus are also the same as before. Visually, it's the same old thing. However, there *are* changes lurking just under the surface. For example, double-clicking on the Trash Can results in an alert-box opening that states, "You cannot open the Trash Can icon into a window." The older TOS versions said the same, but continued: "To delete a folder, document or application, drag it to the Trash Can." This info was eliminated because it is a repeat of what is stated if you highlight the Trash icon and select Show Info from the file drop-down.

Speaking of the Trash, the older TOS had a Trash Info box which contained an error. It stated "The Trash Can is the destination to which you drag the *disks*, folders, documents or applications that you want to delete PERMANENTLY." The italics on "disks" are mine, but they serve to note the error. You cannot drag a disk icon to the Trash to erase it. Doing so will result in the following message: "You cannot drag a disk to the Trash Can." The new TOS's Trash Info reports, "The Trash Can is used to permanently delete files or folders." Period. Not only is it correct, but it's much simpler and to the point than the old message.

Okay, that was the most obvious surface change. Other changes that appear on the Desktop proper have to do with many of the dialog boxes that appear as the result of using drop-down menus. One nice touch that makes these boxes easier to look at is that (generally) they no longer use underlines to lead numeric values but use spaces instead. Therefore, rather than seeing "Bytes used: _ _ 165492," you would see "Bytes used: 165492." Compare the illustrations accompanying this article with the same dialog/alert/etc. boxes on your version of TOS.

Info-mation

Show Info under the file menu looks a little different in Figures 1, 2 and 3; mostly it's just easier on the eyes. The main difference is that now, for the first time in TOS history, you can change the name of a folder (hooray!). Before there was no way to do this from GEM itself. Thus, if you want to rename a folder, you can change its name just like you can on a file. Furthermore, as with much of the new TOS, things are much clearer. For example, notice that the box in Figure 3 is labeled "FILE INFORMATION." The old TOS box for this was labeled "ITEM INFORMATION," which isn't as accurate. There is also more consistency between similar boxes. Show Info on disks, folders and files all return dialog boxes which are laid out more or less the same, with the same terms, etc. However, note that while the same dialog appears for files and folders, the "Attributes" are only active for files. Thus, you can't "lock" a folder.

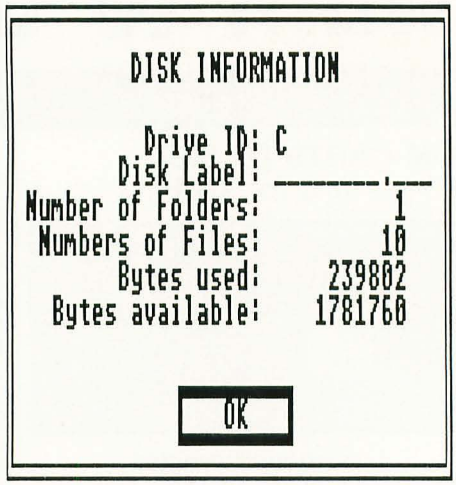


FIGURE 1

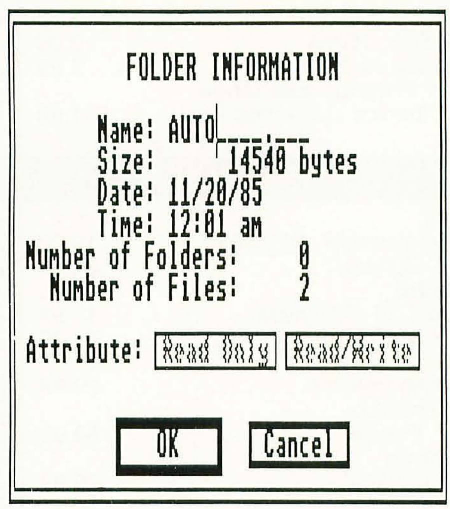


FIGURE 2

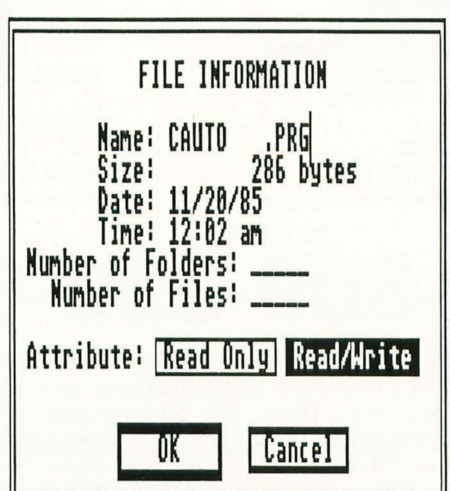


FIGURE 3

Installation Charge

The Install Disk Drive selection under Options has been changed only insofar as that the default button (the one that responds to the Return key) is now Install rather than Cancel.

In Figure 4 Install Application looks much the same as well, the key differences being the addition of Boot Status options and a Remove button. Remove allows you to un-install an application. (Previously the only way to do this was to reboot or edit the DESKTOP.INF file [if you had saved the Desktop with the application in question installed] and *then* reboot.)

One major change is that Install Application now retains the full pathname of the installed file, including drive identifier and folder names. Previously it did not, and installing a program to run from a specific path used to involve editing DESKTOP.INF.

The Boot Status options are the most interesting. "Normal" means the installed program runs just like always. However, selecting "Auto" (and then saving the Desktop) will result in the selected program (just one, mind you) auto-running whenever you boot-up. This program can be on any drive and, better yet, doesn't have to be in the AUTO folder on your boot drive (in fact, if it is, it'll run *twice*). Even better, since the auto-boot program is run after TOS runs the GEM Desktop, you can now auto-boot GEM programs (impossible to do from the AUTO folder because GEM isn't activated until afterward). You can run TOS and GEM programs but not those of the TTP variety.

A note for beginners: Install Application allows you to designate a type of data file that will, when such a file is opened or double-clicked on, automatically run the program which uses it. You do this by selecting (single-click to highlight) a program file on the Desktop, and selecting Install Application. In the dialog box that appears, you enter the three-letter extension of the type of file the program in question uses. Thus, to install *Neochrome* you would enter NEO as the Document type. After that, double-clicking on any NEO file would run Neochrome. To avoid having to reinstall the application every time you reboot, you must save the Desktop.

A note for Desktop customizers: When you save the Desktop with a file installed to auto-boot, a line like the following is included in DESKTOP.INF:

#Z 01 C:\FOLDER\PROGRAM.PRG@

When you select a program to auto-boot, it wipes out any other program previously set to auto-boot. Thus, you are limited to one program to auto-boot from the Desktop.

The maximum size I was able to make a DESKTOP.INF file, under the new ROMs, was just under 1,600 bytes. If I tried to install any more applications after that point, I received a message stating that I could not install any more applications.

Selections

Set Preferences under Options has also received a small face-lift with the addition of a choice for confirmation of file overwrites. If you want the system not to prompt you when overwriting files, you can do so (though it's beyond me why anyone would want to risk the danger of accidentally overwriting something important).

One thing I griped about in an early *Step 1* was the GEM file selector. This is the box that you see whenever you are given a choice of a file to load, save, etc. The original was maddeningly ill-designed, with an obscure title (ITEM SELECTOR) that didn't tell you if you were loading, saving, deleting or whatever. Also, the only way to select a different disk drive for the directory path was to manually edit the Directory line in the selector. Figure 5 features the old selector and the new selector.

The new selector has buttons for each of the 16 possible drives the ST can access (A-P), and switching from one drive to another is as simple as clicking on the appropriate button. To reread a disk no longer requires clicking on the "grip" bar at the top of the directory box; just clicking on the appropriate drive button. Furthermore, a new AES call (FSELEXINPUT) has been added which allows programmers to replace the File Selector label at the top of the selector with the message of his/her choice. Before, tricky programming was needed to do this.

Another nice touch when you edit the pathname in the selector: Pressing Return now forces the box to display the new pathname parameters, rather than acting as the OK button. Pressing Return at any other acts as OK. When editing a path, you can click on the gripper bar to update the files, but clicking within the file display area no longer works.

Unfortunately, the new selector does not feature any way to check free disk space, or to toggle between a limited search string and an all-files search (as in toggling between searching for files matching *.P?1 or *.*), as do some third-party selector replacements. Furthermore, the selector still limits viewing to only nine files at a time (*sigh*).

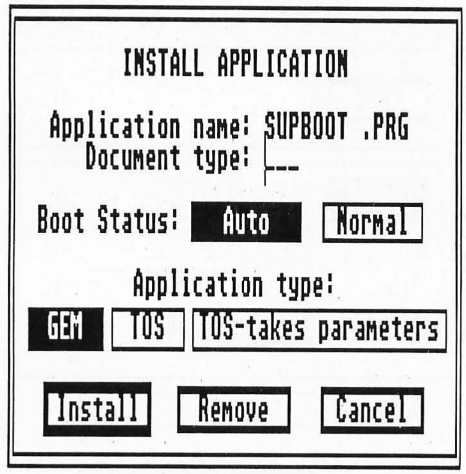


FIGURE 4

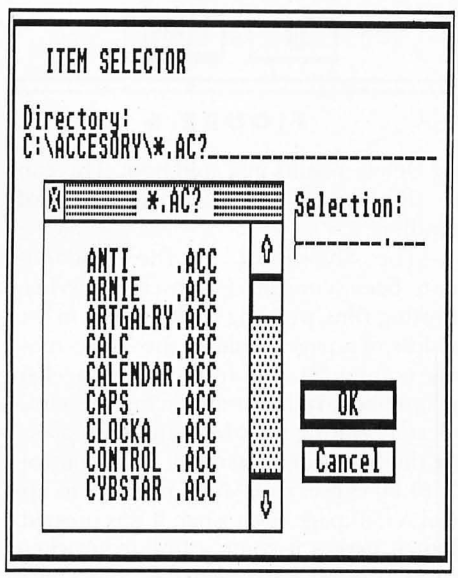


FIGURE 5 (OLD)

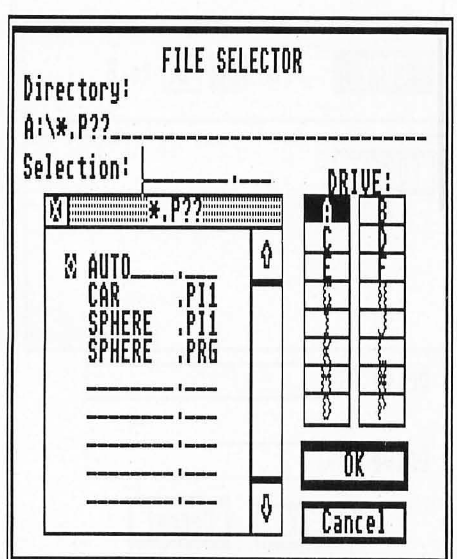


FIGURE 5 (NEW)

Disk Copy and Format

Highlighting a floppy-disk icon and clicking on Format under the file menu results in a totally new dialog box as you can see in Figure 6 than the one seen before. This is a copy/format box, which allows you to do either or both from the same place. If you drag one floppy-disk icon to another, the box in Figure 7 will appear. If you choose Format from the Desk drop-down, it will come up looking like Figure 6. To switch from Copy to Format, or vice versa, one merely has to click on the appropriate buttons. Thus, if you were going to copy a disk but forgot to format the destination disk first, you can do it from here, rather than having to Cancel the operation and select that option for the file menu. Handy.

Furthermore, when disk copying you can use the A and B buttons to toggle the source and destination drives. Thus, if you decided you'd rather have the other drive as the source, switching is as simple as clicking the appropriate button (you can toggle the drive to format as well). The Cancel button is now the default, making it less likely you'll accidentally click on OK and format or copy a disk when you were trying to exit.

A welcome addition for PC users: Disks formatted from the Desktop in this manner now have MS-DOS compatible boot sectors, eliminating the need for many of those little MS-DOS boot utilities.

File/Folder Copying and Deletion

When copying files and folders, things work just about as normal. The main difference is, again, a much better user interface. Figure 8 shows the alert box that appears when you are copying files. Notice that, unlike the old box which merely showed the number of files and folders remaining to be copied, this box displays the name of the file being copied, and its destination folder (if any). When deleting files and/or folders, a similar alert appears, as in Figure 9.

A file can be "moved," as well as copied, by holding down the Control key while dragging a file(s) to its destination. If the source and destination directories are on the same drive or partition, the file is moved by renaming it (changing its pathname to or from that of a folder, etc.). If the source and destination are different devices or partitions, the file is copied and the original deleted.

In the event of your mistakenly starting a copy, move or delete operation, holding down Undo on your keyboard will cause an alert to appear stating, "Abort this operation?" with Yes and No buttons. This will halt the process, but will not undo any action already done. (If you let it trash a few files, they're gone; if you copy or move a few files, the copies remain or some file will have been moved.) Furthermore, the time/date stamp on files are preserved when copied or moved.

If a name conflict occurs during a copy procedure, a NAME CONFLICT! box appears as in Figure 10. Unlike the old box, whose only options were OK (overwrite) and Cancel (skip), this new box contains three buttons: Copy (overwrite), Skip (go to next file, if any) and Quit (abort copy procedure). The addition of Quit is welcome as it allows you to abort, which was previously impossible. If the File Overwrites option of Set Preferences is set to No, NAME CONFLICT! will not appear when duplicate filenames are encountered—they will just be overwritten. NAME CONFLICT! will appear for conflicting folder names regardless of the status of the File Overwrite setting.

Miscellany

There are quite a number of other changes. I'll list some of them here:

—Warm and Cold resets are available from the keyboard. Holding down Alternate and Control while pressing Delete, causes a warmstart (just as in the PC world), while holding down Alternate, Control and the right Shift key while pressing Delete, results in a cold boot. This can be disabled by user-defined keyboard handlers.

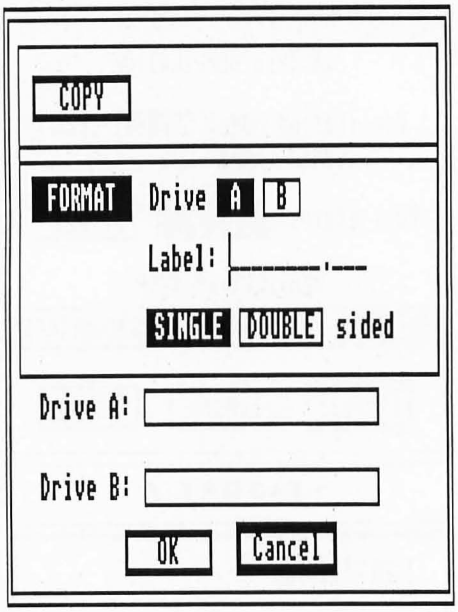


FIGURE 6

—The Show/Print text-file functions have been completely rewritten. When showing files, pressing the spacebar in the middle of a page results in the -More- message coming 24 lines from when spacebar was pressed, rather than having the same effect as waiting for -More- and then pressing the spacebar. Pressing d, D, or Control-D (^ D) causes the -More- message to appear a half page from when it was pressed. Return makes it come a line from when it was pressed. Q, q, and ^ C and Undo cause the output to stop immediately. When printing files the keyboard is polled every 16 characters. Like Show, Q, q, ^ C and Undo abort printing.

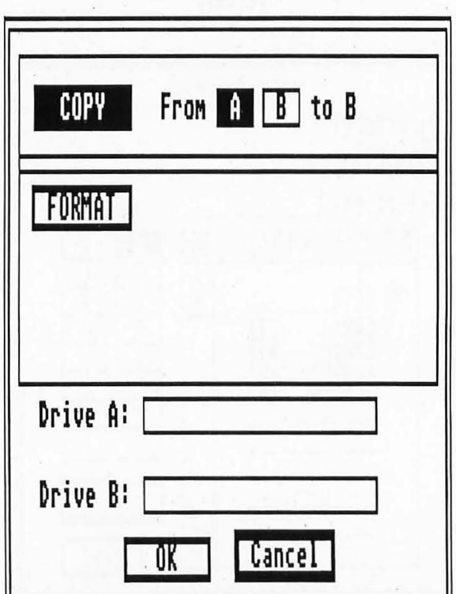


FIGURE 7

—When copying files, all available memory is used to minimize disk swapping on single-drive systems (at last!).

—All background windows are updated after a file copy, move, delete, disk copy or format operation.

—Pressing Escape forces the Desktop to read the directory of the active/top window. Before, it would only do so if a media change was detected. Furthermore, the media change detection routines are better, making it less likely that the system will miss the fact that you have swapped disks.

—The Desktop now shows as many files as possible in each window. Previously, there was a limit of 400 files that could be shown. The only limit now is the available free RAM (used to store the file information read from the directory).

—The Desktop does a much better job of opening the next window in the location where the last one was closed. In the older TOS, the system oftentimes got confused. Furthermore, window handling is improved. If an error occurs when a drive is "opened," the system does not open a blank window as it previously did.

—GEMDOS has been rewritten, and much of it has been improved. The FAT (File Allocation Table) search routines (used for finding data on disks) are much faster. The infamous 40-folder bug has been corrected. Limits still exist, mainly on the depth of folders and the accumulated depth of open files, but these limits are much harder to reach than on the old ROMs, and can still be eliminated using the FOLDRXXX.PRG program to add more folder capacity.

—The size of TOS has been brought down to the size of the original ROMs, so some programs that did not work with the larger Mega/Blitter ROMs may work again.

—The "archive bit" of a file's attributes is now supported. It denotes if a file is new (new or modified) or old (unchanged). Support of this archive bit will now allow programmers to check if a file has been modified since a certain date/time. For example, a program that backs up hard disks could be set to back up only those files which have archive bits set to the "new/modified" status. Once the file has been backed up, the program would set the bit to "old/backed-up" status. Thus, the program would know by itself which files you have created or modified since the last back up was made, and only back up those files, saving time and trouble.

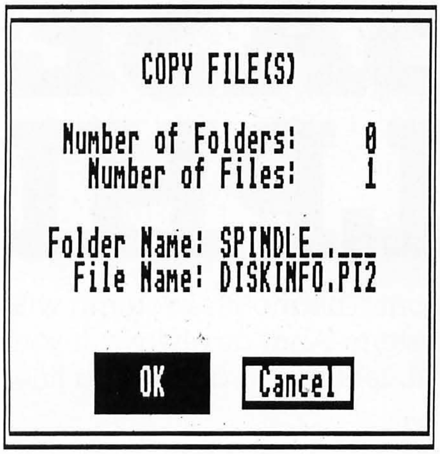


FIGURE 8

FIGURE 9

FIGURE 10

Conclusion

Before wrapping up this month, I'd like to acknowledge the assistance of John Townsend at Atari Corp., who provided me with the technical help and research materials used for writing this article. Thanks, John! Also, thanks to John Eidsvoog for his contribution of interesting facts I'd probably have otherwise missed.

Summing up: It has taken Atari a long time to come up with a significant update to TOS, and while some may say this new version is "too little too late," I beg to point out that many of the changes made in the new TOS make it much easier to live with. The messages are clearer, overall speed is improved, and there are ways to abort things that were previously un-abortable. We even got the long-awaited ability to auto-boot a GEM program. And, while it *would* be nice to have TOS support more Desk Accessories, true multitasking, more colors (one could easily have 16 colors in medium res if the appropriate changes were made), alternate disk formats (say, Twisted), etc., at the very least we've finally got *some* improvement. And the nice thing is, many of the changes made were what we users have asked for. So, if there's something on your TOS wish list, drop a line to Atari and let them know about it. Who knows? It may end up in the next TOS. (Hopefully we won't wait three years for that one!).

This *Step 1* has been surprisingly free of puns and bad humor so, to end this on an appropriately wacko note, I offer this little tidbit: If one pokes around through the ST OS, it's possible to find some interesting things. Case in point: With TOS in ROM, viewing memory in ASCII form from hex FCF3C8 to FCF407 in the old TOS, or FC2400 to FC243F in the Mega/Blitter TOS (and it appears in the version of the new TOS I have now), reveals a message which reads, "Dave StaUgas loves Bea Hablig" (sic).

It could only happen at Atari.

Addenda

In "Playing Games with Graphics" (ST-Log #24) there was a slip-up in regard to the figures. What was labeled as Figure 5 was actually Figure 6, and Figure 5 got lost somewhere between me and the magazine. Sorry if that caused anyone any confusion!

Maurice Molyneaux studies classic cel and modern computer animation, deadens his eardrums with overloud classical music, and further damages his already questionable sanity by listening to recordings of Monty Python and Tom Lehrer. Otherwise he just makes a nuisance of himself. His DELPHI username is MAURICEM.

DATABASE DELPHI

by
Andy Eddy

A new year begins and with it comes a lot to report on the DELPHI front. This month's column will pass on a few helpful hints, unique tidbits and new features on the system. And, as always, if you have something you want to see covered in a future Database DELPHI, feel free to drop me a line at the magazine or through DELPHI E-mail at ANALOG2.

Power Trips

There's no doubt that DELPHI is a powerful system. Unfortunately, most of its users don't realize this until they wander away from the familiar areas, such as the ST SIG. But the diversity of offerings spread throughout the system may surprise you. Let's take a glimpse at some areas outside the Atari sections.

Online Photo-Opportunity

A new SIG was recently added for photo and video buffs. Named the *Video and Photography Forum SIG* (or *ViP* for short), this area is run by Bob Gorrill (MRBOB), and will offer tips from professionals, discussions on the latest in hardware and software releases, as well as provide a place for users to get in touch with each other through the Forum and Conference areas.

For instance, a recent stop to the forum in the ViP area brought to light opinions on the pros and cons of autofocus cameras, the status of Konica as a camera manufacturer and discussions surrounding the production of the *Max Headroom* TV series.

In the future, the ViP hopes to make it possible for online visitors to purchase products from the Shopping section inside the SIG. Combining the information you receive in Forum with a mini-mall inside the SIG will make for more worry-free purchases.

This SIG is available from the Groups and Clubs section of DELPHI. To get there, just type *GRO VI* from the MAIN> prompt. The layout of the SIG is similar to the ST SIG—databases, forums, polls, etc., are all available—so you should be comfortable getting around. And if you stop in, tell them ST-Log sent you.

Just the FAX, Ma'am

Facsimile, or FAX, as it is more commonly referred to, is becoming more and more popular around the world. After all, what better way is there to get a document from point to point? These days, there are cases where even overnight delivery won't make the grade.

For example, the L.A. offices of ST-Log often send copies of the magazine's pages to the Connecticut office for verification, and those proofread pages can then be FAXed back or discussed over the phone in the span of a couple of hours—actually minutes, provided Clay (ST-Log's editor) isn't busy slamming down pizza slices.

For those unaware of the process, FAX optically scans the page you wish to send, converts the image into binary (black and white) code, then transmits it over standard phone equipment in seconds, usually at 9600 baud. The receiving FAX station takes this data, much like standard telecommunications data, then reconverts the signal into a replica of the original document. With this method, text and graphics are treated the same way, almost as if you were photocopying the original. Depending on the quality of the receiving machine, the copy may come very close to the original.

DELPHI has a FAX service available from the DELPHI mail section that lets you send text messages to any standard FAX machine from your PC. Due to many graphic formats available, there isn't currently any way to send pictures. Perhaps in the future, graphics will also be supported.

To send a FAX through DELPHI, you first must have a text file to send. This file can either be uploaded to your workspace (WOR from a SIG menu or from the Main prompt) using whatever transfer method you choose from the list, or it can be typed in directly to the FAX service "live."

Keep in mind that this fancy service comes with a price. According to DELPHI's help screen for FAX (type *HELP FAX* at the DMAIL> prompt; type *GO DELP* from most any prompt to get there), the rates are:

Destination	First Page	Additional Half Pages
United States	$1.25	$0.50
Canada	$2.00	$1.00
International	$7.00	$2.00

In this chart, a page is defined as 2,500 characters, and a half page is defined as 1,250 characters. You can also send the same FAX to more than one number, but each message sent will result in extra charges.

DELPHI sends FAXes every 15 minutes so your message will get to its destination almost immediately.

Roll With It

There's an interesting command that can be found once you are in a conference "room" or group. It is the ROLL command, and it electronically rolls dice.

First we'll start off with the basic ROLL. Once in a group (from the CONFERENCE> prompt, type *JOIN TEST*, if you just want to try it out) type out */ROLL*. The slash is important; otherwise, DELPHI will think you are entering the word ROLL as text. What you'll get back is:

```
/roll
ROLL by ANALOG2> (2d6) 1/6 6/6
```

The symbol, if you haven't figured it out already, translates as two dice, each with six sides, with the first die reading "1" and the second reading "6." The default is for two six-sided dice to be rolled. That command shows everyone the roll you made. You can keep it to yourself by typing */PROLL*:

```
/proll
Private ROLL> (2d6) 2/6 3/6
```

This brings us to customization. If you want, you can change how many dice to roll. Just put a number with a "d" after the */ROLL*, and DELPHI will read that as the number of dice. For example:

```
/roll 3d
ROLL by ANALOG2> (3d6) 4/6 3/6 5/6
```

The other customization varies the number of sides on the die/dice. Figure 1 shows how you'd roll three two-sided dice *and* two three-sided dice:

FIGURE 1

```
/roll 3d2 2d3
ROLL by ANALOG2> (3d2 2d3) 1/2 1/2 1/2     3/3 2/3
```

In this manner you can set it up to test coin-flipping probability, making DELPHI think you are rolling two-sided dice. *Dungeons and Dragons* players can match their special dice configurations and possibly play long distance games. Maybe you'd choose to get a group together for a game of Yahtzee. Sure, this is one of the less businesslike features, but it can bring a bit more fun to your online sessions.

I'll Have Mine *Scramble-d*

As we mentioned last month, *Scramble* is a new word game on DELPHI. You can access it from the GAMES area (type *GO GAME* from most prompts to get there) or from the main conference area (type *GO CON*, then do a */WHO* to find out where the game is played).

When you enter the Scramble room, you'll be shown the latest high scores and then be allowed to play. Type a *GO* and it'll start: (Figure 2)

FIGURE 2

```
go
ANALOG2> Get ready for a round of SCRAMBLE!
*** Starting a SCRAMBLE Round!
The letters are:
   C  T  A  R
   I  O  E  I
   E  N  G  Q
   L  I  G  U

reel
tar
tarring
 TARRING
    ^ is not ok.
talon
lacing
lace
gig
quiet
quieting
quite
quit
queen
 !!Time's up !!
SCORE:
  ANALOG2      :  266 (QUIETING for 64 points!)
That was quite a game... 266 is my best score this week.
ANALOG2> That was quite a game... 266 is my best score this week.
```

As you can see, a four-by-four grid of letters is shown, and you have to come up with as many words in 90 seconds as you can; the bigger the word, the higher the score. The score is achieved by the number of letters squared, and all word scores in the 90 seconds are added.

Scramble is loads of fun because you don't just play—you play against *others.* Better yet, you can talk about the game (or anything you want) with the other players after each contest. DELPHI is thinking about putting this in each SIG's Conference area so you can play with your online friends, but as of this writing it wasn't enabled. We'll keep you informed on this in future columns.

I can see by the clock on the wall that it's time for champagne and noise makers. I'd like to wish everyone a Happy New Year. Till next month, C U online. ■

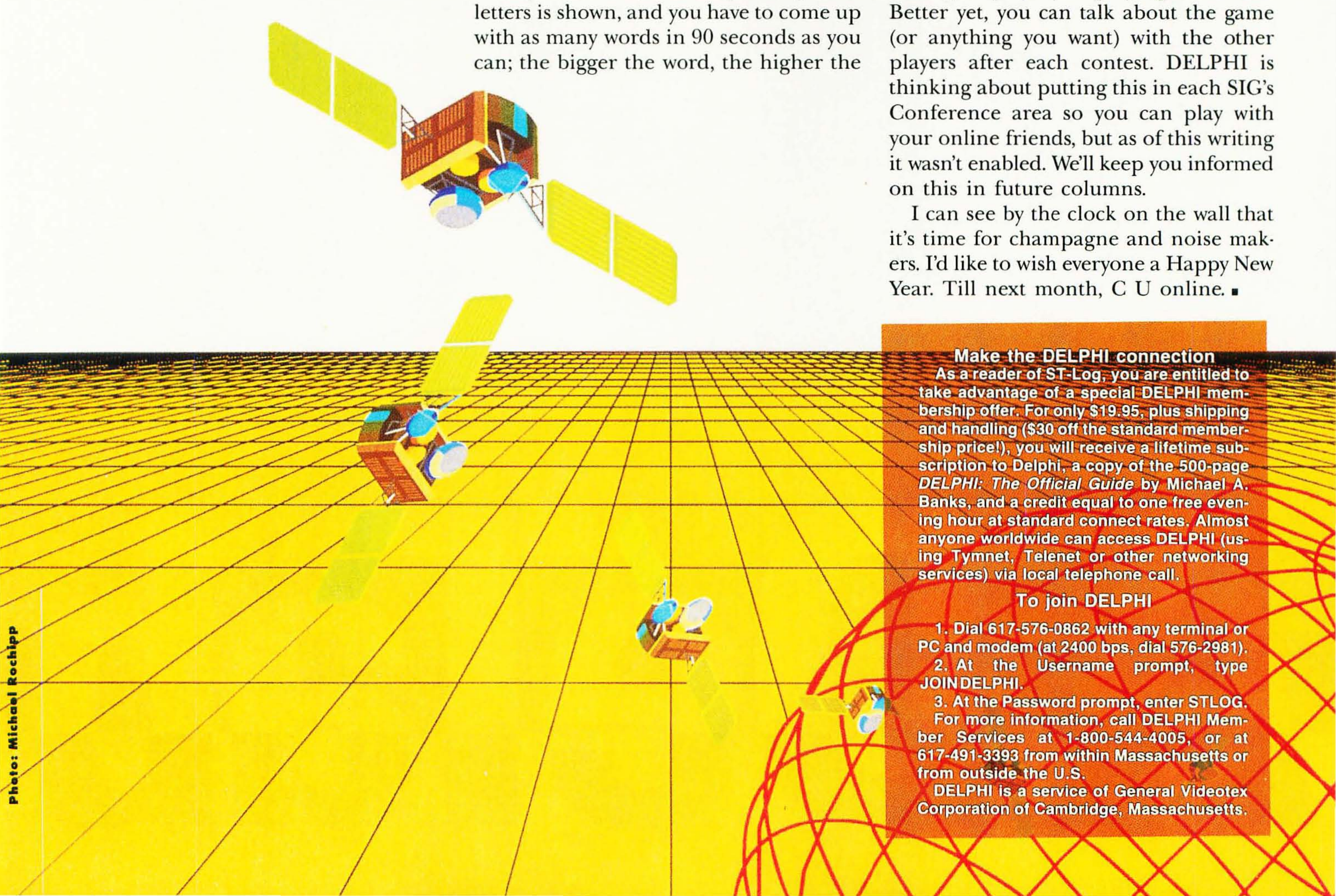


Photo: Michael Rochlpp

Make the DELPHI connection

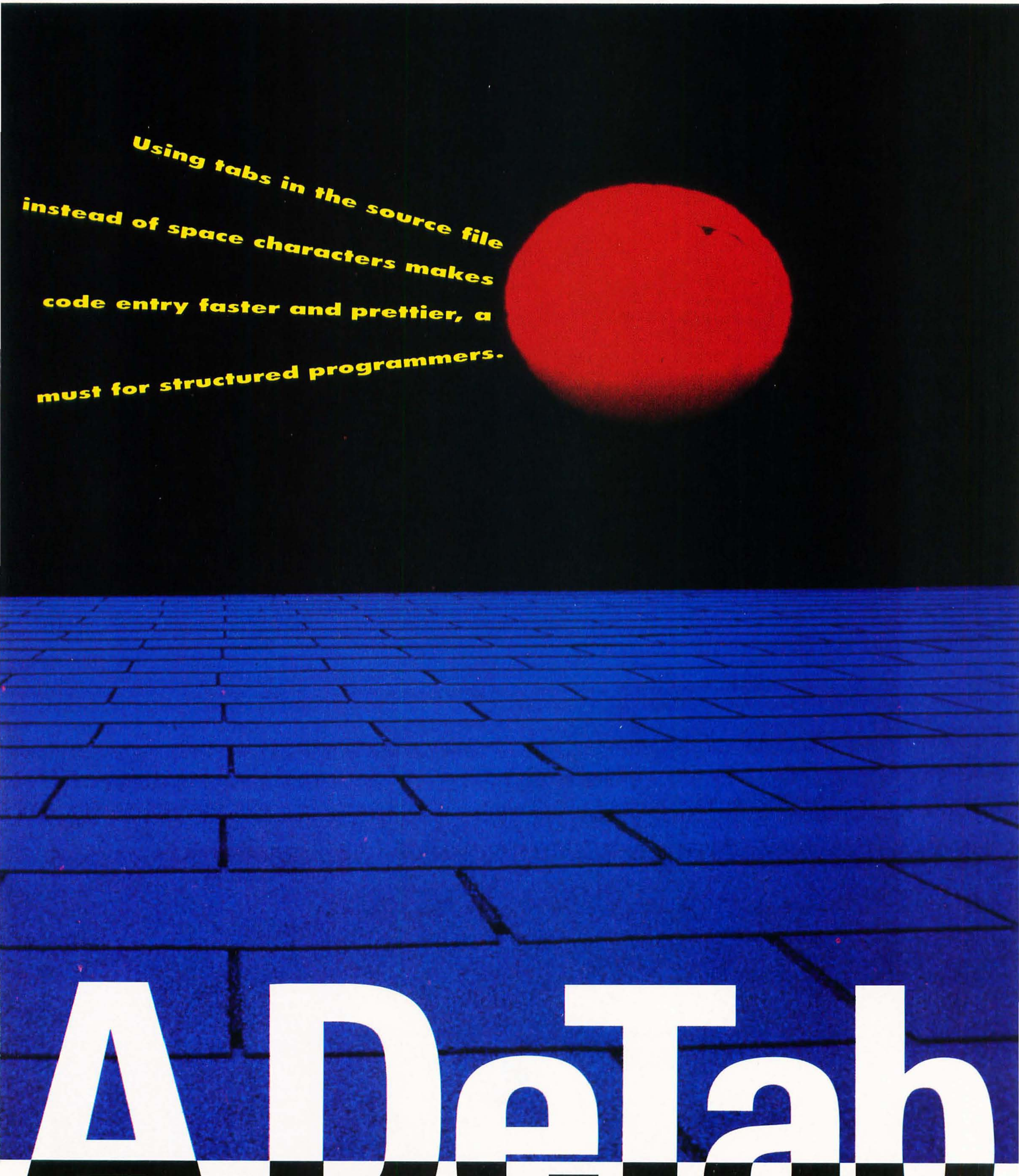
As a reader of ST-Log, you are entitled to take advantage of a special DELPHI membership offer. For only $19.95, plus shipping and handling ($30 off the standard membership price!), you will receive a lifetime subscription to Delphi, a copy of the 500-page *DELPHI: The Official Guide* by Michael A. Banks,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

To join DELPHI

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

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

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



Using tabs in the source file instead of space characters makes code entry faster and prettier, a must for structured programmers.

A DeTab

BY MATTHEW J.W. RATCLIFF

Many programmers soon realize, for several reasons, the advantages of using *MicroEMACS* or other relatively simple editors for code development. First, these text editors are fast, without all the frills of GEM's windows and menus, which clutter the screen and significantly slow down the user interface. The second advantage is the use of tabs, or programmable tabs.

Utility

Using tabs in the source file, instead of space characters, makes code entry faster and prettier, a must for structured programmers. Often as many as eight spaces are replaced by a single tab character. This improves the efficiency of file storage and can actually speed up compile times (since the compiler has to skip over only one "white space" tab character instead of eight space characters).

The use of tabs presents problems when it comes to printing your files, however. Most printers will recognize a tab character and move to the next eighth character position (i.e. 1, 9, 17, etc.). This can really spoil your "pretty listing" if your editor has tab stops at every four positions (the default used by Russ Wetmore's version of MicroEMACS), for example. Some printers are dumber in that when they see a tab they automatically spit out eight space characters, completely oblivious of the current print column (e.g. jump from position 6 to 14, instead of 9). I've run into these problems frequently, resulting in listings that are annoying and difficult to read.

One solution is to load the file into a word processor such as *1st Word* or *Word Writer*. Word Writer uses the "typewriter standard" of five spaces for each tab stop, while 1st Word defaults to eight. If your editor doesn't use either of these settings, then you must remember to edit your tab stops before loading the source code.

The above nuisances can be easily eliminated with this utility, DETAB.TTP. When you double click on DeTab, a dialog box pops up, allowing you to enter the filename of your source code followed by a number separated by a space character. The number specifies tab size. If none is specified, a default of eight is assumed. If no filename is entered in the .TTP entry dialog, then DeTab will remind you of the proper call format and prompt you for the filename and tab size.

DeTab will then generate a .DET (detabbed) file on the same drive, in the same directory. If PROG.C is detabbed, the output file will be PROG.DET. It will be a copy of the original, with all tabs expanded to space characters. You can then load the file into your word processor and do a printout, without all the annoying tab character problems.

The DeTab utility has saved me a lot of aggravation, and I hope it proves useful to you too. ■

DeTab Utility
LISTING 1
C

```
/****************************************
** Detab utility by Mat*Rat
** calling format: DETAB file tabsize
** If no tabsize is specified
** a default of 8 is assumed.
*/

/****************************************
** INCLUDE files
*/

#include <stdio.h>
#include <osbind.h>

/****************************************
** Constant Declarations:
*/

#define BACK_SLASH 92
#define NAME_MAX   64
#define LINE_MAX   132
#define FALSE      0
#define TRUE       1
#define TAB        9
#define NULL       0
#define SPACE      32
#define CR         13
#define LF         10
#define BELL       7


/****************************************
** GLOBAL VARIABLES:
*/

int Tabs,Tab_Sum;
char Filename[NAME_MAX];

char Oops[] = "DeTab call format: filename [tab size]\n\0";
char Ferr[] = "Unexpected file ERROR.\0";
char Done[] = "DeTab FINISHED, exit now.\n\n\0";
char Temp[] = ".DET\0"; /* Detabbed filename */

/****************************************
** MAIN PROGRAM
*/


main ( argc, argv )
int argc;
char *argv[];

{ /* main begin*/

int j, i, eof;
FILE *fd, *fd1;
char outfile[NAME_MAX], line[LINE_MAX], c, k;
char *name;

for (i=0; i < NAME_MAX; i++)
  outfile[i] = 0;

/****************************************
** fetch filename from input line,
** if none is given remind user
** of proper call format and exit.
*/

Tabs = 8;
```

Matthew Ratcliff is an Electrical Engineer at McDonnell Aircraft in St. Louis with the perfect job. He gets paid to program—in C and PL1—software for a high-tech imaging system used to evaluate ultrasonic test data taken on fighter aircraft parts. When he isn't programming at McDonnell or hacking at home, he's teaching his two young sons how to play video games and the art of computer-speak.

```
printf("DeTab by Mat*Rat.\n\n");

if (argc < 2)
  {
  printf( Oops );
  printf("Enter filename and optional tab size:  ");
  name == 0;
  while (name == 0)
    name = gets(Filename);
  i = Clfind(SPACE, Filename);
  if (i > 0)
    {
    Tabs = atoi(&Filename[i+1]);
@Filename[i] = 0;
@}
  }
else
  strcpy ( Filename, argv[1] );


if (argc >= 3)
  Tabs = atoi( argv[2] );

if (Tabs == 0)
  Tabs = 8;

i = Clfind( '.', Filename );        /* Check for EXTENDER */

if (i < 0)
  i = strlen(Filename);

for(j=0; j < i; j++)
  outfile[j] = Filename[j];

outfile[i] = 0;

strcat(outfile, Temp);

printf("Tab Size = %3d\n\n", Tabs);
printf("Detab file: %s\n\n", Filename);
printf("Output to: %s\n\n", outfile);

Tab_Sum = 0;

fd = fopen ( Filename, "r\0" );

if (fd == NULL)
  {
  Prompt( Ferr );
  exit(1);
  }

Fdelete ( outfile );     /* Erase if already exists */

fd1 = fopen( outfile, "w\0" );

if (fd1 == NULL)
  {
  Prompt( Ferr);
  exit(1);
  }

eof = FALSE;

while (eof == FALSE)
  {
  eof = Get_Line ( line, fd );
  if (eof == FALSE)
    {
    Detab ( line );
    Write_Line( line, fd1 );
    }
  }

fclose( fd );
fclose( fd1 );

printf("Total tab characters removed: %d\n\n", Tab_Sum);
Prompt( Done );

exit(1);

} /* main end */
```

```
/****************************************
** Clfind ( c, str ) - find the LAST
** occurance of the character C in the
** string STR.  If found, return the index
** pointing to it. If not, return a -1.
*/

Clfind ( c, str )
char c, str[];

{ /* Clfind begin */

int i, j, ptr;
char b;

j   = strlen ( str );

ptr = -1;

if (j > 0)
  { /* begin if */
  i = j-1;
  while( (i >= 0) && (ptr < 0) )
    { /* begin while */
    if ( ( b = str[i] ) == c)
      ptr = i;
    i--;
    } /* end while */
  } /* end if */
return ( ptr );

} /* Clfind end */


/****************************************
** Detab( line ) - convert tab characters
** in the line to the proper number
** of space characters according to the
** global tab size of Tabs.
*/

Detab ( line )
char line[];

{ /* Detab begin */

int i, j, tsz;
char str[LINE_MAX], c;

strcpy(str, line);

j   = 0;
i   = 0;
tsz = Tabs;

while ( j < LINE_MAX)
  { /* while begin */
  c = str[i++];
  switch ( c )
    { /* switch begin */
    case ( NULL ):
      line[j] = NULL;
      j = LINE_MAX;
      break;

     case ( TAB ):
@   Tab_Sum++;
        while ( ( tsz > 0) && (j < LINE_MAX) )
          {  /* while begin */
          line[j++] = SPACE;
          tsz--;
          } /* while end */
        tsz = Tabs;
        break;

      default:
        line[j++] = c;
        tsz--;
        if (tsz <= 0)
          tsz = Tabs;
        break;
      } /* switch end */

  } /* while end */
```

PROGRAM LISTINGS

```
} /* Detab end */

/****************************************
** Get_Line ( line, fd ) - fetch line from
** open input file pointed to by fd. Return
** eof flag. TRUE if EOF reached, else FALSE.
*/

Get_Line ( line, fd )
char line[];
FILE *fd;

{ /* Get_Line begin */

int i;
char *str;

str = fgets(line, (int)LINE_MAX, fd);

i = Clfind( CR, line);

if (i >= 0)
  line[i] = (char)NULL;

i = Clfind( LF, line);

if (i >= 0)
  line[i] = (char)NULL;

i = feof( fd );

return( i );

} /* Get_Line end */

/****************************************
** Write_Line ( line, fd ) - Write the
** line to the file pointed to by fd.
*/

Write_Line ( line, fd )
char line[];
FILE *fd;

{ /* Write_Line begin */

int i;

i = NULL;

i = fprintf(fd, "%s\n", line );

} /* Write_Line end */

/****************************************
** Prompt ( line ) - print line to display
** and then wait for RETURN to continue.
*/

Prompt ( line )
char line[];

{ /* Prompt begin */

char c;

printf("%s", line);

printf("\nPress [RETURN] ?");

putchar ( BELL );

c = getchar();

} /* Prompt end */
```

A DeTab Utility END

ST USER

by Arthur Leyenberger

Guess who I've been seeing a lot of on television commercials these days? Our old friend, Alan Alda. You know, "Mr. Personality," "Mr. appeal to all genders, ages and demographic groups." He's pushing IBM PS/2 systems.

The last time we saw Mr. Alda in a TV computer ad, it was 1984, and he was talking about Atari XL computers and how easy they were to set up and use. This was way back when Atari used to *run* commercials, not just say they were *going* to run commercials. In another words, this was in the days of Atari, Inc. not the Atari Corp.

And, boy oh boy, were those exciting days. The 1450XLD computer was almost out, there was a CP/M box shown at the June CES (Consumer Electronics Show), there was even Mindlink, XL expansion boxes, and a host of other products too numerous or painful to mention. Also, those were the days when Warner's Atari was losing money by the carload.

Anyway, Mr. Alda is once again using his supposed appeal to peddle the new IBM PS/2 line of computers. In this "slice of life" commercial, Mr. Alda plays an office worker listening to a coworker extol the virtues of the new computers, supposedly to a computer-naive Mr. Alda. After the guy finishes talking about all these features, Mr. Alda mentions a blurb about OS/2 (the soon-to-appear operating system for ATs and 386 PCs), letting us know that he isn't naive after all. The other guy is surprised, we're surprised—and I'm ready to throw up.

Why does IBM, or any manufacturer for that matter, use these inane commercials to try to get their message across? IBM did the same thing within the last year on radio. They had the *MASH* TV-show actors talking about the PS/2 line of computers in radio ads. These actors have no reason to know anything about computers, are not necessarily credible when talking about computers and fail to add anything to the commercials as far as I'm concerned.

I'm not picking on just IBM. Also in the last year or so, Dom DeLouise was the central figure in a rash of TV commercials for some computer maker. He played the fool in a foolish commercial. The ads were so bad, I was embarrassed to watch, and as a result, can't even remember who the company was.

I feel the same about Alan Alda's Atari computer commercials of 1984. What was the point? I have no problem if a famous actor or celebrity is used as a mouthpiece for a company in their commercials. Let them read the lines; they may be pleasant to look at and that's that. But when the famous person is supposed to be knowledgeable in the area, the commercial falls flat on its face and in fact begins to work against itself.

Someday, if I live long enough to see it, Atari Corp. will run computer TV commercials. It will be interesting to see what type of ads they will be. The sports celebrity tie-in commercials with video games look good and I imagine have been effective. If the same folks who produced these ads do the computer TV ads, maybe we'll see some good stuff. Maybe, someday.

Standard Wars

Not standard wars really, but wars about standards. The computer industry is less then a decade old officially and there have been a number of fights about computer standards. In the early days of the late 1970s, there was a standard called the S-100 bus. This was a hardware standard used in the early computers so that any manufacturer's add-in board would fit and work in a user's computer, assuming it used the S-100 bus. The S-100 bus standard lasted a long time until IBM introduced the PC in 1982. If you search the electronic mail order houses, you can *still* find S-100 boards.

There is a debate currently being waged in the PC arena. It centers around the standard bus used for add-in cards for IBM PCs and clones. In one corner, is IBM itself, with their new line of computers: the PS/2. The PS/2 uses a new type of add-in card (and interfacing architecture), called the MicroChannel Architecture (MCA). MCA is used on each and every IBM PS/2 computer and when PS/2 clones become available, they too will use the MCA.

In the other corner is the AT-standard bus that has been used for several years by IBM itself as well as all of the PC clone manufacturers in the existing AT-class computers. Existing AT-class computer users have very little reason to switch to the PS/2 machines since they can upgrade to 386-class machines and use many of their existing add-in boards.

What does all of this have to do with ST users or Atari in general, you ask? Plenty. If Atari plans to remain competitive, it will eventually introduce new computers. In fact, it is already rumored that Atari is working on a 68030 microprocessor-based computer. Whether this machine uses either of the above mentioned bus standards or uses its own proprietary bus is anyone's guess. In any case, the type of bus selected will influence third-party developers' decisions to support or not support the new machine.

The fight over the future of the AT-class and PS/2 machine's bus and architecture standards is also an example of what can be done when manufacturers work together. Atari is well noted for not working with anybody. If Atari were to have an open architecture, meaning that other company's add-in boards could be inserted into Atari computers to provide more memory and other functions, the Atari user would benefit. Further, if Atari would actually cooperate with third-party developers so that additional products could work with the Atari computer, the introduction of a new computer with the Atari badge on it would be received by potential users much more favorably.

One of the major complaints about the existing ST line of computers is that the architecture is closed. For example, if you want more memory for your computer, Atari suggests you buy the newest machine. Or, you can kludge a memory addition to the computer at the risk of voiding the warranty. Not only does adding memory void the warranty, but it also requires an electronics technician or electronics knowledge to perform. The Atari ST could be a much more successful computer if there was an easier way to add memory to the computer.

Happy Anniversary

It may be difficult to believe but this month marks the fourth anniversary of the "new Atari." It was January 1985 when Atari debuted the ST computer at the Winter Consumer Electronics Show. It seems like eons ago, doesn't it?

The Tramiels bought Atari from Warner in July 1984. For the rest of that year, very little information was forthcoming from the new company other than (shades of the movie *2010*) "something wonderful is going to happen." And it did! Not only did the ST computer debut in January, 1985, but so did the reworked 65XE and 130XE 8-bit computers.

Many of us were dying to get hold of an ST. They started to become available in the Spring of 1985. For the rest of the year, most of us who were fortunate enough to have an ST, ran demos of everything from the "bouncing ball" to 4xForth, to music programs.

Toward the end of that year and well into the following year there was still a dearth of ST software. *DEGAS* was one of the few programs available, and a good one at that. We were scrambling to get a word processor and telecommunications program. One of the first word processors, *Express*, was a joke. *ST Intercom* was a terminal program that was flawed by its copy protection and minimal company support. *ST Talk* was a good, if minimal, telecommunications program that many of us used for a long time. One of its strengths was a rock solid Xmodem downloading capability.

The ST initially came with a couple of programs. *1st Word* was a usable program, especially since it was given out free with the computer. ST Basic was, and still is, a poor excuse for a computer language. At least there was one game for the new computer and it was fun to play: *Asteroids* in black and white.

So here we are, four years after the introduction of the computer that was billed as having "Power Without The Price." The future of the ST in the United States is being shaped right now. Atari Corp. has, once again, the opportunity to demonstrate what they are capable of doing. The introduction of the ST blew everyone away four years ago. Now it'sup to Atari to do it again by concentrating their efforts in the U.S. market. I hope Atari is able to do it again.

The introduction of the ST blew everyone away four years ago. Now it's up to Atari to do it again by concentrating their efforts in the U.S. Market.

Late Breaking News

Just as I was about to upload this month's column to DELPHI, I found out that Neil Harris has left Atari. Neil has been with Jack Tramiel a long time, since the Commodore days. He has had a number of jobs at Atari Corp. including product manager, public relations manager, sales manager, and overall good-will ambassador.

Of late, Neil has spent a considerable amount of time on DELPHI, CompuServe and GEnie representing Atari and, to be honest, taking a lot of abuse from users. He has had a positive influence and has, given the constraints of his position, tried to keep users abreast of what was happening in Sunnyvale.

Neil's departure from Atari comes at an interesting time. It was just a few days before a major on-line conference was to be held on CompuServe with the management of Atari. Further, it comes about a month before the supposed big Atari presence at the Winter COMDEX.

I don't know the particulars regarding Neil's move, but since he was one of the few credible Atari spokespersons, his absence will be strongly felt. In addition, it remains to be seen if Atari will replace him and with whom. Much of the Atari on-line support was Neil's idea, and I have to wonder if Jack and the boys will feel it necessary to continue in this direction.

Neil will be taking a position with the GEnie Information Service. He will apparently no longer be involved with the Atari community but will still be on-line under his own name. I wish Neil the best of luck in his new job and thank him for trying, especially in spite of his former employer, to be a supporter of the Atari user.

How Do You Communicate?

As mentioned above, ST Talk by QMI was in my opinion the first solid terminal program for the ST. QMI is a super company. They first introduced ST Talk in a prerelease form as shareware on several of the national information services. During the time that version .97 was available on-line, users were able to use it, test it and make recommendations for improvement. Once the program was finished, it was sold for a very reasonable $20 with upgrades available for $5.

Now QMI finally has brought out the sequel to ST Talk, and it is called *ST Talk Professional* (QMI, P.O. Box 179, Liverpool, NY 13088). It has a host of new features, including full GEM operation. With mouse control and drop-down menus, all of the commands are readily available. If you don't want to use the mouse, you can use alternative keystrokes and program up to 40 programmable functions.

You can easily send and receive files with all of the popular transfer protocols. The program now includes archive and unarchive capabilities from within the program, and there is even a background file-transfer accessory (available separately) to transfer files while you are using another program. The program features a capture buffer to save incoming text, which can later be edited by the built-in word processor.

ST Talk Professional also features auto-dialer, script language, disk utility and type-ahead capabilities. Built-in help menus make it easy to learn the program and, as usual, QMI offers voice and BBS telephone support as well as on-line support through CompuServe.

ST Talk Professional sells for $40. If you are looking for a capable terminal program that has the features to meet your needs, check out ST Talk Professional from QMI.

There is another telecommunications program that has been around for a while but has only recently started getting some use on my ST. It is *Interlink ST* (Intersect Software Corp., 3951 Sawyer Road, Suite 108, Sarasota, FL 33583). Interlink ST is a *very* complete terminal program and has become one of my favorites.

ST Talk Pro also features auto-dialer, script language, disk utility and type-ahead capabilities with built-in help menus making it easy to learn the program.

One thing I always do when I get a new program is to attempt to use it right away without reading the manual. If I can't seem to make head nor tails out of it, the program, documentation and packaging get thrown into the box under the table for later use. "Later use" usually means maybe I'll get to it before the next ice age.

Interlink passed the test without any difficulty. I copied all of the files from the distribution disk to my hard disk and clicked on the file called "interlin.prg." What appeared on the screen was the first of two main screens. This screen, called the main menu, displays the usual array of drop-down menu titles across the top of the screen and four boxes at the bottom of the screen containing the current status and program option settings.

You can immediately go on-line in one of three ways: select the "File" menu and click on "go on-line", press the F1 function key or click the right mouse button. You'll spend most of your time in the on-line screen, but it is just as easy to get back to the main menu at any time by clicking the right mouse button or pressing the Undo key. Providing both keyboard and mouse methods for navigating through the program shows the attention to detail that this program has.

From the on-line screen, you can see how long you have been on-line, whether you are currently connected to a host system, what baud rate you have selected, the duplex mode, whether the buffer is on or off and how much space is available in the buffer. In addition, the bottom of the display tells you if you are in manual mode or if the recorder (more on that later) is active.

The type-ahead buffer is handy for composing messages or responses while you are still receiving information. Then, the entire buffer can be sent with just one keystroke or held for later use. And you can even set the size of the buffer.

One of the best parts of Interlink ST is the recorder. When activated, the program stores every prompt from the host system and your response to it. Then, when played back, it will send the same response that you did initially. This is a great way to automate the sign-on process of bulletin boards and information services, as well as saving the steps necessary for downloading or uploading files.

Interlink ST has a bunch of built-in commands that simply make life in the ST lane easier. Common disk commands such as delete, copy, format, rename, directory, create folder and drive path are all accessible from within the program. You can even show the contents of files and print them. There is an automatic phone dialer, programmable function keys, translation tables and much more. Further, another program can be run from within Interlink.

> **One of the best things about Interlink is that the program will never be outdated since the menu to select file-transfer protocol has a box mysteriously labeled "?????."**

File transfers can be performed using a number of protocols such as Xmodem, ASCII transfer, etc. One of the best things about Interlink is that the program will never be outdated since the menu to select file-transfer protocol has a box mysteriously labeled "?????."

Clicking on this box allows you to select, from a file-selector box, external protocols that are simply files on the disk. As new protocols are developed and made available, they can be used by Interlink ST.

Terminal emulation supports VT52 or that same "?????." I got on CompuServe just to see how many emulation files were available (the ATARIVEN section of PCS58, Intersect Software) and found a IBM 3101 emulator (useful for some mainframe communications), a VT100 emulator, an ATASCII graphics emulator (for full Atari 8-bit terminal emulation), a VT52 with Quick B file transfer and IBM PC ANSI graphics emulation. Intersect is serious about supporting their product.

I'm out of space for this month but I could go on and on about Interlink ST. Did I mention that it also serves as a BBS? How about using the recorder for making macros? Oh, yes, it was written by Randy Mears. Great job, Randy.

See you all next time. ■

(from page 41)

Read and Status. For starters, let's repeat our last experiment. Select Item 1 (Write) and press Return. The program will prompt you for the data you wish to write. Type 0, press Return and once again check that all data outputs are zero volts. Then, press Return to resume and select Item 1 again. This time, enter FFFF (all entries in hex!) and observe that all data outputs are at five volts.

Next, select Item 3 (Status). The IDR and ODP flags will be read from the interface and displayed. The IDR flag is undefined right now since no data has yet been read from the port, but the ODP flag should indicate "SET."

Now, using the other temporary lead, momentarily ground the ACKnowledge pin on the interface, being careful not to touch anything else with the wire. Repeat the Status function, and ODP should be "CLEAR."

Now, let's test the Read function. Momentarily touch the IDA terminal with the ground wire, and again select Item 3. IDR should now definitely be "SET." Next, select Item 2, and the port data will be displayed. It should be $FFFF since all data input pins were unconnected when you strobed IDA. Remember that with TTL logic chips, an open connection is always considered logic Level 1. Selecting the Status function now shows that IDR has been CLEARed by the read function.

Now the next step is a bit tricky. Disconnect the first temporary lead from the multimeter, so that you have two "ground" probes free. With the first probe, touch the D0 pin on the input data bus, and while holding this wire in place, carefully touch IDA with the second. Now, put down the leads, select Item 2, and the data read should be $FFFE. This procedure will allow you to test all 16 data inputs by verifying that the hex data matches the pin you grounded when IDA is strobed. To help reduce this to a "two hands" operation, you can temporarily add the following line to the BASIC program:

```
735  goto 720
```

This will cause menu Item 2 to continuously read and print the port data, so you can keep testing with the ground probes while you keep your eye on the screen. Testing the output data leads is much easier, since you need only select Item 1 to write data and use the multimeter to check the results.

Suggestions for Further Development

Obviously, to gain maximum speed advantage from our interface, assembly language is needed. You can use Listing 2 as a guide, but note that the code *must* be executed in 68000 supervisory mode. Since BASIC does this for us when the CALL command is used, this listing does not include this task. Also, for maximum speed, it is possible to disable all interrupts by setting the IPL in the 68000 status register to seven.

To use the system from BASIC, all that is really needed are lines 100 through 190, and 1000 through 1040. Then, poke the variable *command* with one, two or three for the functions Write, Read and Status, respectively. For the write function, poke *cdata* with the 16-bit integer you want to write to the port. Finally, enter CALL CARTPORT.

For Read and Status commands, the data returned can be obtained by peeking *cdata*. Never poke the cartridge address range, as this will cause a bus error and crash the system. Let the CALL function do the work, and you'll always be safe.

On the hardware side, there are many additions and modifications possible. For example, wiring in the six unused bits on the 74LS373 chip will enable the Status function to monitor six more external inputs. If this is done, you'll have to change the &h0003 to &h00ff in line 1040 of Listing 1, to allow eight bits to be monitored by the Status function. Also, if handshaking is not needed, LS373 chips can be substituted for the LS374s in the data input section. This way, strobing the IDA pin will not be necessary.

It is also possible to create more functions by adding steering logic to sense word-size access of the ROM4 address range. This, as well as "even byte" access of ROM3, has not been utilized. My only caution here is that you should not make any ST side signal drive more than two or three TTL gates without some kind of buffering.

As far as application goes, the list is endless. Instant data transfers between two STs, PROM burners, data acquisition and supervisory systems are all possible. Don't be afraid to experiment! And, drop me EMAIL if you come up with anything interesting (CompuServe: 73637,317).

PARTS LIST

Item	Quantity	Possible Source
74LS374	4	Local electronic distributor
74LS373	1	Local electronic distributor
74LS74	1	Radio Shack
74LS32	1	Radio Shack
74LS02	1	Radio Shack
40-pin headers	2	Radio Shack
Prototype board	1	Douglas Electronics 718 Marina Blvd, San Leandro, CA 94577

It is possible to create more functions by adding steering logic to sense word-sized access of the ROM4 address range.

CARTRIDGE PORT
Listing 1
ST BASIC

```
10 rem CARTRIGE PORT TEST PROGRAM
20 rem
30 rem Randy Constan, 1987
40 rem
50 rem
90 rem INITIALIZE MACHINE LANGUAGE STR
ING
100 def seg = 0                  'set pee
k/poke for 2 bytes
110 a$ = "                       " '20 sp
aces
115 a$ = a$+a$+a$+a$            'reserve
 80 bytes of space
120 restore 1000
130 b = varptr(a$)               'get add
ress of string a$
140 for i = 0 to 78  step 2      'put mac
hine code program into string
150 read a: poke b+i,a
160 next i
165 rem
170 command = b          'define command
 area
180 cdata = b+2          'define data ar
ea
190 cartport = b+4       'define beginni
ng of asm program
200 rem
300 rem                  'set up menu
310 fullw 2
320 clearw 2
330 gotoxy 10,01:print "Cartrige Port
test program"
340 gotoxy 12,03:print "1 - Write port
 data"
350 gotoxy 12,04:print "2 - Read port
data"
360 gotoxy 12,05:print "3 - Read port
status"
370 gotoxy 10,08:input "Enter command:
 ",x
380 poke command,x
390 if x<1 or x>3 goto 320
400 on x gosub 500,700,800
410 gotoxy 10,16:input "Press return t
o resume",h$
420 goto 320
430 rem
500 rem  DATA WRITE TO PORT
501 rem
510 gotoxy 10,10:input "Enter write da
ta (0-FFFF): $",h$
520 hh$ = "&H"+h$:poke cdata,val(hh$)
530 call cartport:return
540 rem
550 rem
700 rem DATA READ FROM PORT
720 call cartport: x%=peek(cdata)
730 gotoxy 12,10: print"current data:
",hex$(x%)
740 return
750 rem
800 rem STATUS READ
810 call cartport
820 x%=peek(cdata)
830 hh$="CLEAR":if x%>=2 then x%=x%-2:
hh$="SET"
840 h$ ="CLEAR":if x% then h$="SET"
850 gotoxy 12,10:print "   input data
ready: ",h$
860 gotoxy 12,11:print "output data pe
nding: ",hh$
870 return
900 rem
910 rem machine code from listing 2
920 rem
1000 data &h0000,&h0000,&h41fa,&hfffa,
&h0c50,&h0001,&h670e,&h0c50
1010 data &h0002,&h671a,&h0c50,&h0003,
&h6720,&h4e75,&h227c,&h00fa
1020 data &h0000,&h4280,&h303a,&hffdc,
&h1231,&h0800,&h4e75,&h43fa
1030 data &hffd2,&h32b9,&h00fb,&h0000,
&h4e75,&h43fa,&hffc6,&h1039
1040 data &h00fb,&h0001,&h0240,&h0003,
&h3280,&h4e75,&h0000,&h0000
```

CARTRIDGE PORT
Listing 1
Checksum Data

```
10 data 519,724,336,728,730,631,12,589
,646,961,5876
130 data 402,907,637,281,846,454,891,5
58,825,593,6394
310 data 337,376,49,232,862,429,14,321
,4,20,2644
410 data 745,395,838,510,832,700,100,3
70,843,846,6179
700 data 474,428,629,354,850,486,363,3
16,709,285,4894
850 data 735,480,365,839,521,845,918,9
22,986,156,6767
1040 data 715,715
```

CARTRIDGE PORT
Listing 2
Assembly

```
                ;cartrige port test utility for ST BASIC

command: dc.w 0 ;space for command entry
data:    dc.w 0 ;space for I/O data

start:
 lea command(pc),a0         ;a0 -> command word
 cmp #1,(a0)                ;determine command.
 beq.s write
 cmp #2,(a0)
 beq.s read
 cmp #3,(a0)
 beq.s status
 rts                        ;exit if invalid!

write:
 move.l #$FA0000,A1         ;-> ROM4 address range
 clr.l d0
 move.w data(pc),d0         ;fetch word to write
 move.b 0(a1,d0.l),d1       ;fake a write to
 rts                        ;the port and exit!

read:
 lea data(pc),a1            ;fetch data address.
 move.w $FB0000,(a1)        ;get port data
 rts                        ;and exit

status:
 lea data(pc),a1            ;fetch data address.
 move.b $FB0001,d0          ;read odd byte
 and #3,d0                  ;mask off unused bits,
 move d0,(a1)               ;and store status.
 rts
```

TRANS-WARP DRIVE
LIST 1
ST-BASIC

```
100 OPEN"R",#1,"A:TWARP.PRG",16:FIELD#1,16 AS B$
110 A$="":FOR I=1 TO 16:READ V$:IF V$="*" THEN 140
120 A=VAL("&H"+V$):PRINT "*";:A$=A$+CHR$(A):NEXT
130 LSET B$=A$:R=R+1:PUT 1,R:GOTO 110
140 CLOSE 1:PRINT:PRINT "ALL DONE!"
1000 data 60,1A,00,00,09,C8,00,00,01,CC,00,00,00,62,00,00
1010 data 00,00,00,00,00,00,00,00,00,00,00,00,4E,F9,00,00
1020 data 00,0C,4E,F9,00,00,09,3E,2A,4F,2A,6D,00,04,20,2D
1030 data 00,0C,D0,AD,00,14,D0,AD,00,1C,D0,BC,00,00,21,00
1040 data 22,00,D2,8D,C2,BC,FF,FF,FF,FE,2E,41,2F,00,2F,0D
1050 data 42,67,3F,3C,00,4A,4E,41,DF,FC,00,00,00,0C,20,6D
1060 data 00,18,22,6D,00,18,D3,ED,00,1C,20,2D,00,14,53,80
1070 data 6F,06,13,20,51,C8,FF,FC,20,6D,00,10,20,2D,00,1C
1080 data 53,80,42,18,51,C8,FF,FC,28,6D,00,10,D9,ED,00,1C
1090 data 29,4D,FF,FC,2A,6D,00,08,4E,AD,00,06,20,6C,FF,FC
1100 data 48,68,00,80,4E,BA,08,7A,58,8F,2F,2C,FF,F6,3F,2C
1110 data FF,FA,4E,BA,07,9C,5C,8F,3F,3C,00,00,4E,BA,00,10
1120 data 3F,3C,00,41,3F,3C,00,02,4E,41,58,8F,4E,75,00,00
1130 data 00,00,20,3C,00,00,01,FF,20,6F,00,04,22,6F,00,08
1140 data 12,D8,51,C8,FF,FC,4E,75,4E,56,00,00,20,2E,00,08
1150 data 52,80,20,40,10,10,48,80,C0,7C,00,FF,E1,40,20,6E
1160 data 00,08,12,10,48,81,C2,7C,00,FF,80,41,4E,5E,4E,75
1170 data 4E,B9,00,FC,1C,BE,70,F5,4E,B9,00,FC,1A,34,4E,B9
1180 data 00,FC,1C,14,4E,F9,00,FC,15,B4,4E,B9,00,FC,1E,BE
1190 data 70,F5,4E,B9,00,FC,1C,48,4E,B9,00,FC,1E,14,4E,F9
1200 data 00,FC,17,98,4E,B9,00,FC,1C,BE,70,F6,4E,B9,00,FC
1210 data 1A,34,30,2D,09,C6,53,40,80,6D,09,C4,80,6D,09,C8
1220 data 66,08,70,02,4E,B9,00,FC,1C,F6,4E,B9,00,FC,1C,14
1230 data 4E,F9,00,FC,16,A8,4E,B9,00,FC,1E,BE,70,F6,4E,B9
1240 data 00,FC,1C,48,30,2D,09,C6,53,40,80,6D,09,C4,80,6D
1250 data 09,C8,66,08,70,02,4E,B9,00,FC,1E,F6,4E,B9,00,FC
1260 data 1E,14,4E,F9,00,FC,18,84,4E,56,FF,EE,48,E7,0F,10
1270 data 3D,7C,FF,FF,FF,F6,28,7A,FF,16,0C,6E,00,02,00,12
1280 data 6D,22,3F,2E,00,12,3F,2E,00,10,3F,2E,00,0E,2F,2E
1290 data 00,0A,3F,2E,00,08,20,6C,FF,F2,4E,90,DE,FC,00,0C
1300 data 60,00,05,CE,20,6C,FF,D2,30,10,66,0A,30,3C,FF,FE
1310 data 48,C0,60,00,05,BC,20,2E,00,0A,66,18,30,2E,00,0E
1320 data 20,6C,FF,DA,D0,EE,00,12,10,80,30,3C,00,00,48,C0
1330 data 60,00,05,9E,0C,6E,00,02,00,08,6C,32,3F,2E,00,12
1340 data 20,6C,FF,E6,4E,90,54,8F,2D,40,FF,FC,0C,80,00,00
1350 data 00,00,67,1A,0C,AE,00,00,00,02,FF,FC,66,08,2D,7C
1360 data FF,FF,FF,F2,FF,FC,20,2E,FF,FC,60,00,05,64,30,2E
1370 data 00,12,C1,FC,00,20,20,6C,FF,E2,D0,C0,41,D0,26,48
1380 data 20,2E,00,0A,C0,BC,00,00,00,01,0C,80,00,00,00,01
1390 data 57,C0,C0,7C,00,01,3D,40,FF,FA,30,2B,00,16,66,0C
1400 data 30,3C,00,09,37,40,00,16,37,40,00,18,60,00,05,14
1410 data 30,2E,FF,FA,67,06,20,2C,FF,D6,60,04,20,2E,00,0A
1420 data 2D,40,FF,F2,3E,2E,00,10,48,C7,8F,EB,00,16,3A,2E
1430 data 00,10,48,C5,8B,EB,00,16,48,45,30,05,B0,6B,00,18
1440 data 6C,04,42,46,60,08,3C,3C,00,01,9A,6B,00,18,30,2E
1450 data FF,FA,67,06,38,3C,00,01,60,18,30,2B,00,18,90,45
1460 data B0,6E,00,0E,6C,08,38,2B,00,18,98,45,60,04,38,2E
1470 data 00,0E,52,45,30,2E,00,08,C0,7C,00,01,67,00,02,36
1480 data 20,2E,FF,F2,B0,AE,00,0A,67,0E,2F,2E,FF,F2,2F,2E
1490 data 00,0A,4E,BA,FD,BE,50,8F,30,04,B0,6B,00,18,66,00
1500 data 01,9E,0C,45,00,01,66,00,01,96,30,07,0C,6B,00,09
1510 data 00,18,66,06,32,3C,00,04,60,04,32,3C,00,05,48,C0
1520 data 81,C1,48,40,3D,40,FF,F8,0C,6B,00,09,00,18,66,22
1530 data 30,2E,FF,F8,E3,80,41,EC,FF,C6,D0,C0,3D,50,FF,F0
1540 data 30,2E,FF,F8,E3,80,41,EC,FF,BE,D0,C0,3D,50,FF,EE
1550 data 60,20,30,2E,FF,F8,E3,80,41,EC,FF,B4,D0,C0,3D,50
1560 data FF,F0,30,2E,FF,F8,E3,80,41,EC,FF,AA,D0,C0,3D,50
1570 data FF,EE,30,2E,FF,F6,B0,47,66,00,00,84,30,2C,FF,CE
1580 data 67,3E,30,2B,00,18,90,6E,FF,F0,52,40,3F,00,3F,06
1590 data 3F,07,3F,2E,FF,F0,3F,2E,00,12,42,A7,20,2E,FF,F2
1600 data 32,2E,FF,F0,53,41,C3,FC,02,00,48,C1,D0,81,2F,00
1610 data 4E,BA,FD,A4,DE,FC,00,12,48,C0,2D,40,FF,FC,60,3C
1620 data 30,2B,00,18,90,6E,FF,F0,52,40,3F,00,3F,06,3F,07
1630 data 3F,2E,FF,F0,3F,2E,00,12,42,A7,20,2E,FF,F2,32,2E
1640 data FF,F0,53,41,C3,FC,02,00,48,C1,D0,81,2F,00,4E,BA
1650 data FD,34,DE,FC,00,12,48,C0,2D,40,FF,FC,60,3E,30,2B
1660 data 00,18,90,6E,FF,F0,52,40,3F,00,3F,06,3F,07,3F,2E
1670 data FF,F0,3F,2E,00,12,42,A7,20,2E,FF,F2,32,2E,FF,F0
1680 data 53,41,C3,FC,02,00,48,C1,D0,81,2F,00,3F,3C,00,09
1690 data 4E,BA,04,D0,DE,FC,00,14,2D,40,FF,FC,0C,AE,00,00
```

PROGRAM LISTINGS

TRANS-WARP DRIVE

```
1700 data 00,00,FF,FC,66,56,30,2E,FF,E
E,67,50,30,2C,FF,CE
1710 data 67,26,3F,2E,FF,EE,3F,06,3F,0
7,3F,3C,00,01,3F,2E
1720 data 00,12,42,A7,2F,2E,FF,F2,4E,B
A,FC,EC,DE,FC,00,12
1730 data 48,C0,2D,40,FF,FC,60,24,3F,2
E,FF,EE,3F,06,3F,07
1740 data 3F,3C,00,01,3F,2E,00,12,42,A
7,2F,2E,FF,F2,4E,BA
1750 data FC,94,DE,FC,00,12,48,C0,2D,4
0,FF,FC,60,22,3F,04
1760 data 3F,06,3F,07,3F,05,3F,2E,00,1
2,42,A7,2F,2E,FF,F2
1770 data 3F,3C,00,09,4E,BA,04,4C,DE,F
C,00,14,2D,40,FF,FC
1780 data 3D,47,FF,F6,20,2E,FF,FC,66,4
6,20,6C,FF,DE,30,10
1790 data 67,3E,3F,04,3F,06,3F,07,3F,0
5,3F,2E,00,12,42,A7
1800 data 2F,2C,FF,D6,3F,3C,00,13,4E,B
A,04,18,DE,FC,00,14
1810 data 2D,40,FF,FC,20,2E,FF,FC,66,1
6,2F,2C,FF,D6,4E,BA
1820 data FB,C8,58,8F,4A,40,67,08,2D,7
C,FF,FF,FF,F0,FF,FC
1830 data 60,00,01,E6,30,04,B0,6B,00,1
8,66,00,01,9E,0C,45
1840 data 00,01,66,00,01,96,30,07,0C,6
B,00,09,00,18,66,06
1850 data 32,3C,00,04,60,04,32,3C,00,0
5,48,C0,81,C1,48,40
1860 data 3D,40,FF,F8,0C,6B,00,09,00,1
8,66,22,30,2E,FF,F8
1870 data E3,80,41,EC,FF,C6,D0,C0,3D,5
0,FF,F0,30,2E,FF,F8
1880 data E3,80,41,EC,FF,BE,D0,C0,3D,5
0,FF,EE,60,20,30,2E
1890 data FF,F8,E3,80,41,EC,FF,B4,D0,C
0,3D,50,FF,F0,30,2E
1900 data FF,F8,E3,80,41,EC,FF,AA,D0,C
0,3D,50,FF,EE,30,2E
1910 data FF,F6,B0,47,66,00,00,84,30,2
C,FF,CE,67,3E,30,2B
1920 data 00,18,90,6E,FF,F0,52,40,3F,0
0,3F,06,3F,07,3F,2E
1930 data FF,F0,3F,2E,00,12,42,A7,20,2
E,FF,F2,32,2E,FF,F0
1940 data 53,41,C3,FC,02,00,48,C1,D0,8
1,2F,00,4E,BA,FB,3C
1950 data DE,FC,00,12,48,C0,2D,40,FF,F
C,60,3C,30,2B,00,18
1960 data 90,6E,FF,F0,52,40,3F,00,3F,0
6,3F,07,3F,2E,FF,F0
1970 data 3F,2E,00,12,42,A7,20,2E,FF,F
2,32,2E,FF,F0,53,41
1980 data C3,FC,02,00,48,C1,D0,81,2F,0
0,4E,BA,FA,E4,DE,FC
1990 data 00,12,48,C0,2D,40,FF,FC,60,3
E,30,2B,00,18,90,6E
2000 data FF,F0,52,40,3F,00,3F,06,3F,0
7,3F,2E,FF,F0,3F,2E
2010 data 00,12,42,A7,20,2E,FF,F2,32,2
E,FF,F0,53,41,C3,FC
2020 data 02,00,48,C1,D0,81,2F,00,3F,3
C,00,08,4E,BA,02,B4
2030 data DE,FC,00,14,2D,40,FF,FC,0C,A
E,00,00,00,00,FF,FC
2040 data 66,56,30,2E,FF,EE,67,50,30,2
C,FF,CE,67,26,3F,2E
2050 data FF,EE,3F,06,3F,07,3F,3C,00,0
1,3F,2E,00,12,42,A7
2060 data 2F,2E,FF,F2,4E,BA,FA,84,DE,F
C,00,12,48,C0,2D,40
2070 data FF,FC,60,24,3F,2E,FF,EE,3F,0
6,3F,07,3F,3C,00,01
2080 data 3F,2E,00,12,42,A7,2F,2E,FF,F
2,4E,BA,FA,44,DE,FC
2090 data 00,12,48,C0,2D,40,FF,FC,60,2
2,3F,04,3F,06,3F,07
2100 data 3F,05,3F,2E,00,12,42,A7,2F,2
E,FF,F2,3F,3C,00,08
2110 data 4E,BA,02,30,DE,FC,00,14,2D,4
0,FF,FC,3D,47,FF,F6
2120 data 20,2E,FF,F2,B0,AE,00,0A,67,0
E,2F,2E,00,0A,2F,2E
2130 data FF,F2,4E,BA,F9,BE,50,8F,0C,A
E,00,00,00,00,FF,FC
2140 data 6C,4C,3F,2E,00,12,20,2E,FF,F
C,3F,00,20,6C,FF,EA
2150 data 20,50,4E,90,58,8F,2D,40,FF,F
C,0C,6E,00,02,00,08
2160 data 6C,26,0C,AE,00,01,00,00,FF,F
C,66,1C,3F,2E,00,12
2170 data 20,6C,FF,E6,4E,90,54,8F,0C,8
0,00,00,00,02,66,08
2180 data 2D,7C,FF,FF,FF,F2,FF,FC,3D,7
C,00,63,FF,F6,0C,AE
2190 data 00,01,00,00,FF,FC,67,00,FB,7
C,0C,AE,00,00,00,00
2200 data FF,FC,6C,06,20,2E,FF,FC,60,2
6,30,04,48,C0,E3,80
2210 data E1,80,D1,AE,00,0A,30,04,D1,6
E,00,10,30,04,91,6E
2220 data 00,0E,30,2E,00,0E,66,00,FA,E
8,30,3C,00,00,48,C0
2230 data 4C,DF,08,F0,4E,5E,4E,75,4E,5
6,00,00,42,78,04,44
2240 data 42,B8,04,20,42,B8,04,3A,0C,B
9,11,20,19,85,00,FC
2250 data 00,18,67,12,0C,B9,04,22,19,8
7,00,FC,00,18,67,2C
2260 data 42,6C,FF,D0,60,66,29,7C,00,F
C,10,00,FF,E6,29,7C
2270 data 00,00,4D,CE,FF,E2,29,7C,00,0
0,4D,B8,FF,DA,29,7C
2280 data 00,00,16,7A,FF,D6,42,6C,FF,C
E,60,26,29,7C,00,FC
2290 data 11,E4,FF,E6,29,7C,00,00,75,7
0,FF,E2,29,7C,00,00
2300 data 75,5A,FF,DA,29,7C,00,00,16,D
A,FF,D6,39,7C,00,01
2310 data FF,CE,20,6C,FF,EE,29,50,FF,F
2,41,FA,F9,6C,20,08
2320 data 20,6C,FF,EE,20,80,39,7C,00,0
1,FF,D0,4E,5E,4E,75
2330 data 4E,56,00,00,41,FA,F8,78,20,8
C,48,7A,FF,5C,3F,3C
2340 data 00,26,4E,BA,00,BE,5C,8F,30,2
C,FF,D0,66,1C,48,6C
2350 data 00,00,3F,3C,00,09,4E,BA,00,C
6,5C,8F,3F,3C,00,01
2360 data 3F,3C,00,4C,4E,BA,00,B8,58,8
F,48,6C,00,3A,3F,3C
2370 data 00,09,4E,BA,00,AA,5C,8F,48,6
C,00,6E,3F,3C,00,09
2380 data 4E,BA,00,9C,5C,8F,48,6C,00,A
0,3F,3C,00,09,4E,BA
2390 data 00,8E,5C,8F,48,6C,00,D2,3F,3
C,00,09,4E,BA,00,80
2400 data 5C,8F,48,6C,01,04,3F,3C,00,0
9,4E,BA,00,72,5C,8F
2410 data 48,6C,01,36,3F,3C,00,09,4E,B
A,00,64,5C,8F,48,6C
2420 data 01,68,3F,3C,00,09,4E,BA,00,5
6,5C,8F,48,6C,01,9A
2430 data 3F,3C,00,09,4E,BA,00,48,5C,8
F,20,6C,FF,FC,20,28
2440 data 00,0C,D0,A8,00,14,D0,A8,00,1
C,D0,BC,00,00,01,00
2450 data 3F,3C,00,00,2F,00,3F,3C,00,3
1,4E,41,4E,5E,4E,75
```

```
2460 data 4E,75,29,49,FF,A6,29,4A,FF,A
2,29,5F,FF,9E,4E,4E
2470 data 22,6C,FF,A6,24,6C,FF,A2,2F,2
C,FF,9E,4E,75,29,49
2480 data FF,A6,29,4A,FF,A2,29,5F,FF,9
E,4E,41,22,6C,FF,A6
2490 data 24,6C,FF,A2,2F,2C,FF,9E,4E,7
5,29,7C,00,00,04,76
2500 data FF,EE,29,7C,00,00,04,04,FF,E
A,29,7C,00,00,04,44
2510 data FF,DE,29,7C,00,00,4D,B8,FF,D
A,29,7C,00,00,16,7A
2520 data FF,D6,29,7C,00,00,04,A6,FF,D
2,43,EC,FF,C6,32,FC
2530 data 00,01,32,FC,00,03,32,FC,00,0
5,32,FC,00,07,43,EC
2540 data FF,BE,32,FC,00,00,32,FC,00,0
2,32,FC,00,04,32,FC
2550 data 00,06,43,EC,FF,B4,32,FC,00,0
1,32,FC,00,03,32,FC
2560 data 00,05,32,FC,00,07,32,FC,00,0
9,43,EC,FF,AA,32,FC
2570 data 00,00,32,FC,00,02,32,FC,00,0
4,32,FC,00,06,32,FC
2580 data 00,08,4E,75,54,72,61,6E,73,2
D,57,61,72,70,20,44
2590 data 72,69,76,65,20,6E,6F,74,20,6
C,6F,61,64,65,64,2E
2600 data 0D,0A,57,72,6F,6E,67,20,54,4
F,53,20,52,4F,4D,20
2610 data 69,6E,73,74,61,6C,6C,65,64,2
E,0D,0A,00,00,0D,0A
2620 data 2A,2A,2A,2A,2A,2A,2A,2A,2A,2
A,2A,2A,2A,2A,2A,2A
2630 data 2A,2A,2A,2A,2A,2A,2A,2A,2A,2
A,2A,2A,2A,2A,2A,2A
2640 data 2A,2A,2A,2A,2A,2A,2A,2A,2A,2
A,2A,2A,2A,2A,0D,0A
2650 data 00,00,2A,20,20,20,20,20,20,2
0,20,20,20,20,20,20
2660 data 20,20,20,20,20,20,20,20,20,2
0,20,20,20,20,20,20
2670 data 20,20,20,20,20,20,20,20,20,2
0,20,20,20,20,20,2A
2680 data 0D,0A,00,00,2A,20,20,20,20,2
0,20,20,54,72,61,6E
2690 data 73,2D,57,61,72,70,20,44,72,6
9,76,65,2C,20,56,65
2700 data 72,73,69,6F,6E,20,35,2E,31,3
0,20,20,20,20,20,20
2710 data 20,2A,0D,0A,00,00,2A,20,43,6
F,70,79,72,69,67,68
2720 data 74,20,31,39,38,37,2C,31,39,3
8,38,20,44,61,6E,20
2730 data 4D,6F,6F,72,65,20,26,20,44,6
1,76,65,20,53,6D,61
2740 data 6C,6C,20,2A,0D,0A,00,00,2A,2
0,20,20,20,4E,65,77
2750 data 2C,20,49,6D,70,72,6F,76,65,6
4,20,61,6E,64,20,4D
2760 data 65,67,61,20,43,6F,6D,70,61,7
4,69,62,6C,65,21,21
2770 data 21,21,20,20,20,20,2A,0D,0A,00,0
0,2A,20,20,20,20,20
2780 data 20,20,52,65,73,65,74,73,20,6
6,6F,78,65,64,2E,20
2790 data 46,6C,6F,70,70,79,20,76,65,7
2,69,66,79,20,6F,66
2800 data 66,2E,20,20,20,20,20,2A,0D,0
A,00,00,2A,20,20,20
2810 data 20,20,20,20,20,20,20,20,20,2
0,20,20,20,20,20,20
2820 data 20,20,20,20,20,20,20,20,20,2
0,20,20,20,20,20,20
2830 data 20,20,20,20,20,20,20,20,20,2
A,0D,0A,00,00,2A,2A
2840 data 2A,2A,2A,2A,2A,2A,2A,2A,2A,2
A,2A,2A,2A,2A,2A,2A
2850 data 2A,2A,2A,2A,2A,2A,2A,2A,2A,2
A,2A,2A,2A,2A,2A,2A
2860 data 2A,2A,2A,2A,2A,2A,2A,2A,2A,2
A,2A,2A,0D,0A,00,00
2870 data 00,00,00,02,06,00,00,00,00,0
0,00,00,00,00,00,00
2880 data *
```

TRANSWARP DRIVE
LISTING 1
CHECKSUM DATA

```
100 data 639,544,391,421,536,662,562,7
75,906,170,5606
1050 data 861,776,873,11,77,14,31,765,
688,941,5037
1150 data 876,875,143,117,100,141,898,
1,192,921,4264
1250 data 51,69,85,771,995,21,778,962,
715,45,4492
1350 data 876,260,41,695,945,617,961,9
94,837,752,6978
1450 data 835,800,711,117,955,692,696,
895,320,351,6372
1550 data 187,328,168,892,51,32,222,87
2,48,102,2902
1650 data 115,901,148,916,123,140,935,
206,98,988,4570
1750 data 115,937,72,162,819,41,239,34
0,791,661,4177
1850 data 783,991,333,226,330,352,54,9
06,153,69,4197
1950 data 34,53,34,246,940,14,90,881,1
06,47,2445
2050 data 902,173,28,178,893,907,166,9
65,196,41,4449
2150 data 892,887,867,378,881,93,896,8
33,896,875,7498
2250 data 790,63,100,57,8,41,254,46,8,
24,1391
2350 data 844,937,896,982,913,884,872,
871,978,891,9068
2450 data 802,152,136,229,998,948,56,2
36,874,27,4458
2550 data 16,38,873,802,885,831,828,84
3,844,836,6796
2650 data 579,575,593,683,814,731,779,
743,827,751,7075
2750 data 851,834,628,809,882,661,576,
577,642,851,7311
2850 data 852,798,494,221,2365
```

TRANS-WARP DRIVE END

Attention Programmers!

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

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

Send your programs and articles to:
ST-LOG Magazine
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

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.

P.O. BOX 16927, N. HOLLYWOOD, CA 91615

Name ____________

Street Address ____________

City ________ State ______ Zip ______

ATTACH LABEL HERE

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

NeoDesk

Gribnif Software
P.O. Box 350
Hadley, MA 01035
(413) 584-7887
Medium or High Resolution
$29.95

Reviewed by Andy Eddy

It goes without saying that using a utility program should provide some benefit beyond that which you can gain normally. By the same token, it shouldn't cause more work or hardship than it alleviates. Using those rules as a yardstick, *NeoDesk* meets the former requirement reasonably, yet sometimes falls victim to the second.

For those who haven't heard of it yet, NeoDesk, by Gribnif Software, bills itself as an "alternative desktop." Loading NeoDesk after booting your ST brings up an entirely different look—the shape of the icons, the operation of disk copies, the basic function of the desktop is all changed.

A simple example is the rubber-banding effect when you drag your mouse pointer. With the original desktop you can only open the box downwards and to the right; NeoDesk allows stretching of the box in any direction. Not a drastic change, but an improvement just the same.

Another change—and certainly more of a change for the better—is the provision to pull program icons to the desktop. Your most frequently used files become an integral part of the desktop, and double-clicking on the icon operates just as if you had the window that contained that piece of software open and then double-clicked the program. This saves your having to open window after window (if your programs happen to reside within multiple folders) to find a program or document.

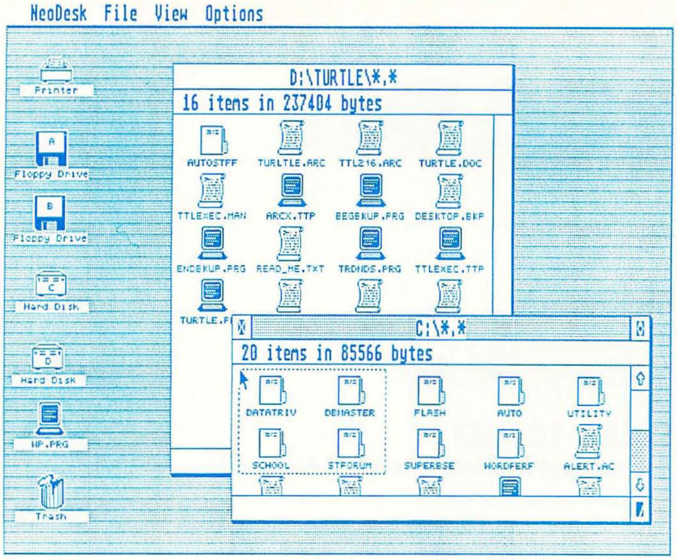
Icons are more descriptive as well. NeoDesk displays floppy drive icons, *in addition to* hard drive and RAMdisk symbols, making it look much different than the standard GEM desktop. Folder icons are the fanciest, appearing to have writing on them and opening up when highlighted. Also included in the package is an icon editor, for altering the icon's appearances to your own design.

There is an additional symbol on the screen, a *printer* icon, that works hand in hand with the Neoqueue print accessory. This program, when installed, lets the user drag up to ten files to the printer icon for spooling (handy when using another GEM program), as well as select the order of printing.

Windows themselves are changed also. The horizontal slider has been eliminated, with all partially displayed file icons wrapped similar to a word processor. If less than half of an icon can't be displayed in a window, it is carried to the next line. You are also permitted to have up to the GEM limit of seven windows open simultaneously.

The top line of the window shows the path as with any standard window; but if you highlight a file, the information on it is shown at the top, similar to the "Show Info" selection on the normal desktop—file size, read/write status and time/date stamp. In the same respect, highlighting multiple files will show combined data on the file group. This is handy for copying files, so you'll know ahead of time how much space is required. As a negative note, highlighting a file and a folder doesn't include the folder's size in the information line, so it can be somewhat misleading; this is no improvement over the standard desktop.

Copying with NeoDesk is done *smartly*. For example, when dragging one icon to another, NeoDesk analyzes whether the source and destination media match in size. If they do, NeoDesk may opt to copy with formatting; if not, it will treat the copy as if you dragged a group of icons to the destination drive. Copies and formatting can also be



"It must be remembered that NeoDesk is simply a *replacement* for the desktop; its functions don't carry into other programs."

halted with the UNDO key. And while on the subject of formatting, support is in place for a few different configurations of track and sector numbers, but strangely enough, the popular *Twister* format isn't one of those supported.

If you utilize a CLI (Command Line Interpreter), you can use batch files to do some things automatically; this is similar to using a script file under *Flash*. If you want to accomplish a particular procedure at boot time, for instance, you can set up a batch file called NEOAUTO.BAT. Similarly, NeoDesk adds another type of file type to the list: the .BTP, or Batch Takes Parameters, file. The .BTP file is much the same as the .TTP file in that you can enter a string of parameters that will be passed on at execution.

Another plus is that many of NeoDesk's functions can be called with Control key combinations, to reduce the time it takes to enact a process. An example of this comes from clicking on a file or group of files with the Control key held down simultaneously, which lets you rename files quickly. Using the Shift key to pick multiple files, in tandem with the Control key, will pop through all of the files to rename them in sequence.

The Ups And Downs

All of this is well and good, right? Not quite. You see, with every silver lining comes a cloud—in this case, a dark cloud. For starters, the loading of NeoDesk chips away at your available RAM to the tune of *145K*. That won't affect many Mega 2 or 4 users too adversely, but most of the ST contingent are using 520s and 1040s. The loss of 145K in one swoop—independent of other accessories or AUTO programs you may also use—is a major chunk, to say the least.

As an aside, a few experienced ST programmers have told me that a program of this scope shouldn't use nearly as much RAM. There are certain memory wasters that were programmed in, when simpler and leaner methods could have been used. Earlier I noted the use of Control key combos to enact NeoDesk functions. Strangely, the NeoDesk information screen that normally is brought up under the Desk header of the menu bar can also be caused to surface with a Control-A.

As a similar example, the sizeable custom dialogs, like the Edit Environment box or the running of a .TTP/.BTP file, surely take up a good slab of silicon. And NeoDesk doesn't have any check on whether it is in place already or not, so conceivably it could be run on top of itself, at a loss of 145K each time, until you run out of RAM space. It's not likely to happen accidentally, but proper error checking would be a better use of RAM than some of the other uses.

Another bug, though more obscure, was pointed out and explained by ST-LOG West Coast Editor Charles Johnson. Apparently, the SHEL_WRITE command in the AES (Application Environment Services)—the section of the operating system that oversees the menu bars, dialog boxes, windows and such—isn't handled properly by NeoDesk. This command tells the desktop to run another program when the current one is complete; for example, *Easy Draw* uses this function to run its OUTPRINT.PRG program when you select "Output" from the menu bar. For this reason, when you exit from Easy Draw, it does indeed drop back to NeoDesk; but when exiting NeoDesk to the regular desktop, OUTPRINT.PRG *then* tries to run. This results in bombs and a crash.

Finally, NeoDesk has a bunch of features that frankly aren't of much use to most users. For example, the dialog box that comes up when you run a .TTP or .BTP file lets you enter up to 128 characters of parameters. You can also add one command to the parameters to have the output of these files redirected to a file or the printer. Not everyday computing aids and not very useful to all but the most dedicated power users.

There is also an "Edit Environment" process that will pass information, such as paths for file locations, to another program. To date, only a few C language compilers support such a technique. With NeoDesk, you can pass up to ten environment strings from the desktop, so conceivably you could compile from the desktop. This segment of the program is only useful for those few who use compatible compilers, and even then is not likely to be used that heavily. Regardless, the feature takes up precious RAM whether it's used or not.

It must be remembered that NeoDesk is simply a *replacement* for the desktop; its functions don't carry into other programs. Many people spend little time at the desktop level, especially with so varied a selection of desk accessories (many in the public domain, available on local BBSs that support the ST, as well as in the databases of the ST SIG on DELPHI) that remain resident, can be called up from most applications and, most importantly, duplicate a portion of NeoDesk's features. Besides, from a program that bills itself as the "complete desktop alternative," it seems strange that it doesn't support low resolution usage.

Other limitations—which Gribnif acknowledge—actually serve to make NeoDesk *less* than the original desktop. A few examples are the ceiling of 18 icons you are limited to having on the desktop at once, a maximum of ten file icons that can be put on the desktop, and, most restricting, the condition of not more than 112 viewable files in a window.

On top of that, many seasoned ST users find that icons are onerous and clumsy, so they've switched to text mode to display window contents. Unfortunately NeoDesk, in its initial incarnation, doesn't support a text mode. The reason I say "initial incarnation" is because Gribnif Software's Mike Cohan, Rick Flashman and Dan Wilga noted in a DELPHI conference that a new version would be released in the fall to pacify some of the critics by adding many features—text mode among them. Of course, the additions would expand the RAM grabbing; but in their defense, they also intend to have a configuration accessory that would enable the user to minimize the memory usage to only those functions that are needed.

Granted, the previous description doesn't describe all that NeoDesk is capable of, and $29.95 is certainly a small sum to pay for the noble effort Gribnif has undertaken.

If you collect ST utilities religiously, then by all means pick up a copy. Personally, I see it as a program that you'll use a few times, then put on the shelf. If you do plan on buying it, keep in sight that it doesn't come without its negative points, coupled with the fact that Gribnif is planning on a substantial upgrade (which may not be out by the time you read this). As always: *caveat emptor*, let the buyer beware. ■

DynaCADD

DynaCADD
ISD Marketing, Inc.
2651 John St. Unit #3
Markham, Ont., Canada L3R 2W5
(416) 479-1880
One megabyte 512ST, 1040ST
or any Mega ST, Monochrome only $695.00

Reviewed by Ian Chadwick and Thom Weeks

***DynaCADD* is one of a rare breed of soft**ware. It is designed and implemented uniquely for the Atari ST, for the professional user. It isn't a crossbreed or a port from another system, with features and commands sorely limited to suit the impression that the ST market isn't a serious place to develop in. It is easily the most complex, fully featured program available in its category. In fact, its real competition doesn't even come from within the ST market, but from *AutoCAD*, the best-selling PC/MS-DOS program. At $695, DynaCADD is also in the stratosphere of ST pricing.

The price alone should be a warning to casual users. And if that doesn't deter you, the almost-bewildering array of commands and features should convince you. DynaCADD is meant for the professional market: engineers, architects, designers, draftsmen. It makes no concessions for amateurs, tinkerers or hackers.

DynaCADD is both a two- and a three-dimensional CAD package. In 2-D mode, it supports up to 256 layers. The screen can be subdivided into four windows, each of which can be used to display a different "view." Views can be rotated, moved or zoomed independent of one another.

The user interface follows all of the familiar GEM conventions, including windows, scroll bars and the mouse. It makes good use of the ST keyboard for Help and Undo features, although the latter is only good for the last action and does not permit backward stepping. Basically, the user interface is quite straightforward, though it takes some getting used to, since the screen is visually very "busy."

Due to the complexity of the program, the typical GEM menu tree isn't large enough to contain all of the commands and subcommands. Only global commands and utilities are available through the pull-down menus.

Most of the commands are displayed as icons on the "menu pads" along the left of the screen. These pads can be hidden to show the drawing screen at full size, if desired. The pads work in a tree hierarchy. That is, when you click on one, it changes the display of the lower pad(s) to show appropriate subcommands. The top pad doesn't change. There are also several unchanging icons along the top of the drawing screen.

While the icon approach makes the user interface easy, it can also lead to great confusion as you try to remember what the pictures stand for. To counter this, as you pass the pointer over any icon, the text version of the command is echoed in the top left corner, above the drawing area. Finally, for those who prefer, commands can be typed in. The command line interpreter not only accepts input, but records the program's messages and the X/Y/Z coordinates of the cursor (in the units of the drawing). The text screen can be viewed anytime, or input can be echoed to the printer or a disk file.

Going from one level of the interface to another is sometimes time-consuming, especially when you're deep within one part of the tree and want to back go to another branch. It requires backing up, then going forward several steps to the right command. This process can be avoided by typing the commands directly.

Another part of the user interface is a scientific calculator. This always appears whenever DynaCADD expects a numeric value. Numbers can be entered and manipulated as on any calculator, or stored in four calculator memories. This seems like overdoing things when a simple numeric entry is required. It would be better to offer a two-step process: a box for simple entries, and the calculator called up by the user only when the user desires it.

It would be easy to simply list DynaCADD's features, but that would consume the rest of this article. The program simply has more features than any of its competitors: 360 commands at the last count. For the professional, it does everything you need, plus a lot more.

Text is treated as an entity, much the same as the other "primitives": point, line, circle, arc, fillet (an arc placed between two lines), polygon, bezier curves, ellipse and ellipse arc, except for the special commands required to deal with it. It can be moved, rotated, scaled (along either or both axes), slanted, centered and placed by relative or exact location. Individual letter spacing can be adjusted or the text set as proportional.

DynaCADD does have a very sophisticated font editor (we were only able to see the developmental version for this review), but it requires you to build a font much in the same way as you build a 2-D drawing, a somewhat laborious method but rewarding in the end results. Because DynaCADD fonts are stored as vectors, bit-mapped fonts such as those used in GDOS or output by programs like *Fontz*, are not compatible.

The other file format available, DynaCADD's own, is called DEF. This is a text (ASCII) file that can be read and edited by most word processors.

The file format is described in the manual but best understood by creating and saving a drawing, then examining the DEF file.

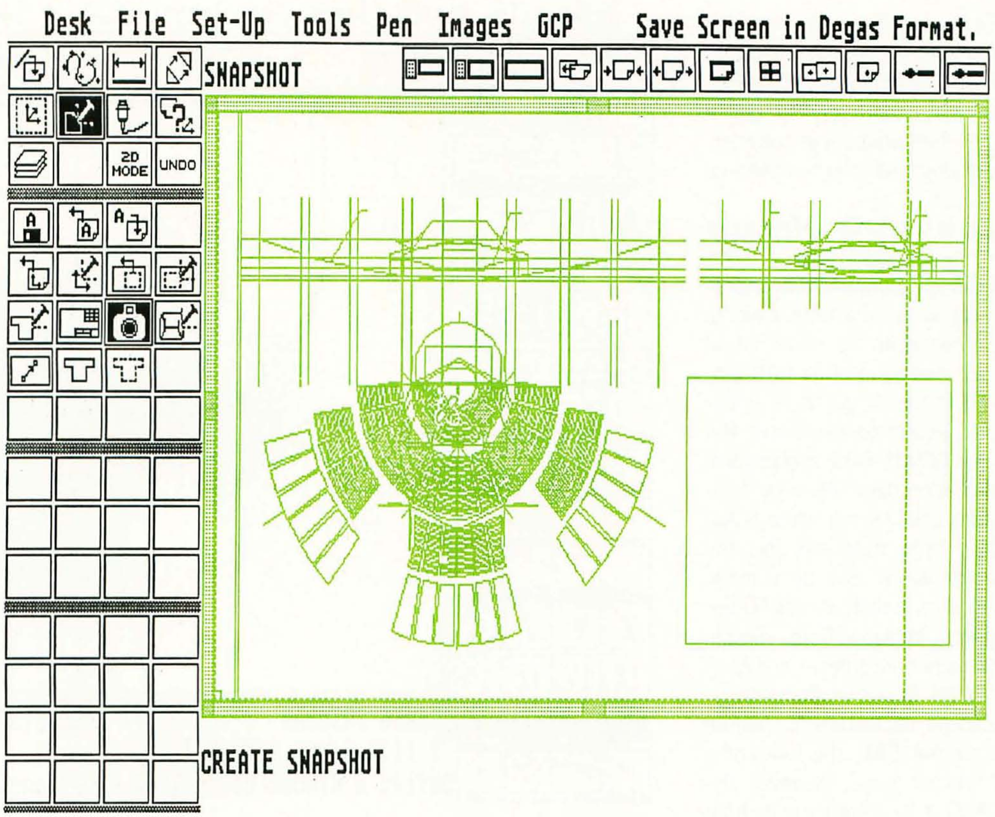
Dimensioning allows numerous formats and parameters for both 2-D and 3-D drawings, including arrowhead types, tolerances, precision and text format dimensions. You can also determine real-world scaling, such as making an inch in your drawing equal to eight feet, for exact measurements and scaling. You can measure distances between points, including the radius and diameter of circles, angles and distances between lines and points.

Transformation covers a wide range of commands to manipulate entities, including delete, divide (break an entity into two or more separate entities), copy, rotate, mirror, modify, move, stretch, scale, project (extrude a 2-D entity along the Z axis), revolve (3-D only), sweep (generate entities along the Z axis), move and copy along an axis, mask, trim an entity, and transform 3-D entities into 2-D. Entities can also be grouped as figures or formats and saved to disk.

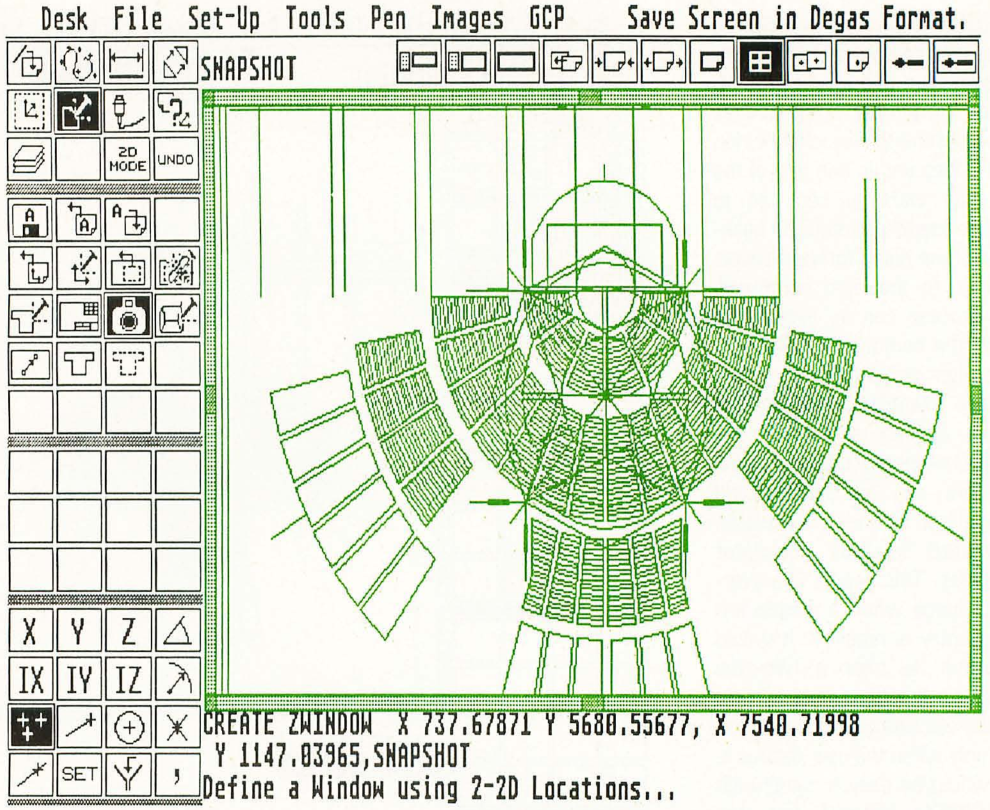
Drawings can be done with a grid (visible or not) and points snapped to it. You cannot, however, add cross-hatching or other GEM fill patterns to a drawing.

Views can be manipulated in a number of ways, changing not only the view location, but the distance from the entities being viewed. You can zoom or rotate a view, scroll it or hide selected entities within it. The dynamic rotation option is particularly easy to use although it unfortunately does not use the cursor (arrow) keys. A view can be stored as an image and recalled later, rather than having to work through various commands to achieve that view.

DynaCADD is copy protected, using a cartridge slot key. This is simple and effective, but limits the program to single-system use. There is no provision currently for site licensing or the purchase of multiple keys.



DynaCADD is one of a rare breed of software. It is designed and implemented uniquely for the Atari ST, for the professional user.



For the small architectural firm or the professional who wants to work at home, DynaCADD provides a superb opportunity to get a master-level program at considerable savings over the competition.

DynaCADD Versus AutoCAD

Although DynaCADD's ancestors are in the mainframe world, it competes in the micro world against dozens of other CAD programs. In the ST galaxy, it is unrivalled as a professional tool. But in the PC/MS-DOS professional marketplace, there are many competitors, chief among which is *AutoCAD*, from AutoDesk Inc, the industry leader and benchmark. Contenders include *VersaCAD Design* from Versacad Corp.—first in features but not sales— and *FastCAD* from Evolution Computing.

Stacked up feature for feature against AutoCAD, the two come out roughly equal. However, DynaCADD is far superior in its handling of 3-D entities, and far faster in redrawing screens.

The biggest difference between the two, vocabulary aside, lies in the way they handle the commands and the processing to attain similar results. This is neither positive or negative: both programs do much the same, but often approach the same concept from different viewpoints. It takes some getting used to the differences and the manual—the weakest element in the package—fails to help the user over them (see below).

With a little perseverance, the professional who understands the basics of CAD can figure out how to achieve with DynaCADD what he or she can do with AutoCAD. If you're not steeped in preconceptions from using AutoCAD, then learning DynaCADD will prove a trifle easier.

DynaCADD also reads AutoCAD's DXF files and can display everything in them except the text and (of course) the color information. We had no difficulty porting DXF files between the ST and PC. However, the loss of text from PC DXF files is disconcerting, especially if important information is contained in the text. It would be nice to offer a utility to translate AutoCAD fonts over to DynaCADD so the text could be imported as well.

AutoCAD supports on-screen color definition. This is important for identifying layers and entities, something DynaCADD lacks, since it works in monochrome only (although you can print in color, if your printer or plotter supports it).

AutoCAD lists for about four times the price of DynaCADD: $2,850 for version 9. If you consider the cost of a high-resolution color graphics system, based on at least an 80286 system, the hardware costs are another $10,000. Given this, DynaCADD compares more than favorably to AutoCAD as far as price for power. If Atari ever manages to make inroads into the business world, DynaCADD might go head-to-head with AutoCAD for market share.

ST Versus the Mac

If the PC/MS-DOS systems are oranges to the ST's apples, then the Macintosh is another system of the same breed. Right now, the best CAD package on the Mac is MGMStation, at $495 similar to AutoCAD version 9, but it only supports 56 layers, compared to DynaCADD's 256. It also doesn't have a macro capability and falls short on the isometric displays.

VersaCAD should have released its Mac version by now, but initial press releases say it only supports 2-D.

Despite the hullabaloo about the Macintosh, nothing currently available for Apple's frontline computer can match DynaCADD for features. ST users can rejoice in their superiority.

Documentation

The 300+ page manual is the weakest part of the DynaCADD package. It was written by the programmer, Dave Fletcher, and edited by the company's president, Nathan Potechin. It's admirable that they tried so hard, but they failed to provide documentation equal to either the program or the price being asked for it.

It has a terribly inadequate index, gives no command examples, has far too few illustrations and the descriptions are obscure, often to the point of opacity, when explaining complex commands. What illustrations are available are not accompanied by the commands used to create the results. Figuring out how to get something equivalent is often very confusing. The order of the chapters is sometimes confusing and not conducive to clarity.

The tutorials (2-D and 3-D) are mistakenly placed at the end of the manual (they should be at the front), but they are meager and sparse. Very little help is given to the user in either of the tutorials.

Many areas of the program are poorly documented or even ignored, especially since the manual was written for an earlier version with fewer features than are currently available. The language is acceptable but generally poor, mostly using the passive voice and a spartan style that fails to explain things properly.

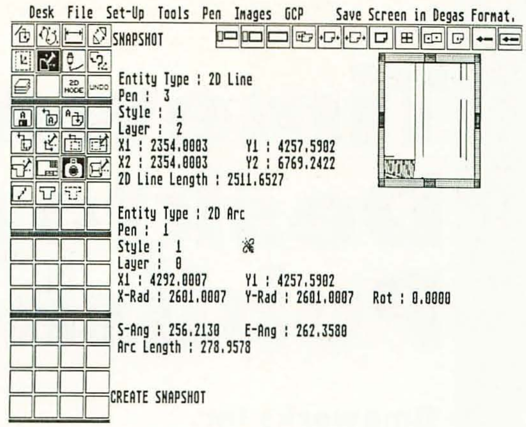
A complete rewrite should be undertaken. At $695, the buyer deserves much better in a manual.

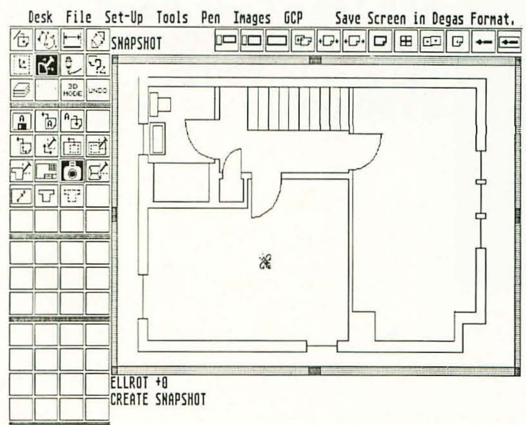
Strengths

Installation is easy and painless.

You can suspend DynaCadd or save it and return to that same drawing later.

There are many information commands which provide considerable information about the system and the drawing.





There are a high number of complex commands, with much flexibility and control over drawings.

There is a large, context-sensitive Help system, although the writing leaves something to be desired. However, the help is comprehensive and is almost a second manual in itself.

Good upgrade policy, on-going program development and ISD appears to be willing to listen to the users and incorporate their requests and suggestions into the program.

All commands are available through the keyboard (but see below about the space bar).

Commands can be concatenated and stored as macros.

The screen can be saved as a DEGAS format picture file, although the menu pad, menu bar, etc., are also included. There is no way to save just the drawing screen.

There are many information commands which provide considerable information about the system and the drawing.

There is considerable support for plotters and laser printers, including the Atari SLM804. However, only Epson 9- and-24 pin or compatible dot-matrix printers are supported. Drawings can be saved to disk as GEM image files.

There is easy switching between viewing windows.

The screen display is crisp, clear and distinct.

Filter (not in AutoCAD) permits simple selection of particular types of entity for manipulation or viewing. For example, you can alter only circles in a drawing.

ISD is very open to suggestion and listens to its users. They will incorporate suggestions and ideas, where possible in upgrades.

Weaknesses

(Note: Neither of us consider any or all of these weaknesses sufficient reason not to buy the program. We believe that, due to the high level of dedication and user-support, ISD will make changes and continue to add features and improve the program to compensate for any perceived flaws.)

When you load the program, you must "activate" a part or drawing first. In order to load a new drawing, you must exit the program to the desktop, load DynaCADD again and activate a new drawing and/or part. You can't close a drawing and start a new one from within the program. This is incredibly annoying and no reason is given for it in the manual.

You activate a command by pressing the space bar, rather than Return. This is hard to remember, since it does not follow.

The program is not entirely bug-free. Transform (offset) locations, for example, seems to be at the computer's arbitrary choice, rather than where the user determines. The program sometimes crashes, although, to be fair, not very often.

The cursor is hard to see and sometimes hard to position. It would be better if a small square surrounded the crosshair or if DynaCADD used a variable cursor to identify function or zoom mode.

DXF files lose their text when imported. However, when DynaCADD reads a DXF file with text, it still takes time to process the text vectors, even though it ignores the text in the drawing later. Dealing with the missing text is slow.

Printing is slow, but by most CAD standards, it's not excessively slow. There should be a printer driver construction utility for non-Epson printers.

Monochrome is the only resolution supported. It is difficult or impossible to visually distinguish layers without color information.

It can't detect intersections between a circle or arc and a straight line (it does easily with straight lines).

AutoCAD allows you to select the intersection, not the lines involved.

It doesn't support the mini/mainframe IGES file format. It should have a file conversion utility, since this format is supported by all major PC/MS-DOS CAD programs.

There is no freehand sketch feature nor are there any fill patterns or cross-hatching.

Undo only goes back one step, rather than backing step by step. It should be able to step backwards.

The manual is mediocre, at best (see above).

Many of the prompts and alerts are cryptic and none are explained in the manual. It's too easy to destroy or delete a drawing because the alerts don't explain the severity of the commands.

There is no "crossing" command (to select any entity the window crosses) as there is in AutoCAD. ■

Ian Chadwick is a freelance technical writer and editor who specializes in desktop publishing and word processing, and is a columnist for ST-Log.

Thom Weeks is an architect for the firm of Zeidler Roberts Partnership, Toronto, Canada. He and his architect wife, Jennifer, both use AutoCAD in the office and DynaCADD at home. They bought their Mega ST specifically for CAD and paint programs.

Our thanks to ISD's Nathan Potechin and David Fletcher for their time and efforts and the hours spent in patient demonstration and answering questions.

Timeworks Desktop Publisher

Timeworks Inc.
444 Lake Cook Road
Deerfield, IL 60015
(312) 948-9202
$130.00

Reviewed by Ian Chadwick

Desktop publishing essentially combines the tasks of production manager, layout and paste-up artists and typesetter. The term describes a category of software dedicated to page layout and design. Before importing text or graphics, the page is laid out to present a comprehensive document structure: page size, margins, columns, headers and footers. Edited text and completed graphics are brought into the DTP program where they are assembled on the page for publication. In the design process, the text is also assigned various attributes such as bold, italics and fonts. Graphics are placed and scaled to suit the layout. Final editing for finished output is completed, then the document is printed. Of course, to be even marginally effective, DTP programs must be fully WYSIWYG.

Timeworks has a reputation for producing well-designed software. Their products fill the needs of the ST market: simplicity of use carefully balanced with a wide range of features and functionality. It's hard to find fault with their approach and, if particular products may be open to specific criticism, overall they do an excellent job of filling the major software niches with solid, bug-free products.

Publisher ST, written by GST, adds considerably to Timeworks' existing line, bringing a desktop publishing program (DTP) to challenge *Publishing Partner*'s hold on that market. Both programs are quite similar in their strengths and weaknesses. Modelled after Xerox's popular program *Ventura Publisher* for the IBM PC market, Publisher is not merely a clone, but provides its own features and has several functions Ventura lacks. However, anyone accustomed to Ventura will find themselves on familiar ground with Publisher. Owners of Timeworks' *Word Writer* will also find that their text converts automatically to Publisher format, including all type attributes and styles.

There are several basic elements in Publisher:

Master page: This is the underlying page layout that will be carried through from page to page. To allow for two-sided printing, right and left master pages alternate margins to accommodate binding or hole punching.

Frame: Everything imported goes into a frame. Each column of text in a multi-column document is a frame that must first be drawn. Frames can be created on the master page(s) to repeat through the entire document, or as needed on each page. Frame outlines can be made visible and given various thicknesses and shades.

Paragraph: A paragraph is defined as text that ends with a carriage return. This can be as large as your entire document, or as small as a word or a character. Each paragraph can be assigned its own style, with unique justification, font, spacing and tabs. Any number of paragraphs can be assigned a single style.

Style sheet: A document consists of the common elements (paragraph styles, headers, footers, master pages, empty frames) and the specific elements (text and graphics assigned to the frames). The common elements can be saved as a style sheet and reused by other documents.

Text: Publisher provides limited text editing capability, so it's best to use a word processor to edit before importing text. However, you can do final editing in Publisher and assign specific words or characters bold, italics, underlining, or select a font different from the rest of the paragraph.

Graphics: In Publisher you can edit an imported graphic on a pixel level, as well as manipulate the size and shape of it. The object-oriented draw package permits line drawing, rectangle, circle, ellipse and freehand drawing. Although not as sophisticated as a CAD program, you can use the built-in graphics for labels, flow charts, arrows and simple but elegant illustrations.

Publisher supports several file types: Word Writer, 1st Word, 1st Word Plus and Word Perfect for text, as well as ASCII. Unfortunately for the thousands among us who use it, ST Writer is not directly supported. For graphics, you can import DEGAS, Neochrome, Easy Draw, and GEM Paint and GEM Draw, neither of which DRI has released for the ST.

The Publisher's screen consists of three sections:

Menu bar: Pull-down menus include file (load, save and print functions), edit (cut, copy and paste text, frames or graphic objects), options (various display options, column guide settings, hide pictures, and more), page (zoom modes, go to and add or delete pages), style (text style, font and point size), text (search and replace, hyphens, kerning, headers and footers), graphics (scale, crop, edit, grid size and snap) and help (online help features).

Tools: The left-hand side of the screen displays the tool access: icons for frame, paragraph, text and graphics allow you to switch to their unique functions, also indicated by icons. A selection in the Options menu allows you to hide the tool area and expand the workspace.

Workspace: This is the display of the page at the selected zoom (magnification) level.

It uses GEM scroll bars for positioning and can show rulers (in imperial, metric or pica/point units) along the top and left borders. All editing is done in the workspace.

All Publisher commands and features are activated by the mouse through either the menu or the tool icons, but most can also be called through keyboard equivalents.

Ventura users may be surprised to find that Publisher supports the Epson LQ series as well as several other popular printers, which Xerox fails to do, limiting its dot-matrix support to the MX/FX series. Also, the graphics editor in Publisher has more flexibility and

Timeworks has a reputation for producing well-designed software. Their products fill the needs of the ST market: simplicity of use carefully balanced with a wide range of features and functionality.

Technical Requirements:

- Atari 520; 1040; MEGA ST computers with one or more single- or double-sided disk drives. (Hard drives are also supported.)
- Supports color or monochrome monitors.
- Printer Requirements:
  - Most popular dot matrix pritners, including: 9-pin and 24-pin Epson, Epson MX/FX/RX/LX, and all Epson-compatible printers; and NEC P6/P7; Atari SMM804.*
  - Atari SLM 804 Laser Printer; all Postscript-compatible printers, and Okidata Laserline 6 (512K).
  - All printers compatible with those listed above.

you can edit imported graphics, something Ventura doesn't offer. And while Ventura's built-in fonts are, except for Postscript printers, very limited, the Publisher comes with seven fonts, which can, depending on your printer, be displayed in sizes between six and 72 points. Finally, Publisher comes with 23 files of 200 ready-to-use graphic images.

On the minus side, the Publisher has no means to create text columns *within* frames. Columnar text must be manually designed by creating a separate frame for each column, then assigning text to flow from the first into the next frame. This can lead to a lot of agonizing tinkering to get precise columns and locations. Ventura's method of selecting the number of columns, then having the program figure the default widths (which can be manually adjusted), is far easier and sophisticated.

Here's a typical example, from the type of document I work on every week: on a two-column page, a three-column frame must be created to hold separate tabular material. In Ventura, this is an easy task, requiring only a few moments of effort. In Publisher, it is a demanding exercise in which excruciating care must be taken to insure that the frames are precisely placed and are the proper size.

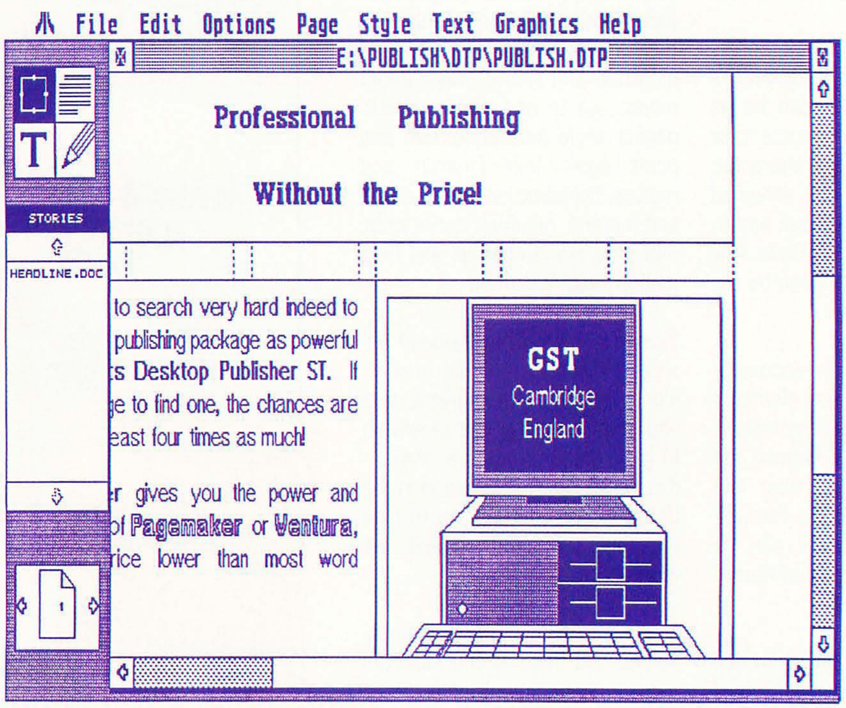
There are other features available in Ventura and not in Publisher which offset the seeming advantages of Publisher's extras. For example, although you can start a paragraph with a bullet, you cannot start with a drop cap (a large initial character), nor can you specify automatic page breaks within the text. There is no means to do automatic indexing or table of contents, nor can documents be chained for any purpose. Frames cannot be anchored to specific text points nor can they be given captions or automatically incrementing figure numbers. Page numbering is in Arabic only, with no provision for Roman numerals. If the comparison between the two appears unfavorable, remember that Ventura costs roughly five times what Publisher costs!

The biggest factor that makes the Publisher fall below the professional use category is the manner in which it handles text flow between frames. Since each frame is independent, you can assign any text or graphic file to it. However, if the text file is too large to fit into the frame, it does not automatically flow to the next. There is no simple means to link frames and tell the Publisher to fill contiguous frames with a file. You must click on each frame, then on the file, then on the next frame into which the text will flow, then on the file, and so on until the text is exhausted. Although this gives you moderate flexibility in determining where text will go, it is a tedious process with a large file. In my work, I often deal with 200 or more page documents, and I can't spend my time pointing and clicking over several hundred pages.

You also can't create a frame to handle text out of order. That is, if an article doesn't fit the frame on page seven, I can't create a new frame on page six to handle the overflow. What will happen is that the text will start flowing from page six, although it was initially assigned to page seven. Frames have a strict front to back, left to right sequence.

Also, because of their independence, once you establish a frame on a page, you cannot change it for the whole document if anything has been added to it, even if you alter the master page frame itself. This means that, if you have just finished designing and editing your 12-page newsletter and want to increase the sizes by 1/4 inch, you have to start all over again, because the frame sizes are fixed for good.

Going to a new page requires confirmation that you indeed want to create a new page. In a large publication, this simply adds unproductive time to the process without returning anything in equal value.



"The biggest factor that makes the Publisher fall below the professional use category is the manner in which it handles text flow between frames."

Kerning—minute adjustments between letters—is another issue where the Publisher is inferior to Ventura. You can kern individual letters but there is no provision for automatic document or paragraph kerning. Even had they simply provided automatic kerning for the 20 most commonly kerned pairs, it would have vastly improved the program. Lack of kerning is most noticeable in Postscript and typesetter output, where fonts carry correct kerning information in their data file. A quick glance through the Publisher's manual shows that they did not use their own program to create it, because the manual is properly kerned throughout.

This, unfortunately, also drags the Publisher's output down below the requirements of professional publishing, since no one is likely to engage in the Herculean task of manually kerning a 300-page document, much less a single chapter. And, simply put, professional documents demand kerning.

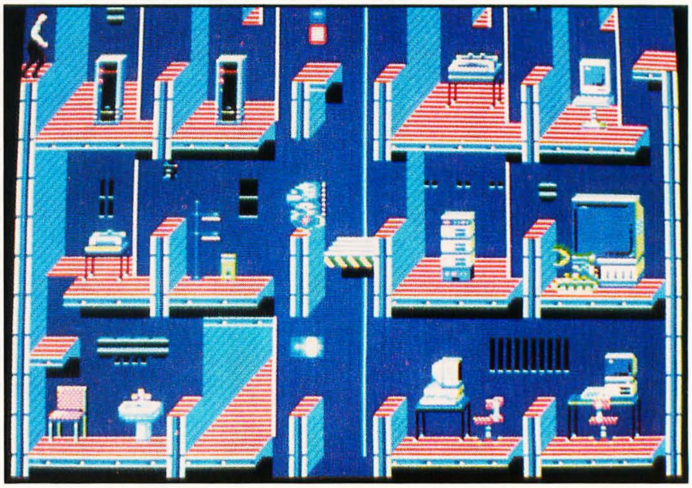
Despite these drawbacks, Publisher is *not* at all a bad program. It is simply better suited for noncommercial or non-professional efforts, such as user-group newsletters, small in-house publications or hobbyist publications. I wouldn't suggest you use it for magazines, books or anything larger than about a dozen pages. But for less ambitious efforts, it's more than suitable.

I've tried my best to confuse Publisher and make it crash, without success. It seems pretty ruggedly built and able to handle my erroneous input, wrong file types and overloaded memory. The printed output is clean and the dot-matrix printout is better than Ventura's modest effort. All in all, it does a job that, if not particularly suited for printing your next novel, is certainly better than anything you can do with a word processor! ■

Impossible Mission II

by Novotrade Studios
Epyx
600 Galveston Drive
Redwood City, CA 94063
(415) 366-0606
Low Resolution
$39.95

Reviewed by Andy Eddy



Another nice feature they provide is the ability to run Impossible Mission II off a hard drive.

Elvin is at it again. The evil Professor Elvin Atombender, the genius with a long streak of mental instability, is plotting to destroy the planet. Ensconced in his heavily guarded tower, protected by robot killers, he appears to be too isolated to stop. This is where you come in.

The name of the contest is *Impossible Mission II*, Epyx's sequel to the Impossible Mission game that came out a while ago. You are a government field agent who must set out to break Elvin's defenses, get to the pieces of music strewn throughout, then enter the control room to stop the launch of the missiles that will end the world. One thing to remember is that the game is very aptly named: It may indeed be a task impossible.

Skyscraper Caper

Elvin has filled his fortress with many obstacles and implements of destruction; his best shot to keep you away from his control room. A slew of robot warriors pace back and forth through each room, protecting all the pieces of the puzzle you have to gather. You'll have to search behind each chair, picture frame and, most of all, locked safes (where the music is stored), to keep your quest alive. Hidden by these items, you'll find such necessities as mines (to take out the robots and blast holes in the floor), light bulbs (to illuminate darkened rooms) and bombs (for blasting open a safe).

By far the most important gathering you make is colored numbers. The correct set of three numbers must be completed before you can exit the tower you are in; each of the eight towers must be finished in the same manner. Lastly, you must assemble the six different snippets of music (there are two duplicates to throw you off even more), which will allow you to pass through to Elvin's computer room. Once there, you have to pick one of the three terminals and disarm the rockets; only one will do the job, and the other two are electrified in an attempt to further hinder you.

Is That All?

I guess you can tell that you have a lot ahead of you before you finish. Oh, did I tell you that you are also under the constraint of a time limit? Yes, eight hours are all the time you have to do your dirty work. And one thing the documentation doesn't tell you is that each time your character dies—either by being fried by a robot's laser or falling to his demise—a few minutes are extracted from the time limit.

All of the impediments in your path would seem to be enough to keep you from getting anywhere near the end of your mission. No doubt, if you could make a careful map of each room, plotting out a strategy and path to get around each robot, and where each important item is located, much more could be completed. Unfortunately, they even scramble object locations, so you're out of luck. The key is not to stay in one place too long, wasting ticks of the clock. There is a game-save feature, so you can creatively help yourself out anyway.

Epyx has, through the programming talents of Hungary's Novotrade Studios, put together a challenging test of a player's ability.

Thank the Interior Decorator

Just the same, as discouraging as the constant downfalls can be, it is a ton of fun to tromp around the towers, leaping and running all over the place. Each room offers a separate challenge, all the while accompanied by lushly designed graphics. Little details will leap out at you, such as when your character strolls in front of a mirror, and you see an accurately depicted reflection in it. The animation of your agent flipping over things, sliding up and down or back and forth on moving platforms and falling through holes in the floor—the latter accompanied by a frightening, digitized bellow—are well done.

Another nice feature they provide is the ability to run Impossible Mission II off a hard drive, though you must use the original floppy as a key when loading it in that way. Software companies should take this path in bringing their games to market, what with more computer users going the hard-drive route, particularly on the basis that many of the new generation contests require a large chunk of time to load.

As a final note, the documentation is simple but gives you enough background and detail to make the game play easy to understand. Though you can't simply boot-up the game and play—there are certain keys that have an integral purpose, and knowing how to search for objects is not easily discovered without docs—there isn't much to keep track of.

Epyx has, through the programming talents of Hungary's Novotrade Studios, put together a challenging test of a player's ability and will. Though most gamers will be frustrated and often discouraged by the program's difficulty, it will stretch your talents in addition to being sharp to look at. ■

BB/ST

by Steve Grimm
Quantum Microsystems Incorporated
P.O. Box 179
Liverpool, NY 13088
(315) 451-7747
$49.95 Color or Monochrome

Reviewed by Blake Arnold

***BB/ST* is a relatively new entry into the Atari** ST bulletin-board market, and probably one of the most unique bulletin-board programs available. It allows much more customization than most BBS programs, and its features offer a radical departure—features such as its "tree" type message base and its mini-language.

The hardware requirements for BB/ST are typical of most of the Atari ST's bulletin-board system programs: a modem (I recommend a Hayes compatible), at least a 520ST with a single-sided disk drive and monochrome or color monitor. Although BB/ST will run from a single-side floppy-disk system, you'll need two double-side drives or a hard drive, to maximize use of the program. BB/ST itself doesn't use much disk space, but the user log, message bases and file areas can grow large quickly. The BB/ST program itself arrives on a single-side 3.5-inch diskette that contains all the necessary system files (and a few online games) for the program. BB/ST is not copy protected, so it can be run from a hard drive if your setup includes one.

The documentation is very complete, even going so far as to explain the correct use of file pathnames on the ST. As with any program, it's wise to read through the manual a couple of times before you use the program. BB/ST's instruction manual is relatively straightforward and written in a way that is easy to understand. The program itself presents no real difficulties, but if you have a question, a quick flip through the manual should clear it up.

One of BB/ST's unique features is its message system. Instead of using a linear message system like most other bulletin-board programs, BB/ST uses a "message tree" (actually it resembles an upside-down tree). In a linear message base, messages string along one after another without much consideration for their subjects, but in a tree-based system messages fall under a certain "branch" of the tree which contains related subjects. There is one message at the top of the tree; Under this message one might enter messages that would correspond to normal message bases on other BBS programs (such as "General Topics," "Computers," "For Sale," etc.). Under these branches the users may enter their own messages, creating even more branches and sub-topics. In this way a "Computers" message branch could be created, with a "Hot Computer News" branch attached to it, and a "Atari introduces transputer" branch attached to that. In that situation the message tree would look like this:

```
1. Top of Tree
  2. General Topics
  3. Computers
    5. Hot Computer News
      6. Atari introduces transputer
  4. For Sale
```

Each message may have up to 128 "children" (replies) attached to it, and it is possible to keep users from entering messages directly under the top of the message tree. The message tree is a powerful system, but in this case power breeds a certain amount of complexity. The message tree does take a little while to get used to, especially its advanced commands. It contains commands that are radically different from a linear message base's, but after posting a few messages and "climbing" around the tree a little bit, it isn't that difficult.

The tree structure makes it easy to read a message and follow all of its replies, then read another message and follow all of its replies, etc. Messages read in this way follow a logical order instead of being jumbled together as in a normal, linear message base. The tree does have a disadvantage though. In a linear message system users that don't call often can simply go into a message base and post a message; but in the tree system it is proper to look for a related message to post a new message under. In other words, the tree can be more work and more confusing to the infrequent caller. As was stated earlier, the tree is powerful and flexible, but it is also much more complex than a linear message base.

There is a rather large flaw in the way the message file is handled by the bulletin-board program: There is no way to set a limit for the message file's size. The same is true with the user-log files (the files that contain the callers' names and passwords) and a few other files. It wouldn't be too hard for a mischievous user using an advanced terminal program to write a macro to apply for passwords endlessly, which could theoretically result in a disk filled with a large user-log file. The same would hold true for the message base files. Even if no one ever tried to damage the system, it would be nice to have an option to set a maximum file size for those files that currently have no upper limit.

BB/ST's system editors are easy to use, almost to the point of not even requiring the manual. The editors are built-in to the BBS itself, which means you don't have to exit the program to access editing functions. The main SYSOP (System Operator) menu contains commands to access most of the other editor menus, and also to perform disk operations; the other editors are a little more specialized.

The SYSOP message menu contains commands that are used for maintenance of the message tree. It allows the SYSOP to save entire message branches to disk to be reloaded later, change the privilege flags of branches, delete unused branches (by days old or straight deletion), and to move branches to different parts of the message tree. These commands are powerful and convenient, but BB/ST's message tree lacks something that most other BBS programs provide: automation. Although the tree may be automatically compacted (deleted messages are erased and file area is recovered for new messages during compaction), it is still up to the SYSOP to perform the dead branch deletion functions.

The user-log editor is relatively straightforward, and contains options to set privileges such as call time per day, message base access, commands for terminal setup and other functions that affect what areas of the BBS each user has access to. One thing missing from the user's information displays is a list of passwords. This is a feature that should be included in any bulletin-board system program, and this is the first program that I have seen that does not allow a SYSOP to view the user's passwords. Of course if you have a sector-editor program such as *Disk Doctor*, you can view the passwords, but this should not be necessary.

The bulletin-board system's configuration editor contains information such as pathnames to files (message tree file, download libraries, text files, etc.), security levels to the message tree and file-transfer areas, modem setup commands and other system-specific configuration items. It is possible to change aspects of the configuration file while the BBS is online, which can be handy for remote SYSOPs.

The menu editor is used to change traits of the bulletin board's menus. There are commands to do such things as send E-mail and display text files, along with commands to move around the message tree and commands to download/upload files. In short, there are commands for all of the functions of the BBS, and they can all be edited. With the menu editor it is possible to add items to the menus, create new menus and to chain almost any non-GEM program into the BBS through use of the program's mini-language.

BB/ST's mini-language can be used to write small programs for the BBS, and the manual contains a few examples of mini-language programs such as a lottery-number generator program and a database program that allows users to display certain selected text files to the screen (useful for bulletins). The mini-language is similar to a tiny implementation of BASIC mixed with BBS-specific commands and contains statements such as IF and GOTO. Intermediate level BASIC programmers should have no problems with the mini-language, but beginners may want to enlist the help of a more experienced programmer.

Through the mini-language it is also possible to pass parameters to a TTP (TOS Takes Parameters) program. It would also be possible to write a utility for users to view the files in an ARChived (ARC) file, extract the files they wanted and then download those files. The mini-language combined with the ability to pass parameters and execute TOS, TTP and other non-GEM programs is definitely one of BB/ST's strong points. With this ability it is possible to customize the system to do exactly what you want.

You don't have to be a great programmer either; compiled GFA BASIC programs work very well, and I don't think you'll find an easier language to work with. Another nice feature of BB/ST's ability to execute external programs is that it continually monitors the modem's carrier detect line; should the modem lose the connection the BBS will simply reset itself for the next caller (it also automatically redirects I/O to the modem).

For those who would like to write more in-depth programs for BB/ST, there is a "Programmer's Guide to BB/ST" available from QMI that has details on how to communicate with the BB/ST program from another (external) program. With the mini-language and external program ability, it is very easy to implement suggestions from users, which is an important feature when considering any BBS program.

File transfers are also an important part of any bulletin-board system program, and BB/ST has an easy-to-use, yet powerful file-transfer menu. There may be a maximum of 16 file areas, which should be more than enough for most setups. Included in BB/ST's file-transfer menu are options to view files in an ARChive, extract files from an archive into a temporary archive that may then be downloaded, and upload and download files via Xmodem, Xmodem CRC, Ymodem, Ymodem Batch, Fmodem and ASCII. Also in the file-transfer menu are options to view new files, and to change the current file area the user is in, along with a few other useful commands.

BB/ST has a terminal mode that the SYSOP may use to call other bulletin boards or telecommunication services without having to exit the BBS program.

It is strictly a no-frills terminal, but does have options to change duplex, baud rate and to transfer files via any of BB/ST's implemented protocols.

The program also allows the use of voting polls for users, and even has an option to allow users to create polls or add selections to existing polls. When a poll is viewed, BB/ST will display the total number of votes and the percentage of the total for each selection. If certain users create problems with the polls (such as adding unnecessary voting selections), it is easy to remove their access to the poll section. The same holds true with all of BB/ST's areas (message bases, etc.).

Another of BB/ST's features is the ability to display Vidtex graphics in almost any text file. Vidtex graphics may be used in menus, messages and just about anywhere else the program displays text. Other BBS programs may allow Vidtex, but most of them require a separate set of Vidtex files to do it; BB/ST needs only one set of files for Vidtex and ASCII modes. BB/ST will simply skip over any Vidtex portions of the file if the user is not in Vidtex mode.

Obviously there isn't room here to go into a detailed description of all of BB/ST's features, but I've tried to reveal most of its high and low points. BB/ST is a unique program with several unique features, but it also contains a few oversights such as the user log and message files not having an upper size limit and the inability of the SYSOP to easily view the password file. On the other hand, features such as its mini-language and configurable menus are powerful and allow the SYSOP to create a nicely customized BBS. If you'd like to see a BB/ST system online, call the QMI Customer Service BBS at (315) 457-7216. ■

JANUARY 1989 DISK LISTING

The ST-Log #27 diskette contains 16 magazine files. They are listed below.

```
FILENAME.EXT    FILE TYPE     COMMENTS
------------    ---------     --------

\ASMLINE\
  DUMP    .PRG  RUN FILE      ASSEMBLY LINE
  DUMP    .S    ASSEMBLY      SOURCE CODE

\DETAB\
  DETAB   .TTP  RUN FILE      DETAB UTILITY
  DETAB   .C    C             SOURCE CODE

\DRAMACDE\
  DRAMA   .BAS  ST BASIC      DRAMA-CIDE

\INTERFCE\
  CARTPRT1.BAS  ST BASIC      INTERFACE TEST
  CARTPRT2.S    ASSEMBLY      SOURCE CODE

\OMNILIFE\
  OMNILIFE.PRG  RUN FILE      OMNILIFE
  OMNILIFE.RSC  RESOURCE      OMNILIFE RSC
  OMNILIFE.DFN  DATA          RCS FILE
  OMNILIFE.C    C             SOURCE CODE
  OMNILIFE.H    C             SOURCE CODE
  LIFEASM .S    ASSEMBLY      SOURCE CODE
  STEREO  .O    OBJECT        STEREO MODULE

\TRANSWRP\
  TWARP   .PRG  RUN FILE      TRANS-WARP
  TWARP   .C    C             SOURCE CODE
  README  .DOC  TEXT          DISK INSTRUCTIONS

Disk instructions:
   Only those files with PRG, TOS, or TTP
   extensions may be run from the GEM desktop.
   Other programs may require additional software
   as shown below.

WARNING: Be sure to read the appropriate magazine
         article before attempting to run
         programs.  Failure to do so may yield
         confusing results.

                 .EXT  DESCRIPTION
                 ----  -----------
                 .BAS  Requires ST BASIC
                 .GFA  Requires GFA BASIC
                 .C    Requires C compiler
                 .S    Requires 68000 assembler
                 .PAS  Requires Pascal compiler
```

INDEX TO ADVERTISERS

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher connot be responsible for inadvertent errors.

How to Buy a Computer

by Kevin L. Pehr

There are many possible reasons for buying a personal computer: You run a small business and all your competitors have one; you are a yuppie and all your friends have one; or you just think they're real neat.

Whatever the reason, be ready for some unpleasant times ahead. Choosing a personal computer is far more complicated and nerve-racking than choosing a house or a car or a spouse. Be prepared for condescending computer store clerks babbling incomprehensible jargon. Be prepared to spend weeks snapping and snarling at your friends and family as you agonize over the options. In short, be prepared to enter the world of high tech.

Which one to buy? The easiest method of deciding is to ask friends who already own computers which type they think is best. The answer is unequivocally an AtariAppleIBMcompatibleCommodoreMacintoshgenuineIBM.

Product loyalty is fierce. People who simply smile and look tolerant if you insult their ethnic group, religion or football team will fly into a homicidal rage at the suggestion that their computer can't lick any other one on the block with one disk drive tied behind its back.

The crucial difference is the operating system, which is sort of the language the computer speaks. Arguing the relative merits of MS-DOS versus TOS as an operating system is like arguing French versus Swedish: It's all a matter of cultural preference.

Finally, some well-informed friend lets you in on a secret. (You know the friend I mean? "Four Eyes," the scrawny kid everybody made fun of in your junior high school class? The one who started his own computer company at age 19 and is now a multimillionaire?) He'll tell you that the most important thing is to get a machine that will run your software.

Software is to a computer what clothing is to a Barbie doll: It's not the initial investment that gets you, it's the hundreds of accessories. The problem with your friend's theory is that you probably don't have any software unless you already own a computer. After all, how many people have a complete prom gown plus accessories for Barbie before they buy the doll?

So there you are again, on your own, trying to decide on the ideal system for you. The best way really is the old "making a chart of all the pros and cons." You will need to do a little research for this.

First, go to your public library and read the last 12 issues of all the computer magazines. (This should take about a week, assuming you don't have to waste time working for a living.)

Then, down one side of a very large piece of paper list all the existing machines. Subdivide these by the different models available for each one such as Colonial, French Provincial and Scandinavian Modern. Subdivide further by number of "K" memory, configuration of disk drives and type of monitor. Across the top, list options on operating systems, number of expansion slots and number and type of interface ports.

Next, call every computer dealer within driving distance of your house (a WATS line can lessen the shock to your phone bill). This will enable you to fill in the boxes on your chart with notes about price, availability and whether it matches your living-room decor.

When you are through, you will have all the necessary information on a chart roughly 2,014 lines long by 185 columns wide from which you can easily select your best choice. As this is a little large to handle in your head, buy some database management software and plug your information into that. Any of the popular ones will do; just make sure it is one that will run on your own personal computer.

ST-Log *invites all authors to submit essays for possible use in the* Footnotes *column. Submissions should be no longer than 1,500 words and may be on any aspect of Atari computing. Any style or type of essay is acceptable—opinion, humor, personal experience—but creativity is a plus. Submissions should be sent to: Footnotes, C/O* ST-Log, *P.O. Box 1413-M.O., Manchester, CT 06040-1413.*

HOWARD MEDICAL COMPUTERS

1690 N. Elston • Chicago, IL 60622 • ORDERS (800) 443-1444 • INQUIRIES AND ORDER STATUS (312) 278-1440

★ 5 STAR FINAL NOV'88 CLEAR

HMC SAVES BUDGET

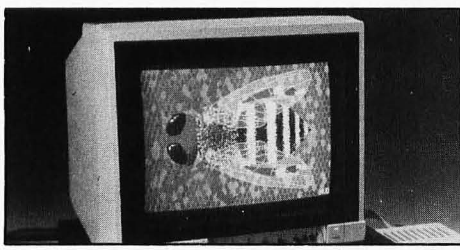
MONITORS

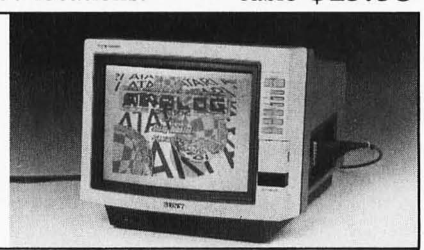
MAGNAVOX color **8CM515** (14 ship) **$269** 3 inputs offer maximum versatility for this 13" color monitor; RGB Analog for the ST; composite for the 800/XL/XE or VCR and RGB TTL for IBM and Clones. The 640 × 240 resolution with .42 MM dot gives vivid colors and a green screen override for 80 column text. ST RGB cable $19.95 with monitor or $29.50 separate.

MAGNAVOX 8CM 505 $220 13" Color Monitor 390 × 240 resolution

SONY color **KV1311CR** (15 ship) **$499** The Ultimate 13" color monitor includes a trinitron TV with remote control; 640 × 240 resolution with a .37 MM dot and a Micro Black screen that is vertically flat for distortion free viewing. Four inputs includes all of those from the Magnavox 515 plus a digital RGB. The colors are incredibly brilliant.

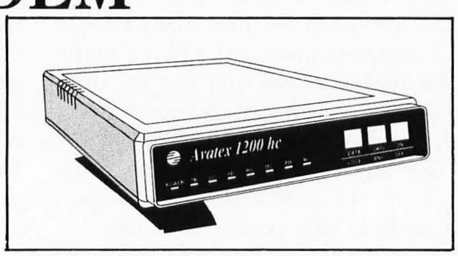
ZENITH green **123A** (7 ship) **$67.50** This close out from Zenith features a 12" non-glare composite Amber screen with 640 × 240 resolution. 90-day warranty valid at our 1200 locations. cable **$19.95**





MODEM

AVATEX 1200 HC (2 ship) **$119.95** This Avatex modem is fully Hayes compatible and operates at 300 or 1200 baud. We include express 3.0, a public domain communications program free with each modem.



DRIVE A+ ($178.45)
Howards Drive A+ features our DB-1 drive box and our DD-3 MPI double sided double density, 40 track full height drive with case and power supply for full 360K storage.

DB-1 DRIVE BOX ($85.00)
Howards drive box hooks up to a ST 1040 to give 2 external drivers. A 3 pole switch will configure an 80 track DSDD (double sided double density) drive or a 40 track DSDD drive or the new 3½ drive. Use with PC Ditto for IBM software compatability.

SIO ($64.45)
IDC's PR. Connector gives 2 RS 232 ports for a modem or serial printer and 1 parallel port for a parallel printer like EPSON or STAR. Ask for our special price on cables.

SPARTA DOS COMBO ($50.00)
This combo gives you IDC's SPARTA DOS OPERATING SYSTEM and the 2 chips ROM set that allows double sided access for the 1050's.

HMC's Guarantee– A Promise you can take to the Bank.

Howard Medical's 30-day guarantee is meant to eliminate the uncertainty of dealing with a company through the mail. Once you receive our hardware, try it out; test it for compatibility. If you're not happy with it for any reason, return it in 30 days and we'll give you your money back (less shipping.) Shipping charges are for 48 states. APO, Canada and Puerto Rico orders are higher.

CIRCLE #121 ON READER SERVICE CARD.

hotline number

DON'T MISS OUT, ORDER TODAY!
800 / 443-1444
WE ACCEPT VISA
• MASTERCARD • AMERICAN EXPRESS • C.O.D. OR CHECKS
DISCOVER • SCHOOL P.O.

SWITCH BACK

REVOLUTIONARY NEW PRODUCT

REQUIRES at least 1 meg. of RAM
(or a Megadisk 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

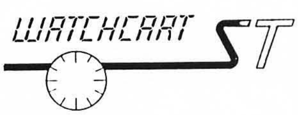
Finally ST Copy protection techniques are revealed. This complete book and disk package details the state of the art in ST Protection methods and much, much more.

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**

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***

Price Subject to change

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

CIRCLE #122 ON READER SERVICE CARD.

COMING
NEXT·MONTH

FEATURES:

- FLAG TRIVIA
- VISUAL INTERFACE GUIDELINES
- DESK SWITCH

REVIEWS:

- GFA BASIC 3.0
- MASTER CAD
- WIZBALL

ST LOG
THE ATARI ST MONTHLY MAGAZINE