ST LOG

THE ATARI ST MONTHLY MAGAZI

FDC 64736

August 1988 ISSUE 22

U.S.A. $3.50
CANADA $4.75

MIDIMON
A complete MIDI analyzer

Microcompositions
Teach your ST to sing

Software Engineering

REVIEWS:
Softsynth
OCP Art Studio
Dungeon Master

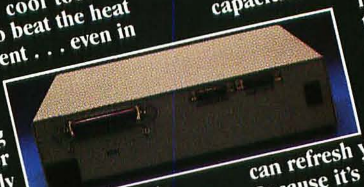
0 73361 64736 8 08

Refresh Your Memory

And Keep Your Cool.

Introducing the ST Hard Drive System from ICD that refreshes your memory better than any other ST hard drive around. *No problem.*

It's the drive that not only looks cool, but stays cool too. All because of a built-in fan that knows exactly how to beat the heat and maintain a calm, cool and collected environment . . . even in *your* most heated situations. *No sweat.*



And, it's the hard drive that takes a refreshing approach to aesthetic case design as well. See for yourself. It's easy on the space, fitting perfectly under the monitor. And it's easy on the eyes, tailored to look great in the company of your Atari ST. With adjustable front legs, your monitor gets the lift it needs for comfortable viewing. *No strain.*

Despite a sleek and compact exterior, the ICD ST Hard Drive System is packed full of overwhelming enhancements. Like an internal clock that tags each file with up-to-the-minute time and date information. Not to mention expansion capabilities that welcome the connection of up to six SCSI devices and daisy-chaining Atari's DMA Bus (ACSI). It's available in more memory capacities than you can imagine. With storage ranging from 20 megabyte systems up to 280 megabytes. And, there's dual drives too, that double your protection and double your confidence. *No stress.*

So, the next time you think about a hard drive for your Atari ST, think about the countless ways we can refresh your memory. It's the only drive worth remembering. Because it's from ICD. *No wonder.*

For further product information, please call or write for our catalog today.

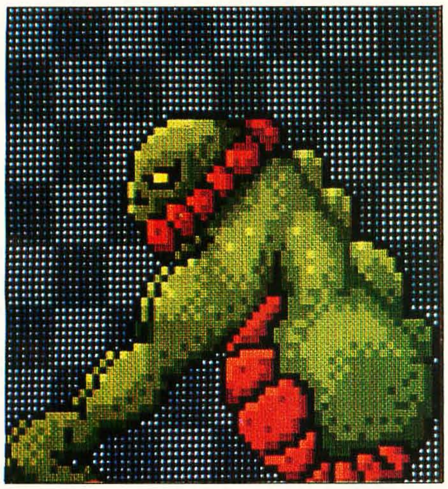
ICD

1220 Rock Street
Rockford, IL 61101-1437
(815)968-2228
MODEM: (815)968-2229
FAX: (815)968-6888

Atari ST is a trademark of Atari Corporation.

CIRCLE #101 ON READER SERVICE CARD.

ON THE COVER: Andrew Todd and Bill Paxton, a/k/a Martini Ranch, whose debut album was produced in its entirety using a MIDI. Photo by Chris Cuffaro/Computer Graphics by Digital Art.


ON PAGE . . . 82

ON PAGE . . . 90

ST-LOG (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1988 L.F.P., Inc. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. All rights reserved on entire contents; nothing may be reproduced in whole or in part without written permission of the publisher. 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.

SUBSCRIBE TO ANALOG

SAVE $14 OFF THE COVER PRICE

1 YEAR FOR ONLY $28—SAVE $14
1 YEAR *WITH DISK* ONLY $105

☐ 1 YEAR @ $28—SAVE $14!
FOREIGN—ADD $7 PER YEAR MCHWY
☐ 1 YEAR WITH DISK @ $105
FOREIGN—ADD $15 PER YEAR DCHWY
MONEY BACK—On all unused portions of subscriptions if not satisfied.

Your first issue will arrive in 6 to 8 weeks **WATCH FOR IT!**

☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MC #__________

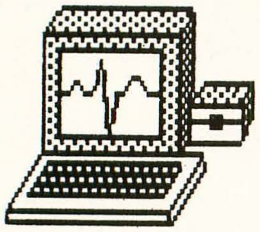
EXPIRATION DATE SIGNATURE

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16927, N. Hollywood, CA 91615. OFFER EXPIRES OCTOBER 26,1988



An incredible simulation

Cardiac Arrest!

with binder and manual, $69.
See discounted package price.

Cardiac Arrest! is a unique product. In this mathematically-based simulator, you interpret the history, on-screen EKG, lab data, and vital signs, then give treatment orders in plain English. While many computer users enjoy Cardiac Arrest! as a challenging medical adventure game, it's a sophisticated product used world-wide for ACLS (Advanced Cardiac Life Support) education. **IBM, Apple II+/c/e, Atari ST, Atari XL/E.**

Antic: "impressive and amazingly complete"
ST World: "both highly educational and fun to play"

We support our products. Updates will be available to users for $6 each when ACLS recommendations change. Our software is NOT copy-protected.

Cardiac Arrest!	$69
ACLS Protocols	$29
EKG Teaching	$29
CardioQuiz	$19
Blood Gases	$24
QuizPlus	$29
Demo	$7

Ask about the four-disk ACLS Package (includes Cardiac Arrest!) for $109. Order direct!

Mad Scientist Software
2063 N. 820 W., Pleasant Grove, UT 84062
Visa/MC orders call 801-785-3028

CIRCLE #102 ON READER SERVICE CARD.

ProCopy ST BACKUP UTILITY

You can't backup your software because copy protection locks you out. **ProCopy** is the key!

- Protects against the accidental loss of expensive software
- Works with all Atari STs
- Copies both S/S & D/S disks
- Use with 1 or 2 disk drives
- Not copy protected
- **FREE** shipping and handling
- **TOLL-FREE** telephone number
- Updates available to registered owners
- Orders shipped same day
- Disk analyze function included

Dealer Inquiries Welcome

VISA MasterCard and C.O.D. orders

Call (800) 843-1223

$34.95

Send check for $34.95 (overseas add $2.00 for air mail) to:

PROCO PRODUCTS

P.O. BOX 665, CHEPACHET, RHODE ISLAND 02814 USA
(401) 568-8459

Available in Europe

THE MOVING FINGER CO.
Building 2
Shamrock Quay
Southampton, SO1-1QL
England
Tel. 0703-229041

TECH-SOFT
COMPUTER WHOLESALE
324 Stirling Highway
Claremont
Western Australia 6010
Tel. 09-385-1885

CIRCLE #103 ON READER SERVICE CARD.

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 16928, N. HOLLYWOOD, CA 91615

Name ______

Street Address ______

City ______ State ______ Zip ______

ATTACH LABEL HERE

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

ST-Log Staff

Publisher: Lee H. Pappas. **Executive Editor:** Clayton Walnum. **Art Director:** Andy Dean. **Managing Editor:** Dean Brierly. **East Coast Editor:** Arthur Leyenberger. **Midwest Editor:** Matthew J.W. Ratcliff. **West Coast Editor:** Charles F. Johnson. **Contributing Editors:** Ian Chadwick, Frank Cohen, Andy Eddy, Maurice Molyneaux, Steve Panak, Douglas Weir. **Copy Chief:** Katrina Veit. **Copy Editors:** Sara Bellum, Anne Denbok, Pat Romero. **Typographers:** David Buchanan, Klarissa Curtis, Judy Villanueva. **Contributors:** Gordon Billingsley, David Duberman, Eugene R. Gobby, Michael Gogins, TG, Larry Herzberg, Brian James Parry, Kelly Schreiner, Karl Wiegers. **Production Director:** Donna Hahner. **Production Assistant:** Steve Hopkins. **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, 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.
San Francisco: (415) 864-3252. Chicago: (312) 445-2489.
Denver: (303) 595-4331.
6855 Santa Monica Blvd., Suite 200, Los Angeles, CA 90038.
New York: (212) 724-7767.

Address all advertising materials to: Paula Thornton — Advertising Production, **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 to 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 (617) 797-4436.

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

EDITORIAL

by Clayton Walnum

It seems that Atari Corp. is going through a major reorganization these days. Most will agree that Atari needed to do something to focus their computer business, and it seems that Atari has come to the same conclusion. Recently, they set up a new computer division called—simply enough—Atari Computer. Whether this means that we'll see the employment of some new marketing strategies, no one is really sure, but the word "advertising" has been bandied about quite a lot lately, and it could be that—at last—ST computers will be getting the exposure they need to ensure success.

According to Atari's Neil Harris, the Sunnyvale company's marketing organization is getting revamped. Mentioned in a recent DELPHI conference were the formation of a telemarketing group and the taking on of a new public-relations agency, one that specializes in computers. Changes for the future also include a new advertising agency.

"Reading the tea leaves," commented Harris, "I have to think this means that corporate management is getting very serious about the USA. They would not be letting us spend the money to do all this, otherwise."

This statement, while providing hope for Atari's supporters in the U.S., concedes the fact that Atari has been sidestepping the U.S. market in favor of the higher European sales—something that we've all come to realize over the last couple of years. Of course, one can't *blame* Atari for going where the paper is greener. They are, after all, a small company with limited resources, able to spread themselves only so thinly before they cease to be able to function. The lack of attention granted the U.S., however, has made a great many people wary of the STs and has made others—people who have already purchased their STs and fear for the future of their investment—downright bitter.

While many past promises from Atari have come to naught, I have a feeling that this time we may see some change. Atari has to have realized by now that if they ignore it much longer, the U.S. market will slip away from them. They've played the waiting game long enough. It's time to tackle the U.S. with aggressive marketing strategies—time to grab every American contemplating a computer purchase and scream, "Power without the price!" into their faces so loudly that they have to respond. In this media-saturated society, the winners are those who scream the loudest.

You and I have known for a long time that the ST is something special, and it's only natural for us to want to share it with our fellow citizens. It's ST-Log's hope that Atari is now ready to bring the rest of the U.S. in on our little secret, that they have at last realized that the time has come to display the ST to the American people in a way that cannot be ignored.

READER COMMENT

VIP Service

As the frustrated owner of a 1040ST unable to videotape my creations, I was thrilled when I received a call from Practical Solutions, asking if I was still interested in the Videokey converter. My overwhelming "Yes!" still bellowing over the phone lines, I received my Videokey a week later.

I promptly hooked it up, and after a minute adjustment, per the included instructions, I was in business. Most of the colors looked fine, but I had a problem with oversaturation of certain blues on my composite monitor. The problem looked worse upon viewing the videotape back. These blues were from the CAD 3-D demos I had made.

A call to Practical Solutions returned me a call from Mark, one of their engineers. I explained my problem to a friendly, sympathetic and knowledgeable person who kindly told me to send it back for a modification. I included a videotape to help describe my problem with the oversaturated blues.

Ten days later, I received my Videokey along with my videotape and a personal letter thanking me for my description and cooperation. Practical Solutions also videotaped some pictures onto my tape after their modification, so I could see what they did. The next day at work, their product service rep called me to follow-up on my problem. He asked if I had received the Videokey and if I was satisfied. He also stressed that if there were any other problems to call right away so they could help out.

I cannot remember ever getting this VIP service from any company before. Practical Solutions is a first-class, professional, customer-oriented organization that really cares about the consumer. I would recommend its products and services without reservation, and its support of the Atari ST is a benefit for all of us. It deserves our support. Thanks to all at Practical Solutions for the help and sincerity.

—Barry Summer
Brooklyn, NY

Thank you for sharing your experience. It's always a pleasure to print this type of letter. Because we feel that the quality of a company's customer support is just as important as the quality of the products they sell, we believe that outfits such as Practical Solutions should be publicly applauded when it shows its customers consideration over and above the norm. Companies like this do more than satisfy their customers; they also help build up the ST's reputation.

The Adventure Continues

I read your review of my adventure book in the May issue of ST-Log and wish to personally thank you for the kind words. With reference to some of the statements in the review, I thought that a few words of clarification are in order.

First of all, the reference to the previous Arrays adventure books was done strictly as a matter of courtesy. For many years, I myself was an active reviewer of entertainment and educational software for Arrays, Inc., since at that time it was also publishing *The Apple Book of Software,* a series of softcover books dedicated to Apple II software reviews. During that period of time, I made personal acquaintance with Kim Schutte, the author of *The Book of Adventure Games, Vol. I & II*, and, in fact, was acknowledged in Volume II for my help thereof. The reason I speak

READER COMMENT

in the past rather than present tense of my relations with Arrays is that it is no longer in the book publishing business. Thus, the reference to a volume III of *The Book of Adventure Games* is not a reality. The fact is that I have maintained my relations with Kim and thus know of his activities. At times, he has entertained the notion of another adventure game book but, to the best of my knowledge, has not gone forth with the venture. If he does indeed decide to do so, it certainly won't be with Arrays. Thus, in summary, there is no one at Arrays "jumping out of a window with volume III" although, if it had any smarts, it would have continued to put out such books. For a long time, that was the only profit-making venture it had and what kept it solvent for so long.

Although my book resembles the Arrays books in format, it differs in some significant ways. The idea of scrambling the clues in the simple-to-use and efficient way in which it is handled is something I am very proud of. Indeed, the review in the January issue of *A+ Magazine* gave credit to this feature. Also, the inclusion of "walk thrus" is a new feature which I have personally received many compliments for. At the least, it saves on a lot of fan mail coming from people asking for more clues than those I have provided.

With regard to the production of the book itself, you are correct in assuming that it was done on an Apple II. The book was produced by a friend of mine, Jerry Jones (see the acknowledgments), on a Macintosh Plus using a combination of MacDraw, MacWrite and PageMaker. I myself prepared (and typed) the rough draft on an Apple II but the final camera-ready copy was done as described above.

Finally, if you liked Book I of *Keys to Solving Computer Adventure Games* you will surely like Book II, which was released early this year and is doing quite well thus far. Although the format is the same, the games contained in the second book are, of course, different and much more current (most are of 1987 vintage). Also, the overall appearance of the production is much more elegant (just look at the title pages, and the clue sheets, and you will immediately see the difference).

By the way, Book III is in the works. I have played, finished and prepared the material for 14 new games (e.g., Dondra, Beyond Zork, Sherlock Holmes in Riddle of the Jewels, King's Quest III, Plundered Hearts, etc.) thus far and essentially am at the point of waiting for more adventure games to be issued.

—Marvin K. Simon, Author
Keys to Solving Computer Adventure Games

Thank you for the extra information and clarifications, Marvin. We wish you the best of success with your future projects and can assure you that reviews of your new books will grace the pages of ST-Log. *There's only one question left unanswered: How the devil do you fit enough hours in the day to finish all those games?*

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

SAVE TIME AND MONEY SUBSCRIBE TO ST-LOG

SAVE $14 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ST-LOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS! GET THE MOST OUT OF YOUR COMPUTER

SUBSCRIBE TO ST-LOG TODAY!

☐ 1 YEAR @ $28 — SAVE $14! MCHWW
FOREIGN — ADD $7 PER YEAR

☐ 1 YEAR WITH DISK @ $105 DCHWW
FOREIGN — ADD $15 PER YEAR

☐ PAYMENT ENCLOSED ☐ BILL ME

CHARGE MY: ☐ VISA ☐ MC #________________

EXPIRATION DATE ________ SIGNATURE ________

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME ________________

ADDRESS ________________

CITY ________ STATE ________ ZIP ________

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires OCTOBER 26,1988. Your first issue will arrive in 6 to 8 weeks.

WATCH FOR IT!

ST NEWS

Michtron Introduces New Titles

Slaygon is a new arcade-style maze game from Michtron. The game offers a graphical 3-D maze game along the lines of Dungeons & Dragons. As a voyager you travel through a building of corridors and rooms picking up objects. The game has good graphics and a price of $39.95.

GFA Artist, $79.95, is an animation and graphics program. Similar to DEGAS Elite, images are developed using a standard painting metaphor. Animation scripts are created to animate the sprites. The program can generate more than 1,000 colors per screen.

Michtron is now also an educational software supplier. The new Kidprogs is geared towards preschool children. For $39.95, it offers elementary interaction with music, pictures and the alphabet. ABZOO ($29.95), is a letter recognition program for toddlers. The screen shows pictures of objects that correspond with keys on the ST keyboard. Invasion is a $29.95 typing tutor program where letters and words appear at the top of the screen and tumble downward until the corresponding key is pressed on the keyboard.

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

CIRCLE #130 ON READER SERVICE CARD.

Supra Corp. has begun shipping a 45-Megabyte hard disk that is installed inside a Mega ST.

CompuServe Vendors Forum

A new service is being offered to CompuServe users. The online modem service now has a forum devoted to Atari software and hardware vendors. The ATARIVEN forum is split into several company topics, with the manufacturers supporting their own topics. Currently the following companies can be found in the Vendors Forum:

Atari Explorer Magazine 74710,13
Data Pacific 76004,1612
ICD, Inc. 76004,1600
Intersect 76004,1577
Michtron 76004,1607
QMI, Inc. 76004,1601
Regent Software 76004,1573

ANALOG and ST-Log magazines will also be participating in the new forum. Type GO ATARIVEN for more information on CompuServe.

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

CIRCLE #131 ON READER SERVICE CARD.

Publishing Partner 2

The 1986 release of Publishing Partner (PP) was heralded as the answer to the missing Atari desktop publishing system. At $149.95, PP was a bargain. Soft Logik has now released Publishing Partner Professional. At a high retail price of $199.95, the new system sports auto-text flow around graphic images, auto hyphenation, kearning, an UNDO command, special text effects like slant, twist and rotate, and more included fonts. Word-processor files may be imported directly from WordPerfect, First Word, Regent Word II and Word Writer files. Upgrades for PP 1.1 owners are available directly from Soft Logik at a cost of $99. PP 1.1 will continue to be marketed at a lower price, so you will have the option to try the lower priced package first, then upgrade to PP Professional later.

Soft Logik
11137 S. Town Square -C
St Louis, MO 63123
(314) 894-8608
CIRCLE #132 ON READER SERVICE CARD.

Jefferson Software is now shipping more than 1,000 pages of information on GEM and the ST packaged to look like a fishing tackle box.

Desktop Publisher ST

Timeworks has introduced a desktop publishing system for the ST. At $129.95, Timeworks' new product becomes the third DTP package for the ST after the successful Publishing Partner and Fleet Street Publisher DTPs. Timeworks' Desktop Publisher ST is different from the other DTPs, as it includes a word processor to enter your text. Page layout, graphic setups and typesetting abilities are all included, allowing the user to move freely between all of its functions. The program is GEM based and was written by the same people that developed 1st Word in England. DEGAS, Neochrome, GEM Draw and Easy Draw graphic files may be imported. Text from Word Writer, 1st Word, WordPerfect, Regent Word II and ASCII files may be imported. Kearning and leading are supported with automatic text flow.

Timeworks
444 Lake Cook Road
Deerfield, IL 60015
(312) 948-9200
CIRCLE #134 ON READER SERVICE CARD.

Tackle Your ST

Jefferson Software is now shipping the Tackle Box ST, literally more than 1,000 pages of information on GEM and the ST packaged into what looks like a fishing tackle box. The package sells for only $69.95 and even includes GDOS support. Jefferson sells their own GEM version of Modula 2—Modula is an interesting language that is the successor to Pascal. JS Modula 2 includes source code for GEM and TOS application development at a cost of only $49.95. Jefferson also sells XPRO Prolog, a fifth generation artifical intelligence language that conforms to the Edinburgh standard syntax for only $59.95.

Jefferson Software
12416 N. 28 Drive -18-236
Phoenix, AZ 85029
(602) 243-3106
(602) 276-6102 (JS BBS)
CIRCLE #133 ON READER SERVICE CARD

Supra Ships New Drives

Supra Corp. the leading hard-disk supplier to the Atari market, has released the FD10, a 10-megabyte removable floppy-disk drive that runs about as fast as a normal hard disk. With the FD10 you can change disks, impossible for a hard disk, so backing up your information is easy. The FD10 sells for $895.00.

Supra has also begun shipping a 45-Megabyte hard disk that is installed inside a Mega ST. The unit is installed by Supra dealers. At a cost of $1,095, the new Supra unit is quite a bargain.

Finally, Supra has announced that all of their hard-disk drives will use a redesigned controller which includes output to SCSI hard disks and DMA devices (such as the Atari hard disk or Atari Laser Printer). The new controller also has a battery-powered clock to reset your ST's internal clock whenever you turn on your ST. Upgrading your existing Supra controller will cost approximately $100. Contact Supra for more information.

Supra Corp.
1133 Commerical Way
Albany, OR 97321
(503) 967-9075
CIRCLE #135 ON READER SERVICE CARD.

GEM Directory

Digital Research of the United Kingdom has published a directory of GEM applications for the IBM PC and the Atari ST. The directory gives an overview of what GEM is and how the standard graphical user interface has evolved. Then more than 200 listings of GEM applications are given. The listings include summaries of features, file format support, national language support and minimum system configuration. The directory cleaves between ST and PC products. However, since it is published in the UK, many of the ST products are those unusual European programs you hear so much about.

Digital Research (UK)
Dept GEM, 70 West Way, Botley
Oxford OX2 9JT, ENGLAND
CIRCLE #136 ON READER SERVICE CARD.

ST GOSSIP FROM HOLLYWOOD USA

by TG

Did you hear? At least a half a dozen companies have been trying to license Dr. T's multi-tasking operating system for use on their non-music applications. The Dr. is thinking about it—more to follow next month.

If someone hasn't already told you, Atari will be shipping the Imagen Postscript software clone for their laser printer within the next 30 days. The product is technically called **UltraScript** and is a full Postscript clone. Imagen is a division of QMS corporation and its reputation is one of quality products and full compatibility with the postscript standard as set by Adobe, Corp. Imagen has had a fast text rotation and display system for quite some time now, and their new Postscript compatible version is said to be quite a bit faster than the routines provided by Adobe systems. The emulator will load off disk much as GDOS does. By using this disk-based system Atari will be able to continue to offer full support for products already released designed to run under the current GDOS system.

This makes the story that Atari is preparing to offer bundled systems considerably more interesting. The story goes as follows: Atari will be supplying Mega dealers with a **complete desktop-publishing system** which consists of a monochrome Mega 4, an Atari laser printer and **Publishing Partner II**. Atari is also considering several other bundled systems featuring their Megas and software. In effect, these will allow the dealer to set himself up as a VAR (value added dealer) without any heavy investment in either hardware or custom software.

Rumors concerning the new **STL** (ST Laptop) have reached a fever pitch. It seems a forgone conclusion that the project is being worked on at Atari and the only major questions left concern the type of screen display to be offered and, of course, the big one. . .*when*?

Among the strong possibilities for the fall season is a cartridge containing a **PC emulator** to be marketed by Atari Corp. This cartridge is expected to be quite a bit faster than the current version of **PC Ditto** since it will be ROM-based rather than requiring you to load and run software from a disk. While we are on the subject of a faster PC Ditto type program, we hear from Bill Teal, the author of PC Ditto, that the newest version is *much* faster than the current one, but there are still a few bugs to be ironed out. Bill is the kind of guy who will not release (or even promise to release) something before he knows it's ready.

The ST is now the hottest computer in the world for two specific applications. One that I'm sure you already know about is music; the second is *chess*.

The ST is now the hottest computer in the world for two specific applications. One that I'm sure you already know about is music; the second is **chess**. When the Russian world chess champion played an exhibition match in Norway (playing against 30 players with master ratings) about two months ago, he was shown a new chess database. After viewing the information on their games as stored and displayed in this database program, he remarked that he would be willing to return to play again the following year only if he were given a copy of the program and an ST to run it on in lieu of pay. The news of his remarks has swept the chess underground (yes, of course there is a chess underground). The package is at this point not available in English, but expect that to change shortly.

Where is the new **Exel** clone that Atari was showing in Hanover last April? Several reps from Michtron were at the show and saw a copy of this package which they called extraordinary.

The *final* revision of **WordPerfect** has been out a while now, and it's working the way we all knew it would. The company admitted that they may have released the product early but made up for this eagerness by sending out countless free upgrade disks to all registered owners. Quality will out!

As many of you already know, **David Small** of Magic Sac fame has moved on from Data Pacific to other things. AT&T offered him an outstanding opportunity, with a chance to work on much larger systems. Too bad Atari didn't have a chance to grab him and put him to work in their overworked programming dept.

It's official: **Dungeon Master** from FTL is the best-selling ST software of all time. . . .

Is **shareware** worth writing for the ST? We have talked to several authors who have products in the shareware market for both the Atari ST and other makes of computers. They report that the owners of Ataris seem to be the cheapest PC owners they write software for. This is too bad, since much of the shareware for other computers is actually better than the commercial software it competes with. I would guess that a product like ARC.TTP is used by at least 50,000 people on a fairly regular basis, yet the author has not received contributions that would equal even 10¢ per user.

Heard about the new GEM for the ST? A rather seedy looking individual with a strong foreign accent sat down at my table last night as I was finishing off an evening drink. He whispered in my ear that **GEM 2.2** is now being marketed in Germany. It seems that another company (in other words: not Atari) is offering the package, and it's a beauty. You can buy it with GEM Write, GEM Paint and GEM Draw or any combination of those titles, and it appears that all the bugs built into the original GEM have been fixed. And fast! It's written in 100% machine language and performs like the Tempus text editor. The German price is around 300dm. A quick calculation told me that this is around $148.00 U.S., and I had my checkbook out in a flash. Unfortunately, there is no U.S. importer for the product yet. Anyone with overseas connections want to make a small fortune?

The corporate restructuring at Atari continues with titles and responsibilities being moved around like playing pieces on a checkerboard. What will come of it? *Advertising*! Look for those that wind up with the responsibility for making **Atari Computer Corp.** fly (yes, I mean a new company created under the banner of Atari, Corp.), finally deciding to spend a little money to tell the non-ST owners what we've been keeping secret for so long—that the Atari is one of the most powerful machines in the under $3000 class.

Best guess is that the total number of software titles on sale in Europe is about double that of the titles in the U.S. With a little advertising support from Atari, the chances are good that more of these titles would find their way here and with them would come an increased interest in the ST and its "broad software base." This would of course lead to more sales of the ST, which would lead to more software being written for the ST, which would lead to more ST sales, which. . . Is it any wonder that the owners of STs are the ones pushing for more advertising for the product?

The new high-speed dBMAN blows away some of the biggest names in the PC world and does it at a price that makes the competition turn white with fear.

Networking for the ST (under GEM) is at least six months away. When the ST was first released the idea of a MIDI network seemed natural for the machine, since MIDI is faster than the RS232 port that is used in many low-cost systems. Unfortunately there is a bug in the GEM system which has beaten everyone we know of who has tried to write a reasonably fast MIDI network system. As far as help from Atari goes, there seems to be little they can or will do. Could this be because they have announced that they will be shipping their own **PromisedLan** sometime? P.S. "Sometime" looks like the beginning of '89 at the earliest.

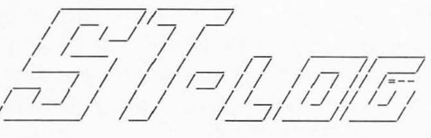
The new high-speed **dBMAN** compiler might make the dBMAN version IV the single most-popular programing environment available on the ST for small VARs. This little database blows away some of the biggest names in the PC world and does it at a price that makes the competition turn white with fear. It is a source of constant amazement to me that a package like dBMAN—which is a complete dBASE—clone does not do better in the ST world than it does. dBASE has become one of the standards of the business world with less power than dBMAN. There are literally thousands of business applications written in dBASE currently being sold to IBM owners. As an extra bonus you can buy the ST and dBMAN for a low-cost development system, and then use the IBM version of the compiler to compile your code to run on the IBM's. The features and syntax are identical.

TG can often be found skulking the turf around Hollywood and Vine. He is often seen frequenting bars in the area, carrying what he calls the only true portable computer: an ST plugged into a heavy-duty truck battery. He writes this column to make a living until he breaks into film and to provide the cash he needs to recharge his truck battery every month. Heard anything good? Write it down and stick it with used gum on the underside of the payphone at the address above. (Don't live in La-la land? Then send TG's mail to: ST-Log, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.)

DATABASE DELPHI

by Andy Eddy

As we noted last month, the ST contingent on Delphi has been moved to a separate area. You may access the new ST SIG by typing "ST" at the Group > prompt. Here's how the entry to the ST area looks (thanks to Maurice Molyneaux (MAURICEM) for the logo).



```
welcomes you to the DELPHI Atari ST SIG

Announcements          Request Free Upload
Conference             Set Preferences
Databases              Topic Descriptions
Entry Log              Who's Here
Forum (Messages)       Workspace
MAIL (Electronic)      Help
Member Directory       Exit
Poll

Atari ST>What do you want to do?
```

The menu is much the same as in the 8-bit area, so familiarity should reign. For obvious reasons, the topics within the ST SIG—for forum messages and database entries—have been changed from the topics in the ANALOG/Atari SIG. Here's what we've got to work with now:

```
General Information      Desktop Publishing
Applications             Educational
Games & Entertainment    Telecommunications
Art                      News & Reviews
Programming              Current Issue
Utilities                Recent Arrivals
Sound & Graphics
```

The recent "recent arrivals" arrival

On the list, you'll see a category called "Recent Arrivals." This category was requested by popular demand (as voted on by the masses after a Poll was created to settle the debate), based on MADMODIFIER's (Lloyd Pulley) recommendation.

His view was that with all the topics available, it could take a while to select a topic, run through the entries until you caught up with your last search, repeat the process with the next section, and so on. With a Recent Arrivals area, you have to pop only into *that* section for new files; if you're looking for older entries, they can be found in their designated areas. Files will only reside in the Recent Arrivals topic for a couple of weeks, at the most, giving everyone a chance to scan and download them from there.

The "Applications" topic will cover most of what was in the "ST Programs" area prior to the split; the other topic names are mostly self-explanatory. Keep in mind that we are *always* looking for new public domain files to offer to our online visitors, so if you get an interesting or helpful file from another user or local BBS (or have written it by yourself), please submit it to us for inclusion in the libraries.

Okay! I submit!

If you are new to submitting software, it can be somewhat intimidating. Here's an overview of the process:

The first thing to do is request time for uploading. Most of the major online services now offer free uploading, as an incentive to building up the software libraries. In Delphi's case, they prefer to keep file repetition down to a minimum by having the SIG managers confirm what a user is going to upload and when. To request free time, simply type "REQUEST FREE UPLOAD" (or "REQ" for short) at the main SIG prompts—that's what I love about Delphi: its easy-to-understand, English command set. Completing a short online form will leave a note for the SIG manager, who will then set up an appointment (based on your request form). After the appointment is logged in, you'll get a confirmation through E-mail.

After you're given the go-ahead, the next step is to upload the file to your Workspace. Entering "WOR" (short for WORKSPACE) at either the ANALOG> or Atari ST> prompt will bring you there. This Workspace is where Delphi stores your mail file (MAIL.MAI) and large mail messages, and where you can store files and messages of interest. This is your personal storage area and no one else has access to it. Remember there is a charge for storage over 50 blocks (Delphi's blocks are 512 bytes in size), which is billed at the start of each month, so be sure you frequently peruse your Workspace and delete unwanted files. To get the current charges, type "RAT POL" (short for RATES POLICY) from the "Using Delphi" selection off the main menu; at the time of this writing, the surcharge was 16 cents for every two additional blocks.

The actual transfer of the file(s) will be taken care of by your terminal software. Virtually all commercial telecommunications packages—*Flash!* and *Interlink* for the ST, for example, are very popular and strongly recommended—are programmed for a variety of protocols, as there are many methods of doing this: ASCII (for text files), Xmodem, Ymodem (for binary files), Ymodem Batch, Kermit (for multiple file transfers), etc. Further information on these can be found by typing "OTHER" and "NEW" at the WS> prompt. Make sure you use the same transfer protocol as you select from the Workspace menu.

Once the file resides in your Workspace, you are ready to move it to the databases. From any Database topic prompt or the WS> prompt, type "SUB" (short for SUBMIT) and follow the prompts—just be sure when you submit, that you are in the SIG the file should be posted in. Figure 1 shows a sample submission, with my entries in capital letters and notations in [ ].

Holding out until the house is finished

Once that procedure is completed, don't fret if your file and description are not immediately visible in the selected area. The reason for this is that it will first go to a holding area for checking by the Sysops. This ensures that the file is not a duplication of another user's entry and that the file is intended for public viewing. Once this previewing is completed, the file will be moved into the public areas for all to see.

```
Name: SUBMISSION RUN-THROUGH
Type: MISCELLANEOU
Date: 21-APR-1988 21:14 by ANALOG2
Size: 8034 Count: 0

THIS IS A TEXT FILE OF HOW TO SUBMIT FILES FOR THE
DATABASES...

Topic: News & Reviews

Keywords: OTHER, TEXT, SAMPLE
```

One more thing to note: When you're finished with your transfer and submission process, you'll have some free time to peruse the area and download some files on our nickel. Consider this a reward for your hard work. The databases are one of the most heavily accessed areas of any SIG, which means an investment return for the SIG managers. Your contribution is valuable to everyone, especially if the program is one that helps the users save time or accomplish something they normally couldn't as easily. A good example is Charles F. Johnson's (CFJ) *ArcShell* program (originally published in the April 1987 ST-Log, but it has gone through a number of *substantial* upgrades since that time), which frees you from remembering the parameters to enter into ARC.TTP.

Let your fingers do the talking

We're doing our best to bring some of the major ST software developers online for product support and formal conferences. The latter lets the users conduct a press conference, with the manufacturer in the hot seat. By the time you read this, we'll most likely have had representatives from Michtron, FTL and Atari in CO (the abbreviation for conference) to tackle your questions in real-time.

To keep up with the events of this sort and when they will take place in the Atari SIGs, be sure to read the brief notices that come up when you enter. As well, there is an Announcements area that lets you read through these notices. By typing "AN" at either the ANALOG> or Atari ST> prompts, you can pick a specific area—Conference, Database, etc.—to read what is developing.

The most active of these areas is the What's New section, which contains the brief messages you see at entry to a SIG. Typing "SCAN" will show the list of messages, in case you missed them the first time. Generally these messages are shown to a user once and not repeated.

We've covered a lot of ground this session. Till next month, C U online. . . .

```
How many files will you be submitting? 1  [Files can be
grouped; for example, a few picture files may belong under
one description.]

Which of the following BEST describes the file(s) you are
submitting:
  Program or Program Pack
  Newsletter
  Article(s)
  Transcript
  Documentation
  Data (Graphics, etc.)
  Miscellaneous Text

Filetype: MIS

Topic? NEWS  [Hitting a ? here would bring a list of the
topics, but we are targeting the News and Reviews section.]

Enter a BRIEF description of the file you are submitting.
Press Control-Z when complete or type /HELP for help.
THIS IS A TEXT FILE OF HOW TO SUBMIT FILES FOR THE
DATABASES...
^Z

Primary Keyword: ?  [This ? gives a list of available
keywords for the topic you entered.]

The file you are submitting for publication in News &
Reviews must be given one of the following PRIMARY keywords:
  Games
  Utilities
  Applications
  Programming
  Other


Primary Keyword: OTHER
Primary Keyword: Other

You may now supply any optional keyword(s) of your choice,
subject to review by the database manager.  Please enter one
keyword or phrase per line, and enter a blank line or CTRL/Z
when complete.
>TEXT
>SAMPLE
>
Workspace filename: TEST.TXT
Now enter the "name" of the file as you wish it to appear in
the directory.
Display Name: SUBMISSION RUN-THROUGH
Please wait.

Would you like the file "TEST.TXT" deleted from your
workspace? (Y/N) Y

Your submission is complete.  Thank you.
```

Figure 1.

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 ($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 service) via a local telephone call. Make the Delphi connection by signing up today!

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 Video-tex Corporation of Cambridge, Massachusetts.

TUTORIAL

Sprite PROGRAMMING

by Kelly Schreiner

For those of you who grew up programming on the 8-bit Ataris as I did, ST sprite programming should be a fairly routine task, because the ST's sprites are analogous in many ways to the Player/Missile graphics we were accustomed to on the 8-bit machines. And for those of you that have never programmed either sprites or Player/Missile graphics, this is the perfect time to learn sprite programming for the ST computer. But first, let's make some comparisons between ST sprites and Player/Missile graphics, just so everybody knows what a sprite is to begin with.

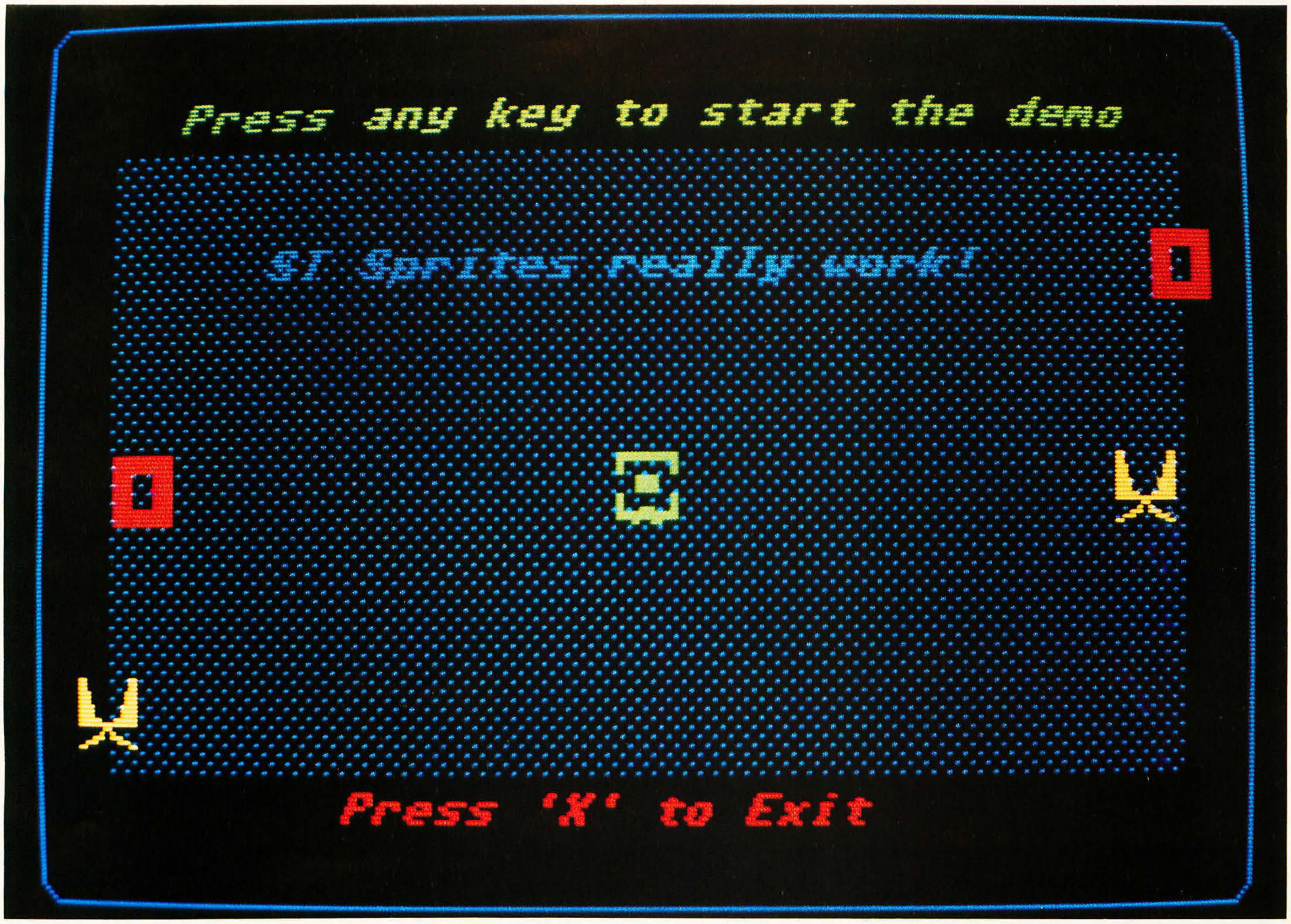
ST sprites and Player/Missile graphics are both user definable bit-block images stored in memory that can easily be moved around the screen independent of the background drawing. They both can have depth, (meaning that the first object drawn can pass in front of the second object, the second in front of the third and so on), and they both can have only one color. (Unless, of course, two or more objects of differing colors are overlayed on top of each other). Both Player/Missile graphics and ST sprites are commonly used in games for arcade-type action on their respective computers.

Now that the similarities between ST sprites and Player/Missile graphics have been looked at, let's examine the differences between the two. One of the major differences between ST sprites and Player/Missile graphics is that the ST sprites are purely software-based sprites, not hardware-based, like Player/Missile graphics. The ST sprites are also limited in size; they measure 16x16 pixels instead of 8 pixels wide by either 1-128 or 1-256 pixels high for Player/Missile graphics. The ST sprites are considerably easier to create than Player/Missile graphics though, and you can have practically as many sprites in a program as you want compared to the four players along with their missiles that Player/Missile graphics offers.

To be able to program sprites on ST computers, you need to understand the use of Line A opcodes.

As I said earlier, sprite programming on the ST computer is a relatively easy task to accomplish. That is, once you know how! The main reason for this is the fact that most of the work related to sprite programming is already done for you in the Line A opcodes, $A00D (DRAW__SPRITE) and $A00C (UNDRAW__SPRITE).

By using the DRAW__SPRITE opcode, ($A00D), the sprite's X/Y coordinates on the screen can be established along with the sprite to be seen and its associated background buffer. The background buffer is needed to temporarily store the background behind the sprite, so that when the UNDRAW__SPRITE opcode is called to erase the sprite, the background can be restored to its original condition, just like



it was before the sprite was drawn over it.

That's basically all there is to programming sprites on the Atari ST. Now we'll get down to the heart of the matter and take an in-depth look at how to program the various structures needed to produce ST sprites. (This is also the code to use for drawing and erasing the sprites normally, instead of the macro functions used in the program listing, just in case you don't have a macro assembler.)

One of the most important things you need to understand to be able to program sprites on the ST computers is the use of the Line A opcodes. These opcodes are the very heart of the ST's powerful graphics capabilities.

The software developers of the ST have made use of the fact that the 68000 processor used in the Atari ST computers has two groups of opcodes which it does not understand, and which consequently generate a software interrupt when encountered in a program. These opcodes begin at memory locations $Axxx and $Fxxx. The Atari ST uses the $Axxx opcode trap, referred to as the Line A Handler, to access its graphics routines. The trap handler that processes this trap recognizes word opcodes that begin with the bits 1010 (hexadecimal A) as unimplemented instructions and then jumps through a special exception vector, which in the ST's case points to the Line A routines. The lower four bits of the word opcode used contain the number of the routine to be accessed. Only values between $0 and $E are allowed here. That means that a total of 15 different opcodes are available for the Line A graphics routines. This number includes the initialization opcode, though, so there are actually only 14 graphics routines. In this demo program only the word opcodes $A000, $A00C, and $A00D are used, so I will confine myself to describing those.

In order to use the Line A opcodes you must first label the word opcode functions you want to use in your program. Here is the way these functions are labeled in the demo program SPRITES.

```
LINE_AI       = $A000 ;Initialization opcode
DRAW_SPRITE   = $A00D ;Draw sprite opcode
UNDRAW_SPRITE = $A00C ;Erase sprite opcode
```

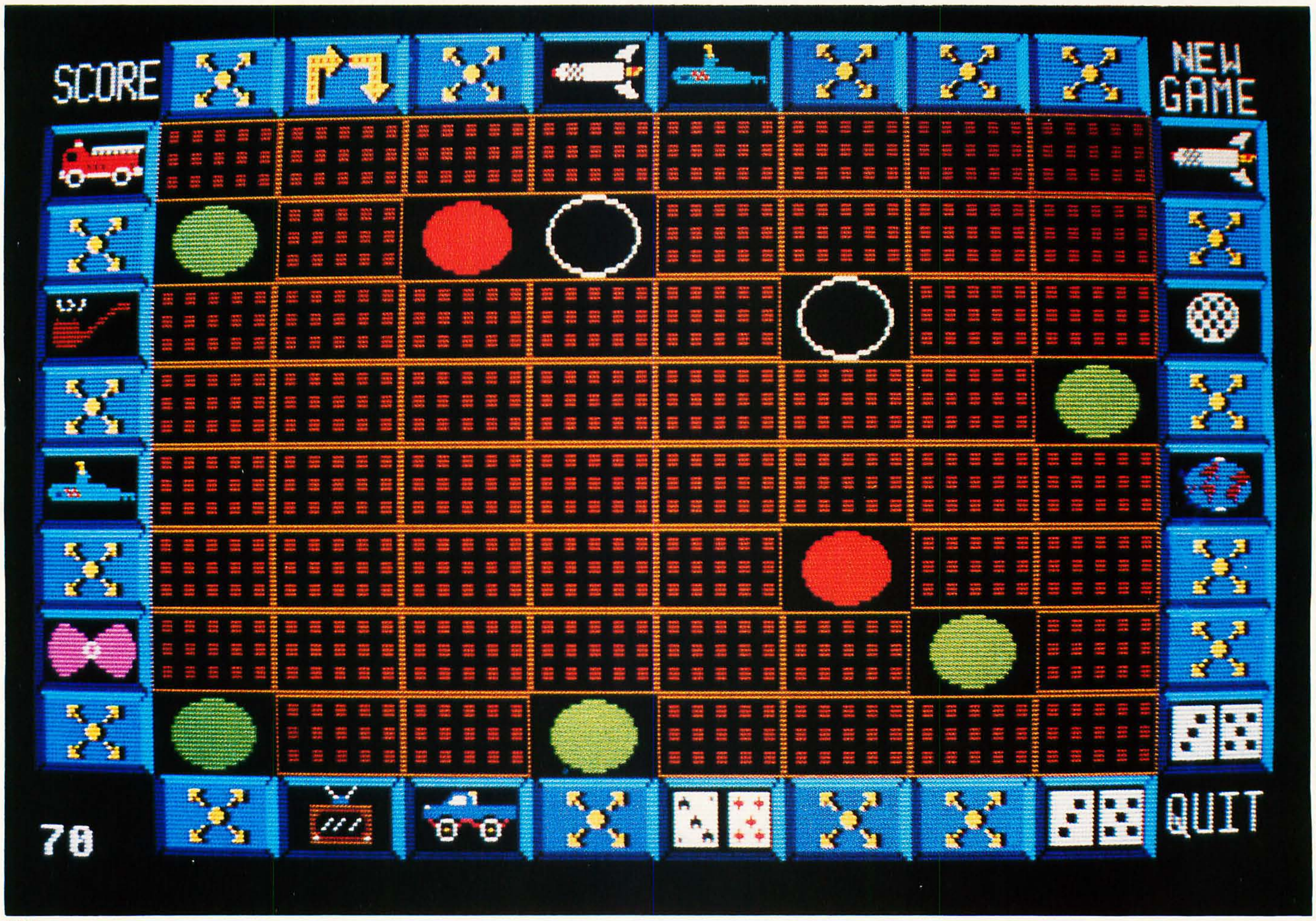
The next step is for you to initialize the Line A routines for use in the program. The opcode $A000, Initialize, is used to do this; it determines the address of the Line A routines. After calling this function, data register D0 and address register A0 point to a table with the starting address of the Line A variables. Address register A1 points to a table with the starting addresses for the three system font headers, and address register A2 points to a table that specifies the starting addresses of the 15 Line A opcodes. There's no parameter required for this function, so all you have to do is call the word opcode label that you specified for the $A000 (Initialize) function.

In order to draw a sprite onto the screen there are three things that need to be discussed. These are the DRAW__SPRITE opcode, the sprite definition block, and the sprite's background buffer. We will look at each of these structures individually.

The Line A opcode $A00D, DRAW__SPRITE, draws the desired sprite to the screen at the specified X/Y coordinates. To use this opcode the first thing you have to do is move the sprite's X coordinate into data register D0. Then, you have to move the sprite's Y coordinate into data register D1. After setting

continued on page 62

GAME MER BOX



LIN'S

by Brian James Parry

Merlin's Box is a single-player game in which you try to locate several glass spheres. The spheres are hidden within Merlin's Box and cannot be seen from the outside. Clues to where the spheres are located are obtained by shooting a light into the box and noting where it exits, if at all. When you believe you have located all the spheres, their correct locations will be revealed. Based on the number of correct choices, you will be given a score from 0 to 100, 100 being the best.

To create your copy of Merlin's Box from the program listings included here, you must first type and compile each listing. (Listing 1 requires Personal Pascal compiler and Listing 2 requires an assembler.) After compilation and assembly, you should have two files with a .O extension. Use the Personal Pascal linker to link these files. You will then have an executable .PRG file that can be run from the desktop.

Playing the Game

Merlin's Box is a low-resolution color game, so make sure you are in low-resolution mode. Double click on MERLIN.PRG. When the title screen appears, click on the OK in the title box. Next, you will be asked to choose a skill level, 1 through 3. In skill Level 1 you must locate three spheres, with Level 2 locate five spheres, and Level 3 locate seven spheres. Level 1 is pretty easy because there is less deflection from other spheres to confuse you. The seven spheres in Level 3 tend to bounce light around before it leaves the box, making for a few misjudgments as to the locations of the spheres.

Once a level has been chosen, the spheres are randomly placed and the box is drawn. On each side of the box are eight positions from which to shoot light into the box. To shoot light click on one of these outside squares. A symbol will mark the entrance, and you will hear the light enter the box, bounce around and exit.

When the light enters the box it can exit one of three ways. First, it can be deflected or go straight through, in which case a matching symbol will be displayed at its exit position. Second, it can exit where it entered. Here, a U-turn symbol will be placed. Last, the light can directly hit a sphere and be diffused throughout the box and not be strong enough to exit. In this last case a "Hit" symbol will appear. Please refer to Figure 1 for an illustration as to how the light can act inside the box. One situation not yet mentioned is a Stall. Here the light is not able to enter the box because of an adjacent sphere. Stalls can be determined because no exit position is shown and neither a U-turn nor a hit is displayed.

To guess where a sphere might be, click on one of the 64 squares in the box. A circle will appear marking your guess. If you wish to change your guess, click again on the circle, and it will disappear. You can then reposition your guess. In the lower left corner of the screen is the number of spheres in the box. You must make a guess for all of them before obtaining your score.

To get your score click on "SCORE" in the upper left of the screen. One by one the spheres will appear as either green or red. Green spheres mark a correct guess while red ones mark an incorrect guess. Your score will appear in the lower left of the screen. After a few games you will get the hang of it and then it will be time to move up a skill level. You can quit any time by clicking on "QUIT" in the lower right corner. Lastly, a hint. For those of you who are good listeners, try to listen to how and when the light deflects. The sounds come fast but can aid in locating tricky spheres.

Brian J. Parry is a computer science student at the California State University, Chico. He has published several programs for the Apple II and this is his first Atari ST program.

Merlin's Box — Listing 1
Pascal

```
{
*******************************************************************
*******************************************************************
** MERLIN'S BOX                                                  **
**                                                               **
** by Brian J. Parry              Copyright (C) 7,1987           **
**                                                               **
**                                                               **
**                                                               **
**                                                               **
*******************************************************************
*******************************************************************
}

program MERLINS_BOX;

        const   {$I gemconst}
        type    {$I gemtype}
        var     txt1,txt2,txt3:string;
                m_box, m2_box : array [0..9,0..9] of integer;
                la : array [1..4,1..8] of integer;
                count,pattern,button,guess_num,
                i,j,dir,msx,msy,skill_level,dummy : integer;
                msg : message_buffer;
                quit,new_game:boolean;
                desk_colors : array [0..15] of integer;

{$I gemsubs}

{*PE*}
procedure DELAY (t:integer);
var i,j:integer;
begin
     for i:=1 to t do
     for j:=1 to 1000 do begin end;
end;

function LOGICAL_BASE : long_integer;
xbios(3);

function RANDOM : long_integer;
xbios(17);

function GET_RES : integer;
xbios(4);

function s_color (cn, cl : integer) : integer;
xbios(7);

procedure STDRW (screen_adr : long_integer; shape_num : integer);
external;

function CALC_SCRN_ADR (x, y : integer) : long_integer;
begin
     calc_scrn_adr:=logical_base+y*160*18+x*16+1600;
end;

procedure DRAW_SHAPE (x, y, sn : integer);
begin
     stdrw(calc_scrn_adr(x,y),sn-1);
end;

function GIA_READ (d,r:integer):integer;
xbios(28);

procedure GIA_WRITE (d,r:integer);
xbios(28);

{*PE*}
procedure DO_SOUND (pitch,time:integer);
var state:integer;
begin
     state:=gia_read(0,7);
     gia_write(state&(~7),135);
     gia_write(15,136);
     gia_write(pitch&$FF,128);
     gia_write(shr(pitch,8),129);
     delay(time);
     state:=gia_read(0,7);
     gia_write(state|7,135);
end;

procedure INIT_PALETTE;
begin
     set_color(0,0000,0000,0000);
     set_color(1,1000,1000,1000);
     set_color(2,1000,0000,0000);
     set_color(3,0000,1000,0000);
     set_color(4,0000,0000,1000);
     set_color(5,1000,1000,0000);
     set_color(6,0000,1000,1000);
     set_color(7,1000,0000,1000);
     set_color(8,0000,0500,1000);
     set_color(9,0000,0125,0375);
     set_color(10,0625,0625,0000);
     set_color(11,0000,0625,0625);
     set_color(12,0250,0125,0000);
     set_color(13,0500,0000,0000);
     set_color(14,0500,0250,0000);
     set_color(15,0625,0375,0000);
end;

{*PE*}
procedure DRAW_BOARD;
var i, j :integer;
begin
     for i:=1 to 8 do
      for j:=1 to 8 do
      begin
           draw_shape (i,j,26);
           m_box[i,j]:=0;
           m2_box[i,j]:=0;
           do_sound (500,1);
           delay(1);
      end;
     for i:=1 to 8 do
     begin
          draw_shape(0,i,27); m_box[0,i]:=1; m2_box[0,i]:=1;
          draw_shape(9,i,27); m_box[9,i]:=1; m2_box[9,i]:=1;
          draw_shape(i,0,27); m_box[i,0]:=1; m2_box[i,0]:=1;
          draw_shape(i,9,27); m_box[i,9]:=1; m2_box[i,9]:=1;
     end;
     m_box[0,9]:=1;
     m_box[0,0]:=1;
     m_box[9,9]:=1;
     m_box[9,0]:=1;
     draw_shape(0,0,5);
     draw_shape(9,9,8);
end;

procedure PLACE_BALLS;
var i,j,x,y : integer;
begin
     for i:=1 to skill_level do
     begin
          repeat
                x:=int(random mod 10);
                y:=int(random mod 10);
          until ((x>0) and (x<9)) and ((y>0) and (y<9)) and (m_box[x,y]=0);
          m_box[x,y]:=-99;
          do_sound(200+75*i,4);
          delay(7);
     end;
end;

{*PE*}
procedure init_logic;
begin
     {1 -> }
     la[1,1]:=1; la[1,2]:=0; la[1,3]:=1; la[1,4]:=-1; la[1,5]:=1;
     la[1,6]:=1; la[1,7]:=3; la[1,8]:=4;

     {2 <- }
     la[2,1]:=-1; la[2,2]:=0; la[2,3]:=-1; la[2,4]:=-1; la[2,5]:=-1;
     la[2,6]:=1; la[2,7]:=3; la[2,8]:=4;

     {3 v }
     la[3,1]:=0; la[3,2]:=1; la[3,3]:=-1; la[3,4]:=1; la[3,5]:=1;
     la[3,6]:=1; la[3,7]:=1; la[3,8]:=2;

     {4 ^ }
     la[4,1]:=0; la[4,2]:=-1; la[4,3]:=-1; la[4,4]:=-1; la[4,5]:=1;
     la[4,6]:=-1; la[4,7]:=1; la[4,8]:=2;
end;

function GET_SKILL : integer;
begin
     button:=do_alert('[2][ Choose skill level ][ 1 | 2 | 3 ]',1);
     get_skill:=button*2+1;
end;
```

continued on page 67

MIDI

MIDIMON provides a detailed, easy-to-understand analysis of the MIDI data output from any synthesizer, drum machine, sequencer or similar controller. A necessity for MIDI programmers, a problem-solver for musicians, MIDIMON is also a fun and friendly program for anyone caring to unravel the mysteries of MIDI. Due to this program's large size, it's available only on this month's disk version or on the ST-Log SIG on Delphi.

by Larry Herzberg

A few words about MIDI

Have you ever wondered what was really coming out of the MIDI port of your synthesizer, drum machine or sequencer? Although in recent years there has been a great deal of progress in standardizing MIDI (which, in case you don't know by now, is an acronym for Musical Instrument Digital Interface), there are still wide gaps in the transmission and reception capabilities of various instruments. For instance, some synthesizers read "aftertouch" information—the changes in key pressure after a note has been triggered—while others don't. And some transmit variations in velocity data—just how quickly (or hard) a note has been hit—while others can't.

Sometimes such incompatibilities create performance problems for musicians who need to play several synthesizers with one physical keyboard. As a matter of fact, many synthesizers today have no keyboard of their own, and can only be played via MIDI, either from a "remote" keyboard or some other external controlling device (sequencer, MIDI guitar, MIDI drum pad or MIDI wind instrument; the list just keeps on growing). It can often be useful to determine which controls on the controlling device output MIDI data and which don't, and when they do, what controller numbers they have been assigned. Then, if the receiving synthesizers are flexible enough, they can be configured accordingly.

There are basically three types of MIDI information: Channel, System Common/Real Time and System Exclusive. Channel messages include note-on, note-off, control change, program change, pitch wheel, poly key pressure and channel pressure. Let's take a closer look at these kinds of messages and see how MIDIMON can be helpful in clearing up some of the more common problems associated with them.

Note-on and note-off messages both include velocity data, but many synthesizers don't transmit note-off messages at all. Instead, they turn a note off by transmitting a note-on message with zero velocity. This doesn't matter in most cases, but if you're trying to play a sound on a synthesizer sensitive to release velocity from a controller which doesn't transmit actual note-off commands (with their added capability of variable release velocities), you're not going to hear all the expressive possibilities programmed into the last part of the sound. MIDIMON can quickly alert you to this problem.

MIDIMON is a fun and friendly program for anyone caring to unravel the mysteries of MIDI.

Control changes include events associated with levers, sliders, pedals and buttons. Sometimes controls will surprise you. I have an older Yamaha DX-7 which I use as my main keyboard. It would be handy if its volume pedal output some kind of controller information, but I've discovered with MIDIMON that it does not output MIDI data of any sort. On the other hand, the "YES" and "NO" buttons have been assigned MIDI controller numbers of their own (#96 and #97, both outputting a value of 127 when pressed). Although this might be useful when programming other Yamaha synths from the DX-7, it is

MON

of limited utility when dealing with different brands of synthesizers.

Patch changes often yield unexpected results as well. For instance, when I press patch button #1 on my DX-7, it outputs a patch #0 message through MIDI. In other words, it's "off by one" (as far as MIDI is concerned) throughout its entire patch range. Other synths, with multiple banks of sounds, sometimes assign nonconsecutive patch numbers to adjacent banks. Fortunately, MIDIMON can help you map the patch layout of each of your synths so you can coordinate patch changes efficiently.

Pitch wheel data is different from other controller data in that its center (no-effect) position should have a decimal value of 8192. Going sharp increases this value, while going flat decreases it. Check your pitch wheel with MIDIMON to make sure it conforms to this MIDI standard. You can tell by looking at the last value the pitch wheel outputs as it returns to its home position.

Poly key pressure and channel pressure are two ways of transmitting aftertouch information. The difference is that poly key pressure transmits separate values for each note played, whereas channel pressure does not. MIDIMON will tell you which kind of aftertouch your synth outputs.

Finally, Channel Mode Select messages—Omni on and off, Poly and Mono—all concern the relationship be-

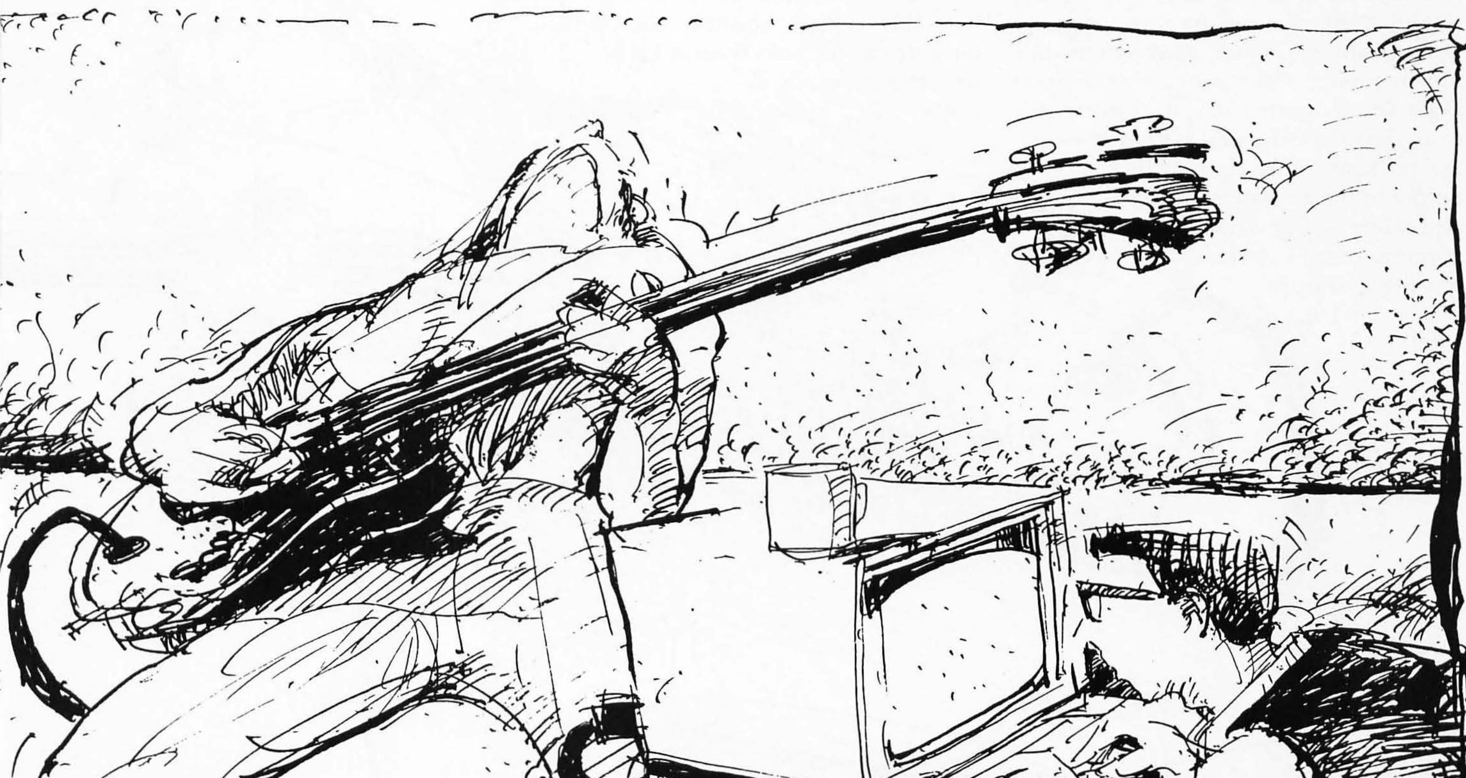
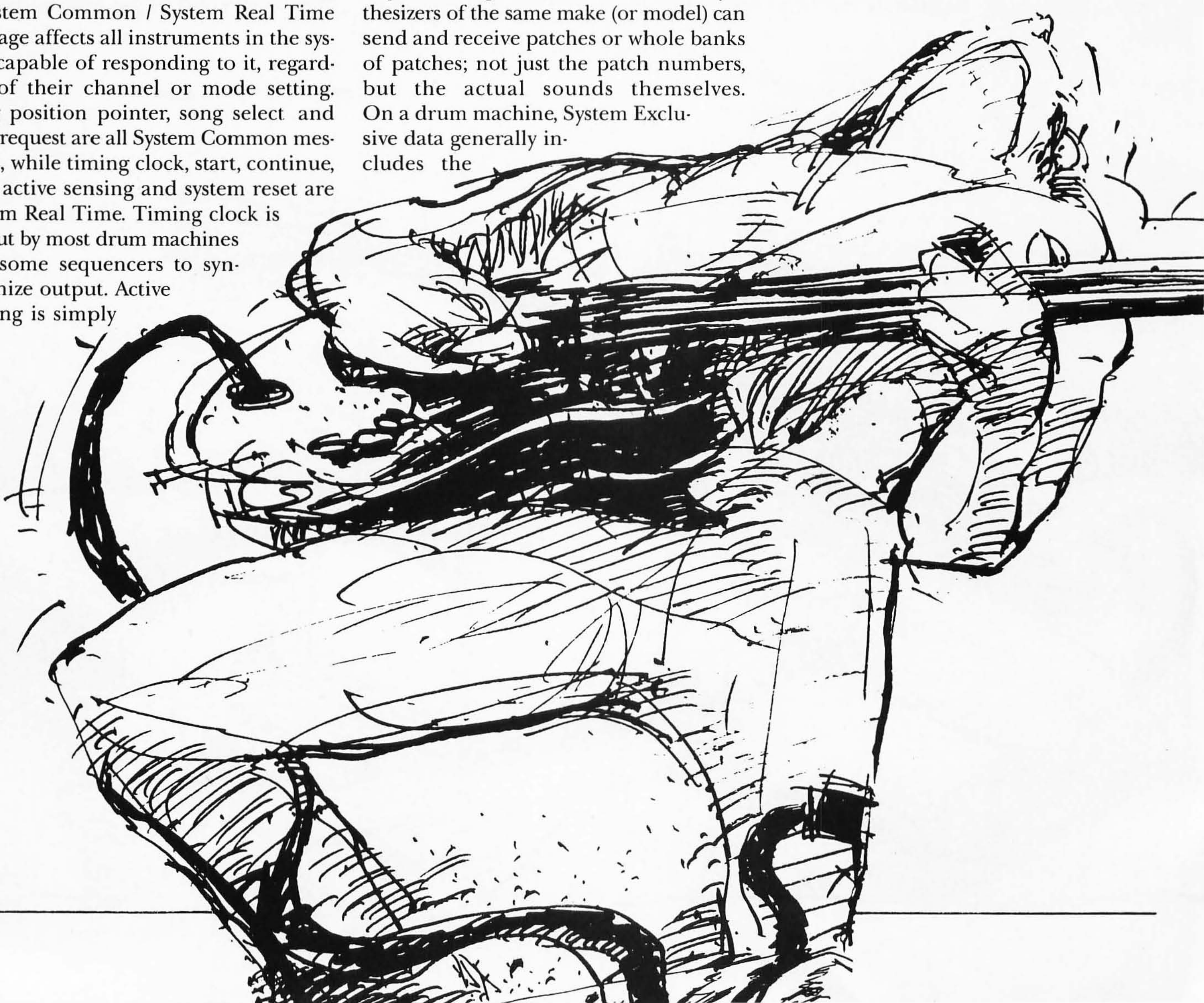


Illustration by Brent Watts

tween MIDI's 16 channels and the synthesizer's voice assignments. When Omni is on, for instance, the synth can receive messages over any channel, rather than just its basic channel (which can usually be set by the user). The way in which Poly and Mono interact with Omni gets a little complicated, and depends in part upon whether the synth is transmitting or receiving; we'll leave such details to another article. What you should know, however, is that these messages, along with Local Control on and off, are handled as control changes by MIDI; that is, they do not have separate status bytes of their own (see below), but instead are identified by control numbers 122 thru 127. MIDIMON identifies all control changes by their assigned controller numbers and specifies the values they are outputting, but identifies these special Channel Mode Select messages by their "proper" names as well.

The second category of MIDI information, System Common/Real Time, differs from Channel in one important respect. Whereas Channel messages affect only those instruments set to a particular channel (or to all channels when Omni is on), a System Common / System Real Time message affects all instruments in the system capable of responding to it, regardless of their channel or mode setting. Song position pointer, song select and tune request are all System Common messages, while timing clock, start, continue, stop, active sensing and system reset are System Real Time. Timing clock is output by most drum machines and some sequencers to synchronize output. Active sensing is simply a byte (hex FE) which is output every 300 milliseconds by some synthesizers when nothing else is happening. Over the years, active sensing has been found to be of limited utility, and has now been dropped from many instruments. MIDIMON filters out active sensing bytes by default, but allows you to turn the filter off.

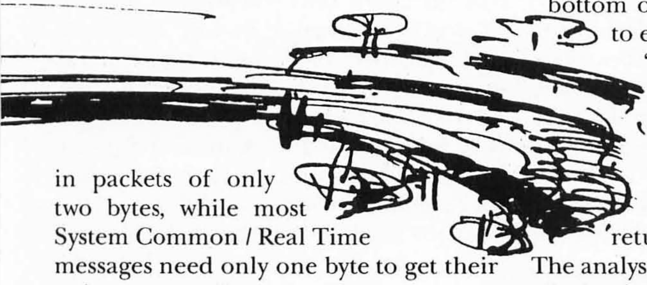
There are two other messages that can be properly thought of as System Common, but which really function only as "bookends" for the third category of MIDI information: System Exclusive. Although there is no standardization of System Exclusive data, it is always headed by an $F0 byte and tailed by an $F7 (End Of Exclusive) byte. In between can be any number of data bytes, but the first byte after $F0 should always contain the manufacturer's ID number. This ID number is what allows you to single out a particular instrument in a chain for System Exclusive operations ($F0 and $F7 being channel-independent messages). On a synthesizer, System Exclusive data usually includes the values the instrument assigns to its internal components in order to produce a given sound. This is how synthesizers of the same make (or model) can send and receive patches or whole banks of patches; not just the patch numbers, but the actual sounds themselves. On a drum machine, System Exclusive data generally includes the



coding by which it remembers patterns or songs. MIDIMON accepts System Exclusive data, and properly identifies it as such, but to analyze all the data in detail it is necessary to consult your particular instrument's System Exclusive specifications.

MIDI data formats

In its simplest form, MIDI data is output in packets of one, two or three bytes. It's most easily readable in hexadecimal form, each byte consisting of two hex digits. Any byte with a value greater than or equal to $80 (128 in decimal) is called a "status" byte. Status bytes identify the type of operation which the instrument is being called on to perform. For example, when a synthesizer receives the value $90 through its MIDI-In port, it recognizes this value as a note-on command for all synths set to receive information on channel #1 (the second nibble of the status byte in a Channel message is always the channel number minus one; this convention limits MIDI to 16 channels). However, it does not yet know which note to turn on or how loudly to play it. The following two bytes—known as data bytes—give it this information in values ranging from $00 to $7F (0 to 127). The second byte of the packet is the note number, the third its velocity (loudness). Program change and channel pressure messages are transmitted



in packets of only two bytes, while most System Common / Real Time messages need only one byte to get their points across. I'm not going to go into much detail on this subject; there's plenty of information available if you wish to delve further into the mechanics of MIDI, and you'll be able to learn almost everything you need to know just by observing how MIDIMON interprets MIDI output. The important thing to understand here is that the meaning of data bytes depends entirely upon the status byte which precedes them.

So far so simple, right? Well, not quite. Suppose you play a couple of notes into MIDIMON. When you look at the MIDIMON analysis screen, on the far left side where the raw data is listed you might see a series of 3-byte packets, but you might also see one 3-byte packet followed by a bunch of 2-byte packets. Don't let that throw you; it only means that your synthesizer outputs MIDI data in the "running status" format. This means that once a status byte is output, it need not be output again as long as the same kind of MIDI event follows. This is a faster, more efficient way of transmitting MIDI data, and seems to be gaining in popularity. Some instruments use a combination of standard / running status formats. In any case, all instruments should be able to correctly interpret both.

Using MIDIMON

Make sure your computer's MIDI-In is connected to the synthesizer's MIDI-Out, and vice-versa. Turn the synthesizer on first, then run MIDIMON off the desktop like you would any other program. After the title is displayed you'll be asked whether you want to filter out active sensing bytes or not. Unless you have some reason for wanting to see them, I recommend that you do filter them out. Now you're in capture mode; as soon as you play a note or fiddle with any MIDI-transmitting controller, it will show up on the capture screen. Although the capture screen clears after you fill it up each time, don't worry about losing the data; it's all being stored in a buffer as large as the free memory in your computer.

All the functions available from capture mode are listed on the menu line at the bottom of the screen. When you want to enter analysis mode, simply hit "A." An analysis will then be printed to the screen, and the menu line will show you which keys to press in order to page back and forth through the analyzed data, return to capture mode or quit. The analysis includes the raw data in hexadecimal form, packet by packet, a description of each status byte, channel number if appropriate, and whatever details are relevant. If you return to capture mode, the buffer will remain intact unless you formally clear it with the "C"lear Capture command.

Do you want to print the analysis to a printer or disk file? Just strike "P,"and you will be prompted as to which device to print to. Printouts to the printer can be aborted by striking "A" at any time.

In order to send a string of bytes to the synthesizer, press "S." You can then enter a series of up to 32 bytes (64 hex nibbles) in hexadecimal form. Any characters other than hex digits (except spaces) will not be accepted, and since it takes two hex digits to make a byte, odd numbers of hex digits will not be transmitted. Spaces are not transmitted, but are counted as characters in the send buffer.

Here's an example. To turn on middle "C" (note #60, decimal) with a velocity value of 64 (decimal), you would enter:

90 3C 40

To turn it off, you could use the note-on, zero-velocity method by sending:

90 3C 00

Alternately, you could send a formal note-off (a status byte of $80 on Channel 1) and experiment with a release velocity of, say, 32 (decimal):

80 3C 20

Or if you wanted to see how your synth reacts to "running status," you might try turning off the note with no immediate status byte at all:

3C 00

It's probably a good idea to first learn how your synthesizer handles MIDI by analyzing some of its output. Then you'll know the best way to communicate with it. In regards to sending your synthesizer System Exclusive commands, remember that you must preface all such messages with an $F0 byte followed by the manufacturer's System Exclusive ID number. If you initiate a patch dump from the synthesizer itself, MIDIMON will analyze this ID number for you. The model ID number is also necessary sometimes, as is an $F7 at the end of the command.

A note about MIDIMON's Echo feature. You can toggle this function on and off by pressing "E" in capture mode. When you press it the screen will clear, and the letter next to "E"cho on the menu line will switch from "N" to "Y" or back again. "N" means no echo; "Y" means, yes, the echo feature is on. While it is on, everything entering the computer's MIDI-In port is immediately echoed to its MIDI-Out port. This acts like a simple Thru box, and can be quite handy for checking synthesizer compatibilities while monitoring input. Remember, however, that if the synth you are playing is also receiving data from the computer's MIDI-Out port (along with the synth you intend to echo to), notes on the transmitting synth will be triggered twice, once from the keyboard and again from the computer. This can create strange, "out of phase" sounding effects. There is an easy way to remedy this problem, however. Reconfigure the keyboard to receive on a different channel than it's transmitting on.

Larry Herzberg is a freelance musician, writer and MIDI programmer who has been based in Los Angeles for the last ten years.

UTILITY

ALTERED STeights

by Matthew J.W. Ratcliff

Altered STeights is a graphics display utility for ST owners like me, who still have a particular affection for the old reliable 8-bit machines. Altered STeights will load and display Koala pictures, 8-bit graphics mode 9 and mode 8 screens as well. If you have Computer Eyes for your 800XL, why buy a new version for the ST? Just port your pictures over to big brother and load them up with Altered STeights (more on "porting" later). You can load pictures in quarter- (four Koala pictures on the ST display at once), half- or full-screen formats. It is more than a picture viewer utility however, since it allows you to save your composite creations in DEGAS uncompressed format. Note that due to the large size of the source code, this program (and all associated files) is found only on this month's disk version or on the Delphi ST SIG.

When you double click on ALTSTE.PRG, the program will run and a brief demonstration will be presented. You must be in low or high resolution to run this program. The demo gives you a quick preview of just what this utility is designed to do. Three demo files are searched for, DEMO with an extender for each 8-bit screen type, .PI9, .PI8 and .PIC respectively. If found each is loaded and displayed. In monochrome, the picture is mapped to the display in all seven possible display combinations. In color mode, Altered STeights uses VDI plot calls instead of direct screen-memory writing, so it is much slower. The low-resolution demo loads the .PI9 file and displays it in "split" screen format, on the left of the display. The .PI8 and .PIC demonstrations are shown on the right half of the display in "quad" format, top and bottom. The mode 9 pictures, digitized photos and such, look best in low resolution, while the mode 8 and Koala pictures look better in high. If you will be using this utility to create "clip art" files for Publishing Partner, for example, then it is best to work in high resolution if possible.

After the demo, the first dialog (or more accurately, form alert) is presented, giving you the option to continue or exit. You are next prompted to clear the workscreen before continuing. The first time through you'll want to answer YES, unless you would like to continue looking at the last demo screen while loading other pictures.

The next dialog prompts you for the picture type you're going to want to load: mode 9, mode 8 or Koala. The extenders for these pictures are expected to be .PI9, .PI8 and .PIC, respectively. Once you select the picture type, if you load a different type of picture, the display is likely to look a bit odd. If a Koala load is requested, the file will be uncompressed from the Micro Illustrator format if necessary. If it's a simple screen dump, or a "header error" occurs, it's mapped to the screen verbatim. (If it goofs up the display, you can always load another picture over it.)

Once you select the picture type, a file selector is brought up with a directory of the current drive, path and the proper extender expected. Double click on the file (or once and OK) you wish to load, or cancel if you want to do something else.

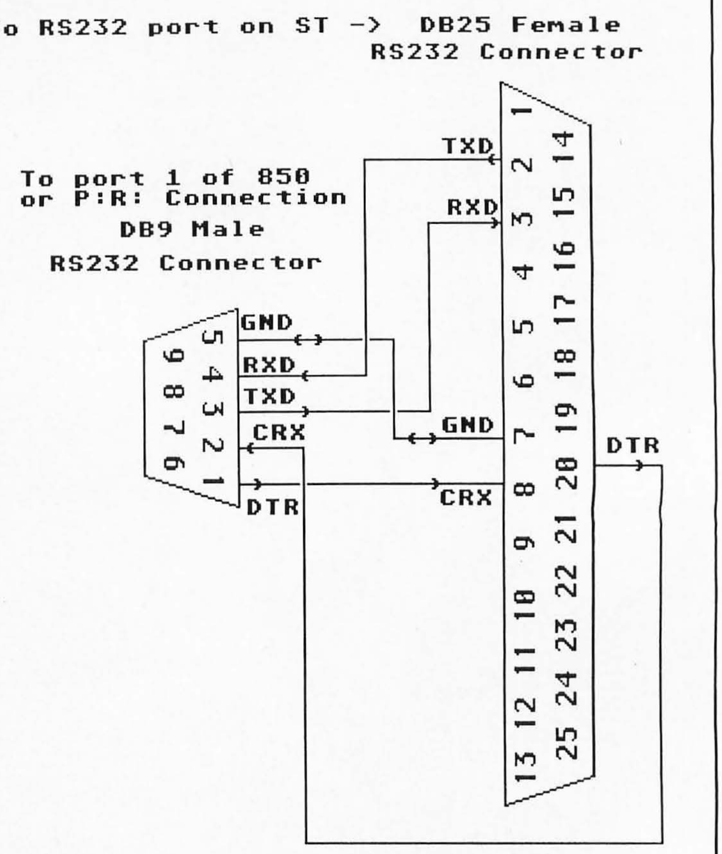
If you are in the full-screen format, the picture is mapped to the display immediately. If you had selected half screen, you will be prompted to point and click on the side of the screen to map the picture on. Similarly, if you're in the "quad" format, the display will be divided into quarters, so you can point and click on the proper quadrant.

After the picture is on the display, click the left mouse button to continue or the right button to "invert" the picture. Most mode 8 pictures, for example, will look fine in a black-on-white display format. This includes screen dumps from Solid States or Super 3D Plotter II. If the picture loaded was a mode 8 Computer Eyes picture it will look like a "negative." Clicking the right mouse button will correct for this.

Continuing after the load will bring up another alert box with the following prompts:

AGAIN—Load another picture of any of the three types supported in the same screen layout format.

Schematic for NULL MODEM cable connection between 8bit Atari and ST.



Connection Summary

8bit		ST
3 TXD	→	3 RXD
4 RXD	←	2 TXD
5 GND	↔	7 GND
2 CRX	←	20 DTR
1 DTR	→	8 CRX

Legend

DTR	- Data Terminal Ready
CRX	- Carrier Detect
TXD	- Transmit Data
RXD	- Receive Data
GND	- Ground

figure 1

SAVE—Bring up a file selector, with the extension of .PI3 or .PI1, to save the current display screen as a DEGAS uncompressed picture file.

RESTART—Restart the program, then exit or continue. Continuing allows you to clear the screen and select a new screen format.

You can create some interesting composite pictures this way. For example, you can load two pictures in quad-screen format on the left side of the display. Then restart and select half screen to load a picture on the right side of the display, without clearing the screen first. You can freely load and display mode 8, 9 and Koala pictures on the same display.

If the Koala pictures are in the Micro-Illustrator compressed format, Altered STeights looks at the color data in the file. It then generates a look-up table based on the luminance values for each of the foreground colors. The end result is proper shading in your Koala pictures. If the luminance values are the same, then a default look-up table (LUT) is generated. Sometimes your Koala pictures will need to be inverted because of this. You can always go to DEGAS and adjust colors of any picture you've saved in color mode.

The 8-bit GRAPHICS 9 pictures are made up of 80x192 pixel, 16-gray- level data. Altered STeights emulates this with an 8x2 pixel grid of varying dot densities. In the quarter-screen format you lose 50% of the original picture's horizontal *and* vertical resolution. To compensate, I tried averaging four pixels of the original data for every 8x2 block on the ST's display, but the end result was a little washed out and fuzzy. Altered STeights just tosses out what won't fit, and most pictures look pretty good anyway.

In the split-screen format, 50% of the horizontal resolution must be sacrificed. Several different shading patterns are used for the Koala pictures to retain 100% of its resolution in all screen formats. For the mode 8 pictures, no shading is required, and there's no loss of resolution—even in the quad-screen format. In low resolution you'll lose some image data in quad-screen formats for mode 8 and Koala pictures.

If you still have a complete 8-bit system, you may want to transport your graphic files from it to the ST. The easiest way to do this is with a "null modem" cable and terminal software at both ends. I use Flash on the ST and Keith Ledbetter's Express 3.0 at 2400 baud, both in ASCII mode, half duplex, 8 bits per character and one stop bit. Once cabled up, I set up the 8-bit to send Xmodem. Next the ST is set to Receive Xmodem and I start the transfer. The rest is automatic. If you have the 8-bit ARC utility, it'll save a lot of time if you compress all of your pictures into one big file and get the job done in one transfer. Then just de-ARC them on the ST.

Figure 1 is a schematic for the 8-bit to ST null modem cable. A 9-pin D type male connector is needed for your P:R: Connection or 850 interface. A 25-pin D type female (RS232) connector is needed for the ST. It isn't absolutely necessary to carry DTR through to CRX (this is just used by the computers to make sure the other is ON). You can simply short pin 1 to 2 on the 9-pin 8-bit connector and pin 20 to 8 on the 25-pin ST connector.

Of course, you can always download Koala pictures from Delphi or most any public Atari BBS. In the Atari SIG on Delphi we have a Koala database. You can also find many Koala and digitized pictures in the Micro Artists SIG.

The program was developed with Megamax C and the ever-patient assistance of Charles F. Johnson, Dan Moore, Tim and Jeff Randall of Randall's Home Computers and Clay Walnum's C-manship series. My thanks to them all!

STPlus•STPlus•STPlus•STPlus

P.O. 1197, Berkeley, Ca. 94701

We all want the ST to grow, so let's BUY MORE SOFTWARE so the ST grows!

BUSINESS

DBMan 4.0	175.00
Datamanager	56.00
Superbase	104.95
Trimbase	69.95
Phasar	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	56.00
EZ Calc by Royal	48.95
Analyze Spread	25.95
Final Word	99.95
PublishingPartner	140.00
T-works Publish It	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	35.00
Max Pack	35.00
Stuff	27.95
Flash 1.5	21.00
SBT accting ea.	275.00
Omni Res	27.95
Turbo ST(~blitter)	35.00
Dollars & Sense	69.95

GRAPHICS

Degas Elite	55.95
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
Athena 2	69.95

GAMES

Gunship	35.00
Shadowgate	35.00
Uninvited	35.00
Mouse Quest	14.00
Slaygon	27.95
Barbarian	27.95
Obliterator	27.95
Guantlet	35.00
Dark Castle	27.95
F-15 Strike Eagle	27.95
Star Trek-Rebel U.	27.95

MUSIC

Passport	
Master Tracks	280.00
MasterTracks Jr.	104.95
Midisoft Studio	69.95
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	139.95
Copy3-Postscript	299.95

How would you like to be an ST dealer? If you are interested, I am looking for a few limited partners to work with in areas which lack ST support. This is not a solicitation by Atari nor to circumvent Atari's network but an invitation to work with an established dealer to set up new dealerships. I am especially interested in college and business areas. Call (415)841-9183 to discuss

PROGRAMMING

GFA Basic	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	69.95p

GAMES

Tanglewood	27.95
Test Drive	35.00
Chessmastr2000	32.95
StarGliderbw&cl	32.95
Hunt for Red Oct	35.00
Police Quest	35.00
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

HARD DISKS

20 Meg Supra	558.00
20 Meg w/clock	599.95
30 Meg Supra	749.95
60 meg Supra	1249.95
33 Meg UnderMtr	699.95
51 Meg UnderMtr	839.95
60 Meg BMS RRL	1039.00
20 Meg SH205	639.95
250 Meg	3250.0

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

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 #107 ON READER SERVICE CARD.

GEMKIT

by Gordon Billingsley

GEMKIT is a set of ST, BASIC subroutines that gives you easy access to the power of the ST Graphics Environment Manager. GEMKIT uses simple, one-word GOSUB calls to manipulate graphics and text, create dialog boxes, use the mouse and perform many other functions with the lightning fast speed we all expected of the ST. All of the features work in any screen resolution.

Since GEMKIT is intended for use in all resolutions, some GEM features that work only for color monitors are not included. Also, because ST BASIC is, itself, a GEM application many GEM features that duplicate BASIC statements and functions are not included. Often BASIC statements already are making direct GEM system calls.

In other ways, GEMKIT expands on what GEM has to offer. For example:

1) GEMKIT uses an algorithm not available in GEM to automatically center lines of text in any of four type sizes ranging up to twice normal size;

2) GEMKIT contains an algorithm to automatically assign individual characters in a text string to an array allowing pixel-accurate placement of the whole string. (Without the algorithm you'd have have to put the text string on the screen essentially one letter at a time.)

Listing 1 contains not only GEMKIT itself (Lines 60000 on), but also a short program that demonstrates its features. To use GEMKIT in your own programs, delete Lines 10-410 (the demo program) in Listing 1.

Reading the mouse

GOSUB READMOUSE tells you where the mouse pointer is located and whether one of the mouse buttons has been pressed. The x- and y-coordinates (*mx, my*) are returned as pixel locations on the screen.

Mouse button presses may be detected in a variable called *button*, with values as follows: 0=no button pressed; 1=left button pressed; 2=right button pressed.

Here's a program example that calls the subroutine to read the mouse status and then checks for the x-coordinate location of the pointer and the status of the right button:

```
10 GOSUB READMOUSE
20 if mx>150 and button=2 then 40
30 goto 10
40 REM program continues here
```

Pointer shapes

Each of the following routines automatically changes the form of the mouse pointer to the shape the name of the subroutine implies:

GOSUB ARROW: the default pointer
GOSUB BEAM: an I-beam shape useful in word processing
GOSUB BEE: the famous busy bumblebee
GOSUB FINGER: a hand with a pointing finger
GOSUB HAND: the back of a hand or grabbing hand
GOSUB CROSSHAIR: a thin-line version of the crosshair
GOSUB FATHAIR: a thickened version of the crosshair
GOSUB HOLLOWHAIR: an outline version of the crosshair

Pointer visibility

GOSUB HIDE allows you to make the pointer invisible at any time you want to ensure it does not appear on the screen. GOSUB SHOW allows you to ensure that the pointer is visible at all times it is needed.

Text manipulation

Each of the following routines automatically changes the typeface as it appears on the screen to the form implied in the subroutine name. A change in type style remains in effect until explicitly changed again.

GOSUB NORMAL: system default typeface
GOSUB BOLD: a thickened version of the default face
GOSUB HATCH: a hatched look, sometimes called ghost letters
GOSUB ITALICS: a skewed face
GOSUB UNDERLINE: underlines the system default face
GOSUB OUTLINE: a hollow typeface

Calling a type-style subroutine does not affect type styles of characters already on the screen.

Adventurous programmers may use GOSUB NEWFACE to create faces that are combinations of the six standard faces. For instance, it is possible for a typeface to be both bold and italic at the same time. To do this you assign an appropriate value to a variable called *font*, then call GOSUB NEWFACE.

Each type style has a unique number value as below:

Normal = 0	Bold = 1
Hatch = 2	Italics = 4
Underline = 8	Outline = 16

To create a combined style, add the values for the styles you wish to have combined. Bold/italics would equal 5, for instance. This is the same method used to combine type styles in ST-Writer.

As an example, the command sequence to set the type style to bold/italic/underlined would be:

```
10 font = 13:GOSUB NEWFACE
```

Four type sizes are available in GEMKIT for the creation of showy title and menu screens: small, regular, large and extra large. A line of any size type may be placed on the screen with pixel accuracy by using the mnemonic GOSUB routines.

GOSUB STYPE: small type, ½ system-default type size
GOSUB RTYPE: regular type, system-default type size
GOSUB LTYPE: large type, 1.5 times system-default type size
GOSUB XLTYPE: extra large type, two times system-default size

Each of these subroutines must be preceded by program lines to indicate the desired text string and to set xy pixel coordinates for placing the first letter of the text. The text string is set in a variable called *text$*. The pixel locations are set with variables called *tx* and *ty*. For instance:

```
10 text$ = "Hello World!"
20 tx = 50:REM x-coordinate
30 ty = 150:REM y-coordinate
40 GOSUB XLTYPE
```

When using any of the type-size subroutines, each single line of text must have its own pixel-precise coordinates and separate GOSUB call. You cannot use combinations of GOTOXY and PRINT to place nonstandard character sizes on the screen because PRINT does not space nonstandard letter sizes correctly. As much as half of each of the letters may disappear because of overlapping when PRINT is used.

Of course, the standard or default type style, may be placed on the screen using PRINT statements, but it is not necessary in that instance to use GOSUB RTYPE. GOSUB RTYPE should be used only when you want pixel-accurate placement of the standard type size.

Text and graphics in ST BASIC are normally limited to the output window. With GEMKIT placement routines, however, the entire screen is available. You could, for instance, place scoring information in the upper right corner of the screen, outside the playing window of a game.

BASIC views position 0,0 as being the upper left corner of the output window. GEM views position 0,0 as the upper left corner of the screen. You should be careful about obliterating menu bar items and such unless you really want to.

A line of any size of text may be automatically centered in the BASIC output window. Simply assign the desired y-coordinate pixel position (vertical position) to the variable called *ty*, set the text string in the variable called *text$* and call the appropriate centering subroutine.

A subroutine is available for each of the four type sizes and each has been given a mnemonic name. You do not have to set the type size first. The centering routine automatically sets the type size and centers the line of text. It also adjusts the point at which it centers for any screen resolution.

If you desire screen-centered text you should change the values assigned to the variable called *rez*, which can be found in the TYPESIZE subroutine.

continued to page 58

TUTORIAL

Software Engineering: System Specification and Analysis

by Karl Wiegers

What's the first thing you think about when you decide to begin a new software project? I hope your answer is, "What's the intended output of this program?" The output of a program might be described very differently, depending on whether you're contemplating a weather-predicting program or a flashy arcade-style game. But all computer programs produce *some* kind of output, and it helps immeasurably during the programming if you know exactly what that output is.

In the olden days of computer programming, the issue of specifying precisely what a software system was to accomplish often was skirted, rather than being confronted head-on and beaten into submission. The results were predictable:Systems were delivered to end users far behind schedule, they cost a lot more than anyone expected, and all too often their performance bore only a vague resemblance to what the end user really had in mind. "Don't worry about the details now," the system analysts said. "This baby will be so flexible that we can change it later on when you decide what you really want." The analysts were wrong; their customers were unhappy.

Contemporary software development efforts rely more and more on systematic "software engineering" methodologies. A vital aspect of those methodologies is a structured, comprehensive approach to analyzing the problem that the software system is being constructed to solve, thereby defining the system's intended functions. This is a critical first step, to be taken before any code is written if you truly wish to create a high-quality system. In fact, good specifications are fundamental to *any* problem-solving venture, from building a box to keeping your firewood dry to landing a man on Mars.

"But," you say, "I successfully programmed for years on an Atari 8-bit computer without using any of these newfangled ideas." So did I. And when you're dealing with small memories and low-powered computers, you can get away with more casual development methods much of the time. But the new generation of microcomputers, such as the Atari STs and their big brothers of the Mega persuasion, can run programs vastly larger than could be handled by the 8-bits. These programs often are written by teams of programmers, and they might involve dozens of program modules all spliced together to create the final product. A systematic, structured approach can give tremendous improvements in productivity, quality and reliability for such systems.

Some software engineering gurus believe that about 40% of the effort put into a complex software system should be devoted to defining the system through structured analysis and design activities. Only 20% of the time should be spent on the implementation step of actually writing code, with the remaining 40% used to integrate the pieces of the system and test for proper behavior. In this article we'll begin a discussion of the software engineering process by examining the principles and tools of structured analysis and system specification.

Where do I begin?

Amazingly enough, at the beginning. Many programmers gloss over the beginning and dive right into a source code editor. Only later do they realize how much easier life would be if they had a plan, a design. I look at it this way: If I don't know what I'm trying to accomplish, how will I know when I'm done?

Admittedly, a thorough, structured analysis isn't something most hobbyists want to spend their time on. They (including me) want to write code. And it's true that the problem of system specification is less serious when only one person (you) is working on the project. But believe me, if you read the rest of this article and try to apply some of the ideas to your next project, I think you'll recognize the benefits. Software engineering methodologies can do for complete systems what structured programming methods do for individual program modules. The long-term payoff comes during the maintenance phase of a software project, when you're trying to add additional features or eradicate bugs. A high-resolution roadmap of your system makes navigation a whole lot easier.

The goal of system analysis is to come up with a requirements document which will serve as the guiding beacon for the rest of the development effort. In the olden days, these often belonged in the Victorian novel category: huge, unwieldy and incomprehensible. Nobody ever read the whole thing. In the 1970s, though, new methods were invented that use graphics for much of the requirements document. The focus is on communication, and we've all learned that visual communication through graphics can be much more effective than reading volumes of text. So, modern requirements documents contain lots of pictures. How do you draw a picture of a software system? We'll see shortly.

Of course, it's unrealistic to expect that the system requirements will never change once they're written down. Most projects go through many iterations of refinement and enhancement. You probably won't be able to permanently freeze the requirements, but the more you know about the needs before you begin writing code, the better off you are. The idea, then, is to construct the system such that it can readily accommodate changes without collapsing under its own weight. This is the goal of structured analysis (in the early phases), structured design (in the middle stage), and structured programming (in the later phases).

To explore some of the ideas and techniques of structured analysis, let's use a real example, an educational chemistry game called *Reaction Time* that I wrote once upon a time on an 8-bit Atari in assembly language. In Reaction Time, a player moves chemical formulas and numbers from specific areas on the screen into an empty line representing a chemical equation. The idea is to build valid chemical reactions by properly combining four elements or compounds from a specified list of formulas. Each reaction set contains 15 different formulas, and between nine and 16 valid equations can be constructed in each set. The program has seven different reaction sets, representing different aspects of basic chemistry. The player gets ten points for each correct equation and six points for each equation in which the formulas are correct but the coefficients are not; he loses five points for each incorrect equation.

See? You just read the entire statement of purpose for Reaction Time. And you didn't even have to know anything about computers (and not much about chemistry). Notice that this description of the system didn't say anything at all about the programming language that would be used, the kind of hardware the program would run on, or any other details. We're beginning with a fairly abstract statement of what the system will do.

Now we must refine this idea into successive levels of detail, such that we not only gain a thorough understanding of what we wish to accomplish, but also get enough information so that we can eventually write a program. Along the way, I like to create a written, numbered list of all the functions the system must perform. Then, as I design the system, I'm less likely to overlook something inadvertently.

Model making

With the statement of purpose firmly in mind, we want to begin building "models" of our software system. These models will graphically represent the system at various levels of detail. An important goal of building these models is to break the system down into logical pieces that fit together so that there aren't any gaps when we assemble the final product. Come to think of it, this isn't all that different from building a plastic model of an F-15 jet fighter. Obviously, we don't use molded plastic parts and glue for constructing models of software systems; we'll get to the model components in just a bit. First let's think about the different kinds of models we might want to create.

In the olden days of computer programming, the issue of specifying precisely what a software system was to accomplish often was skirted, rather than being confronted head-on and beaten into submission.

During the analysis phase, you begin by building a "logical" model of the system. This is a conceptual depiction of what the software system is supposed to do, showing its connections to the outside world (that is, everything in the universe that isn't a part of your system, yet which interacts with it in some fashion). The logical model is independent of any implementation details, showing just *what* happens and not *how* it happens.

After creating a clear picture of your system functions on a logical basis, you build a "physical" model of the system. (I don't mean a physical model in the sense of ice cream sticks and playing cards, but rather a graphical model of the ultimate physical system you intend to create.) This physical model will include implementation details such as hardware- and software-specific features. An example might be a program that will use GEM on the Atari ST for the user interface. The same program written for an IBM PC might have a rather different physical model, because of the different user environments. However, these two physical models would be based on the same logical model, since the system functions would be the same in both cases.

Actually, you may well build more than one physical model. After all, there's almost always more than one way to solve a particular problem. It's highly advisable to consider all the approaches you can think of early on, and then choose the one that makes the most sense. Also, this makes you feel much less foolish than if you dive right in and then discover a far better approach after you're neck-deep in design flaws.

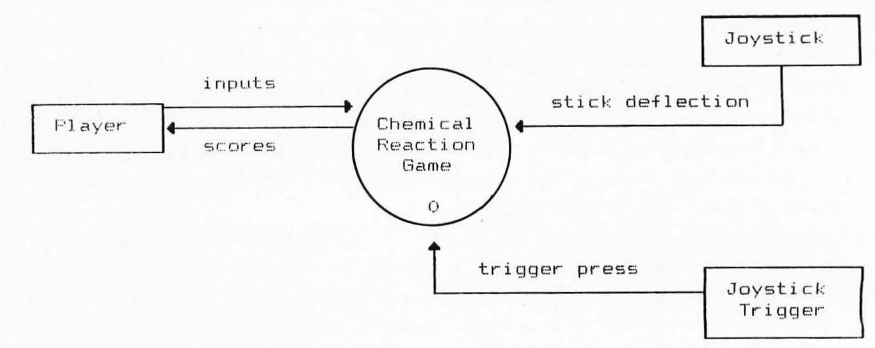
Sometimes (although not usually when writing for microcomputers) the system being developed represents an automation of an existing system. For example, converting a process that is currently performed manually into one in which a computer performs all or part of the tasks of the process fits into this category. In cases like this, the proper starting point is to build a physical model of the current, manual process. This helps to identify all the functions and components of the current system, in terms of the documents, people, and equipment actually involved in the execution of the process.

From that specific, physical model you can derive a logical model of the current system. The logical model is basically the physical model with the implementation details removed. Who needs to know that Ralph sorts the mail in room 145? All you need to know on a logical basis is that the mail gets sorted. Then, the logical model of the *current* system can be used as the starting point to generate the logical model of the *new*, automated system. We won't worry anymore about this aspect of system specification, since I suspect you won't be starting with an existing manual system of any kind when you sit down to write the ST arcade game of the year.

Context diagram

Okay, let's get started with the logical model for Reaction Time. The modeling tool used to represent our system at its most abstract level is called the "context diagram." Figure 1 shows the context diagram for Reaction Time. Our system is represented by the circle labeled "Chemical Reaction Game." (Remember, this is a logical model, so I won't even give it an official name here.) The objects in rectangular boxes are external to our system, yet they have some communication with the system. These are the player, a joystick and a joystick trigger.

Figure 1. Context Diagram for Reaction Time



The lines with arrowheads indicate the communication between the external objects and the system itself; they represent data flowing from one object to another. In this case the player is supplying something called "inputs" to the system, and the system gives him back something called "scores." The joystick's contribution is labeled "stick deflection," and the joystick trigger supplies a "trigger press."

The context diagram is a special case of a very important modeling tool called a "data flow diagram" or DFD. There are really only four kinds of objects in a DFD: processes (shown as circles, or sometimes as rounded rectangles; sometimes called a "bubble"); externals (shown as rectangles; sometimes called "terminators"); data flows (shown as lines with arrowheads); and data stores (but not on the context diagram; stay tuned). All of these objects must be labeled, as they are in Figure 1. On the context diagram only one process is shown, which represents the entire system.

The context diagram often is labeled with a zero, as in Figure 1. This indicates that, on another piece of paper, we are going to peer inside this one bubble and see what it contains in more detail. The magnified view of object "Chemical Reaction Game" will be labeled the zero-level diagram, and it will be another DFD containing the four kinds of objects mentioned in the previous paragraph.

I can hear you now. You're saying, "I'm never going to bother drawing context diagrams and data flow diagrams, because I don't need to go through this analysis jazz for *my* programs. I've got it all in my head." Well, that may be. However, read on, because these same diagramming tools are also used in the structured design phase, and I'll guarantee you that you don't have the entire design for a program of more than 100 lines in your head. DFDs really do help you understand both what you want your system to accomplish (during analysis) and how to go about accomplishing it (during design).

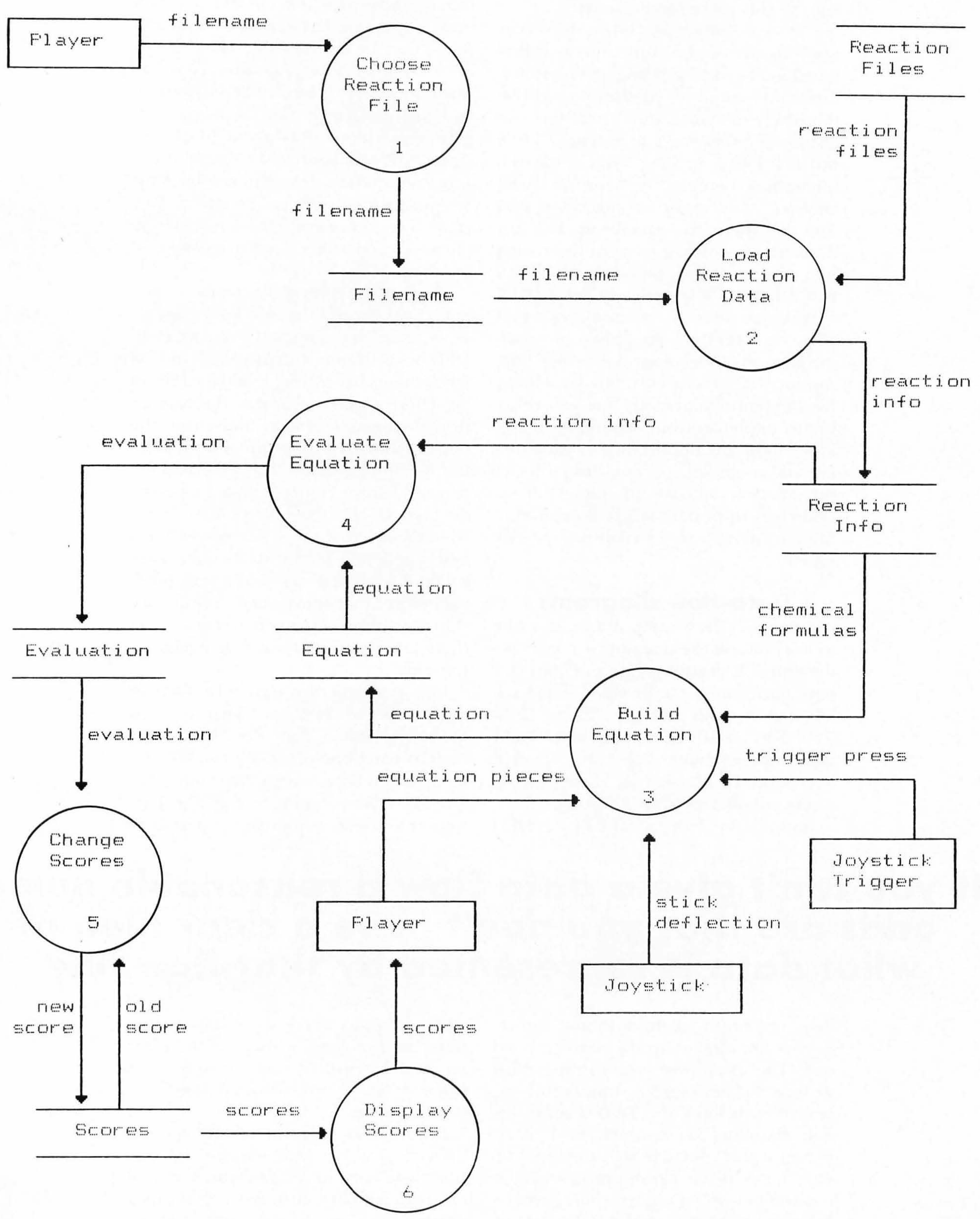
Okay, I admit it, this context diagram has a little bit of physical model flavor. After all, I did mention physical "things" like a joystick and a joystick trigger, yet I didn't say anything about a mouse. It's sometimes difficult to completely separate the *what* of the system from the *how*. My own experience has been that, for smallish software projects, only one model is really necessary at the analysis step, and that is the physical model. In the case of Reaction Time, there wouldn't be an enormous difference between the logical and physical models of the new system. And, since I'm not really converting an existing system, there's no need to create physical or logical models of a current system. Your choice of which models to build during system analysis should be dictated not by dogma, but by the scope, nature, and complexity of your application.

Partitioning

The next step of analysis is to start breaking down your system into smaller pieces, by turning your microscope on to the context diagram. Begin by drawing the zero-level data flow diagram; the one for Reaction Time is shown in Figure 2.

Notice that Figure 2 contains several processes, numbered with integers. The fundamental definition of a process is something that converts inputs into outputs—nothing fancier than that. Basically, this is what all computer programs do, so we must be on the right track. Every function performed by the Reaction Time system must be represented by a

Figure 2. Zero-Level Diagram for Reaction Time



process on the zero-level diagram. After all, the zero-level DFD is just an expansion of the single bubble on the context diagram, which contained the entire system, so this makes some sense.

The step of breaking the entire system into separate logical functions is called "partitioning," and it is mighty important. A certain amount of partitioning will be intuitively obvious. Some functions will just jump out at you and exclaim, "I'm a process, I'm a process!" But often the separation between one process and another won't really be clear. You may find that there are several ways you can think about splitting parts of the system into separate logical processes. Don't feel bad if it takes a few tries to get it right.

We'll go through partitioning again when we enter the design phase, in which the physical model of the new system (one product of structured analysis) will be used to figure out exactly how we are going to implement our system. We'll talk more about the importance of partitioning and modular program design in future articles, but trust me when I say yet again how important it is to do a careful, thoughtful job of partitioning your system.

Data-flow diagrams

Since they are so important, let's take a closer look at the anatomy of a data flow diagram. Please turn again to Figure 2. I have partitioned the Reaction Time system into six main processes: Choose Reaction File, Load Reaction Data, Build Equation, Evaluate Equation, Change Scores and Display Scores. These processes are numbered from 1 through 6.

Important rule No. 1 for DFDs: A DFD does not imply anything about the sequence in which processes are carried out. (There's an exception to this, which we'll get to later on.) The numbering system in each level of a DFD is arbitrary. You shouldn't be tempted to look at Figure 2 and conclude that process 1 always takes place before process 2. Of course, some of your partitioning will be based on just such sequential thinking, but the general rule is to not attempt to infer sequence information from a DFD. This is a very important distinction between the data flow diagram and a diagramming tool with which you may be more familiar, the flowchart.

Important rule No. 2 for DFDs: All externals shown on the context diagram must appear on the zero-level diagram. Recall that the external objects are shown in rectangular boxes; Reaction Time has the three externals labeled Player, Joystick and Joystick Trigger. The zero-level diagram shows more detail about exactly *how* these externals connect to the different processes in the system. You see that Player appears in two places in Figure 2. This is just a matter of graphical convenience, so the data flow lines don't cross over one another.

Think of the context diagram as a "parent" diagram, and the zero-level diagram as its "child." It's important that each child DFD be consistent with its parent DFD as far as things like externals and data stores go. Those objects represent the connections between a process and either the outside world, or other processes within the system. For example, if we expanded process 1 from Figure 2 into yet a more detailed DFD, it had better have input from the external called Player and it had better generate output to the data store called Filename. You could invent other data stores that functioned entirely within the subprocesses of process 1, but these two key objects absolutely must be present.

Important rule No. 3 for DFDs: Bubbles don't talk to bubbles. That is, you shouldn't have a data flow line going directly from one process to another. Instead, it should go into a data store, with a second flow coming out of the data store into the second process. Some DFD advocates don't worry about this convention, but I've come to appreciate it, for reasons that will become more obvious when we talk about structured design. Patience, please.

If you can't give a data flow a reasonable name, odds are that you don't have a clear idea of what data is represented by that flow line.

Just what is a data store? Nothing more than a logical grouping of some information that's used in your system. In a logical model, a data store just represents a block of data. In a physical model of the current system, a data store could represent a paper form that's filled in by one person in your organization and handed to another person. In a physical model of the new system, a data store might be a disk file. At this stage of our analysis, the nature of each data store isn't as important as its contents. And we'll wait to think about actual file formats until much later.

The data stores in Figure 2 have labels like Filename, Reaction Files, Reaction Info, Equation, Evaluation and Scores. Notice that many of the data flows have exactly these same names. This should come as no surprise. After all, it is data that is flowing from one process to another, and data stores are simply collections of data. However, sometimes a process needs only part of the data in a particular store.

For example, in Figure 2 there's a data flow named Chemical Formulas coming out of a data store named Reaction Info. How is this possible? Well, we can conclude that the thing called Chemical Formulas must be just one part of the thing called Reaction Info. On the other hand, process 4 wants *everything* out of Reaction Info, so the flow has the same name as the store. Make sense? In our next software engineering installment, we'll talk about exactly how to define all the objects in a system model.

Important rule No. 4 for DFDs: Label all data flows. If you can't give a data flow a reasonable name, odds are pretty good that you don't have a clear idea of what data is being represented by that flow line. Repartition your system until all the flows can be logically named. And things like "Data" and "Information" do not in themselves constitute reasonable names, so don't think you can be sloppy at this stage.

Some DFD conventions allow you to leave a data flow unlabeled if and only if the flow is connected to a data store. In such cases, the flow label is assumed to be the same as the name of the store. Of course, if the flow is supposed to represent only a portion of the contents of the store, as we just saw, you'd better give it a separate name. In my DFDs, any unlabeled flow will indeed indicate that the flow name is the same as the name of the data store to which it is connected.

We'll leave data flow diagrams for now, but they'll be back. There are some other important rules for DFDs we still need to talk about.

Data Dictionary

Even in this tiny model of this little system, we have already introduced no less than six processes, three externals, six data stores and 18 data stores (if my count is correct). How in the world are we going to keep track of all these names and what they mean? And how are we going to keep track of things like the fact that the data flow Chemical Formulas is a part of the data store Reaction Info? We need another tool, called a "Data Dictionary." The name is pretty self-explanatory.

But, we don't have time to talk about data dictionaries today. Please tune in next time, when we'll continue our discussion of structured analysis and system specification. Once you learn about data dictionaries, your life will be transformed. Well, maybe that's a slight exaggeration, but it's certainly true that using these analysis and design tools will speed your metamorphosis from a casual programmer to a genuine software engineer.

Bibliography

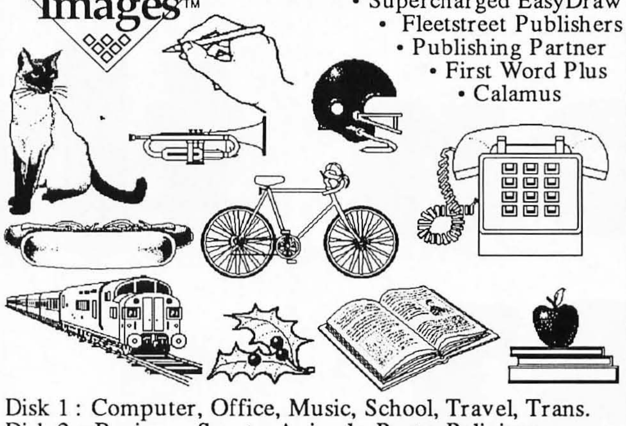
The main reference for structured analysis and system specification is a book called (guess what) *Structured Analysis and System Specification,* by Tom DeMarco. If you think you'll use any of these methods, you definitely want to have this book!

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.*

Magnetic Images™

CLIP ART !

Quality Images for use with:
- Supercharged EasyDraw
- Fleetstreet Publishers
- Publishing Partner
- First Word Plus
- Calamus



Disk 1 : Computer, Office, Music, School, Travel, Trans.
Disk 2 : Business, Sports, Animals, Party, Religious
Disk 3 : Food, Borders, Medicine, Old West, Newsletter
Disk 4 : Hands, Seasons, Pirates, Tools, Personal, America
Disk 5 : Theater, Corners, Zoo, Menu, Outdoor
Disk 6 : Ad man's Special: Computer Products

$19.95 per disk (Add $2.50 P & H per order)

Magnetic Images Co.
P.O. Box 17422, Phoenix, AZ 85011
(602) 265-7849
Dealer inquiries welcome.

CIRCLE #106 ON READER SERVICE CARD.

Vertical Blank Interrupts From C on the ST

by Eugene R. Gobby

The primary purpose of this article is to show how to install a C routine or function as an interrupt driver. The interrupt that we will use is our old friend, the vertical blank interrupt (VBI).

In the course of doing this, I will show how to link to an external module (developed in assembly language) from C and how to get into the supervisor mode of the 68000.

Many of you are aware that C has pointers which can be used to point to the memory locations of its variables; well, C also has pointers to functions. If we put the value of this pointer into an interrupt vector, then our function will be executed every time that the interrupt is called.

The ST VBI system has what it calls a queue, where users can put their own interrupt routines. In color systems this happens exactly 60 times a second (50 Hz in Europe); in monochrome, 70 times a second (I'm not sure if this is exact).

Although there are BIOS or XBIOS calls for installing interrupts in many of the ST's various systems, there is apparently none for the VBI. In this case, we must put the 68000 into the supervisor mode so that we can access the memory locations involved. Since initially I had some trouble getting the call (*Super()*) provided for doing this to work, I wrote and linked my own trap handler to do it. I have since had success with Super( ) but I have left the *Trap1()* routine in this program for educational purposes.

The Trap 1 handler (Listing 2) is modeled on the *Hitchhiker's Guide* trap 14 handler; however, this one works with C. The two changes needed are: 1) declare the *_trap1* label as a globl (notice that the assembler leaves out the "a"), and 2) declare the storage allocaton by .ds.L 1, not .ds.1. The global declaration allows us to call the "function" from C by the label name. The storage must be allocated as a long because we will be saving the address of the user stack.

In the C program (Listing 1), we first declare the function, *trap1( )*, then we call it. We pass the arguments *0x20* (i.e., 20 hex) and *0L*. We thus called a 68000 trap #1 exception. The trap handler then calls the operating system routine indicated by the code *0x20*. This routine puts the 68000 into supervisor mode.

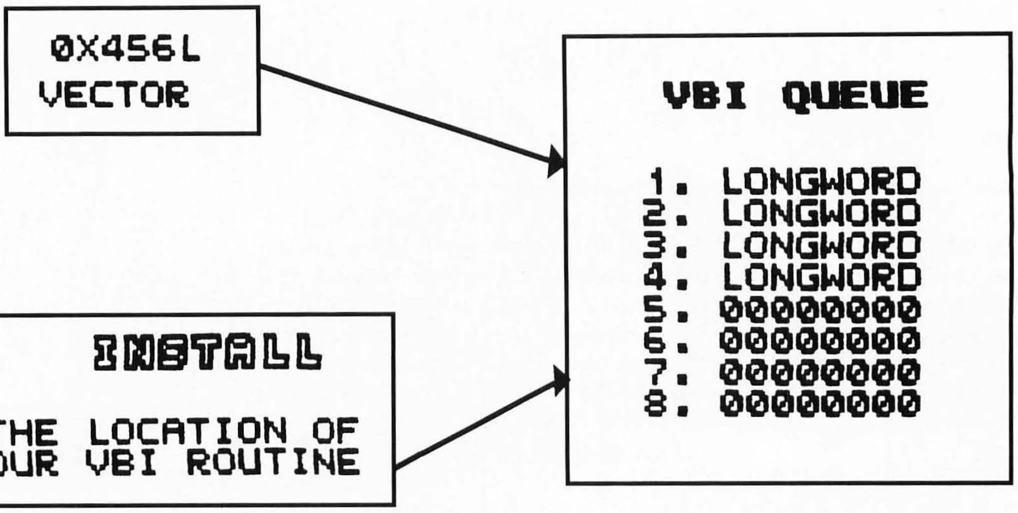
The supervisor stack pointer is loaded with the current value of the user stack pointer when the second argument is *0L*; *L* meaning a longword. (I use uppercase L because the lowercase is often confused with the numeral 1.) The trap handler then passes the old, supervisor stack-pointer value back to us in *save_ssp*. When we return to user mode, we will pass *save_ssp* in place of *0L*, so as to return to where we left off. While in supervisor mode, you should be careful of using the BIOS or XBIOS routines because many of them put the 68000 into the supervisor mode during execution. Thus, calling these routines often hangs the system. (Exactly why is not clear to me, since many of the normal VBI functions must be executed from the supervisor mode. Perhaps the VBI is more careful in checking to see if the 68000 is already in that mode or not.)

Since it is declared as a pointer, the compiler sees to it that it's incremented as a pointer and not as, say, an integer when we increment it.

Installing the VBI

To install our VBI, we must find the queue and then find an empty space in

Figure 1



that queue—indicated by a longword of all 0s (see Figure 1). The queue has only enough space for eight interrupt vectors, so conservative programming demands that we test for the number of vectors in the table.

The function *vbiset()* is a general routine for installing VBIs in the ST. It will install our routine on the bottom of the queue if we enter with an ASCII passed to the variable process. If we pass an "n" instead, it will disengage our routine by writing a 0L to the queue. Note that if you have put other routines under the one being disengaged, they must be moved up.

The interrupt routine *vbiroutine()*, must be declared in *vbiset()*. The pointer to *vbiroutine()* is then simply *vbiroutine*. I declared *vbiroutine* to be an integer because I am using integers in the routine but, since I am not actually returning any values out of *vbiroutine()*, I suppose that it could just as easily have have been declared as returning a char or a long. However, it must be declared as something.

Two other pointers are used in *vbiset()*: *vbiqueue* and *vbiempty*. *Vbiqueue* is set equal to 0x456L. This is the memory location of the pointer to the queue. Thus *vbiempty = vbiqueue*, makes *vbiempty* equal to the first vector in the table of interrupt vectors. Since it is declared as a pointer, the compiler sees to it that it's incremented as a pointer and not as, say, an integer when we increment it.

Note that since the 68000, unlike the 6502, changes the whole address in one operation, we never have to worry that another interrupt will occur while we are changing the address. Thus there is no need to turn off interrupt processing during this instruction. In fact, when I set the TOS variable *vbisem* (hex 452) to 0 and tried to install a routine, it hung the system. *Vbisem* is used as a flag by the TOS during the VBI process itself, and it should probably not be altered by the user.

The ST VBI system has what it calls a queue, where users can put their own interrupt routines.

Our little VBI routine simply increments points at the rate of 60 (or 70) times a second. The *while* loop in *main()*

just prints out the value of points until it has reached 600 (when ten seconds have passed). Then the vbi queue is returned to normal and we end the program. Since I wanted to increment points on successive interrupt calls, I declared it to be an external (equivalent to a global) variable. Thus, it remains in existence between calls to *vbiroutine()* and is also accessible to our print loop in *main()*.

While in supervisor mode, you should be careful of using the BIOS or XBIOS routines because many of them put the 68000 into the supervisor mode during execution.

The *vbiset()* routine can be made more general by putting the supervisor entry and exit within it and by eliminating the calls to *printf()* and *getchar()*.

Typing it in

Now I'll explain some of the mechanics of writing the program. (I used Atari's Developer's Kit.) First type Listing 2 into the editor and save it. Then find the utilities disk and call up COMMAND.PRG. The assembler is on the compiler disk. If you have a single drive system, you will have to copy the "trap1( )" source code to it. Then type "as68 – L trap1.s trap1.o." Single drivers: transfer the object file to the linker disk. Type in the new linker batch file (Listing 3). This may or may not be the approved order of linking but it works.

I had a lot of trouble with the compiler when my programs got above a certain size. Since I was compiling to a second double-sided drive with an almost empty disk, I was going crazy. It turns out that the assembler writes an intermediate file to its own disk, regardless of what drive your source code is on. The batch programs in Listing 4 and 5 will force the intermediate file to be written to either a floppy B (type C2 filename in the batch dialog widow) or drive C (C3 filename), a hard disk or a ramdisk. When I C3 with a ramdisk, the assembler just flies.

A final note: I find that when the compiler or linker fails, my source (.C or O) file often gets wiped out, so I always keep a backup of each file. By the way, pressing a key during the various operations (there are about four for the compiler), will call the abort query, allowing you to save some time. For instance, if the preprocessor has found some errors, you can exit without going through the assembly operation.

VBI's From C — Listing 1
C

```
 /*vbiart.c v.2 */
/* copyright 1988 by ST-Log */

#include "stdio.h"
#include "define.h"
#include "osbind.h"
#include "gemdefs.h"
#include "vdibind.h"
#include "obdefs.h"
#include "easyc.h"

int points;

main()
BEGIN
 extern int points;
 char process,c;
 long save_ssp;
 long trap1();

 save_ssp = trap1(0x20,0L); /* enter supervisor mode */


printf("press the s key to begin ");
   WHILE ((c=getchar()) NE 's')
   ENDWHILE

      points = 0;
      process = 'i';            /* set up vbi */
       vbiset(process);
   WHILE(points LE 600)
       printf("points= %d \n",points);
   ENDWHILE

/* restore normal vbi */
   process = 'n';
   vbiset(process);


trap1(0x20,save_ssp); /* EXIT SUPERVISOR MODE */


   printf("press any key to exit \n");
     getchar();
   END


 vbiset(process)
  char process;  /* an 'i' to install the vb routine, */
                /* an 'n' to return to normal */
       BEGIN
        int vbiroutine();
        int i;
        long  *vbiqueue ,*vbiempty;
        int *sshiftmod, *timer_ms;
        int x =0;
        char *px;

       /* addresses must be long */
     vbiqueue=0x456L;
     vbiempty = *vbiqueue;
        printf("process= %c  \n", process);

             WHILE (x<=8 AND (*vbiempty NE 0x0L))
              vbiempty = vbiempty + 1;
              printf("vbiempty= %Lx, *vbiempty = %Lx \n", vbiempty,*vbiempty);
               x=x+1;
              ENDWHILE
         printf("press to change vector \n");

            IF(process EQ 'i') THEN
              printf("installing  vector, vbiroutine= %Lx \n",vbiroutine);
              printf("go\n");
              getchar();
              *vbiempty = vbiroutine;    /* install vector */
            ELSEIF ( (*(vbiempty-1) EQ vbiroutine) AND (process EQ 'n'))
              printf("disengage interrupt\n");
              getchar();
             *(vbiempty-1) = 0x0L;
            ENDIF
END

vbiroutine()  /* TOGGLE OUTPUT, INCREMENT POINTS */



 BEGIN
extern int points;

      points = points +1; /* just increases points by 1 every vbi */

END
```

VBI's From C — Listing 2
Assembly

```
      .globl _trap1
_trap1:
       move.l (sp)+,tr1ret ; pop ret addr
       trap #1             ; do BIOS function
       move.l tr1ret,-(sp) ; return to
       rts                 ; caller
       .bss
tr1ret:   .ds.l    1        *   saved ret. addr
```

VBI's From C — Listing 3
Batch File

```
link68 [u] %1.68k=gemstart,%1,vdibind,aesbind,osbind,gemlib,libf,trap1
relmod %1
rm %1.68k
wait
```

VBI's From C — Listing 4
Batch File

```
cp68 %1.c %1.i
c068 %1.i %1.1 %1.2 %1.3 -f
rm %1.i
c168 %1.1 %1.2 %1.s
rm %1.1
rm %1.2
as68 -F b: -l -u  -p %1.s %1.o
rm %1.s
wait.prg
```

VBI's From C — Listing 5
Batch File

```
cp68 %1.c %1.i
c068 %1.i %1.1 %1.2 %1.3 -f
rm %1.i
c168 %1.1 %1.2 %1.s
rm %1.1
rm %1.2
as68 -F c: -l -u  %1.s %1.o
rm %1.s
wait.prg
```

VBI's From C — Listing 6
Header File

```
          /* EASYC.H */

/* LOGIC OPERATORS */

#define AND &&  /* logical and */
#define  OR ||  /* logical or */
#define NOT  !  /* logica not */
#define  EQ ==  /* equ val comp */
#define  NE !=  /* not equal val comp */
#define  LE <=  /* less than or equal to */
#define  GE >=  /* greater than or equal to */

/* bitwise operators */

#define BAND  &  /* bitwise AND */
#define  BOR  |  /* bitwise OR */
#define BXOR  ^  /* bit exclusive OR */
#define BNOT  ~  /* bitwise NOT */
#define LSHF <<  /* left shift */
#define RSHF >>  /* right shift */


/* arithmetic ops */

#define INC ++  /* increment */
#define DEC --  /* decrement */
#define MOD  %  /* modulo devision */


/* if_then_esleif_else */

#define IF(e)         { if(e)
#define THEN             {
#define ELSE          } else {
#define ELSEIF(e)        }  else if(e) {
#define ENDIF            ;}}

/* CASE */

#define CASE(e)          { switch(e) {
#define CASEOF(e)       case e: {
#define DEFCASE          default: {
#define ENDCOF        } break;
#define ENDCASE          }}

/* WHILE */

#define WHILE(e)      { while(e) {
#define ENDWHILE      ;}}

/* FOR */

#define FOR(e)        { for(e) {
#define ENDFOR        ;}}

/* BEGIN */

#define BEGIN         {
#define END           }
```

Illustration by Brent Watts

MICROCOMPOSITIONS

by Michael Gogins

The term "computer music" evokes images of a PC running a sequencer program and controlling a bank of synthesizers—layering tracks, instantaneously transposing and correcting, and finally printing out a meticulous transcription of the human intuition. Yet the first computer musicians were interested in much more than simply automating the tedium of traditional arranging and copying. They wanted to hear sounds which had never been heard before, and to apply algorithmic and mathematical techniques to the actual process of composition. For these pioneers, a musical score did not have to be black squiggles on a page—it could as easily be a program for generating music.

Today, due perhaps to its academic heritage and association with the avant-garde, compositional programming is not as familiar to musicians and programmers as sequencers and transcription software. Yet on the Atari ST, with its built-in MIDI interface, some quite interesting compositions can be written even in that resource everyone has—ST BASIC!

To illustrate three simple but powerful techniques of compositional programming, I have written an ST BASIC program, *MICROS* (see Listing 1). MICROS contains three sub-programs, each of which consists of one or two pages of code and generates an entire piece of music from either a short sequence of notes or a few numerical constants:

OC1	Overlap canon	Lines 2000-2490
KC1	Koch curve	Lines 3000-3370
CD1	Complex dynamical system	Lines 6000-6230

These programs do not store sequences. All they do is play a synthesizer in real-time by computing numbers and sending them to the MIDI-Out port. (If you want to record a sequence, you can, of course, plug the MIDI-Out port of your ST into the MIDI-In port of another sequencer.) You can load MICROS.BAS into your ST, hook up your Casio CZ, and immediately run these compositions to hear some very different music. The program menu is self-explanatory. You can stop the music at any time by pressing the S key.

Hierarchical structures

KC1 and OC1 use multiple real-time "processes" to build up complex compositions. This is made possible by using a master timing loop which constantly reads the ST system timer. Within the loop is an event timing list of conditional branches which can execute several independent processes at the same time, based on note-on and note-off times generated by the processes themselves. Each process, when called, computes and plays only its next note before returning to the timing loop. (This is the closest one can come in BASIC to multitasking!) Therefore, a second process can compute its notes based on the notes returned by the first process, a third process can compute its notes based on the notes returned by the second process, and so on. In this way hierarchical or recursive structures, such as musical fractals, can be constructed.

Hardware and software requirements

Despite its faults and the fact that it is an interpreted language, ST BASIC is fast enough for some simple real-time compositional programming. (You can always use a BASIC compiler such as GFA BASIC if you want to speed things up, attain more precision, or do modular programming.)

These programs are written to play the Casio CZ-101 synthesizer. The ability of this common, inexpensive machine to play up to four different timbres at the same time makes it suitable for realizing, all by itself, fairly sophisticated compositions. Therefore each program contains mode and program change messages for accessing favorite patches I have stored in the Casio's internal memory. Experiment with the program change messages to find the best sounds on your machine.

If you do not have a CZ, you will have to change the syntax to control your own

On the Atari ST, with its built-in MIDI interface, some interesting compositions can be written even in ST BASIC!

synthesizer. Or, if you have no synthesizer at all, you can still get three notes at a time out of the sound chip in the ST. Replace the note-on and off statements in Lines 380-480 with SOUND statements, and replace the various program change statements with WAVE statements.

KC1—a musical fractal

This program (Lines 3000-3370 in Listing 1) was inspired by VARIATN.BAS, a program written by Curtis Bahn to generate musical fractals for the Yamaha CX5-M, an MSX computer with a built-in synthesizer. (See Charles Dodge and Curthis R. Bahn, "Musical Fractals," *Byte*, June 1986, p. 185.)

If you want to understand fractals in depth, read Benoit Mandelbrot's book, *The Fractal Geometry of Nature* (W.H. Freeman and Company, 1983). Basically, a fractal is a curve which has wiggles or spikes on top of wiggles or spikes, *ad infinitum*, so that the curve actually fills up a measurable fraction of its region of the plane (hence the term "fractional dimension"). (A circle, square, or other ordinary curve, being only one point wide, fills only an infinitesimally small part of the plane.) For example, to construct a Koch or snowflake curve, take an equilateral triangle, stick three smaller equilateral triangles on its sides, stick three yet smaller triangles on the sides of those triangles, and so on to infinity.

A musical Koch curve is constructed by taking a simple melody as a generator. On top of each note in the generator is then played a tiny copy of the generator, which makes a second and faster layer of music. On top of each note in the second layer is then played a yet tinier copy of the generator, which makes a third and yet faster layer of the "snowflake," and so on. KC1 has four such layers.

KC1 is called by the master timing loop in Lines 160-240 of MICROS. Lines 3110-3160 of KC1 are an event timing list for the four layers of the musical snowflake. Lines 3010-3100 of KC1 are an initialization section, which switches itself off after the first call from the timing loop. (For a detailed understanding of the MIDI statements which program the CZ, consult the *MIDI 1.0 Specification* of the International MIDI User's Group, or see John Jainschigg, "Sound Chip Part 3: A Nuts and Bolts Guide to MIDI," *Atari Explorer*, Summer 1987, p. 78.) KCNOTES is the number of notes in the "generator" melody, KCLEN is the total duration of the piece of music in 200ths of a second, and KCSTART is the MIDI note number upon which the snowflake curve is built. The data in Line 3080 specifies the generator in terms of alternating pitch movements and durations as fractions of the total duration (i.e., add three half steps to the starting point and play the resulting note for .2 of the total duration, add seven half steps to the starting point and play the result for .2 of the total, and so on).

Experiment with different durations, pitches and numbers of notes in the generator to create new compositions. Even a small change to the generator makes a big change in the final music!

On each call from the master timing loop, control passes to the event timing list in Lines 3110-3160. On the first pass, all note on times are 0, and each process is executed immediately in turn.

The first process, in Lines 3170-3210, increments a counter which steps through the generator one note at a time. The appropriate number of half-steps are added to the starting point to obtain NOTE1, which is turned on by calling a subroutine. NOTE1's duration is then computed by multiplying its fractional duration by the total duration, and the next note on time is computed by adding NOTE1's duration to the current note-on time. Control then returns to the event timing list.

The second process works exactly the same way as the first, except that NOTE1 is used for the starting point, and the duration of NOTE1 is used instead of the total duration of the piece, so that a copy of the generator is played (as NOTE2) on top of each note of the generator. The third process is similarly built up (NOTE3) on top of the second, and the fourth (NOTE4) on top of the third. Control then returns to the master timing loop. On each subsequent pass through the timing loop, each process will be called only at its appropriate time, and will compute only its next note.

Experiment with different durations, pitches, and numbers of notes in the generator in Lines 3050-3080 to create new compositions. Even a small change to the generator makes a big change in the final music! Just make sure all the fractional durations add up exactly to 1.

OC1—an overlap canon

A canon, or round, is simply a melody which is played against itself after a certain delay. Like the musical snowflake, a canon can have several voices. OC1, however, is an overlap or differential canon. (See Lines 2000-2490 in Listing 1.) Each voice of the canon plays one less note of the generator than the last voice. Therefore, as the layers repeat, the amounts of overlap or delay are constantly shifted, and the piece as a whole does not repeat for a very long time—more than a day, in fact, if you let it play that long!

The initialization section and event timing list are similar to KC1, except that OCLEN specifies the fraction of each note duration for which the note will actually be on. This is the degree of *staccato* or *legato* to be played. The processes, however, are simpler; each process simply steps through the generator melody one note at a time, plays its note, and computes its next note on time. The only difference between the processes is the number of notes to be read from the generator before repeating.

Experiment with different melodies and rhythms, and especially with different synthesizer patches. It's interesting to use the same timbre for at least two of the voices, so that the shifting overlaps will interlace to generate constantly changing melodies. You may want to choose a generator whose notes do not all clash with each other, because at one time or another each note will be played against every other note.

Experiment with different melodies, rhythms and synthesizer patches. It's interesting to use the same timbre for at least two of the voices, so that the shifting overlaps with interlace to generate changing melodies.

CD1—a complex dynamical system

CD1 (Lines 6000-6230 in Listing 1) takes a totally different approach to music. The entire composition is generated by a single, very simple equation in Lines 6160-6170, and the music is completely determined by a choice of four numbers.

The mathematics is based upon arithmetic in the complex plane, where each point is represented by two numbers, a real (or X) component and an imaginary (or Y) component. For musical purposes, each point in the plane can also be thought of as representing two notes, one on an X keyboard and one on a Y keyboard. Complex arithmetic specifies the rules for adding, subtracting, multipying, and dividing points.

The equation Z< = Z ∧ 2 – M means: take the point Z, square it and subtract from it a constant point, M. Then make Z equal to the result, and repeat the procedure. As the equation is iterated, Z hops about on the plane—playing, according to our musical interpretation, two-voice counterpoint. This is a complex dynamical system. And the behavior of Z depends very sensitively on the value of M. For some values of M, Z hops very quickly off to infinity; for other values of M, Z spirals down into some stable point and stays there; for yet other values, Z whirls about until it settles into a hopping orbit with two, three or however many points.

If each point on the plane is taken as M and colored black if Z=0 never goes to infinity, or some other color according to how fast Z=0 does go to infinity, a map of the Mandelbrot set will be produced. The Mandelbrot set is one of the most complex and fascinating objects in all mathematics, because no matter how much it is enlarged, it reveals more and more detail, never exactly repeating itself as most fractals do. I used the set to choose an interesting value of M for CD1. Ms from the black region, the inside of the set, but close to the edge, produce relatively complicated orbits of Z. For more information about the Mandelbrot set and other fractals, with awesome color pictures, see H.O. Pietgen and P.H. Richter, *The Beauty of Fractals: Images of Complex Dynamical Systems* (Berlin, Springer-Verlag, 1987). You may be able to find public-domain programs for generating maps of the Mandelbrot set on the ST.

Experiment with CD1 by changing CDMR and CMI, the value of M, and/or CDZR and CDZI, the initial value of Z, which does not need to be 0. To guide your explorations, you may want to use one of the public-domain ST programs for mapping the Mandelbrot set.

Michael Gogins is 37 years old, has a B.A. in comparative religions and is supporting himself by word processing and dBASE programming while he tries to get a career going writing science fiction. He plays the flute, writes music and will further pursue compositional programming.

Micro Compositions — Listing 1
ST Basic

```
10 MICROCOMPOSITIONS:' Michael Gogins
/ June 30, 1987
20 '
30 DIM OCM(100,2)' Overlap canon score
 array
40 '
50 OPENW 2:FULLW 2:CLEARW 2:COLOR 1,1,
2,0,0' Menu background
60 LINEF 50,20,548,20:LINEF 548,20,548
,130
70 LINEF 548,130,50,130:LINEF 50,130,5
0,20
80 COLOR 3,3,3,4,2:FILL 0,0
90 '
100 '
110 COLOR 1,0,0,0,0' Menu header and f
ooter
120 GOTOXY 11,3:PRINT "MICROCOMPOSITIO
NS / Michael Gogins / June 30, 1987"
125 GOTOXY 22,5:PRINT "PRESS K>EY TO S
ELECT:"
130 GOSUB MIMENU
140 GOTOXY 22,12:PRINT "Q>uit"
150 '
160 POKE 1210#,0' MASTER TIMING LOOP -
 Reset system timer
170 WHILE MIKEY<>4351' While "Q" not p
ressed
180 MIKEY=PEEK(&HFFFC02)' Peek keyboar
d ACIA register directly for speed
190 IF MIKEY>255 THEN GOSUB MIMENU' Do
 menu if key pressed
200 TIME#=PEEK(1210#)' Read system tim
er
210 ON MIPLAY GOSUB OC1,KC1,CD1' Maste
r event timing list
240 WEND
250 '
260 GOSUB MIOFF:END
270 '
280 MIMENU:GOSUB MIOFF' Menu with sele
ction highlighted
290 IF MIKEY<>6399 THEN COLOR 1 ELSE M
IPLAY=1:OCINIT=0:COLOR 2
300 GOTOXY 22,7:PRINT "O>verlap canon"
310 IF MIKEY<>9727 THEN COLOR 1 ELSE M
IPLAY=2:KCINIT=0:COLOR 2
320 GOTOXY 22,8:PRINT "K>och curve"
330 IF MIKEY<>12031 THEN COLOR 1 ELSE
MIPLAY=3:CDINIT=0:COLOR 2
340 GOTOXY 22,9:PRINT "C>omplex dynami
cal system"
350 IF MIKEY<>8191 THEN COLOR 1 ELSE M
IPLAY=0:COLOR 2
360 GOTOXY 22,10:PRINT "S>top playing"
:RETURN
370 '
380 MIOFF:GOSUB OFF1:GOSUB OFF2:GOSUB
OFF3:GOSUB OFF4:RETURN
390 '
400 REM Note on and off routines for c
hannels 1-4
410 ON1:OUT 3,144:OUT 3,NOTE1:OUT 3,SG
N(NOTE1)*64:RETURN
420 OFF1:OUT 3,144:OUT 3,NOTE1:OUT 3,0
:RETURN
430 ON2:OUT 3,145:OUT 3,NOTE2:OUT 3,SG
N(NOTE2)*64:RETURN
440 OFF2:OUT 3,145:OUT 3,NOTE2:OUT 3,0
:RETURN
450 ON3:OUT 3,146:OUT 3,NOTE3:OUT 3,SG
N(NOTE3)*64:RETURN
460 OFF3:OUT 3,146:OUT 3,NOTE3:OUT 3,0
:RETURN
470 ON4:OUT 3,147:OUT 3,NOTE4:OUT 3,SG
N(NOTE4)*64:RETURN
480 OFF4:OUT 3,147:OUT 3,NOTE4:OUT 3,0
:RETURN
490 '
2000 OC1:' Overlap canon
2010 IF OCINIT=1 THEN GOTO OCON ELSE O
CINIT=1' Initialization
2020 OUT 3,176:OUT 3,126:OUT 3,4' Mono
 mode
2030 OUT 3,192:OUT 3, 37:OUT 3,193:OUT
 3,35' Internal programs
2040 OUT 3,194:OUT 3, 38:OUT 3,195:OUT
 3,35' Next:  time parameters
2050 OCNOTES=18:OCBEAT=50:OCLEN=.75:OC
REPS=10:OCREP=0
2060 OCI1=0:OCI2=0:OCI3=0:OCI4=0
2070 DATA 38,4,41,4,57,2,62,1,65,1,58,
2,67,2,66,4
2080 DATA 39,4,78,4,77,1,69,2,77,1,74,
3,72,3,42,4,44,4,46,4
2090 RESTORE 2070:FOR OCI=1 TO OCNOTES
2100 READ OCM(OCI,1):READ OCM(OCI,2)
2110 OCM(OCI,2)=OCM(OCI,2)*OCBEAT:NEXT
2120 OCONT1#=TIME#:OCOFFT1#=TIME#+OCBE
AT
2130 OCONT2#=TIME#:OCOFFT2#=TIME#+OCBE
AT
2140 OCONT3#=TIME#:OCOFFT3#=TIME#+OCBE
AT
2150 OCONT4#=TIME#:OCOFFT4#=TIME#+OCBE
AT
2160 OCON:' Event timing list
2170 IF TIME#>=OCONT1# THEN GOSUB OCON
1
2180 IF TIME#>=OCOFFT1# THEN GOSUB OFF
1
2190 IF TIME#>=OCONT2# THEN GOSUB OCON
2
2200 IF TIME#>=OCOFFT2# THEN GOSUB OFF
2
2210 IF TIME#>=OCONT3# THEN GOSUB OCON
3
2220 IF TIME#>=OCOFFT3# THEN GOSUB OFF
3
2230 IF TIME#>=OCONT4# THEN GOSUB OCON
4
2240 IF TIME#>=OCOFFT4# THEN GOSUB OFF
4
2250 RETURN' Next are overlapping cano
ns
2260 OCON1:' Turn on Voice 1
2270 OCI1=1+OCI1 MOD OCNOTES:IF OCI1=1
 THEN OCREP=OCREP+1
2280 IF OCREP>OCREPS THEN GOSUB MIOFF:
SHPLAY=0:GOSUB MIMENU:RETURN
2290 NOTE1=OCM(OCI1,1):GOSUB ON1
2300 OCOFFT1#=OCONT1#+OCM(OCI1,2)*OCLE
N
2310 OCONT1#=OCONT1#+OCM(OCI1,2):RETUR
N
2320 OCON2:' Turn on Voice 2
2330 IF OCREP>OCREPS THEN GOSUB MIOFF:
MIPLAY=0:GOSUB MIMENU:RETURN
2340 OCI2=1+OCI2 MOD (OCNOTES-1)
2350 NOTE2=OCM(OCI2,1):GOSUB ON2
2360 OCOFFT2#=OCONT2#+OCM(OCI2,2)*OCLE
N
2370 OCONT2#=OCONT2#+OCM(OCI2,2):RETUR
N
2380 OCON3:' Turn on Voice 3
2390 IF OCREP>OCREPS THEN GOSUB MIOFF:
MIPLAY=0:GOSUB MIMENU:RETURN
2400 OCI3=1+OCI3 MOD (OCNOTES-2)
```

```
2410 NOTE3=OCM(OCI3,1):GOSUB ON3
2420 OCOFFT3#=OCONT3#+OCM(OCI3,2)*OCLE
N
2430 OCONT3#=OCONT3#+OCM(OCI3,2):RETUR
N
2440 OCON4:' Turn on Voice 4
2450 IF OCREP>OCREPS THEN GOSUB MIOFF:
MIPLAY=0:GOSUB MIMENU:RETURN
2460 OCI4=1+OCI4 MOD (OCNOTES-3)
2470 NOTE4=OCM(OCI4,1):GOSUB ON4
2480 OCOFFT4#=OCONT4#+OCM(OCI4,2)*OCLE
N
2490 OCONT4#=OCONT4#+OCM(OCI4,2):RETUR
N
2500 '
3000 KC1:' Koch curve
3010 IF KCINIT=1 THEN GOTO KCON ELSE K
CINIT=1
3020 OUT 3,176:OUT 3,126:OUT 3,4' Mono
 mode
3030 OUT 3,192:OUT 3, 37:OUT 3,193:OUT
 3,38' Programs
3040 OUT 3,194:OUT 3, 41:OUT 3,195:OUT
 3,35
3050 KCNOTES=5:KCLEN=33000' Time param
eters
3060 KCI1=0:KCI2=0:KCI3=0:KCI3=0:KCI4=
0
3070 RESTORE 3080:KCSTART=40' Melody d
ata:  times must sum to 1
3080 DATA 3,.2,7,.2,0,.2,12,.2,14,.2
3090 FOR I=1 TO KCNOTES:READ KCGEN(I,1
):READ KCGEN(I,2):NEXT
3100 KCOT1#=TIME#:KCOT2#=TIME#:KCOT3#=
TIME#:KCOT4#=TIME#
3110 KCON:' Event timing list
3120 IF TIME#>=KCOT1# THEN GOSUB KCO1
3130 IF TIME#>=KCOT2# THEN GOSUB KCO2
3140 IF TIME#>=KCOT3# THEN GOSUB KCO3
3150 IF TIME#>=KCOT4# THEN GOSUB KCO4
3160 RETURN
3170 KCO1:' Layer 1 (slowest)
3180 GOSUB OFF1:KCI1=KCI1+1:IF KCI1>KC
NOTES THEN GOSUB MIMENU:RETURN
3190 NOTE1=KCSTART+KCGEN(KCI1,1):GOSUB
 ON1
3200 KCBEAT1=KCGEN(KCI1,2)*KCLEN
3210 KCOT1#=KCOT1#+KCBEAT1:RETURN
3220 KCO2:' Layer 2 (faster than 1)
3230 GOSUB OFF2:KCI2=1+KCI2 MOD KCNOTE
S:IF KCI1>KCNOTES THEN RETURN
3240 NOTE2=NOTE1+KCGEN(KCI2,1):GOSUB O
N2
3250 KCBEAT2=KCBEAT1*KCGEN(KCI2,2)
3260 KCOT2#=KCOT2#+KCBEAT2:RETURN
3270 KCO3: 'Layer 3 (faster than 2)
3280 GOSUB OFF3:KCI3=1+KCI3 MOD KCNOTE
S:IF KCI1>KCNOTES THEN RETURN
3290 NOTE3=NOTE2+KCGEN(KCI3,1):GOSUB O
N3
3300 KCBEAT3=KCBEAT2*KCGEN(KCI3,2)
3310 KCOT3#=KCOT3#+KCBEAT3:RETURN
3320 KCO4: 'Layer 4 (fastest)
3330 GOSUB OFF4:KCI4=1+KCI4 MOD KCNOTE
S:IF KCI1>KCNOTES THEN RETURN
3340 NOTE4=NOTE3+KCGEN(KCI4,1):GOSUB O
N4
3350 KCBEAT4=KCBEAT3*KCGEN(KCI4,2)
3360 KCOT4#=KCOT4#+KCBEAT4:RETURN
3370 '
6000 CD1:' Complex dynamical system
6010 IF CDINIT=1 THEN GOTO CDON ELSE C
DINIT=1' Initialization
6020 OUT 3, 176:OUT 3,126:OUT 3,4' mon
o mode
6030 OUT 3, 192:OUT 3,41:OUT 3,193:OUT
 3,41' Internal program
6040 CDMR=.636434:CDMI=.38785:CDZR=0:C
DZI=0' Pitch parameters
6050 CDBEAT=52:CDLEN=.10:CDREP=0:CDREP
S=450' Time parameters
6060 CDWR=1/2+1/2*SQR(1+4*SQR(CDMR^2+C
DMI^2))' Whirlpool radius
6070 CDONT1#=TIME#:CDOFFT1#=TIME#+CDBE
AT
6080 CDON:' Event timing list
6090 IF TIME#>=CDONT1# THEN GOSUB CDON
1
6100 IF TIME#>=CDOFFT1# THEN GOSUB OFF
1:GOSUB OFF2
6110 RETURN
6120 CDON1:' Two-dimensional (complex-
valued) dynamical system
6130 GOSUB OFF1:GOSUB OFF2
6140 CDREP=CDREP+1:IF CDREP>CDREPS THE
N GOSUB MIMENU:RETURN
6150 CDZR2=CDZR
6160 CDZR=CDZR^2-CDZI^2-CDMR' Compute
Z<=Z^2-MU
6170 CDZI=CDZR2*CDZI*2-CDMI
6180 NOTE1=36+(CDZR+CDWR)/CDWR*30' Nor
malize whirlpool
6190 NOTE2=36+(CDZI+CDWR)/CDWR*30' rad
ius to MIDI keys 36-96
6200 GOSUB ON1:GOSUB ON2
6210 CDOFFT1#=CDONT1#+CDBEAT*CDLEN' Ti
mes
6220 CDONT1#=CDONT1#+CDBEAT:RETURN
6230 '
```

Micro Compositions — ST Checksums

```
 10 data  848, 354, 731, 358, 503, 614
, 584, 114, 368, 471, 4945
 110 data  182, 263, 697, 119, 926, 48
6, 292, 572, 571, 521, 4629
 200 data  940, 303, 58, 488, 496, 494
, 411, 168, 505, 123, 3986
 320 data  861, 912, 825, 875, 560, 49
6, 149, 502, 308, 189, 5677
 420 data  652, 204, 663, 219, 674, 23
4, 685, 504, 844, 939, 5618
 2020 data  124, 706, 17, 874, 858, 91
5, 957, 925, 230, 813, 6419
 2120 data  223, 230, 237, 244, 889, 8
76, 732, 880, 730, 877, 5918
 2220 data  735, 881, 740, 476, 543, 1
76, 993, 961, 834, 907, 7246
 2320 data  546, 987, 904, 967, 845, 9
21, 556, 993, 908, 973, 8600
 2420 data  849, 928, 559, 992, 919, 9
86, 860, 942, 538, 421, 7994
 3010 data  691, 126, 398, 910, 524, 6
48, 34, 524, 615, 77, 4547
 3110 data  882, 609, 615, 621, 627, 4
50, 570, 976, 434, 13, 5797
 3210 data  171, 223, 660, 15, 202, 18
4, 219, 670, 27, 206, 2577
 3310 data  190, 505, 673, 32, 217, 20
3, 541, 465, 881, 188, 3895
 6030 data  454, 650, 606, 949, 153, 8
86, 827, 267, 451, 295, 5538
 6130 data  323, 973, 893, 315, 451, 9
79, 358, 33, 280, 394, 4999
 6230 data  540, 540
```

A Broad Spectrum

An interview with the team behind Spectrum 512

by Andy Eddy
and
Maurice Molyneaux

A lot was going on at the Northeast Atari Computer Fair, held at the Worcester Centrum, in Worcester, Massachusetts, October 10th and 11th, 1987. Yet, even with attention grabbers like Mega ST4s, laser printers, WordPerfect and others, one particular program seemed to catch almost everyone's eye: *Spectrum 512*, a graphics program which allows the ST to display its full 512 colors on screen simultaneously. Also at the show was *Digispec*, software that links to the *ComputerEyes* video digitizer that makes it possible to digitize pictures into *Spectrum* format. On the first day of the show, Andy Eddy and Maurice Molyneaux interviewed the people behind Trio Engineering, the developers of Spectrum and Digispec. The primary force behind the Trio efforts is Boris Tsikanovsky, who spent his childhood on a Russian island near Japan and moved to the United States in 1979.

▼ANALOG: When did you first start using computers. . .what's your background with them?
TRIO: I don't know. . .for a long time (chuckles).

ANALOG: Did you have any contact with computers in Russia or not?
TRIO: Uh, yes. All different kinds of computers, but, uh. . . .

ANALOG: Mainframes?
TRIO: I guess so. I don't even know what it was. . .it was kind of big (laughing).

ANALOG: You started on mainframes in Russia, and you moved over here in '79. What was your progression? What computers have you had since then?
TRIO: One computer I used to have was the ZX-81. You know?

ANALOG: The Sinclair?
TRIO: The Sinclair. I guess we should say that we've been interested in the 68000 computers, and we've been looking for a niche. You know, to do something in this area. And the Atari seemed to us to be the perfect computer to do something. So we've been looking at a variety of different things...we have concentrated on Atari, and I guess it worked out very well. So far.

ANALOG: It appears so.
TRIO: So that's the main thing right now. Let me add something, too, talking about general engineering experience with computers. Being in the United States, we worked with the IBM mainframes; we worked with the Prime computers, particularly 850s and 9950s and the newer machines...the bigger machines. We worked with the VAXes, DEC computers. We're talking VAXes...750 and up. The new VAXes.

ANALOG: So you have a lot of experience.
TRIO: We have lots of experience with inter-computer communications. That's the *company* profile, not talking work on one particular project. It could go on and on and on, list after list.

ANALOG: As a company, is this really your first delve into the home-based computer or personal-computer market?
TRIO: Yeah, I would say it's not just the first personal computer because we do know Macs and IBMs and IBM PCs, but we never did any home- or entertainment-market software.

ANALOG: And this is your first touch into the ST market?
TRIO: Absolutely.

ANALOG: Also, your first graphics program as well?
TRIO: If you're talking about Spectrum, yeah. Spectrum was the first, and the second product right away is the Digispec, which gives you access to Spectrum [through ComputerEyes].

ANALOG: This kind of thing has never been done with an ST before; what you've done is revolutionary. You're the first ones to pull off something like that. And again, we see Eidersoft with their *Quantum Paintbox*...it doesn't seem to compare.
TRIO: [Quantum's interface] is so inconvenient. It's absolutely inconvenient, I would say. You can't just paint. To me, when you work with painting programs, the only result you want to see is nice pictures on the screen. The Quantum approach may be a good approach—but I haven't seen the results yet. Until you see those results, it's hard to say what is what. It's not because I'm trying to be negative, but I want to understand what's the limitation of that ST hardware.

What this does is put into the hands of people with very inexpensive computers the power to do the kind of graphics work that normally costs a lot more.

ANALOG: Well, how did you come upon your technology?
TRIO: Just lucky, I guess (laughing).

ANALOG: Just lucky?
TRIO: I would say it took a lot of patience. I see the problem with the other guys doing it; they start from the technical point, with interrupts and all of that. But then it remains on a technical level. You just can't have [the software] where you just paint.

ANALOG: What you're saying is that you want to make it simple, for the program not to get in the way of what the artist is trying to do. It should be transparent.
TRIO: Yes.

ANALOG: Where did the idea for this particular program come from? Did you look at a picture on a screen and decide this could be better? Atari has set down what they say are the specifications for the machine. I heard that Leonard Tramiel saw Spectrum and said, "Well, you can't put 48 colors on a scan line...." and sat down and placed 48 colors with the zoom mode, and walked out. But they know the specifications of the machine and said it was impossible.
TRIO: Well, we just didn't pay any attention to that. We worked from the 68000 side. We saw what it can do—how much time it takes to change so and so (laughing). Maybe if we'd asked him, and he said, "No," maybe then we would have gotten discouraged.

ANALOG: It would have changed your mind?
TRIO: Yeah. I want to add that to be a programmer [alone] is not enough to do this. One has to know physics: the way things work, the way things are displayed, to be able to measure the times of a par-

ticular operation, things like that. So you have to know a lot of the technical details from the physics side of it—not just the programmer's side of it—and incorporate it and integrate it. That's the only way to do it.

ANALOG: Sort of like measuring a room before you move the furniture in?
TRIO: Yes, that's the way it was. That's why the hardware is working at its limit now.

ANALOG: I did hear that you actually oscilloscoped how the machine performed and worked from the inside out rather than tried to program something.
TRIO: Right.

ANALOG: Can you go into detail at all about what you did? Your preparation for actually doing the programming?
TRIO: Well, that was mainly related to the [system] timings...measuring the timings. That's all. Look at it as simply a machine that works in real-time and [does] things. You must understand the machine. Programming comes later.

ANALOG: Were you looking at specific sorts of applications, more than just a paint program?
TRIO: It was kind of curious, because we were thinking first about doing some animation stuff. Some flight simulators and those things. When you start to do this, you realize that it is too slow. When you redraw images, it's a lot of pixel movement. Okay, so you've got to move blocks. You move not just a couple of objects, but the background, and it's hard.

ANALOG: Is that a limit of the ST or is that a general limitation?
TRIO: It's general in that the microprocessor is only so fast. There's so much data. And there was an idea that to change the background maybe you shouldn't redraw it...to do it a little better, you have to change colors on the fly.

ANALOG: Because you need more colors?
TRIO: Yeah. And that's where it starts.

ANALOG: So, at that point you decided to do a paint program?
TRIO: Yes, to do a static display program.

ANALOG: Were you thinking about writing a full-featured paint program, or just an enhancement to take pictures from other paint programs?
TRIO: Well, it was clear that we had to do it from scratch, because you have to have all the painting routines and block-moving routines and everything.

ANALOG: What's the maximum number of colors you can have on the screen with Spectrum?
TRIO: Non-interlaced? Atari hardware has 512 particular colors. All this stuff about 4096, and so forth...it's interlaced, it's dithering, it's all kinds of tricks. All we can get is 512, and that's the maximum. By the way, we explored the [interlace] idea—when you show one picture in even numbered frames and another in the odd numbered frames. But, you know, the flicker.

ANALOG: Right. How do you think these other programs—that are coming out with these multiple palettes and these interlaced tricks—are going to affect Spectrum?
TRIO: The thing is that there was a lot of experimentation. But nobody liked the flickering. So the [interlace] idea was dropped right there. In principle, I would say, with not much modification, we can show the same number [of colors] as theirs. But, again, you lose quality. We think that you can do pretty well with 512 colors.

ANALOG: You didn't use interlace because you said that the flicker was unacceptable. Do you think that that will affect the other programs?
TRIO: Let me put it this way: We had our choice of which way to go. I can see two perspectives here: Number one is the final result of the quality of that display, on the screen; and number two is the user-friendliness, and how comfortable you are working with all these tools. Now, we think that from both of these perspectives, our product is on a much higher level. If someone else wants to do something like that, it's perfectly all right to explore the machine or your own ideas or whatever. It has to be convenient. It's a paint program.

ANALOG: And how long did it take to develop this program entirely from the actual starting point? Actually working on it?
TRIO: From about February [1987].

ANALOG: So we're talking about six, seven months.
TRIO: Yeah. It was ready in August, I guess.

ANALOG: That's a really incredible turnaround. I've seen some graphics software sit for six months to a year.
TRIO: If you consider that we have pretty good backgrounds—Ph.D. in physics, and background in electronics, electrical engineering and in software—then you wouldn't be wondering that the technical level of this program is higher than the average.

To do something like Spectrum 512, you have to have knowledge about how the display is working, how the image is created, how the computer is working, about all kinds of signals, about all kinds of software. When we're talking about creating software these days, people think that to write a program you have to know a particular programming language; either you know BASIC or you know C or you know Assembler. To write something like Spectrum 512, you've got to be pretty diversified. You've got to know something about video images; you've got to know a lot about assembly; you've got to know about the machine.

It's like you're working on a mainframe. You've got to be able to work with the equipment available, to use that oscilloscope or analyzer to get those signals which make sense. To do software, you've got to know the languages; to do hardware, you've got to be an electrical engineer. You've got to have a pretty good idea about electronics.

ANALOG: People will look at this program and think, *What routines are they using there?* They're going to be thinking about system calls or calling the graphics chips; they're not actually figuring how you would address the hardware to make it do things it normally isn't designed to when you go outside of the realm of defining Atari possibility as defined by Atari. They see the demos and they think, *How?*"
TRIO: That's right. That was the idea. We are engineers; we're not just software people, and we're not just hardware people. We're an engineering outfit. We consider that the best way to approach a problem is to get a broad scope of that problem; to use the best available state-of-the-art hardware which will work with the 68000 processor, get right to the nitty-gritties of that machine. And you've got to be a bit of an artist, too, to use the imagination as we've tried to do.

ANALOG: I wanted to get into some specifics on Spectrum, per se. Are you addressing certain chips, but not in a normal way, or bypassing the normal video hardware?

TRIO: We are addressing the color palette, what is called the video shifter [chip], and changing the color registers inside it, like with display line interrupts.

ANALOG: It seems like you must be doing it in the middle of a scan line rather than at the end.
TRIO: Yeah, yeah.

ANALOG: Which is why you need all the tight system timing. You have to know exactly how fast the machine is going, so you know how long it takes you to draw an electron beam across the screen.
TRIO: Yup. I think the idea itself is rather obvious, but, implementation. . . .

ANALOG: The most obvious things are often the hardest ones to pin down. George Lucas once said that he spent years trying to find the obvious. Without knowing the hardware as intimately as you do, there's no possible way of doing this. That's how you do something that's technically impossible to many, because they are trying to get around what they *perceive* as limits, rather than seeing what the actual limits of the hardware are and how the hardware actually works.
TRIO: Our idea was like this: We will change the maximum number of colors as is physically possible. It will create a lot of complications (chuckling), but we'll deal with that somehow later.

ANALOG: You want to implement the basic plan first, then work out the little rough spots.
TRIO: Exactly. And there are plenty of them.

ANALOG: And now you've got this working. But with the amount of overhead and the tight timing, I imagine it would be difficult to do very, very complex animation at this point, at the speed that the computer is running.
TRIO: You never know. It certainly is an open question, but we think there is a good potential.

To write something like Spectrum 512, you've got to be pretty diversified. You got to know something about video images; you've got to know a lot about assembly.

ANALOG: How is Spectrum programmed? Was it in C or what language?
TRIO: It's Assembly. . . all Assembly.

ANALOG: Do you think you could have pulled off the speed or the ability with any other language?
TRIO: Well, some, I guess—70% of it has to be done in Assembly. It's real-time alone. You have to know how long each instruction takes.

ANALOG: The machine only goes so fast, you have to do it at the machine's full speed in order to do what you want to do. If you have the overhead of a compiled language, you won't be able to do it.
TRIO: Yeah, in other programs when you use a high-level language, it will just slow down. Here, the picture will be destroyed (chuckle).

ANALOG: In order to do the system timing, you have to be able to go inside and run at the proper speed.
TRIO: Yeah, some other parts of the program could have been done in . . . (pauses) but not too many of them, so better integration on the whole thing, I guess. Assembler is much richer in the number of instructions, and that gives you more. . . .

ANALOG: More power?
TRIO: To make that little machine work perfectly, you've got to beat the hell out of it, so you better go into the instruction level. To illustrate a little bit what I just said—as you can see when we made another step to [Digispec], what we did first, we really improved [ComputerEyes]. We gave them better abilities to use their own hardware, and then we jumped in and made another step further when we improved the software, which was possible to pull from the Spectrum side.

ANALOG: Can you expand a little bit on how Digispec came about? I realize that you approached the Digital Vision people because they had an existing digitizer, that, again, the market was there; there were a lot of people who had the product already.
TRIO: First of all, we don't have anything exclusive with Digital Vision. At this point, well, their product is improved.

The Atari seemed to us to be the perfect computer to do something. So we've been looking at a variety of different things. We have concentrated on Atari, and I guess it worked out. So far.

Particularly, software-wise. It doesn't mean that it's the best hardware available on the market, so we do have something in the works with other people, particularly working from scratch—combination hardware and software. So we think another digitizer could be done, although we don't have anything certain here.

ANALOG: With regard to the ComputerEyes digitizer: Did you dissect that the same way you did the computer by analyzing the actual hardware—to see what it could do?
TRIO: They have specialized hardware. We didn't go into that. Not in the special hardware side. Starting with the video, the raw video data acquired, and what to do with it, we took over from that.

ANALOG: You just used the device to pull it in; and basically once the data came in, you totally handled it differently than what they did originally.
TRIO: Yes, absolutely.

ANALOG: So, the hardware is the same, but you worked with them to have their software enhanced to a level that you could work with it. It improved their product; and also, now you have another branch on the tree—the Spectrum tree, we could call it—because you instantly have an application for the program. People who have the digitizer now have a reason to buy your program. Or people who have the Spectrum program have a reason to buy the digitizer. It handshakes very nicely, doesn't it?
TRIO: It's a . . .what they call these days, a popular expression: a "win-win" situation. They win, and we win, and everybody wins. I think the user will win because you can see the result right there on the screen. Another thing we applied here, you've got to know simple things—simple from the physics standpoint—you've got to know optics, and we planned a few ideas from the area of optics. And that's how we improved their software too.

ANALOG: In a way, by using the techniques you have, you've managed to smooth out the system's graphics that people would otherwise have considered inferior quality for doing certain types of displays. What this does is put into the hands of people with very inexpensive computers the power to do the kind of graphics work that normally costs a lot more.
TRIO: Oh, yeah (chuckle).

ANALOG: For the small fee of a piece of software, users can now generate graphics of a quality that were impossible previously. So do you see this as affecting ST sales potential?
TRIO: The smoothing is done by introducing additional colors, so the ability to have that many colors in one picture will probably affect the sales of the computers and everything else. Because you can do much more.

ANALOG: At any rate, Spectrum again is a paint program, but now you have a basis. Where do you expect to take it from here?
TRIO: CAD 3-D.

ANALOG: So you expect to work that in with some of the other products?
TRIO: Yeah, we've had pretty good luck, I should say—talking about the marketing side of the picture. We're pretty happy working with *Antic*. And we think that we can really improve their line of graphic products. From another hand, we can expand our own experience going with different types of hardware, and going to different kinds of applications like, for example, digitizers. That's another possibility. The first step may be as some kind of an animation market, limited. Further down the road, we could make it available for some people, to a lot of developers.

ANALOG: Is part of what attracted you to working with, in this case, The Catalog, that they are trying to make a broad-based graphics system with all these various parts? You have sort of fit in there, and you have people who have other parts conceived where eventually they can use your program. Because even if you cannot right now use Spectrum images with CAD 3-D, people will be expecting you to add that somewhere down the line. So you have a built-in market there, if you decide to do that. Is that part of the appeal?
TRIO: Yes, I think that's the next thing to do.

ANALOG: So you are going to license the technology for integrating with other products? It's not limited to your paint program or use with a digitizer. Do you think the process that went into Spectrum—of timing the computer and all—will enhance other people's work. . . and the ST market as a whole?
TRIO: Absolutely. We're saying that [the ST is] a good machine. It has the potentials there, and we'd really like to participate in expanding and improving, and have a much better user base.

ST LOG
THE ATARI ST MONTHLY MAGAZINE

NEXT MONTH IN ST-LOG!

Comdex '88. Atari and third party developers displayed business and productivity products at this large show. We have a complete report along with some interesting surprises.

Opus. Not just the finest spreadsheet program ever published in a magazine for your Atari ST, but one of the best ever offered *anywhere*! Don't believe us? Check out the September issue.

DEGAS Fast Loader. Turbo DEGAS Elite is what you'll get with this desktop utility. Now you can load compressed DEGAS pictures in no time at all.

Start The Presses. A guide to desktop publishing with your ST.

As well as our regular columns and new reviews. All in the September issue of ST-LOG.

IAN'S QUEST

by Ian Chadwick

I was sorting out my hard disk the other day. It's a job I do irregularly, out of necessity when it becomes close to full. I back up everything, then remove a passel of programs I don't use. While I was doing this, I realized that most of the programs on the disk are public domain—PD. That gave me pause to consider the whole business of freeware and shareware.

Ever wonder about PD software? A whole lot of people out there work hours and hours and hours at writing the stuff to give it away. Sounds like poor business to me, but bless 'em all! Some of the most useful programs I own are PD.

Who are the authors who spend their time slaving over a hot computer to give us these gems? Charles Johnson, David Betz, James Luczak, David Small, Frank Cohen, David Addison, Tim Purves, Jerry Cole, George Woodside and many, many more. Forgive me if your name isn't on the list. Many are hackers, but a lot of these people are professionals. It takes a lot of commitment to work that hard for no measurable reward. I think it's high time we said *thanks!*

I guess we all take these programs for granted. After all, good or bad, we don't pay for them. Sure, some are shareware—that means the authors ask you to voluntarily send them a donation. Sometimes this gets you an upgrade or enhanced version. Other times it just makes you feel good. I've never seen any figures on shareware; so I haven't the foggiest idea if anyone makes any money at it. Somehow, it looks more like a labor of love to me.

But where would we be without PD programs? Online services like Delphi and CompuServe would be nothing more than message systems. Ho hum. Some PD offerings are, truthfully, pretty much an amateur's effort; sort of a proclamation to the rest of us, "Hey! Look what I did!" But a lot are very well crafted, programmed and designed. And when source code is included, the user-support idea really makes the concept fly.

Look at ST Writer, one of the best free programs I've ever seen and maybe the best word processor for the ST at present. It came out as a port from the 8-bit Atari Writer. Version 2.52 is the latest (as of this writing), and it's terrific. The printer drivers have been improved, it has mouse control and other new features; all thanks to the efforts of a dedicated bunch who get nothing beyond personal satisfaction for their services (and, of course, a good word processor).

George Woodside's Turtle is another delight. I use it whenever I backup the hard disk. And where would we be without the PD archive/de-arc programs that condense files for transfer and storage?

What about games? There are a fair number of good PD games out there. David Addison gave us Monopoly—as professional a game as I've ever encountered—among others. And he provides the source code in GFA BASIC, so I can tinker to my heart's delight. PD games often include clones of popular commercial games which sell for inflated prices: Centipede, PacMan, that sort of thing. And in a lot of cases, the PD versions are even better than the originals.

PD is also the showcase for a lot of new ideas. Many "hidden" features of GEM and TOS have been revealed and exploited first in PD programs. Some of these efforts show the most creative ideas I've encountered in the ST environment.

There are many nifty utilities out there, at only the cost of the download time, that improve the computing environment considerably. There are programs to select which desk accessories to load, GEM item selector replacements, print spoolers, RAMdisks, replacements for desktop background and icons, printer drivers, graphics translators and more. There are paint and CAD programs, slideshow utilities, educational programs, spreadsheets, word processors, debuggers, disk copiers and sector editors, text editors, databases, languages—so much material that you can get just about any type of program, application or utility you'll ever need in the PD world.

This, of course, means that there is no excuse for piracy of commercial software, with so much good material available for free or very close to it. Then again, there never *was* any excuse for piracy. But that's another column. I was surprised to see how much material I had accumulated on my hard disk and how much of it was PD. I was hard pressed to remove a lot of it, since I use many of these programs. They don't merely consume space on my disk.

Here's a parochial problem: Canada. The question of software and magazine distribution up here has arisen so often that I thought I'd address it here.

Where do Canadian stores get ST-Log or ANALOG? Up here, the magazine doesn't (yet) enjoy as wide a distribution as, sad to say, the competition. I've been asked many times by retailers where they can get the magazine locally. The answer is—I don't really know. But I've found one supplier who deals nationally: Micro D Distributors (not associated with the American company of that name). They also handle major software publishers. They can be reached in the Toronto region at 741-9825 or outside at 1-(800) 387-5855.

If you're in your local store (book, computer, software or magazine outlet), and they don't have ANALOG or ST-Log, ask them to start bringing it in. They can either call ST-Log/ANALOG in California for the name and phone number of a local distributor or call Micro D.

I also wanted to mention Micro D because they have been instrumental in providing me with review copies of software—something most distributors seem loathe to do. One of the problems of writing current reviews in the Great White North is that products, when they reach here, often arrive months behind the U.S. release, if they reach here at all. Many publishers don't have Canadian distributors and a lot of their products simply never get onto the shelves up here. And what does is pretty expensive.

The problem has a lot to do with minimum order quantities (here comes the lesson, so take out your pens, there's a test afterwards). A publisher may demand a minimum order of, say five or ten copies to merit a dealer discount. Sometimes they only have one product (or one suitable for the dealer), and the dealer may not want that many copies of an untried product on his or her shelves. Or the item may be just a customer's special order and no one else wants it. So they order from a distributor who can mix titles for the minimum order and get the one copy, along with other items from other publishers. See?

Without the dealer discount, the markup on a product is hideous. Figure that, with our devalued Canadian peso, software is *already* 30% more expensive in Canada. Add tax, freight charges, distributor's markup and what have you, and the end result runs anywhere between 50% and 100% markup. Here's a single, randomly chosen example: Michtron's great new product, GFA Artist, is $79.95, SLP in the USA. In Canada, it's $129.95. ST dealers up here aren't so numerous that they are highly price-competitive, so the pressure to discount is weak. Most stores *don't* discount these prices. That makes it very hard on the consumer. It discourages casual buying (who casually drops $60 or $75 for a game?) and encourages piracy.

What's the solution? I'm not offering any. I don't know if there is one. I just thought I'd tell you this, to air the problem. If you come up with something—let me know!

Ian Chadwick is a technical writer and editor living with his wife in an igloo in Toronto, Canada. He is currently writing a murder mystery set in Mexico so he can claim travel expenses and designing a game on the campaigns of Napoleon so he can write off his book purchases.

continued from page 31

GEMKIT

GOSUB SCENTER: centers a line of small type (half size)
GOSUB RCENTER: centers a line of regular type (default size)
GOSUB LCENTER: centers a line of large type (1.5X)
GOSUB XLCENTER: centers a line of extra large type (2X)

Here is an example of the appropriate sequence:

```
10 text$="GEMKIT SAVES WORK"
20 TY=250
30 GOSUB LCENTER
```

GEMKIT text centering routines center text in the output window. This is not the true center of the screen. If you desire screen-centered text you should change the values assigned to the variable called *rez*, which can be found in the TYPESIZE subroutine. Appropriate values would be 320 rather than 300, and 160 rather than 150. The type size is automatically returned to the default after each text line.

Writing mode determines the conditions under which text and graphics are printed to the screen. You may control, for instance, whether letters appearing on a patterned fill completely block out the pattern in and around the letters or whether the space around the letters is transparent and permits the fill to show through.

The routines are:

GOSUB BLANKOUT: replace mode, covers previous objects
GOSUB TRANSPARENT: see-through mode
GOSUB EXCLUSIVE: XOR mode, cancels out where pixels conflict
GOSUB REVERSE: white on black (transparent only)

BLANKOUT and EXCLUSIVE are not called with their typical system names of replace and xor, because those are reserved words in ST BASIC. Writing mode will remain in the selected mode until explicitly changed to a new mode or back to the default mode.

Graphics

Three types of graphic elements that GEM refers to as bars may be called through GEMKIT. Essentially the bars are boxes of any size and shape in any screen location.

To use the routines you must set x- and y-pixel coordinates for two corners of the box using variables called *barx1* and *bary1* for the first corner and *barx2, bary2* for the second corner. Then use GOSUB to call for the type of box you desire. If you want a specific fill pattern in the box use the COLOR statement in ST BASIC before you call the bar shape with the GEMKIT. The *ST BASIC Sourcebook* contains a chart of the various fills available.

GOSUB BAR: a filled box with square corners
GOSUB ROUNDBAR: a filled box with round corners
GOSUB HOLLOWBAR: unfilled box with round corners; ignores fill pattern set in COLOR statement
Example:

```
10 REM round cornered box with "Easter egg" pattern
20 COLOR 1,1,1,17,2
30 barx1 =50:bary1 =50:barx2 =300: bary2=90
40 GOSUB ROUNDBAR
```

Polymarkers are predefined graphic shapes that can be manipulated by selecting screen locations, writing mode, colors and so on. GEMKIT has established four pro forma sizes for the polymarkers. Each of the sizes corresponds to one of the four available type sizes. There are six types of markers.

GOSUB DOT: a one-pixel dot (size does not change)
GOSUB PLUS: makes a +
GOSUB ASTERISK: makes an *
GOSUB SQUARE: makes a square
GOSUB CROSSBUCK: makes a diagonal cross
GOSUB DIAMOND: makes diamond (horizontal orientation)

After calling one of the polymarker shapes, you must request a marker size. Pro forma sizes are as follows:

GOSUB SMARK: makes marker one-half default type size
GOSUB RMARK: makes marker default type size
GOSUB LMARK: makes marker 1.5X default type size
GOSUB XLMARK: makes marker 2X default type size

You also must set x- and y-pixel coordinates for the location of the polymarker. The variables are "markx," "marky."
Example:

```
10 REM a double-sized diamond-shaped polymarker
20 REM note that the shape is called before the size
30markx=40:marky=120: GOSUB DIAMOND: GOSUB XLMARK
```

Polyline features allow you to determine what lines drawn with BASIC programs will look like. The two features available in GEMKIT are the width of the line and the types of ends on the line.

GOSUB ENDSTYLE: sets one of three patterns for either or both ends of a line
GOSUB LINEWIDTH: sets width of line in pixels

To use GOSUB ENDSTYLE you must assign values to two variables (one for each end of the line) called *leftend, rightend.* For each variable: 0=square end (default); 1=arrow; 2=rounded.
Example:

```
10 leftend=2:rightend=1: GOSUB ENDSTYLE
20 REM sets linestyles so that the left end of a line will be round and the right end will be an arrow.
```

To use GOSUB LINEWIDTH you must assign an appropriate value to a variable called *linew.* The values are integers indicating the number of pixels wide you want the line to be. *Only* odd numbers (such as 1,3,5,7 should be used).
Example:

```
10 linew=5: GOSUB LINEWIDTH
20 REM sets lines to 5 pixels wide
```

There are two pro-forma dialog boxes available in GEMKIT. Each may be called with a one-word subroutine name.

GOSUB FINISHED: ? in corner, message="Finished?" yes/no buttons that may be clicked with the mouse, return key works with "no"
GOSUB CONTINUE: ! in corner, message="Click box or press RETURN", one response box = "CONTINUE"

You also may design your own dialog boxes with GEMKIT. A variable called *box$* is used to tell GEMKIT what graphic feature you want in the dialog box corner, what message you want in the box and the number and messages in the response boxes to be clicked with mouse.

To indicate which button will respond to a press of the return key, assign an appropriate value to a variable called *returnbox*. You can have up to three buttons in a dialog box and they are numbered left to right. To call a dialog box use GOSUB DIABOX.

Here's the syntax to create your own dialog box:

10 BOX$ = "[2][Will you|continue with your lesson?][YES|NO]"
20 returnbox = 2: GOSUB DIABOX

In Line 10, you will note the dialog box is described in three parts set off in brackets. The number 2 in the first set of brackets tells GEMKIT to put a ? in the upper left corner. The three available graphics are 1 = !; 2 = ?; 3 = stop sign.

The second set of brackets contains the main message in the box. Note the use of the OR symbol (Shift + Backslash). It is used to separate lines of text in the box. You may have up to five lines, and the box grows to accommodate longer lines.

The third bracket set contains the text to be included in the buttons. They also are separated by the Shift + Backslash symbol. Three is the maximum. GEMKIT knows how many buttons to make by checking for the number of OR symbols. Button text should be kept short, under 20 characters total.

Notice there are no spaces between the brackets. GEMKIT likes it that way and can return some strange looking dialog boxes if you aren't careful about that.

Once the box is created it will be asking you to make a choice from among the buttons. The variable that detects which button was clicked with the mouse (or return key) is called *inbox*. It reads the buttons and numbers them from left to right.

Using the example above:

10 BOX$ = [2][Will you|continue with|your lesson?][YES|NO]
20 returnbox = 2: GOSUB DIABOX
30 if inbox = 2 then goto 50
40 REM "yes" would fall through to this line
50 REM "no" jumps to this line

Housekeeping features

The title in the middle of the upper bar of the output window may be changed to, for instance, the name of your program. Assign the text you want to appear there in a variable called *title$*. The call GOSUB NEWTITLE.

Example:

10 title$ = "GEMKIT":GOSUB NEWTITLE

Instead of a STOP or END statement to terminate program execution, use the call GOSUB GETOUT. This isn't required, it's just a tidier way of leaving because it changes the name of the output window back to output.

Improvements

Aside from simply including more routines, GEMKIT could be enhanced by creating more algorithms and user routines that make using the GEM features of GEMKIT easier. Programming techniques—such as the one demonstrated in the sampler for making menu selections with the mouse—could make program development much faster. ■

Gemkit — Listing 1
ST Basic

```
10 clearw 2:fullw 2:gotoxy 0,0
20 TITLE$="GEMKIT SAMPLER":GOSUB NEWTI
TLE
30 GOSUB OUTLINE:TY=80:TEXT$="A GEMKIT
 SAMPLER":GOSUB XLCENTER
40 TY=130:TEXT$="By":GOSUB XLCENTER
50 TY=180:TEXT$="Gordon Billingsley":G
OSUB XLCENTER
60 TY=210:TEXT$="press any key to cont
inue":GOSUB LCENTER
70 GOSUB NORMAL:TY=240:TEXT$="Featurin
g programming examples from GEMKIT":GO
SUB SCENTER
80 CH$=INPUT$(1):clearw 2:fullw 2
90 GOSUB REVERSE:TY=80:TEXT$="MOUSE CO
NTROL":GOSUB LCENTER:GOSUB BLANKOUT
100 GOSUB BOLD:GOTOXY 3,10:?"Click lef
t button to advance through pointer st
yles"
110 GOSUB UNDERLINE:gotoxy 3,11:?"Clic
k right button to continue with the Sa
mpler demonstration"
120 GOSUB NORMAL:POINTER=0
130 GOSUB READMOUSE:gosub show
140 IF BUTTON=2 THEN 170
150 IF BUTTON=1 THEN POINTER=POINTER+1
:IF POINTER>7 THEN POINTER=0
160 GOSUB CHANGE:for l=1 to 20:next:go
to 130
170 CLEARW 2
180 COLOR 1,1,1,17,2:BARX1=30:BARY1=70
:BARX2=400:BARY2=200
190 GOSUB ROUNDBAR:GOSUB BOLD:GOSUB TR
ANSPARENT
200 GOTOXY 2,3:?"GEMKIT DIALOG BOXES A
RE EASY TO CALL":GOSUB NORMAL
210 GOSUB CONTINUE
220 CLEARW 2
230 LINEW=25:GOSUB LINEWIDTH:RIGHTEND=
1:LEFTEND=2:GOSUB ENDSTYLE
240 LINEF 20,60,500,60:GOSUB HATCH:GOS
UB BLANKOUT
250 TX=40:TY=104:TEXT$="THIS IS A POLY
LINE":GOSUB RTYPE
260 GOSUB CONTINUE
270 CLEARW 2:FULLW 2:gosub normal
280 GOSUB DIAMOND:markx=20:marky=80:go
sub rmark:GOSUB MARKCOUNT
290 text$="START SAMPLER AGAIN":tx=35:
ty=80:gosub LTYPE
300 gosub DIAMOND:markx=20:marky=180:g
osub rmark:gosub markcount
310 text$="STOP NOW":tx=35:ty=180:gosu
b LTYPE
320 GOSUB FINGER
330 gosub readmouse:GOSUB SHOW
340 if (my<marcy(1))*(my>marcy(1)-size
mark)*(button=1) then J=1:goto 370
350 if (my<marcy(2))*(my>marcy(2)-size
mark)*(button=1) then J=2:goto 370
360 goto 330
370 GOSUB ARROW:on J goto 10,380
380 clearw 2:J=0:gotoxy 0,0:?"THAT'S A
LL SHE WROTE."
390 gosub get.out
400 MARKCOUNT:POLYMARKER=POLYMARKER+1
410 MARCY(POLYMARKER)=MARKY:RETURN
50000 '******************************
*
50010 '*     GEMKIT, version 1.1
*
50020 '*    By Gordon Billingsley
*
50030 '*
*
50040 '*       Copyright 1987
*
50050 '*     by ST-Log magazine
*
50060 '******************************
*
50070 '
50080 '***************
50090 '** READMOUSE **
50100 '***************
```

```
50110 READMOUSE:poke contrl,124
50120 poke contrl+2,0:poke contrl+6,0:
vdisys(0)
50130 mx=peek(ptsout):my=peek(ptsout+2
):'xy coordinates
50140 button=peek(intout):RETURN
50150 '
50160 '**********************
50170 '** NEW POINTER SHAPES **
50180 '**********************
50190 ARROW:pointer=0:gosub hide:gosub
 change:gosub show:RETURN
50200 BEAM:pointer=1:gosub hide:gosub
change:gosub show:RETURN
50210 BEE:pointer=2:gosub hide:gosub c
hange:gosub show:RETURN
50220 FINGER:pointer=3:gosub hide:gosu
b change:gosub show:RETURN
50230 HAND:pointer=4:gosub hide:gosub
change:gosub show:RETURN
50240 CROSSHAIR:pointer=5:gosub hide:g
osub change:gosub show:RETURN
50250 FATHAIR:pointer=6:gosub hide:gos
ub change:gosub show:RETURN
50260 HOLLOWHAIR:pointer=7:gosub hide:
gosub change:gosub show:RETURN
50270 CHANGE:a#=gb:gintin=peek(a#+8)
50280 poke gintin,POINTER:gemsys(78):R
ETURN
50290 '
50300 '****************************
50310 '** HIDE AND SHOW POINTERS **
50320 '****************************
50330 HIDE:poke contrl,123:vdisys(0):r
eturn
50340 SHOW:poke contrl,122:vdisys(0):r
eturn
50350 '
50360 '________________________________
50370 '
50380 '**********
50390 '** FONTS **
50400 '**********
50410 NORMAL:FONT=0:GOTO NEWFACE
50420 BOLD:FONT=1:GOTO NEWFACE
50430 HATCH:FONT=2:GOTO NEWFACE
50440 ITALICS:FONT=4:GOTO NEWFACE
50450 UNDERLINE:FONT=8:GOTO NEWFACE
50460 OUTLINE:FONT=16
50470 NEWFACE:POKE CONTRL,106:POKE CON
TRL+2,0:POKE CONTRL+6,1
50480 POKE INTIN,FONT:VDISYS(0):RETURN
50490 '
50500 '*******************************
50510 '** SELECTING NEW TYPE SIZES **
50520 '*******************************
50530 XLTYPE:SIZE=24:GOSUB TYPESIZE:GO
TO PUT.TEXT
50540 LTYPE:SIZE=18:GOSUB TYPESIZE:GOT
O PUT.TEXT
50550 STYPE:SIZE=6:GOSUB TYPESIZE:GOTO
 PUT.TEXT
50560 RTYPE:SIZE=12:GOSUB TYPESIZE:GOT
O PUT.TEXT
50570 '
50580 '************************
50590 '** CHANGING TYPE SIZES **
50600 '************************
50610 TYPESIZE:rez=300:if peek(systab)
=3 then rez=150
50620 POKE CONTRL,107:POKE CONTRL+2,0:
POKE CONTRL+6,1
50630 POKE INTIN,SIZE:VDISYS(0):RETURN
50640 '
50650 '********************************
*
50660 '* CENTERING TEXT IN FOUR SIZES
*
50670 '********************************
*
50680 XLCENTER:size=24:gosub typesize:
tx=int(rez-(len(text$)*8)):goto put.te
XT
50690 LCENTER:size=18:gosub typesize:t
x=int(rez-(len(text$)*8)):goto put.tex
t
50700 RCENTER:size=12:gosub typesize:T
X=int(rez-(len(text$)*4)):goto put.tex
t
50710 SCENTER:size=6:gosub typesize:tx
=int(rez-(len(text$)*3)):goto put.text
50720 '
50730 '*****************************
50740 '** PRECISION TEXT PLACEMENT **
50750 '*****************************
50760 PUT.TEXT:long=len(text$)
50770 Poke contrl,8:poke contrl+2,1:po
ke contrl+6,long:poke contrl+12,2
50780 for l=0 to long-1
50790 letter%=asc(mid$(text$,l+1,1))
50800 poke intin+l*2,letter%
50810 next l
50820 poke ptsin,tx:poke ptsin+2,ty:vd
isys(0)
50830 size=12:gosub typesize:RETURN
50840 '
50850 '*****************
50860 '** WRITING MODE **
50870 '*****************
50880 BLANKOUT:wmode=1:goto writemode
50890 TRANSPARENT:wmode=2:goto writemo
de
50900 EXCLUSIVE:wmmode=3:goto writemod
e
50910 REVERSE:wmode=4
50920 WRITEMODE:poke contrl,32:poke co
ntrl+2,0:poke contrl+6,1
50930 poke intin,wmode:vdisys(0):RETUR
N
50940 '
50950 '________________________________
50960 '
50970 '***********************
50980 '** BARS AND RECTANGLES **
50990 '***********************
51000 BAR:BARTYPE=1:GOTO DRAWBAR
51010 HOLLOWBAR:BARTYPE=8:goto drawbar
51020 ROUNDBAR:BARTYPE=9
51030 DRAWBAR:POKE CONTRL,11:POKE CONT
RL+2,2:POKE CONTRL+6,0:POKE CONTRL+10,
BARTYPE
51040 POKE PTSIN,BARX1:POKE PTSIN+2,BA
RY1
51050 POKE PTSIN+4,BARX2:POKE PTSIN+6,
BARY2
51060 VDISYS(0):RETURN
51070 '
51080 '***************
51090 '** POLYMARKERS **
51100 '***************
51110 DOT:mtype=1:goto markertype
51120 PLUS:mtype=2:goto markertype
51130 ASTERISK:mtype=3:goto markertype
51140 SQUARE:mtype=4:goto markertype
51150 CROSSBUCK:mtype=5:goto markertyp
e
51160 DIAMOND:mtype=6
51170 MARKERTYPE:poke contrl,18:poke c
ontrl+2,0:poke contrl+6,1:poke contrl+
12,2
51180 poke intin,mtype:vdisys(0):RETUR
N
51190 '------- marker height -------
51200 SMARK:sizemark=12:goto markersiz
e
51210 RMARK:sizemark=24:goto markersiz
e
51220 LMARK:sizemark=36:goto markersiz
e
51230 XLMARK:sizemark=48
51240 MARKERSIZE:poke contrl,19:poke c
ontrl+2,1:poke contrl+6,0:poke contrl+
12,2
51250 poke ptsin,0:poke ptsin+2,SIZEMA
RK:vdisys(0)
51260 '------- draw the marker -------
51270 MARKER:poke contrl,7:poke contrl
+2,1:poke ptsin,MARKX:poke ptsin+2,MAR
KY-INT(SIZEMARK/4)
51280 vdisys(0):RETURN
```

```
51290 '
51300 '*************************
51310 '** POLYLINE END STYLE **
51320 '*************************
51330 ENDSTYLE:poke contrl,108:poke co
ntrl+2,0:poke contrl+6,2
51340 poke intin,leftend:poke intin+2,
rightend:vdisys(0):RETURN
51350 '
51360 '********************
51370 '** POLYLINE WIDTH **
51380 '********************
51390 LINEWIDTH:poke contrl,16:poke co
ntrl+2,1:poke contrl+6,0
51400 poke ptsin,linew:poke ptsin+2,0:
vdisys(0):RETURN
51410 '
51420 '******************
51430 '** DIALOG BOXES **
51440 '******************
51450 FINISHED:box$="[2][|Finished?][Y
ES|NO]":returnbox=2:goto diabox
51460 CONTINUE:box$="[1][|Click box or
 press <RETURN>][CONTINUE]":returnbox=
1:goto diabox
51470 '------- awaken the box  -------
51480 DIABOX:A#=GB:CONTROL=PEEK(A#):GL
OBAL=PEEK(A#+4)
51490 GINTIN=PEEK(A#+8):GINTOUT=PEEK(A
#+12):ADDRIN=PEEK(A#+16)
51500 '------- box contents  -------
51510 poke contrl,122:poke contrl+2,0:
poke contrl+6,1:poke intin,0:vdisys(0)
51520 N#=addrin:poke gintin,returnbox:
 poke n#,varptr(BOX$):gemsys(52)
51530 '------- box response -------
51540 inbox=peek(gintout):vdisys(0):RE
TURN
51550 '
51560 '______________________________
51570 '
51580 '*******************************
51590 '*CHANGE TITLE IN OUTPUT WINDOW*
51600 '*******************************
51610 NEWTITLE:poke systab+24,1:a# = g
b
51620 gintin = peek(a#+8):poke gintin+
0,peek(systab+8)
51630 poke gintin+2,2:s# = gintin+4:ti
tle$ = title$ + chr$(0)
51640 poke s#,varptr(title$):gemsys(10
5)
51650 poke systab+24,0:RETURN
51660 '
51670 '********************
51680 '** LEAVING GEMKIT **
51690 '********************
51700 GET.OUT:title$="OUTPUT":gosub ne
wtitle:end
```

Gemkit — Listing 1
Checksums

```
 10 data  941, 672, 591, 841, 165, 959
, 960, 175, 174, 470, 5948
 110 data  141, 560, 82, 993, 135, 699
, 387, 821, 759, 759, 5336
 210 data  443, 374, 758, 61, 162, 458
, 301, 769, 213, 870, 4409
 310 data  29, 92, 86, 105, 113, 408,
37, 567, 317, 283, 2037
 410 data  872, 336, 378, 31, 896, 865
, 596, 348, 662, 924, 5908
 50090 data  480, 909, 95, 74, 529, 53
9, 659, 677, 672, 681, 5315
 50190 data  936, 536, 295, 759, 546,
276, 998, 517, 513, 939, 6315
 50290 data  668, 45, 419, 49, 654, 71
9, 661, 648, 665, 591, 5119
 50390 data  913, 576, 337, 870, 102,
369, 842, 713, 884, 155, 5761
 50490 data  670, 173, 548, 177, 934,
606, 787, 610, 667, 811, 5983
 50590 data  970, 796, 830, 811, 169,
662, 352, 2, 356, 511, 5459
 50690 data  578, 559, 449, 659, 181,
524, 185, 47, 299, 711, 4192
 50790 data  246, 451, 371, 966, 807,
664, 178, 934, 182, 150, 4949
 50890 data  664, 517, 736, 262, 208,
665, 652, 669, 813, 949, 6135
 50990 data  817, 309, 258, 225, 4, 34
4, 638, 907, 665, 95, 4262
 51090 data  866, 80, 478, 722, 398, 9
75, 427, 752, 514, 259, 5471
 51190 data  876, 346, 354, 352, 200,
473, 704, 165, 716, 913, 5099
 51290 data  671, 670, 686, 674, 181,
516, 664, 346, 253, 350, 5011
 51390 data  275, 164, 657, 171, 790,
175, 561, 740, 39, 763, 4335
 51490 data  711, 871, 808, 223, 903,
726, 666, 653, 670, 318, 6549
 51590 data  72, 303, 797, 784, 404, 6
3, 785, 669, 351, 175, 4403
 51690 data  355, 485, 840
```

GEMKIT
END

CircuitMaker

CircuitMaker is a professional full featured program that enables you to design, construct and test an unlimited variety of digital circuits. Using CircuitMaker, you eliminate the need to purchase breadboards, integrated circuits, wire and power supplies. CircuitMaker allows you to design and test your digital circuits with just a few clicks of the mouse!

CircuitMaker is designed for the professional as well as the student that is just learning about digital logic. CircuitMaker is a must for your electronic projects!

Only $79.95

iliad
Software Inc.

P.O. Box 1144
495 West 920 North
Orem, Utah 84057
(801) 226-3270
Office hours 10:00AM-6:00PM MST

CIRCLE #114 ON READER SERVICE CARD.

continued from page 17

the sprite's screen position, you need to load the effective address of the sprite's definition block into address register A0. Finally, you must load the effective address of the sprite's background buffer into address register A2. Having done all this you simply call the DRAW_SPRITE function by using the word opcode label that you defined earlier. Here's an example of the process:

```
MOVE #150,D0
MOVE #100,D1
LEA SPRITE1,A0
LEA SPRITE1_BUF,A2
DC.W DRAW_SPRITE
```

The sprite definition block mentioned above is a block of continuous memory that contains a total of 37 words that describe the sprite's appearance on the screen. This definition block must contain the following information:

Word 1 :	X offset to sprite's hot spot
Word 2 :	Y offset to sprite's hot spot
Word 3 :	format flag (0 = VDI format, 1 = XOR format)
Word 4 :	background color
Word 5 :	foreground color

The next 32 words contain the desired sprite pattern. The pattern must be in the following order:

Word 6 :	background pattern of top line
Word 7 :	foreground pattern of top line
Word 8 :	background pattern of next line down
Word 9 :	foreground pattern of next line down

The first two words of the sprite definition block as mentioned above are the X/Y coordinates of the sprite's "hot spot." This coordinate set actually specifies the active point of the sprite. This is the point from which the sprite is drawn relative to when the DRAW_SPRITE opcode is called. The next word in the sprite definition block, Word 3 is the format flag. This flag tells the DRAW_SPRITE routine how to display the sprite's foreground and background in relationship to each other. The format flag usage is as follows:

—VDI Format —

FG	BK	Result
0	0	The background (BK) appears.
0	1	The color in Word 4 appears.
1	0	The color in Word 5 appears.
1	1	The color in Word 5 appears.

—XOR Format —

FG	BK	Result
0	0	The background (BK) appears.
0	1	The color in Word 4 appears.
1	0	The pixel on the screen is XORed with the foreground (FG) bits of the sprite.
1	1	The color in Word 5 appears.

Words 4 and 5 of the sprite definition block as stated before specify the sprite's background color and foreground color, respectively. The background (BK), color for the sprites should be set to a value of 0 or else the sprite will appear as your bit pattern design imprinted upon the top of a colored square. The foreground (FG) color can have any value between 1 and 15 depending upon which screen resolution you are currently using. The remaining 32 words of the sprite definition block contain the bit pattern for the image you want to display. Here's an example of a sprite definition block:

```
SPRITE1: DC.W 8,1,0,0,1 ;X hot,Y hot,Format flag,BK color,FG color
  DC.W %0000000000000000 ;#1 BK (Background) pattern
  DC.W %1111111111111111 ;#1 FG (Foreground) pattern
  DC.W %0000000000000000 ;#2 BK
  DC.W %1111111111111111 ;#2 FG
  D@.W %0000000000000000 ;#3 BK
  DC.W %1111111111111111 ;#3 FG
  DC.W %0000000000000000 ;#4 BK
  DC.W %1111111111111111 ;#4 FG
  DC.W %0000000000000000 ;#5 BK
  DC.W %1111100000011111 ;#5 FG
  DC.W %0000000000000000 ;#6 BK
  DC.W %1111100000011111 ;#6 FG
  DC.W %0000000000000000 ;#7 BK
  DC.W %1111100000011111 ;#7 FG
  DC.W %0000000000000000 ;#8 BK
  DC.W %1111100000011111 ;#8 FG
  DC.W %0000000000000000 ;#9 BK
  DC.W %1111100000011111 ;#9 FG
  DC.W %0000000000000000 ;#10 BK
  DC.W %1111100000011111 ;#10 FG
  DC.W %0000000000000000 ;#11 BK
  DC.W %1111100000011111 ;#11 FG
  DC.W %0000000000000000 ;#12 BK
  DC.W %1111100000011111 ;#12 FG
  DC.W %0000000000000000 ;#13 BK
  DC.W %1111111111111111 ;#13 FG
  DC.W %0000000000000000 ;#14 BK
  DC.W %1111111111111111 ;#14 FG
  DC.W %0000000000000000 ;#15 BK
  DC.W %1111111111111111 ;#15 FG
  DC.W %0000000000000000 ;#16 BK
  DC.W %1111111111111111 ;#16 FG
```

Now that we know how to put a sprite on the screen, we must be able to save the backgound behind the sprite so it can be put back once the sprite is moved. The sprite background buffer as mentioned earlier in this article does that very thing; it holds the background behind the sprite, so that when you move the sprite, the background beneath it can be restored to its original condition. This buffer must be 74 bytes long for high resolution, 138 bytes long for medium resolution, and 266 bytes long for low resolution. The difference is caused by the number of bit planes involved for the different screen resolutions. In low resolution the ST's screen needs a total of four bit planes to display the 16 colors normally available at any given time in that mode. The medium-resolution screen requires only two-bit planes to display the four colors available in that mode, while the high-resolution screen needs only a one-bit plane to display the monochrome image it produces. The formula used to determine the buffer length needed is N*64+10 where N is the number of bit planes.

Here's an example for all three resolutions:

```
SPRITE1_BUF: DC.B 74  ;high
SPRITE1_BUF: DC.B 138 ;med.
SPRITE1_BUF: DC.B 266 ;low
```

Once you know how a sprite is displayed on the screen and how to save into a buffer its background for later use, all that's left to learn is how to use that buffer to restore the background behind the sprite to its original condition. The routine used for this function is the Line A opcode $A00C, UNDRAW_SPRITE. This opcode has only one parameter. It is called simply by loading the effective address of the appropriate sprite's background buffer back into address register A2. You then call the UNDRAW_SPRITE function by the word opcode label that you defined earlier in the program.

In the demo program SPRITES, the INITIALIZATION, DRAW_SPRITE, and UNDRAW_SPRITE functions are put into macro form. The format of these calls are as follows:

LINE_AI = $A000 :label the initialization function

DRAW_SPRITE = $A00D :label the Draw_Sprite function

UNDRAW_SPRITE = $A00C :label the Undraw_Sprite function

LINE_A_INIT :do the Initialization function

MOVE_SPRITE X,Y,pointer to sprite definition block, pointer to the sprite's background buffer

ERASE_SPRITE pointer to the sprite's background buffer.

The macro definitions themselves look as shown in Figure 1.

These macros are optional. If you don't have a macro assembler like AssemPro or its equivalent then you will have to use the coding discussed earlier in this article.

The demo program SPRITES is set up to use the low-resolution screen of the ST, although it will work in medium resolution. The program starts by drawing three stationary sprites of differing colors in the middle of the screen and two movable sprites in opposite corners of the screen. A prompt will then appear asking you to press any key to start the demo. After pressing a key, the two moveable sprites will begin to travel smoothly across the screen in opposite directions. They will continue to loop like that for about two minutes, and then you will be asked to press "X" to exit the program.

The smooth sprite animation achieved in this demo is accomplished by using the XBIOS function #37, (WVBL). This built-in XBIOS function is very interesting; it waits for the next picture return to occur and then synchronizes the following graphic output to it, whatever graphic output that might be. In the demo program the WVBL call is made right before the sprite is erased, thus insuring a smooth look for the sprite's movement to its next position. Here's an example of WVBL call:

```
MOVE.W #37,-(SP) ;WVBL function #37
TRAP #14         ;Call XBIOS
ADD.Q #2,SP      ;Restore the stack
```

(The demo program uses a macro for this function too. Since there are no parameters for this call all you have to do is put the WVBL call right before whatever graphics you want to synchronize with the picture return.)

The following macro definition for WVBL was used in the demo program:

```
WVBL:MACRO
 MOVE.W #37,-(SP) ;XBIOS WVBL function
 TRAP #14         ;Call XBIOS
 ADDQ.L #2,SP     ;Restore stack
 ENDM
```

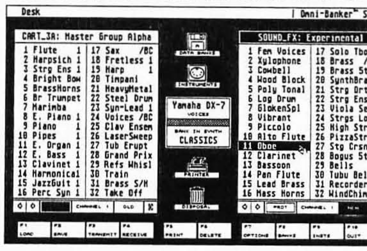
Kelly Schreiner has been programming Ataris for approximately six years. He enjoys programming his 520 ST in both 68000 machine language and C, and is currently studying to become an electronic engineer.

```
LINE_A_INIT:MACRO
 DC.W LINE_AI      ;do the INITIALIZATION function
 ENDM

MOVE_SPRITE:MACRO $\1,$\2,$\3,$\4
 MOVE \1.D0          ;X position parameter
 MOVE \2,D1          ;Y position parameter
 LEA \3,A0           ;pointer to the sprite definition block
 LEA \4,A2           ;pointer to the sprite's background buffer
 DC.W DRAW_SPRITE    ;do the DRAW_SPRITE function
 ENDM

ERASE_SPRITE:MACRO $\1
 LEA \1,A2            ;pointer to the sprite's background buffer
DC.W UNDRAW_SPRITE  ;do the UNDRAW_SPRITE function
 ENDM
```

OMNI-BANKER™ ST



```
Desk                                        | Omni-Banker™ ST |
CART_3A: Master Group Alpha              SOUND_FX: Experimental
 1 Flute      1 | 17 Sax      /BC         1 Fem Voices  | 17 Solo Tbone
 2 Harpsich 1   | 18 Fretless 1           2 Xylophone   | 18 Brass   /BC
 3 Strg Ens 1   | 19 Harp      1          3 Cowbell     | 19 Brass 5ths
 4 Bright Bow   | 20 Timpani              4 Wood Block  | 20 SynthBrass
 5 BrassHorns   | 21 HeavyMetal           5 Poly Tonal  | 21 Strg Qrt 1
 6 Br Trumpet   | 22 Steel Drum           6 Log Drum    | 22 Strg Ens 2
 7 Marimba      | 23 Syn-Lead 1           7 GlokenSpl   | 23 Viola Secn
 8 E. Piano 1   | 24 Voices /BC           8 Vibrant     | 24 Strgs Low
 9 Piano      1 | 25 Clav Ensem           9 Piccolo     | 25 High Strgs
10 Pipes      1 | 26 LaserSweep          10 Alto Flute  | 26 PizzaStrgs
11 E. Organ 1   | 27 Tub Erupt           11 Oboe        | 27 Stg Crsndo
12 E. Bass    1 | 28 Grand Prix          12 Clarinet    | 28 Bogus 5ths
13 Clavinet 1   | 29 Refs Whisl          13 Bassoon     | 29 Bells
14 Harmonical   | 30 Train               14 Pan Flute   | 30 Tubu Bells
15 JazzGuit 1   | 31 Brass S/H           15 Lead Brass  | 31 Recorders
16 Perc Syn 1   | 32 Take Off            16 Mass Horns  | 32 WindChimes
```

"Universal" MIDI Librarian for the Atari 520/1040 ST

- Works as Desk Accessory *and* normal program
- View *entire banks* of up to 128 named patches
- Rename, audition, and drag patches within or between banks, using the mouse or keystrokes
- Manage dozens of *different* synthesizers
- Organize thousands of patches on a single disk
- Inexpensive alternative to slow tapes, costly cartridges, and single-instrument librarians
- Already supports *over* 60 instruments: 23 Yamaha, 10 Roland, 8 Sequential, 6 Korg, 5 Casio, 4 Siel, 2 Ensoniq, 2 Kawai, & more...
- Expanding to support most current *and future* MIDI devices, with free updates issued quarterly
- NOT Copy Protected; Mono or color monitors
- No risk trial, **30 day money-back guarantee**
- Available now! Phone (616) 372-5972

order direct from:

Paradigm™ Software Products
1369 Concord Place, Suite 3-B
Kalamazoo, Michigan 49008

Still Only...
$55
Plus $5 S/H
Check/M.O.

(Mich. resid. add 4% tax, foreign $10 S/H)

CIRCLE #110 ON READER SERVICE CARD.

Sprite Programming — Listing 1
Assempro Macro Assembler

```
 ILABEL TOS\TOS.L                                   ;Include TOS Library
 GEM_INIT                                           ;Initialize GEM

LINE_AI       = $A000                               ;Label INITIALIZATION function
UNDRAW_SPRITE = $A00C                               ;Label UNDRAW_SPRITE function
DRAW_SPRITE   = $A00D                               ;Label DRAW_SPRITE function

 VS_COLOR          GR_HANDLE,#0,BLACK               ;Change color 0 to black
 VS_COLOR          GR_HANDLE,#1,BLUE                ;Change color 1 to blue
 VST_EFFECTS       GR_HANDLE,#4,D3                  ;Set text special effects to Italic
 VST_COLOR         GR_HANDLE,#1,D3                  ;Set text color to 1
 VSL_COLOR         GR_HANDLE,#1,D3                  ;Set line color to 1
 VSF_COLOR         GR_HANDLE,#1,D3                  ;Set fill color to 1
 VSF_INTERIOR      GR_HANDLE,#2,D3                  ;Set fill style to dotted
 VSF_STYLE         GR_HANDLE,#1,D3                  ;Set fill index to 1
 VSF_PERIMETER     GR_HANDLE,#1,D3                  ;Set perimeter fill visibility

 V_HIDE_C          GR_HANDLE                        ;Hide cursor form
 V_CLRWK           GR_HANDLE                        ;Clear screen


 VR_RECFL          GR_HANDLE,TEST_BOX               ;Draw filled rectangle
 V_RBOX            GR_HANDLE,FULL_SCREEN_BOX ;Draw full screen outline
 LINE_A_INIT                                        ;Initialize Line A Opcodes

 V_GTEXT           GR_HANDLE,#61,#59,MSG1           ;Sprites message
 VST_COLOR         GR_HANDLE,#3,D3                  ;Set text color to 3
 V_GTEXT           GR_HANDLE,#37,#25,MSG2           ;Start message

 MOVE_SPRITE #30,#100,SPRITE1,SPRITE0_BUF           ;Stationary sprite1
 MOVE_SPRITE #160,#100,SPRITE2,SPRITE0_BUF          ;Stationary sprite2
 MOVE_SPRITE #289,#100,SPRITE3,SPRITE0_BUF          ;stationary sprite3

 MOVE #20,S1X                                       ;Moveable sprite1's X position
 MOVE #300,S3X                                      ;Moveable sprite3's X position
 MOVE_SPRITE S1X,#50,SPRITE1,SPRITE1_BUF            ;Put moveable sprite1 on screen
 MOVE_SPRITE S3X,#150,SPRITE3,SPRITE3_BUF           ;Put moveable sprite3 on screen
 CONIN_WE                                           ;Check keyboard
                                                    ;Key pressed
 MOVE #0,LOOP_NUMBER                                ;Set a loop counter
MAIN_LOOP:
 ADD.W #1,LOOP_NUMBER                               ;Increment loop counter by 1
START_LOOP:
 WVBL                                               ;Call XBIOS function #37
 ERASE_SPRITE SPRITE1_BUF                           ;Erase moveable sprite1
 ERASE_SPRITE SPRITE3_BUF                           ;Erase moveable sprite3
 ADD.W #2,S1X                                       ;Add 1 to sprite1 X position
 SUB.W #2,S3X                                       ;SUB. 1 from sprite3 X position
 MOVE_SPRITE S1X,#50,SPRITE1,SPRITE1_BUF            ;Redraw moveable sprite1
 MOVE_SPRITE S3X,#150,SPRITE3,SPRITE3_BUF           ;Redraw moveable sprite3
 CMP #300,S1X                                       ;Is sprite1's X position>300
 BHI BACK_AGAIN                                     ;Yes, move the other way now
 JMP START_LOOP                                     ;No, keep moving the same way

BACK_AGAIN:                                         ;Move sprite other way
 WVBL                                               ;Call XBIOS function #37
 ERASE_SPRITE SPRITE1_BUF                           ;Erase moveable sprite1
 ERASE_SPRITE SPRITE3_BUF                           ;Erase moveable sprite3
 SUB.W #2,S1X                                       ;Sub. 1 from sprite1 X position
 ADD.W #2,S3X                                       ;Add 1 to sprite3 X position
 MOVE_SPRITE S1X,#50,SPRITE1,SPRITE1_BUF            ;Redraw moveable sprite1
 MOVE_SPRITE S3X,#150,SPRITE3,SPRITE3_BUF           ;Redraw moveable sprite3
 CMP #20,S1X                                        ;Is sprite1's X position<20
 BLS MAIN_LOOP                                      ;Yes, do another full loop
 CMP #25,LOOP_NUMBER                                ;25 Full loops yet?
 BEQ FINIS                                          ;Yes, check for exit demo
 JMP BACK_AGAIN                                     ;No, do another full set of loops

FINIS:
 MOVE_SPRITE S1X,#50,SPRITE1,SPRITE1_BUF            ;Show moveable sprite1
 MOVE_SPRITE S3X,#150,SPRITE3,SPRITE3_BUF           ;Show moveable sprite3
 VST_COLOR         GR_HANDLE,#2,D3                  ;Set text color to 2
 V_GTEXT           GR_HANDLE,#80,#180,MSG3          ;Exit message
 CONIN_WE                                           ;Key pressed ?
 CMP.B #'X',D0                                      ;Is it 'X' uppercase
 BEQ OUT                                            ;Yes, exit demo program
 CMP.B #'x',D0                                      ;Is it 'x' lowercase
 BEQ OUT                                            ;Yes, exit demo program
 JMP FINIS                                          ;Invalid key, check keyboard again
OUT:
 V_SHOW_C          GR_HANDLE,#0                     ;Show cursor form
 VST_EFFECTS       GR_HANDLE,#0,D3                  ;Set text effects to normal
 VST_COLOR         GR_HANDLE,#1,D3                  ;Set text color to 1
 GRAF_SHRINKBOX SCREEN_CENTER_BOX,FULL_SCREEN_BOX,D3 ;Draw shrinking box
 GEM_EXIT                                           ;Exit GEM
```

```
SPRITE1: DC.W 8,1,0,0,1                          ;Sprite1 definition block
 DC.W %0000000000000000 ;#1 BK (Background pattern)
 DC.W %1111111111111111 ;#1 FG (Foreground pattern)
 DC.W %0000000000000000 ;#2 BK
 DC.W %1111111111111111 ;#2 FG
 DC.W %0000000000000000 ;#3 BK
 DC.W %1111111111111111 ;#3 FG
 DC.W %0000000000000000 ;#4 BK
 DC.W %1111111111111111 ;#4 FG
 DC.W %0000000000000000 ;#5 BK
 DC.W %1111100000011111 ;#5 FG
 DC.W %0000000000000000 ;#6 BK
 DC.W %1111100000011111 ;#6 FG
 DC.W %0000000000000000 ;#7 BK
 DC.W %1111100000011111 ;#7 FG
 DC.W %0000000000000000 ;#8 BK
 DC.W %1111100000011111 ;#8 FG
 DC.W %0000000000000000 ;#9 BK
 DC.W %1111100000011111 ;#9 FG
 DC.W %0000000000000000 ;#10 BK
 DC.W %1111100000011111 ;#10 FG
 DC.W %0000000000000000 ;#11 BK
 DC.W %1111100000011111 ;#11 FG
 DC.W %0000000000000000 ;#12 BK
 DC.W %1111100000011111 ;#12 FG
 DC.W %0000000000000000 ;#13 BK
 DC.W %1111111111111111 ;#13 FG
 DC.W %0000000000000000 ;#14 BK
 DC.W %1111111111111111 ;#14 FG
 DC.W %0000000000000000 ;#15 BK
 DC.W %1111111111111111 ;#15 FG
 DC.W %0000000000000000 ;#16 BK
 DC.W %1111111111111111 ;#16 FG

SPRITE2:DC.W 8,1,0,0,2                           ;Sprite2 definition block
 DC.W %0000000000000000 ;#1 BK (Background pattern)
 DC.W %1111111111111111 ;#1 FG (Foreground pattern)
 DC.W %0000000000000000 ;#2 BK
 DC.W %1111111111111111 ;#2 FG
 DC.W %0000000000000000 ;#3 BK
 DC.W %1100000000000011 ;#3 FG
 DC.W %0000000000000000 ;#4 BK
 DC.W %1100000000000011 ;#4 FG
 DC.W %0000000000000000 ;#5 BK
 DC.W %1100000000000011 ;#5 FG
 DC.W %0000000000000000 ;#6 BK
 DC.W %0000011111100000 ;#6 FG
 DC.W %0000000000000000 ;#7 BK
 DC.W %0000011111100000 ;#7 FG
 DC.W %0000000000000000 ;#8 BK
 DC.W %0000011111100000 ;#8 FG
 DC.W %0000000000000000 ;#9 BK
 DC.W %0000011111100000 ;#9 FG
 DC.W %0000000000000000 ;#10 BK
 DC.W %1100000000000011 ;#10 FG
 DC.W %0000000000000000 ;#11 BK
 DC.W %1100000000000011 ;#11 FG
 DC.W %0000000000000000 ;#12 BK
 DC.W %1100000000000011 ;#12 FG
 DC.W %0000000000000000 ;#13 BK
 DC.W %1100000111000011 ;#13 FG
 DC.W %0000000111000011 ;#14 BK
 DC.W %1111111111111111 ;#14 FG
 DC.W %0000000000000000 ;#15 BK
 DC.W %1111111111111111 ;#15 FG
 DC.W %0000000000000000 ;#16 BK
 DC.W %0000110001110000 ;#16 FG

SPRITE3:DC.W 8,1,0,0,3                           ;Sprite3 definition block
 DC.W %0000000000000000 ;#1 BK (Background pattern)
 DC.W %1100000000000011 ;#1 FG (Foreground pattern)
 DC.W %0000000000000000 ;#2 BK
 DC.W %1100000000000011 ;#2 FG
 DC.W %0000000000000000 ;#3 BK
 DC.W %1100000000000011 ;#3 FG
 DC.W %0000000000000000 ;#4 BK
 DC.W %1110000000000111 ;#4 FG
 DC.W %0000000000000000 ;#5 BK
 DC.W %1110000000000111 ;#5 FG
 DC.W %0000000000000000 ;#6 BK
 DC.W %1110000000000111 ;#6 FG
 DC.W %0000000000000000 ;#7 BK
 DC.W %1111000000001111 ;#7 FG
DC.W %0000000000000000 ;#8 BK
DC.W %1111000000001111 ;#8 FG
DC.W %0000000000000000 ;#9 BK
DC.W %1111000000001111 ;#9 FG
DC.W %0000000000000000 ;#10 BK
DC.W %1111110000111111 ;#10 FG
DC.W %0000000000000000 ;#11 BK
DC.W %1111111111111111 ;#11 FG
DC.W %0000000000000000 ;#12 BK
DC.W %0000000110000000 ;#12 FG
DC.W %0000000000000000 ;#13 BK
DC.W %0000001001000000 ;#13 FG
DC.W %0000000000000000 ;#14 BK
DC.W %0000111001110000 ;#14 FG
DC.W %0000000000000000 ;#15 BK
DC.W %0011100000011100 ;#15 FG
DC.W %0000000000000000 ;#16 BK
DC.W %1100000000000011 ;#16 FG

SPRITE0_BUF:    DS.B 266                    ;Non-moving sprite background buffer
SPRITE1_BUF:    DS.B 266                    ;Moveable sprite1 background buffer
SPRITE3_BUF:    DS.B 266                    ;Moveable sprite3 background buffer
LOOP_NUMBER:    DS.W 1                      ;Variable for loop counter
S1X:            DS.W 1                      ;Variable for sprite1 X position
S3X:            DS.W 1                      ;Variable for sprite3 X position
FULL_SCREEN_BOX:    DC.W 0,0,319,199        ;Coordinates for full screen box
SCREEN_CENTER_BOX: DC.W 315,95,10,10        ;Coordinates for screen center box
MSG1:           DC.B "ST Sprites really work!",0 ;Sprites message
MSG2:           DC.B "Press any key to start the demo",0 ;Start message
MSG3:           DC.B "Press 'X' to Exit",0  ;Exit message
TEST_BOX:       DC.W 20,30,300,169          ;Filled rectangle coordinates
BLACK:          DC.W 0,0,0                  ;Color black
BLUE:           DC.W 0,0,1000               ;Color blue
 END
```

END

HARDWARE ADD–ON's for ATARI ST

MEMORY upgrades:

Solderless "plug in" installation, 1 year warranty

520ST – expand to 1, 2.5 or 4 MB on ONE board – prices start at $129 for the 0K version – or go to 1 MB only, $79 – without ram.

1040/520STfm – upgrade to 2.5 and 4 MB, fully socketed, 0K board $149.

For all our memory upgrades: on board CLOCK module only $30 incl. software!

For more detailed information phone or write to:

tech–specialities Co.
1022 Hodgkins, Houston, TX 77032
☎ (713) 590-2068 and 590-3738

We ship COD or prepaid, sorry no credit cards!

S/H on memory upgrades $5

on Hard Drive Kits – $10/no drive – $20/w. dr.

Texas residents add 8% state sales tax.

EXPANDABLE Hard Drive Kits:

1. 10" x 6.5" x 15" with full SCSI interface – 150 W PC power supply with fan – room for up to 5 half height drives – mounts on floor, under desk or on desktop – can supply power to 520ST and disk drives.

kit with 40 MB full height 30 ms drive $745

2. MEGA footprint, 3.7" high, 1-port SCSI Interface – room and power for 3 half height or 1 each full and half height drive, comes with fan

kit with 10 MB half height drive $395

3. 4.5" wide x 6" high x 13" deep with 1 port SCSI host adapter – ready for 2 half height or 1 full height – 55 W power supply, fan optional

kit with 20 MB 1/2 height $485

Atari 520ST, 1040St, 520STfm and MEGA are trademarks of ATARI Corp.

CIRCLE #104 ON READER SERVICE CARD.

continued from page 21

```
function EVENT : integer;
var d:integer;
begin
     repeat
     event:=get_event(e_button,1,1,1,0,false,0,0,0,0,
                      false,0,0,0,0,msg,d,d,d,msx,msy,d);
     until (msy<191) and (msy>10);
end;

{*PE*}
function SEARCH_BOX (var dir:integer; cpx, cpy : integer):integer;
var i:integer;
begin
     search_box:=0;
     do_sound(70,3);
     delay(2);
     {path end?}
     if (m_box[cpx+la[dir,1], cpy+la[dir,2]]=1) or
        (m_box[cpx+la[dir,1], cpy+la[dir,2]]=2) then
     begin
          search_box:=4;
          do_sound(150,4);
          delay(2);
     end
     else

     {check for direct hit}
     if m_box[cpx+la[dir,1], cpy+la[dir,2]]=-99 then
     begin
           search_box:=1;
           for i:=9 downto 1 do do_sound(i*100,2);
     end
     else

     {check for u-turn}
     if (m_box[cpx+la[dir,3], cpy+la[dir,4]]=-99) and
        (m_box[cpx+la[dir,5], cpy+la[dir,6]]=-99) then
     begin
           search_box:=2;
           for i:=9 downto 1 do do_sound(i*100,2);
           for i:=1 to 9 do do_sound(i*100,2);
     end
     else

     {change direction}
     if m_box[cpx+la[dir,3], cpy+la[dir,4]]=-99 then
     begin
          dir:=la[dir,7];
          search_box:=3;
          do_sound(350,4);
          delay(2);
     end
     else
     if m_box[cpx+la[dir,5], cpy+la[dir,6]]=-99 then
     begin
          dir:=la[dir,8];
          search_box:=3;
          do_sound(550,4);
          delay(2);
     end;
 end;

{*PE*}
procedure CLICK_OUT_BOX (px,py:integer);
var sx,sy,flag:integer;
begin
     sx:=px;
     sy:=py;
     if (px+py>0) and (py-px<>9) and (px-py<>9) then
     begin
          m_box[px,py]:=2;
          draw_shape (px,py,pattern);
          pattern:=pattern+1;
          if pattern>24 then pattern:=9;
          if px=0 then dir:=1 else
          if px=9 then dir:=2 else
          if py=0 then dir:=3 else
          if py=9 then dir:=4;
          flag:=0;
          count:=0;
          repeat
                flag:=search_box (dir,px,py);
                if (flag=0) or (flag=4) then
```

```
                    begin
                         px:=px+la[dir,1];
                         py:=py+la[dir,2];
                    end;
                    if flag=1 then
                    begin
                         draw_shape (sx,sy,1);
                         pattern:=pattern-1;
                    end;
                    if flag=2 then
                    begin
                         draw_shape (sx,sy,2);
                         pattern:=pattern-1;
                    end;
                    if (flag=4) and (count>1) then
                    begin
                         draw_shape (px,py,pattern-1);
                         m_box[px,py]:=2;
                    end;
                    count:=count+1;
               until (flag=1) or (flag=2) or (flag=4);
          end;
     end;

     (*PE*)
     procedure CLICK_IN_BOX (px,py:integer);
     begin
          do_sound(100,2);
          if (m2_box[px,py]=0) and (guess_num<skill_level) then
          begin
               m2_box[px,py]:=-1;
               draw_shape(px,py,3);
               guess_num:=guess_num+1;
          end
          else
          if m2_box[px,py]=-1 then
          begin
               m2_box[px,py]:=0;
               draw_shape(px,py,26);
               guess_num:=guess_num-1;
          end;
          delay(15);
     end;

     (*PE*)
     procedure DO_SCORE;
     var sc,i,j,k,px,py,click:integer;
     begin
          sc:=0;
          for i:=1 to 8 do
           for j:=1 to 8 do
           begin
                do_sound(500,5);
                delay(5);
                if (m_box[i,j]=-99) and (m2_box[i,j]=0) then
                begin
                     draw_shape(i,j,25);
                     do_sound(700,25);
                     delay(5);
                     do_sound(1200,35);
                end;
                if (m_box[i,j]=-99) and (m2_box[i,j]=-1) then
                begin
                     draw_shape(i,j,4);
                     sc:=sc+1;
                     for k:=1 to 10 do
                     begin
                          do_sound(50,4);
                          delay(3);
                          do_sound(200,4);
                          delay(3);
                     end;
                end;
           end;
           if sc=skill_level then sc:=100
           else sc:=sc*trunc(100/skill_level);
           draw_shape(9,0,7);
           write (chr(27),'Y',chr(55),chr(32),sc);
           show_mouse;
           repeat
                 click:=event;
                 px:=msx div 32;
                 py:=msy div 20;
                 if (px=9) and (py=0) then new_game:=true;
                 if (px=9) and (py=9) then quit:=true;
```

```
        until quit or new_game;
        hide_mouse;
end;

{*PE*}
procedure PLAY_GAME;
var i,click,px,py,dummy :integer;
begin
     pattern:=9;
     guess_num:=0;
     new_game:=false;
     quit:=false;
     write (chr(27),'Y',chr(55),chr(32),skill_level,'  ');
     repeat
            click:=event;
            hide_mouse;
            px:=msx div 32;
            py:=(msy-10) div 18;
            if py>9 then py:=9;
            if py<0 then py:=0;
            if (px=0) and (py=0) and (guess_num=skill_level) then do_score
            else
            if px+py<>18 then
            if (m_box[px,py]=1) then click_out_box(px,py)
            else click_in_box(px,py);
            show_mouse;
            if (px=9) and (py=9) then quit:=true;
     until quit or new_game;
end;

BEGIN
     for i:=0 to 15 do
       desk_colors[i]:=s_color(i,-1);
     if init_gem < 0 then
       button:=do_alert('[1][ CANNOT INIT APPLICATION ][ SORRY ]',0)
     else
     if get_res<>0 then
       button:=do_alert('[1][ Game must be run in LOW resolution ][ SORRY ]',0)
     else
  begin
     write(chr(7));
     init_mouse;
     hide_mouse;
     set_mouse(m_point_hand);
     clear_screen;
     show_mouse;
     txt1:='[1][   MERLIN''S BOX   |';
     txt2:=    ' by Brian J Parry ][ OK ]';
     txt1:=concat(txt1,txt2);
     button:=do_alert(txt1,1);
     init_logic;
     init_palette;
     repeat
          hide_mouse;
          clear_screen;
          draw_board;
          show_mouse;
          skill_level:=get_skill;
          place_balls;
          play_game;
     until quit;
     set_mouse(m_arrow);
     exit_gem;
     for i:=0 to 15 do
      dummy:=s_color(i,desk_colors[i]);
  end;
end.
```

Merlin's Box — Listing 2
Assembly

```
.globl  STDRW
STDRW:  move.l  (sp)+,retadr
        move.w  (sp)+,d0
        move.l  (sp)+,a5
        asl     #2,d0
        lea     shptab,a0
        adda    d0,a0
        move.l  (a0),a4
        move.l  #0,a3
        move.l  #17,d0
loop0:  move.l  #2,d1
loop1:  move.l  (a4,a3),(a5)
        adda.l  #4,a3
        adda.l  #4,a5
        dbf     d1,loop1
        move.l  (a4,a3),(a5)
        adda.l  #4,a3
        adda.l  #148,a5
        dbf     d0,loop0
exit:   move.l  retadr,-(sp)
        rts

retadr: .dc.l   1
shptab: .dc.l   s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16
        .dc.l   s17,s18,s19,s20,s21,s22,s23,s24,s25,s26,s27

* 2 by 18 words
s1
DC.L    $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L    $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L    $00000000,$1FFFE000,$00070007,$FFFF0000
DC.L    $00000070,$1FF3E000,$00070E07,$FE7F0000
DC.L    $00000060,$1FE7E000,$00070607,$FE7F0000
DC.L    $00000050,$1FD3E000,$00070A07,$FA7F0000
DC.L    $00000008,$1FF9E000,$00071007,$F3FF0000
DC.L    $00000001,$1FFFE000,$00078007,$9FFF0000
DC.L    $00000003,$1FFFE000,$0007C007,$CFFF0000
DC.L    $00000003,$1FFFE000,$0007C007,$CFFF0000
DC.L    $00000001,$1FFFE000,$00078007,$9FFF0000
DC.L    $00000008,$1FF9E000,$00071007,$F3FF0000
DC.L    $00000050,$1FD3E000,$00070A07,$FA7F0000
DC.L    $00000060,$1FE7E000,$00070607,$FE7F0000
DC.L    $00000070,$1FF3E000,$00070E07,$FE7F0000
DC.L    $00000000,$1FFFE000,$00070007,$FFFF0000
DC.L    $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L    $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s2
DC.L    $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L    $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L    $00000010,$1FF3E000,$00070007,$FFFF0000
DC.L    $00000018,$1FF9E000,$00070007,$FFFF0000
DC.L    $000003FC,$1FFCE000,$00077F87,$FF9F0000
DC.L    $015402AA,$1EAAE000,$55072A87,$2A9F0000
DC.L    $000003FC,$1FFCE000,$00077F87,$FF9F0000
DC.L    $01000298,$1E99E000,$01070287,$FE9F0000
DC.L    $00000390,$1F93E000,$00070387,$FF9F0000
DC.L    $01000280,$1E9FE000,$01070287,$FE9F0000
DC.L    $00000380,$1F9FE000,$00070387,$FF9F0000
DC.L    $01000280,$1E9FE000,$01070EE7,$FEE70000
DC.L    $00000380,$1F9FE000,$000707C7,$FFCF0000
DC.L    $01000280,$1E9FE000,$01070287,$FE9F0000
DC.L    $00000380,$1FFFE000,$00070107,$FF3F0000
DC.L    $00000000,$1FFFE000,$00070007,$FFFF0000
DC.L    $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L    $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s3
DC.L    $00070007,$00070007,$E000E000,$E000E000
DC.L    $00380038,$00380038,$1C001C00,$1C001C00
DC.L    $00C000C0,$00C000C0,$03000300,$03000300
DC.L    $01000100,$01000100,$00800080,$00800080
DC.L    $02000200,$02000200,$00400040,$00400040
DC.L    $04000400,$04000400,$00200020,$00200020
DC.L    $04000400,$04000400,$00200020,$00200020
DC.L    $08000800,$08000800,$00100010,$00100010
DC.L    $08000800,$08000800,$00100010,$00100010
DC.L    $08000800,$08000800,$00100010,$00100010
DC.L    $08000800,$08000800,$00100010,$00100010
DC.L    $04000400,$04000400,$00200020,$00200020
DC.L    $04000400,$04000400,$00200020,$00200020
DC.L    $02000200,$02000200,$00400040,$00400040
DC.L    $01000100,$01000100,$00800080,$00800080
DC.L    $00C000C0,$00C000C0,$03000300,$03000300
DC.L    $00380038,$00380038,$1C001C00,$1C001C00
DC.L     $00070007,$00070007,$E000E000,$E000E000
* 2 by 18 words
s4
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000007,$00000000,$0000E000,$00000000
DC.L     $0000003F,$00000000,$0000FC00,$00000000
DC.L     $000000FF,$00000000,$0000FF00,$00000000
DC.L     $000001FF,$00000000,$0000FF80,$00000000
DC.L     $000003FF,$00000000,$0000FFC0,$00000000
DC.L     $000003FF,$00000000,$0000FFC0,$00000000
DC.L     $000007FF,$00000000,$0000FFE0,$00000000
DC.L     $000007FF,$00000000,$0000FFE0,$00000000
DC.L     $000007FF,$00000000,$0000FFE0,$00000000
DC.L     $000007FF,$00000000,$0000FFE0,$00000000
DC.L     $000003FF,$00000000,$0000FFC0,$00000000
DC.L     $000003FF,$00000000,$0000FFC0,$00000000
DC.L     $000001FF,$00000000,$0000FF80,$00000000
DC.L     $000000FF,$00000000,$0000FF00,$00000000
DC.L     $0000003F,$00000000,$0000FC00,$00000000
DC.L     $00000007,$00000000,$0000E000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
* 2 by 18 words
s5
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $1C711C71,$1C711C71,$CF3ECF3E,$CF3ECF3E
DC.L     $228A228A,$228A228A,$28A028A0,$28A028A0
DC.L     $20822082,$20822082,$28A028A0,$28A028A0
DC.L     $20822082,$20822082,$28A028A0,$28A028A0
DC.L     $1C821C82,$1C821C82,$2F3C2F3C,$2F3C2F3C
DC.L     $02820282,$02820282,$2A202A20,$2A202A20
DC.L     $02820282,$02820282,$29202920,$29202920
DC.L     $228A228A,$228A228A,$28A028A0,$28A028A0
DC.L     $1C711C71,$1C711C71,$C8BEC8BE,$C8BEC8BE
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
* 2 by 18 words
s6
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $113E113E,$113E113E,$40F040F0,$40F040F0
DC.L     $11201120,$11201120,$40884088,$40884088
DC.L     $11201120,$11201120,$40884088,$40884088
DC.L     $11201120,$11201120,$40884088,$40884088
DC.L     $1F3C1F3C,$1F3C1F3C,$40F040F0,$40F040F0
DC.L     $11201120,$11201120,$40804080,$40804080
DC.L     $11201120,$11201120,$40804080,$40804080
DC.L     $11201120,$11201120,$40804080,$40804080
DC.L     $113E113E,$113E113E,$7C807C80,$7C807C80
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
* 2 by 18 words
s7
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $01170117,$01170117,$C880C880,$C880C880
DC.L     $01940194,$01940194,$08800880,$08800880
DC.L     $01940194,$01940194,$08800880,$08800880
DC.L     $01570157,$01570157,$88808880,$88808880
DC.L     $01340134,$01340134,$0A800A80,$0A800A80
DC.L     $01340134,$01340134,$0D800D80,$0D800D80
DC.L     $01170117,$01170117,$C880C880,$C880C880
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $0E1C0E1C,$0E1C0E1C,$44F844F8,$44F844F8
DC.L     $11221122,$11221122,$6C806C80,$6C806C80
DC.L     $10221022,$10221022,$54805480,$54805480
DC.L     $10221022,$10221022,$44804480,$44804480
DC.L     $133E133E,$133E133E,$44F044F0,$44F044F0
DC.L     $11221122,$11221122,$44804480,$44804480
DC.L     $11221122,$11221122,$44804480,$44804480
DC.L     $0F220F22,$0F220F22,$44F844F8,$44F844F8
DC.L     $00000000,$00000000,$00000000,$00000000
* 2 by 18 words
s8
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00000000,$00000000,$00000000,$00000000
```

ST-U.S.E.

THE USED PROGRAM EXCHANGE FOR YOUR ST

Trade Your Old Programs for Exciting New Titles

Buy Quality ST Programs at a Fraction of the Original price.

Over 250 Titles currently in stock
New Arrivals Daily

NOW SELLING NEW SOFTWARE

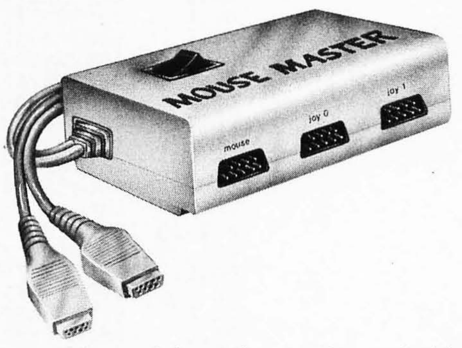
Call or write today for a free price list and membership info.

ST-USE
314 Main Street
Great Barrington, MA 01230
(413) 528-4728 9 a.m. - 5 p.m. Eastern Time
MasterCharge and VISA Accepted

CIRCLE #111 ON READER SERVICE CARD.

MOUSE MASTER

New from the people who brought you Monitor Master!



Are you tired of fumbling under or behind your computer to swap your mouse and joystick cables? Are your cable and computer connectors worn out from all the plugging and unplugging? Then Mouse Master is a must for you!

$39.95*

Practical Solutions®
1930 E. Grant Rd.,
Tucson, AZ 85719

VISA

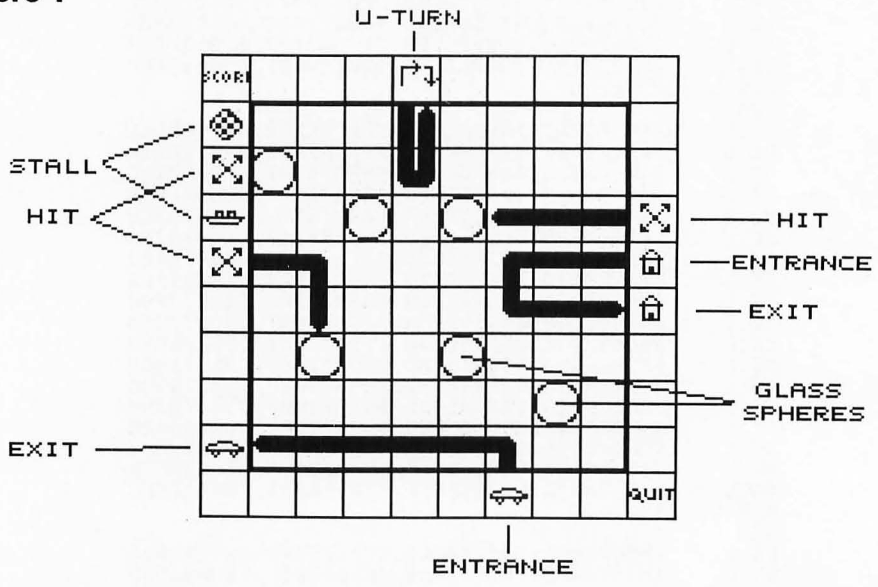
*Retail price does not include shipping & handling.

602-884-9612

CIRCLE #105 ON READER SERVICE CARD.

```
DC.L      $00000000,$00000000,$00000000,$00000000
DC.L      $00000000,$00000000,$00000000,$00000000
DC.L      $0E220E22,$0E220E22,$7CF87CF8,$7CF87CF8
DC.L      $11221122,$11221122,$10201020,$10201020
DC.L      $11221122,$11221122,$10201020,$10201020
DC.L      $11221122,$11221122,$10201020,$10201020
DC.L      $11221122,$11221122,$10201020,$10201020
DC.L      $15221522,$15221522,$10201020,$10201020
DC.L      $15221522,$15221522,$10201020,$10201020
DC.L      $13221322,$13221322,$10201020,$10201020
DC.L      $0F1E0F1E,$0F1E0F1E,$7C207C20,$7C207C20
DC.L      $00800080,$00800080,$00000000,$00000000
DC.L      $00000000,$00000000,$00000000,$00000000
DC.L      $00000000,$00000000,$00000000,$00000000
DC.L      $00000000,$00000000,$00000000,$00000000
DC.L      $00000000,$00000000,$00000000,$00000000
* 2 by 18 words
s9
DC.L      $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L      $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000E000,$00E700E7,$00E700E0
DC.L      $00000000,$0000E000,$03C703C7,$03C703C0
DC.L      $00000000,$0000E000,$07870787,$07870780
DC.L      $00000000,$0000E000,$00870087,$00870080
DC.L      $01AF01AF,$01AFE1AF,$FE07FE07,$FE07FE00
DC.L      $07570357,$0357E357,$FE67FE97,$FE97FE00
DC.L      $07AF03AF,$03AFE3AF,$FE47FEA7,$FEA7FE00
DC.L      $01570157,$0157E157,$FE07FE07,$FE07FE00
DC.L      $00000000,$0000E000,$00870087,$00870080
DC.L      $00000000,$0000E000,$07870787,$07870780
DC.L      $00000000,$0000E000,$03C703C7,$03C703C0
DC.L      $00000000,$0000E000,$00E700E7,$00E700E0
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L      $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s10
DC.L      $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L      $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $0000000C,$000CE000,$00070007,$00070000
DC.L      $00000004,$0004E000,$00070007,$00070000
DC.L      $0000000E,$0004E00A,$00070007,$00070000
DC.L      $0000000F,$0000E00F,$00070007,$00070000
DC.L      $0000000E,$0000E00E,$00070007,$00070000
DC.L      $000007FF,$0000E7FF,$0007F007,$0007F000
DC.L      $00500FAF,$0000EFAF,$0007FF07,$0007FF00
DC.L      $00280CD7,$0000EFD7,$0007FFF7,$0007FFF0
DC.L      $000007FF,$0000E7FF,$0007FF57,$0007FF50
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L      $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s11
DC.L      $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L      $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $000F000F,$000FE00F,$C007C007,$C007C000
DC.L      $00330033,$0033E033,$30073007,$30073000
DC.L      $004C004C,$004CE04C,$C807C807,$C807C800
DC.L      $00CC00CC,$00CCE0CC,$CC07CC07,$CC07CC00
DC.L      $00B300B3,$00B3E0B3,$34073407,$34073400
DC.L      $00B300B3,$00B3E0B3,$34073407,$34073400
DC.L      $00CC00CC,$00CCE0CC,$CC07CC07,$CC07CC00
DC.L      $00CC00CC,$00CCE0CC,$CC07CC07,$CC07CC00
DC.L      $00730073,$0073E073,$38073807,$38073800
DC.L      $00330033,$0033E033,$30073007,$30073000
DC.L      $000F000F,$000FE00F,$C007C007,$C007C000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L      $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s12
DC.L      $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L      $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L      $00000000,$0000E000,$00070007,$00070000
DC.L      $00010001,$0003E001,$80078007,$C0078000
DC.L      $00030003,$001CE003,$00070007,$F8070000
DC.L      $000F000F,$0070E00F,$00070007,$FE070000
DC.L      $001E001E,$00E1E01E,$07070707,$F8070700
DC.L     $00050005,$01FAE005,$1F871F87,$E0071F80
DC.L     $00080008,$01F7E008,$07070707,$F8870700
DC.L     $015E015E,$00A1E15E,$1C071C07,$E3871C00
DC.L     $008E008E,$0171E08E,$3C073C07,$C3873C00
DC.L     $00040004,$00FBE004,$38073807,$C7073800
DC.L     $00040004,$007BE004,$08070807,$F6070800
DC.L     $00000000,$001FE000,$00070007,$F8070000
DC.L     $00010001,$0003E001,$80078007,$C0078000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s13
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $0FFE0FFE,$0FFEEFFE,$7FF77FF7,$7FF77FF0
DC.L     $0FFE0FFE,$0FFEEFFE,$7FF77FF7,$7FF77FF0
DC.L     $0FC60FC6,$0FC6EFC6,$47174717,$47174710
DC.L     $0FC60FC6,$0FC6EFC6,$47174717,$47174710
DC.L     $0FFE0FFE,$0FFEEFFE,$7FF77FF7,$7FF77FF0
DC.L     $0F1E0F1E,$0F1EEF1E,$78F778F7,$78F778F0
DC.L     $0F1E0F1E,$0F1EEF1E,$78F778F7,$78F778F0
DC.L     $0FFE0FFE,$0FFEEFFE,$7FF77FF7,$7FF77FF0
DC.L     $0C7E0C7E,$0C7EEC7E,$47174717,$47174710
DC.L     $0C7E0C7E,$0C7EEC7E,$47174717,$47174710
DC.L     $0FFE0FFE,$0FFEEFFE,$7FF77FF7,$7FF77FF0
DC.L     $0FFE0FFE,$0FFEEFFE,$7FF77FF7,$7FF77FF0
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s14
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $0FFE0FFE,$0FFEEFFE,$3FF73FF7,$3FF73FF0
DC.L     $0DFA0DFA,$0DFAEDFA,$3FF737B7,$37B737B0
DC.L     $08FE08FE,$08FEE8FE,$3FF72317,$23172310
DC.L     $0AFE0AFE,$0AFEEAFE,$3FF737B7,$37B737B0
DC.L     $0FFE0FFE,$0FFEEFFE,$3FF73FF7,$3FF73FF0
DC.L     $0FBE0FBE,$0FBEEFBE,$3FF73EF7,$3EF73EF0
DC.L     $0F1E0F1E,$0F1EEF1E,$3FF73C77,$3C773C70
DC.L     $0F5E0F5E,$0F5EEF5E,$3FF73EF7,$3EF73EF0
DC.L     $0FF60FF6,$0FF6EFF6,$3FF737B7,$37B737B0
DC.L     $0FE20FE2,$0FE2EFE2,$3FF72317,$23172310
DC.L     $0BEA0BEA,$0BEAEBEA,$3FF737B7,$37B737B0
DC.L     $0FFE0FFE,$0FFEEFFE,$3FF73FF7,$3FF73FF0
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s15
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$000FE000,$0007C007,$C0070000
DC.L     $000E000E,$001FE00E,$40074007,$C0074000
DC.L     $001E001E,$003FE01E,$40074007,$C0074000
DC.L     $04000400,$07FFE400,$00070007,$FFF70000
DC.L     $00000000,$07FFE000,$00070007,$FFF70000
DC.L     $00000000,$078FE000,$00070007,$F1F70000
DC.L     $02730070,$0070E070,$8E470E07,$0E070E00
DC.L     $018C0088,$0088E088,$71871107,$11071100
DC.L     $008800A8,$00A8E088,$11071507,$15071100
DC.L     $00880088,$0088E088,$11071107,$11071100
DC.L     $00700070,$0070E070,$0E070E07,$0E070E00
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s16
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00080008,$0008E008,$10071007,$10071000
DC.L     $00040004,$0004E004,$20072007,$20072000
DC.L     $00020002,$0003E002,$40074007,$C0074000
DC.L     $00000000,$0003E000,$00070007,$C0070000
DC.L     $00FF0000,$00FFE0FF,$FF070007,$FF07FF00
DC.L     $00FF001F,$00E0E0FF,$FF07F807,$0707FF00
DC.L     $00E00020,$00C0E0E0,$07070407,$03070700
DC.L     $00C40044,$0084E0C4,$8B078A07,$89078B00
DC.L     $00C90049,$0089E0C9,$13071207,$11071300
DC.L     $00D20052,$0092E0D2,$23072207,$21072300
DC.L     $00E00020,$00C0E0E0,$07070407,$03070700
DC.L     $00FF005F,$00E0E0FF,$FF07FA07,$0707FF00
DC.L     $00FF0000,$00FFE0FF,$FF070007,$FF07FF00
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s17
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $01C00000,$01C0E000,$03870007,$03870000
DC.L     $03F00000,$03F0E000,$0FC70007,$0FC70000
DC.L     $07FC0000,$07FCE000,$3FE70007,$3FE70000
DC.L     $07FE0000,$07FEE000,$7FE70007,$7FE70000
DC.L     $0FFF0001,$0FFFE001,$FFF78007,$FFF78000
DC.L     $0FFF0002,$0FFFE002,$FFF74007,$FFF74000
DC.L     $0FFF0002,$0FFFE002,$FFF74007,$FFF74000
DC.L     $0FFF0001,$0FFFE001,$FFF78007,$FFF78000
DC.L     $07FE0000,$07FEE000,$7FE70007,$7FE70000
DC.L     $07FC0000,$07FCE000,$3FE70007,$3FE70000
DC.L     $03F00000,$03F0E000,$0FC70007,$0FC70000
DC.L     $01C00000,$01C0E000,$03870007,$03870000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s18
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $02440244,$0244E244,$00070007,$00070000
DC.L     $02240224,$0224E224,$01E701E7,$000701E0
DC.L     $01480148,$0148E148,$07E707E7,$000707E0
DC.L     $00000000,$0000E000,$0E070E07,$00070E00
DC.L     $02100210,$01E0E3F0,$38073807,$00073800
DC.L     $07F807F8,$0000E7F8,$70077007,$00077000
DC.L     $07F907F9,$0000E7F9,$C007C007,$0007C000
DC.L     $07FF07FF,$0000E7FF,$80078007,$00078000
DC.L     $07FF07FF,$0000E7FF,$00070007,$00070000
DC.L     $07FC07FC,$0000E7FC,$00070007,$00070000
DC.L     $03F003F0,$0000E3F0,$00070007,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s19
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00600000,$0000E000,$00070007,$00070000
DC.L     $00600060,$0060E060,$00070007,$00070000
DC.L     $01F30003,$0003E003,$FFF7FFF7,$FFF7FFF0
DC.L     $023A0002,$0002E002,$AAA7AAA7,$AAA7AAA0
DC.L     $023F0000,$0000E000,$FFE70007,$00070000
DC.L     $03FF0000,$0000E000,$FFE70007,$00070000
DC.L     $03FA0000,$0000E000,$BFA70007,$00070000
DC.L     $07FF04E0,$04E0E4E0,$FFF70737,$07370730
DC.L     $071F0517,$05F7E5F7,$F8F7E8B7,$EFB7EFB0
DC.L     $01100110,$01F0E1F0,$08870887,$0F870F80
DC.L     $00E000E0,$00E0E0E0,$07070707,$07070700
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s20
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00030003,$0003E003,$E007E007,$E007E000
DC.L     $00040004,$0004E004,$10071007,$10071000
DC.L     $0000000F,$0000E000,$0007F807,$00070000
DC.L     $0000003F,$0000E000,$0007FE07,$00070000
DC.L     $000000FF,$0000E000,$0007FF87,$00070000
DC.L     $002A01EA,$002AE02A,$AA07AAC7,$AA07AA00
DC.L     $000000FF,$0000E000,$0007FF87,$00070000
DC.L     $0000003F,$0000E000,$0007FE07,$00070000
DC.L     $0000000F,$0000E000,$00077807,$00070000
DC.L     $00000020,$0000E000,$00078207,$00070000
DC.L     $00000040,$0000E000,$00078107,$00070000
DC.L     $000001C1,$0000E000,$0007C1C7,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s21
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000007,$0007E000,$0007E007,$E0070000
DC.L     $0000003F,$003FE000,$0007FC07,$FC070000
DC.L     $000000FF,$00FFE000,$0007FF07,$FF070000
DC.L     $000001F9,$01F9E000,$00079F87,$9F870000
DC.L     $000003F9,$03F9E000,$00079FC7,$9FC70000
DC.L     $000003FF,$03FFE000,$0007FFC7,$FFC70000
DC.L     $000003DF,$03DFE000,$0007FBC7,$FBC70000
DC.L     $0000039F,$039FE000,$0007F9C7,$F9C70000
DC.L     $000001E7,$01E7E000,$0007E787,$E7870000
DC.L     $000000F8,$00F8E000,$00071F07,$1F070000
DC.L     $0000003F,$003FE000,$0007FC07,$FC070000
DC.L     $00000007,$0007E000,$0007E007,$E0070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s22
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00010001,$0001E001,$80078007,$80078000
DC.L     $00070007,$0006E007,$E007E007,$6007E000
DC.L     $001F001F,$0018E01F,$F807F807,$1807F800
DC.L     $007F007F,$0060E07F,$FE07FE07,$0607FE00
DC.L     $01FF01FF,$01FFE1FF,$FF87FF87,$FF87FF80
DC.L     $007F0040,$0040E040,$FE070207,$02070200
DC.L     $0063005C,$0040E05C,$FE070207,$02070200
DC.L     $0063005C,$0040E05C,$FE07FA07,$FA07FA00
DC.L     $0063005C,$0040E05C,$FE07FA07,$8A07FA00
DC.L     $007F0040,$0040E040,$FE07FA07,$8A07FA00
DC.L     $007F0040,$0040E040,$FE07FA07,$8A07FA00
DC.L     $007F007F,$007FE07F,$FE07FE07,$FE07FE00
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s23
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00C00000,$00C0E000,$00070C07,$0C070000
DC.L     $01A00080,$01A0E080,$08071207,$12070000
DC.L     $01200000,$0120E000,$00071207,$12070000
DC.L     $00D80000,$00D8E000,$0007CD87,$CD870000
DC.L     $03340051,$0335E010,$01072A47,$22470000
DC.L     $04A40041,$04A5E000,$40072A47,$22470000
DC.L     $05980140,$0598E100,$0007C987,$C1870000
DC.L     $03000044,$0300E000,$00078507,$00070000
DC.L     $00000324,$0000E000,$00076607,$00070000
DC.L     $00000130,$0000E000,$00073407,$00070000
DC.L     $00000190,$0000E000,$0007E8C7,$00070000
DC.L     $0000033C,$0000E000,$0007D577,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s24
DC.L     $3FFF3FFF,$3FFF8000,$FFFDFFFD,$FFFD0000
DC.L     $1FFF1FFF,$1FFFC000,$FFFBFFFB,$FFFB0000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $01C001C0,$01C0E1C0,$03C70007,$03C703C0
DC.L     $03E002E0,$02E0E2E0,$07E701E7,$060707E0
DC.L     $03E00300,$0300E300,$0FE707E7,$08070FE0
DC.L     $03E001E0,$01E0E1E0,$1FC70FC7,$10071FC0
DC.L     $01C000C0,$00C0E0C0,$3F071F07,$20073F00
DC.L     $00000000,$0000E000,$7C073C07,$40077C00
DC.L     $00000000,$0000E000,$F0077007,$8007F000
DC.L     $00010000,$0001E001,$C007C007,$0007C000
DC.L     $00070003,$0004E007,$00070007,$00070000
DC.L     $001E0006,$0018E01E,$00070007,$00070000
DC.L     $000C000C,$0000E00C,$00070007,$00070000
DC.L     $00000000,$0000E000,$00070007,$00070000
DC.L     $00000000,$0000DFFF,$00030003,$0003FFF8
DC.L     $00000000,$0000BFFF,$00010001,$0001FFFC
* 2 by 18 words
s25
DC.L     $00000000,$00000000,$00000000,$00000000
DC.L     $00070000,$00000000,$E0000000,$00000000
DC.L    $003F0000,$00000000,$FC000000,$00000000
DC.L    $00FF0000,$00000000,$FF000000,$00000000
DC.L    $01FF0000,$00000000,$FF800000,$00000000
DC.L    $03FF0000,$00000000,$FFC00000,$00000000
DC.L    $03FF0000,$00000000,$FFC00000,$00000000
DC.L    $07FF0000,$00000000,$FFE00000,$00000000
DC.L    $07FF0000,$00000000,$FFE00000,$00000000
DC.L    $07FF0000,$00000000,$FFE00000,$00000000
DC.L    $07FF0000,$00000000,$FFE00000,$00000000
DC.L    $03FF0000,$00000000,$FFE00000,$00000000
DC.L    $03FF0000,$00000000,$FFC00000,$00000000
DC.L    $01FF0000,$00000000,$FF800000,$00000000
DC.L    $00FF0000,$00000000,$FF000000,$00000000
DC.L    $003F0000,$00000000,$FC000000,$00000000
DC.L    $00070000,$00000000,$E0000000,$00000000
DC.L    $00000000,$00000000,$00000000,$00000000
* 2 by 18 words
s26
DC.L    $FFFF0000,$FFFFFFFF,$FFFF0000,$FFFFFFFF
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $80000000,$BCF3BCF3,$00010000,$CF3DCF3D
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $80000000,$BCF3BCF3,$00010000,$CF3DCF3D
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $9C711C71,$A082BCF3,$C71DC71C,$0821CF3D
DC.L    $80000000,$BCF3BCF3,$00010000,$CF3DCF3D
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $FFFF0000,$FFFFFFFF,$FFFF0000,$FFFFFFFF
* 2 by 18 words
s27
DC.L    $FFFF0000,$FFFFFFFF,$FFFF0000,$FFFFFFFF
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $80000000,$80008000,$00010000,$00010001
DC.L    $FFFF0000,$FFFFFFFF,$FFFF0000,$FFFFFFFF
```

figure 1



ST USER

by Arthur Leyenberger

Arthur Leyenberger is a human factors psychologist and freelance writer living in New Jersey. He has written over 100 articles about computers in the last four years and continues to be an Atari enthusiast. When not computing he enjoys playing with robotic toys.

As a heavy PC user I am familiar with the major word processors for the MS-DOS computers such as *Wordstar* (my favorite by habit), *Microsoft Word, WordPerfect* and others. These programs typically cost hundreds of dollars, have documentation that could strain a pack mule and offer complete word-processing power that is primarily meant for office use. Not since *Final Word* by Mark of the Unicorn have I seen a seemingly "professional" word processor for the ST. As you may already know, Final Word, which debuted soon after the release of the ST, was less than spectacular. It was clumsy to use, error prone, had copy protection and couldn't produce the normal everyday types of hard copy that people need to do.

For the past year or so there have been three ST word processors that perform about equally and seem to be the most popular with knowledgeable users. They are *Regent Word II* by Regent Software, *Word Writer ST* by Timeworks and *1st Word*, the freebie that came with almost everybody's ST. Despite the selection of three (and possibly more) good word processors there still has not appeared on the market what I call a "Macintosh-class" word processor for the ST. By that I mean a word processor with multiple fonts (more than just bold, italic and super/subscript), on-screen formatting, professional quality features such as footnoting, endnoting, top-notch spelling checking and more. The ST computer is a mature, three-year-old product. So where is the mature word processor for the machine?

The most recent entrant in the ST word processing sweepstakes is WordPerfect 4.1 for the ST. It originally debuted in the late fall of 1987 but the company itself admits the first release was somewhat buggy. Since then WordPerfect has cleaned up the bugs in the program and provided several updated releases to registered users. The version of WordPerfect that you buy today is virtually bug-free. However, the question still remains—is WordPerfect ST the ultimate word processor for the ST?

I don't have room in this space for a complete review of WordPerfect. Moreover, I cannot do a thorough comparison of all of the ST word processors. What I *can* do is give an overview of the program as I am actually using it now to write this month's column.

A First Look

What is WordPerfect? In summary, it is a *very* complete word processor. The ST version which I am discussing, is file-compatible with WordPerfect 4.1 for the IBM PC and other computers, allowing for direct document transfer to and from the ST without losing document format. The program offers a plethora of features such as footnotes, endnotes, macros, spelling checker, thesaurus, sorting, paragraph outlining/numbering, a math mode for columnar numeric tables, indexing and table of contents generation. It uses a full GEM interface complete with drop down menus, multiple windows, mouse or keyboard control and access to any currently active Desktop Accessories.

The WordPerfect package comes with six disks, a 600-page manual housed in a slip case, a multi-colored keyboard template (containing WordPerfect's menu) that is placed above the ST's function keys, a quick reference card and a toll-free number for support (available once the registration card is returned). The whole kit and caboodle costs $395 list, which seems steep compared to other "serious" software for the ST. Although the package appears professional and complete, one might legitimately ask if the program is worth almost as much as the price of a monochrome ST system?

I'll attempt to answer that by means of a guided tour of the program and its features. Starting with installation, setup and on to printing, we'll have a look together, at what might or might not be the best word processor for the ST currently available.

The Tour

Well, it only took me about an hour to get WordPerfect installed to the point where I could actually print the first few paragraphs above. Like most computer users, I was eager to get the program going and see some output. I read and reread the documentation, used what I consider to be common sense in these matters and it still wasn't as easy as it should have been. I don't think I like what I've seen so far, but I am willing to hold off final judgment of the program until I put it through its paces.

First, I need to tell you the configuration of my system so that you can follow along and, if you have the same setup, learn from my experiences. I have a 520ST with one megabyte of memory, a Supra 20 megabyte hard disk, one double-sided disk drive and an Epson FX-80 printer. I have been using a monochrome monitor but will also try the color monitor as well. I also have a 5¼ inch, IBM-type disk drive as drive B, but I decided to disconnect it from the system for purposes of this evaluation. Things often get confusing enough when you are using a program for the first time, and I wanted to keep the system as "stock" as possible so as to prevent any additional problems.

The instructions told me to create a separate folder on my hard disk and then to copy the program, spelling, thesaurus and learn files (each on their own disk) to that folder. I attempted to install my printer and ran into several problems. There are two additional disks, labeled Print and Font, and the instructions said to insert the Print disk into drive A when prompted to insert disk in drive B. This repeatedly did not work, and it was only until I copied the entire contents of the Print disk to the "WP" folder on the hard disk that I was able to proceed beyond that point in the installation process. I noticed that during the printer setup, there was one extra step on the screen that the manual did not mention. You can install up to six printer definitions, which I guess might be useful to some people, but I think the screen prompt was asking if I wanted to save all printer definitions, rather than just the one I chose, to the hard disk.

There was nothing in the instructions concerning the Font disk, and I was unable to print anything at first. In fact, every time I tried to print, the program told me it couldn't find the printer file and bombed. After rebooting and going through these shenanigans a couple of times I copied the *entire* contents of the Font disk to the hard disk folder and went back to re-install. During the course of installing the printer, I noticed errors in the manual. For example, Page 548 says "select printer controls (4)" meaning press the number "4" on the keyboard or mouse click on the fourth option in the menu list.

As it turns out, there were only three options presented, and pressing "4" had no effect. This error, the extra step mentioned above and the confusing/incorrect procedure for reading the files from the Print and Font disks suggest that the program changed after the documentation was written. I can understand this, but the normal practice, especially for programs costing this much and purporting to be "professional," is to include an errata sheet listing the errors or additional steps missing from the documentation. There was no errata sheet included with the WordPerfect package.

Dislikes

Let me mention some other things I disliked about the program. When you want to search for a particular text string you must enter the string and then use the mouse to click on OK. There is no default button that normally is selected when pressing the return key, as with many GEM applications. In fact, hitting RETURN causes the ASCII code for the return key to be entered as the search string. The only time using the mouse to click on OK is not awkward is when you do a subsequent search. Then, you can use the mouse to click on the Search menu, choose forward or reverse and then, since the previous string is displayed, just click on OK.

When you use the copy/paste function and the program accesses the file, there is no "busy bee" icon to let you know the machine is busy. The keyboard is locked out, although there is no indication of it. Whenever the program has to access the disk drive (hard disk in my system), access time is extremely slow. This may be as much the fault of the ST as with the program. Also, retrieving a file is not as straightforward as it could be since you must first choose "List Files," click on the filename and then click on "Retrieve." The file selection procedure done with the familiar GEM file list box seems easier.

Here are a couple of snipits. The template that comes with the package is a good idea, but its execution could be better. The plasticized cardboard slips in behind the function keys and there is a small piece that extends to the left which labels the alternate, control and shift key's functions on the template. Since the left side of the template extends beyond the slot, it holds the template in place so that the four rows of functions can be easily read. However, the right side of the template does not extend past the slot, so the template falls down into the slot, making the bottom two rows of the labels on the three right-most function keys difficult to read.

Another minor problem has to do with the design of the documentation binder. There are no pouches on the inside covers of the binder to hold the quick reference guide and other materials. Further, there are no disk holder pages to house the six floppies that come with the program. Granted, these are minor points, but in an otherwise professionally looking package they stand out as being incomplete.

Likes

Although I have some minor complaints about the program, I don't want to give the impression that I hated the program, because that's not true. There is no question that the program is feature-laden, and generally the multitude of features do not get in the way of the primary purpose of the program. Here are some of the things I *did* like during my 20 or so hours of concentrated use. The automatic backup is a useful feature, especially since you can set the frequency at which the backups will occur. I started out with a five-minute backup interval, but it was just too often and soon became annoying. I then tried ten minutes and eventually settled on 15 minutes. When the automatic backup occurs, a small window appears announcing "backup in progress." You do not lose any keystrokes, although when the window appears the cursor freezes where it currently is.

One feature I liked was the Date function. From the Format menu, selecting date will put the current system date wherever the cursor happens to be. Once the date is inserted into the text, it becomes text and will not change. Another feature I liked was the Menu Bar Lock. When this function is enabled it prevents the menus from automatically dropping down from the Menu Bar when the cursor touches one. Instead, the right mouse button is used to enable the drop down menu and the cursor arrow does not have to be near the menu name. Wherever the arrow is horizontally across the screen, will cause the menu directly above it to drop down when the right button is pressed.

Still another feature I liked was the List Files function. When requested, List Files will display files in the current or another folder and allow you to retrieve, delete, rename, print or copy files. In addition to acting on individual files, you can tag them for mass execution, such as when you want to delete a group of files or copy several at a time. Other functions in List Files include searching one or more files for specific words or phrases and displaying the contents of a file on the screen. Having these file management commands available from within a word processor program is quite handy.

WordPerfect has many other features to like as well. Earlier, I mentioned that the ST user is still awaiting a Macintosh-like word processor and wondered if WordPerfect was the product we have all been waiting for. I'm sorry to say it is not. Although it is a what-you-see-is-what-you-get type of word processor, it does not have multiple on-screen fonts in varying sizes, is not as "classy" as MacWrite and contains too many unpolished aspects to be in that class. On the other hand, in terms of features, a street price of about $250 and WordPerfect's support of Atari users, there is no other ST word processor that can match it.

The End

Sometimes a review reveals as much about the company that created the product as it does about the particular hardware or software under review. This has certainly been true with WordPerfect Corp. Their toll-free telephone support lines, free revisions of the program to registered owners and general attitude show that they are a first-rate company that does business the way it ought to be done. If you need the best currently available word processor for your ST, the only choice is WordPerfect.

WordPerfect
WordPerfect Corp.
288 West Center Street
Orem, Utah 84057
(801) 225-5000
$395.00

RE
VIEW

TEMPUS The Editor

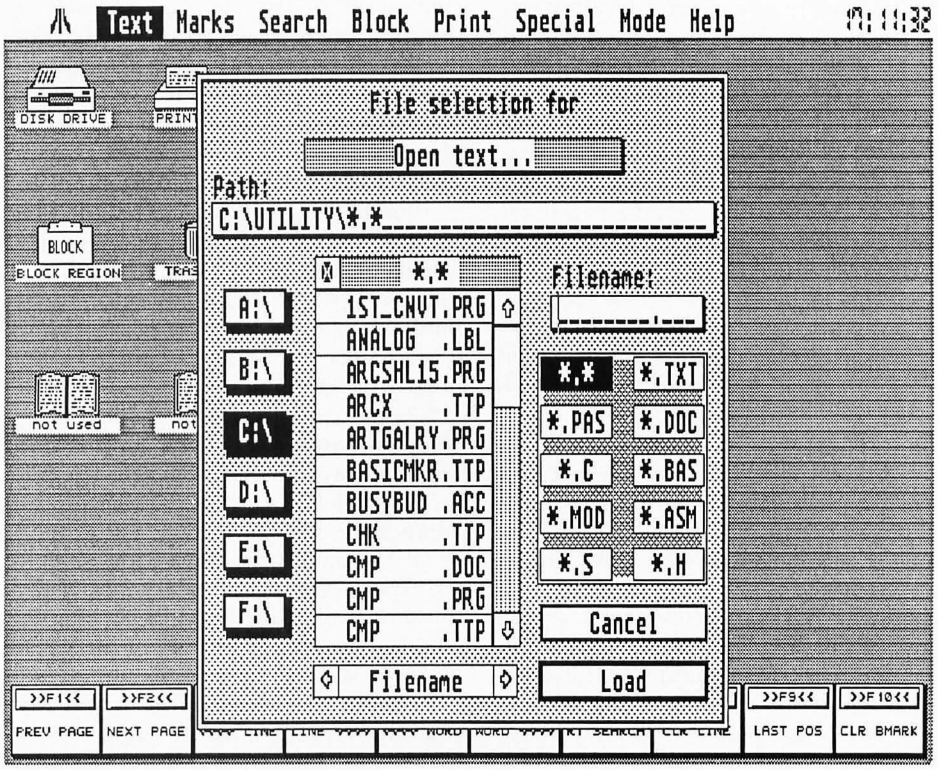
**Creative Computer Design/
Dirk Beyelstein
Eltville, West Germany
Color or Monochrome.
Distributed in the USA by:
Eidersoft USA
P.O. Box 288
Burgettstown, PA 15021
(800) 648-9191
$42.99**

Reviewed by Charles F. Johnson

There's a good deal of ground-breaking new ST software being written in Europe—West Germany, England and Hungary, just to name a few countries. This makes perfect sense when you consider that the ST was officially released in Europe before the USA, and that European programmers had already gotten a head start programming the 68000 chip contained in the ST when Sinclair released its ill-fated, 68000-based QL computer several years ago. The QL vanished into the mists of computer history, but the ST has been a success story since it's release, especially in Europe and the UK. Some of these innovative European programs are beginning to appear in the USA at last, and Eidersoft is responsible for marketing many of them.

One of these is *Tempus*, a product of Creative Computer Design in Eltville, West Germany. First things first—what Tempus isn't: It isn't a word processor. It lacks some of the necessary functions to be a full-fledged word processor, such as "word-wrap." Each line must be entered separately, and ended with a RETURN, just as if you were using a typewriter. With most word processors, you just keep typing without hitting RETURN; the program takes care of keeping your text nicely formatted, on the screen and on the printer. Also, Tempus' printer output is rudimentary at best; although you can create a printer configuration file, there just aren't as many options as a true word processor would normally allow. For example, there is no explicit support for boldface type, italics or different character sizes, although you can imbed printer control codes in your text to achieve these effects.

Now that we know what it isn't, here's what Tempus is—*fast*! The program employs a full GEM interface, the equal of *1st Word*'s (actually, better), but this baby



```
1ST_CNVT.PRG
ANALOG  .LBL
ARCSHL15.PRG
ARCX    .TTP
ARTGALRY.PRG
BASICMKR.TTP
BUSYBUD .ACC
CHK     .TTP
CMP     .DOC
CMP     .PRG
CMP     .TTP
```

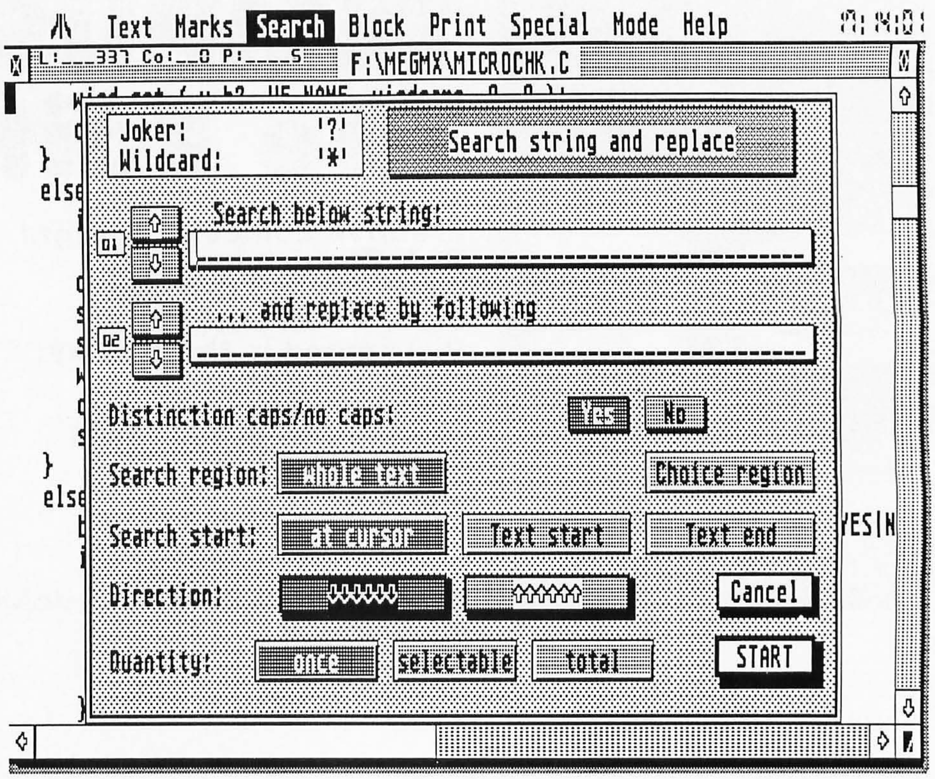
really *flies*! Quite simply, Tempus is the fastest text editor I've ever seen, far faster than any current ST text editor or word processor. It's text output keeps up with the most talented touch-typist with ease. Scrolling through a document is so quick, either with the mouse or with the function keys, that you have to watch very carefully or you'll overshoot. But Tempus is not only fast at displaying the text on-screen—it also is amazingly speedy at loading files, and at searching for words or strings of text.

Tempus is designed primarily for use by programmers (in any language), and has many features specifically aimed at making editing source code easier. It is fully compatible with the *Megamax C* and *Personal Pascal* development shells, and should work with most other shells and CLI interfaces that allow the running of GEM programs. According to the manual, CCD spent over a year developing the program, which is written entirely in assembly language for maximum performance and compactness.

Tempus has lots of other nice features besides its speed, too. You can load up to four documents at once, each in its own GEM (actually, pseudo-GEM) window, and move the windows, resize them or close them. The program can load fonts created with the DEGAS font editor for screen display (not for the printer, unfortunately). You can set up to five "marks" in a document and instantly jump to any of them with a mouse click. You can define up to 20 function key macros, for commonly typed words and strings of text, and save them to disk so they'll load automatically when you run Tempus. Tab spacing can be adjusted to suit your preferences. Any ST character (from ASCII 0 to 255) can be entered into a document. In high-resolution monochrome, each window can display either normal-sized text, or half-height text to see twice as many lines in the window.

Tempus has a comprehensive set of block functions that can be accessed with either keyboard or mouse, like block move, delete and copy. Blocks of text can also be saved to disk and merged into other documents, or sent to the printer. There is a help menu available, in case you're confused about something. And in several weeks of working with the program, I haven't managed to make it crash once (something I wish I could say about certain other ST programs which shall remain nameless).

Tempus even includes a Reverse Polish Notation calculator, which for some inex-


Text Marks Search Block Print Special Mode Help
L:___337 Co:__0 P:____5 F:\MEGMX\MICROCHK.C
Joker: '?'
Wildcard: '*'
Search string and replace
Search below string:
... and replace by following
Distinction caps/no caps: Yes No
Search region: Whole text Choice region
Search start: at cursor Text start Text end
Direction: Cancel
Quantity: once selectable total START

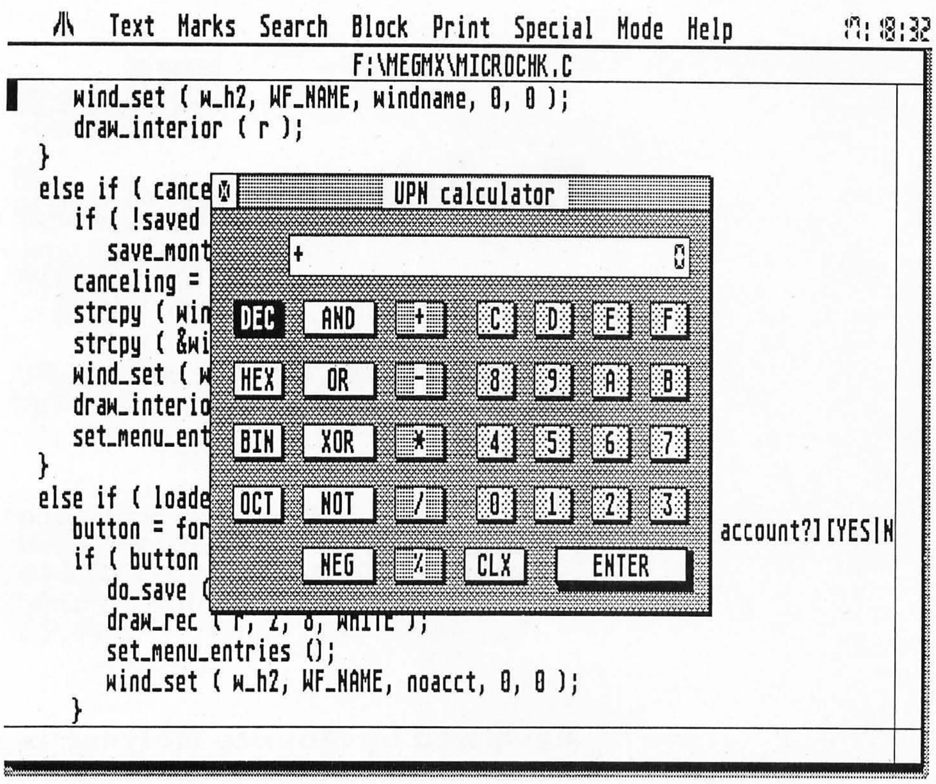
Tempus uses the GEM metaphor to

plicable reason is called a "UPN Calculator" in the program, even though the manual has the initials correct. The calculator can be very helpful, precisely at the time when you need it most. . .during text editing. However, the Tempus calculator is not the best one I've seen for the ST; the screen display bears no resemblance to the arrangement of the keys on the ST's numeric keypad. Also, the RPN method may be confusing to people who are accustomed to the traditional pocket calculator. Fortunately, you can easily install one of the many commercial or public-domain desk-accessory calculators if you wish; desk accessories are fully supported by Tempus.

When loading or saving documents, Tempus uses a customized file selector, not the normal GEM item selector that we've all come to know and love/hate. The Tempus file selector has every feature of its GEM counterpart, along with some extra capabilities that make it much easier to use. Drives are selected by clicking on a button, instead of by editing the directory line. Much more information is available about the files displayed; you can show the size and time/date stamp of each file as well as its name. A list of standard filename extensions is provided as well, such as .BAS, .S, .ASM, .C, .PAS, etc.

Tempus doesn't use real GEM windows, either, although they look and behave almost exactly alike. As with the file selector, Tempus windows are fully compatible with GEM windows. . . but with some improvements. The slider bars and scroll arrows are auto-repeating; in other words, you may click and hold the mouse button on them to scroll horizontally or vertically through the text without repeated button presses. The top line of each window displays the name of its associated document, just as in 1st Word or any other GEM editor, but it also can display information about the current cursor location. And of course, everything about the Tempus windows works at an order of magnitude faster than standard GEM windows. In fact, when you exit Tempus back to the GEM desktop, the feeling is somewhat akin to stepping off a fast escalator. It will give you an appreciation for how slow GEM really is, and how quick it *could* be.

The search routines are very flexible, allowing backward or forward searches with the option to ignore upper/lowercase differences. Up to 20 strings may be en-



```
F:\MEGMX\MICROCHK.C
      wind_set ( w_h2, WF_NAME, windname, 0, 0 );
      draw_interior ( r );
   }
   else if ( cance
      if ( !saved
         save_mont
      canceling =
      strcpy ( win
      strcpy ( &wi
      wind_set ( w
      draw_interio
      set_menu_ent
   }
   else if ( loade
      button = for                                          account?][YES|N
      if ( button
         do_save (
         draw_rec ( r, 2, 0, WHITE );
         set_menu_entries ();
         wind_set ( w_h2, WF_NAME, noacct, 0, 0 );
      }
```

ts fullest, without sacrificing speed.

tered and remembered by the program for subsequent searches. Searches can be limited to a specified region or include the whole text. Search strings can include wildcards that stand for any character. And did I mention that the search routines are *fast*? The only deficiency in Tempus' search capabilities is that it can't search for a string of text that extends across more than one line.

One of Tempus' nicest features is the ability to create a cross-reference list from a group of strings. The strings are entered into the Search dialog box, with up to 20 allowed. When you choose Cross Reference List from the drop-down menu, Tempus will generate a file which includes the line numbers where each string appears in your document, a very handy tool for programmers. But Tempus takes this concept a step farther. The cross-reference list can be loaded into its own window. Then, if you make the document window active and the cross-reference window inactive, you can click with the right mouse button on the line numbers in the list, and the cursor in the document window will instantly jump to that line, ready to start making changes! This is an extremely well-thought-out feature, and can be especially useful to programmers at those times when their compiler/assembler spits out a list of errors categorized by line number.

All Tempus commands can be entered either with the mouse or the keyboard. A slight drawback is that the key combinations used for the commands are rather cryptic and arbitrary. For example, CTRL-C to page down, CTRL-L to search, or CTRL-Y to delete a line. This is a minor point; it's just a matter of learning and becoming accustomed to the keys used. (Although if you've used EMACS, there's a potential "gotcha" lurking here. In EMACS, CTRL-Y is used to *restore* a deleted line!) Unfortunately, function key macros can only contain strings of text, not commands, and so this reduces their usefulness as a substitute for Tempus key commands.

The 83-page Tempus manual is translated from German, and rather poorly; mangled grammar abounds. However, the manual is professionally laid out, and it's reasonably complete except for the lack of an index. All the details of Tempus' operation are covered adequately, with illustrations to help explain the main screens and dialog boxes. The manual comes packaged in a small gray, two-ring binder.

Tempus is not copy-protected, and it may be freely copied to any disk, even a hard disk or RAM disk. Instead of physical copy-protection, Tempus makes you go through an installation process before you can use the program, by running the included INSTALL.PRG. This program asks you to input your name and address, and then (I presume) writes this info along with your disk's serial number into TEMPUS.PRG. After this is done, INSTALL.PRG wipes itself from the Tempus master disk. Although this "registration" method has obvious advantages for the user (no need to constantly keep swapping a key disk), I had a serious problem when I ran INSTALL.PRG.

I followed the instructions for running INSTALL.PRG scrupulously, finished entering my name and address, and clicked on the box labelled "Input Done." The disk spun for a moment, then I exited back to the desktop. Everything seemed fine. . . INSTALL.PRG was gone from the disk, but the manual said that would happen. I ran Tempus and loaded a text file, made some changes to it (thinking, *Wow, this thing is fast!*), and then tried to save the edited file. Lo and behold, a dialog box popped up with the message, "Uninstalled demo version." Something had gone wrong in the installation process, and I ended up with a copy of Tempus that was unable to save or print files. In short, a useless copy of Tempus. And since INSTALL.PRG had deleted itself from the master disk, there was no way to try again! To make matters even worse, after attempting to save my text file with this "demo version," Tempus destroyed my original file, leaving me with a file of zero length. Luckily, I had a backup of my file; otherwise I might have been very peeved. At that point, there was no recourse except to send the disk back to Eidersoft and get a new copy, which worked fine. I have no idea what went wrong with my first attempt at installation, but you'd be well-advised to be cautious the first time you use Tempus.

To sum up, Tempus is an impressive achievement—a program which uses the GEM metaphor to its fullest, without sacrificing speed. Even running on a Mega ST4 with blitter chip enabled, no other text editor comes close to the performance of Tempus. And here's something to look forward to: The manual states that a full-featured word processor based on Tempus is currently in the works. I, for one, can't wait!

REVIEW

The Advanced OCP Art Studio

Rainbird Software
Published in the USA by Firebird
P.O. Box 49, Ramsey, NJ 07446
Low resolution only
$39.95

Reviewed by Maurice Molyneaux

ST "paint" programs are not an endangered species; one might be tempted to say we have an overabundance of them. There's NEO-Chrome, DEGAS, DEGAS Elite, PaintWorks, Art Director, Spectrum 512 and the upcoming (may be out by now) Quantum Paintbox, etc. In the midst of all this, Rainbird—a company well-known to ST users for its excellent games—drops into the market with its own paint software: *The Advanced OCP Art Studio* (OCP stands for "Oxford Computer Publishing," the concept on which Art Studio is based).

At first glance the program is impressive. A lot of attention was lavished on the look of the user interface. The icons on the "control screen" are nicely shaded and drawn. When you point at an icon, instructions appear in the VIEW WINDOW (which otherwise shows a real-size image of the object currently on the editing "grid"), making each icon's function easier to decipher. When you run the program, you are asked if you have the English or German manual. Further, once inside the program you can (at least from the English mode) toggle the icon descriptions to be displayed in French! The program works in low resolution only, allowing you 16 colors for your graphics. You can have two workscreens on a 520ST, and up to ten on a 1-meg machine (no idea of the limit on a Mega!).

The program treats special graphics blocks as "brushes" and "sprites," which can be created on the control screen's grid, or can be "cut" from one of the work pages. Furthermore, these brushes can be placed in a large "store," which itself can be saved and loaded. The store permits instant access to numerous graphic elements, sparing you from having to cut and paste chunks from one screen to another. The items in store can be rearranged, and the whole store can be viewed to see its contents.

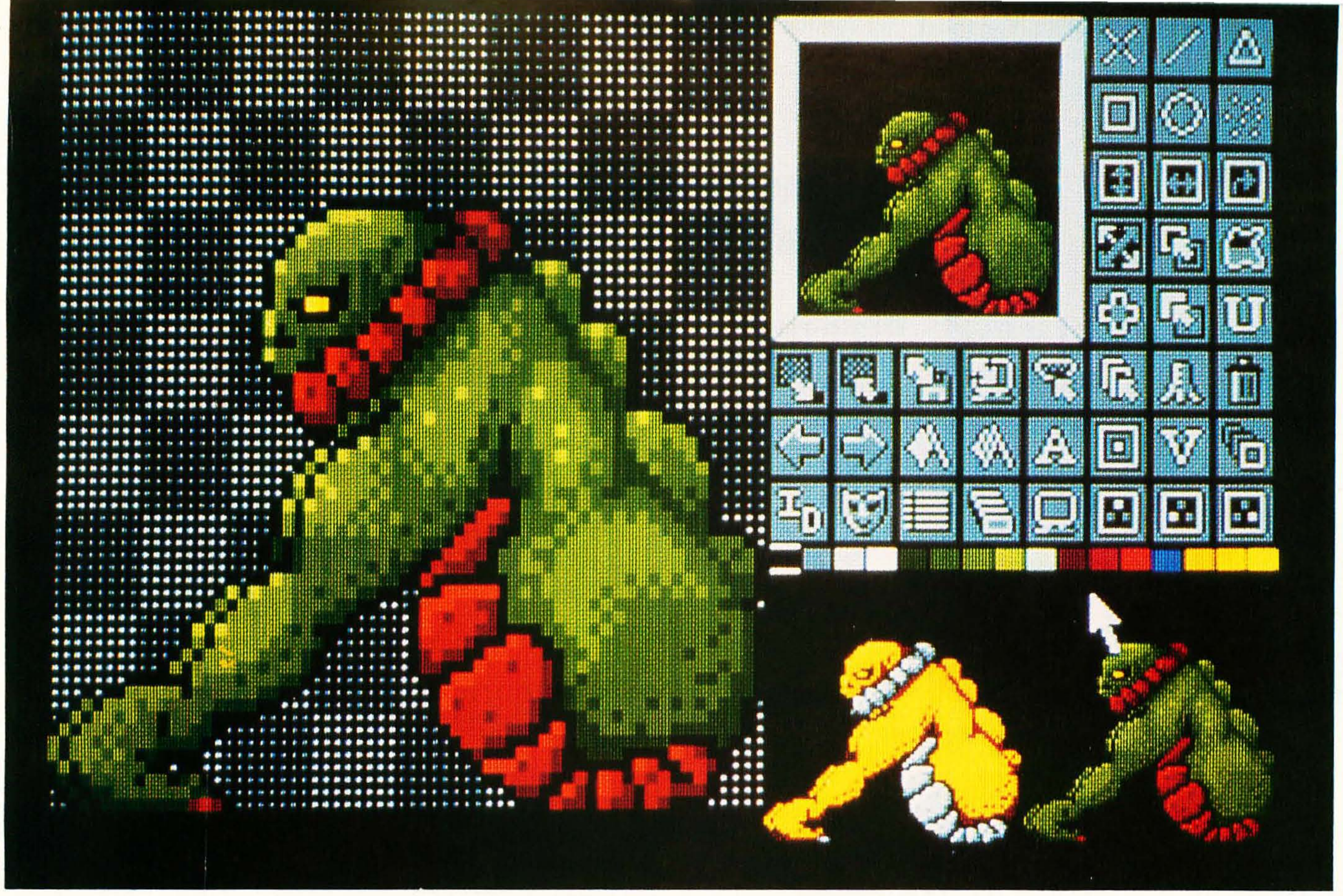
"Fonts" are merely brushes, using as many colors as you like, which are placed in the brush store, and then "typed" on the screen. However, the text does not become your paintbrush, as it does in Art Director.

Portions of the screen or grid can be flipped horizontally or vertically, or turned to the right, 90 degrees at a time. A window mode allows a rectangular area of the screen to be cleared, cut and pasted, edited in grid mode, rescaled or smoothed (anti-alias). The smooth option is quite nice, though it works well only if you have selected a number of mutually complementary colors.

Many operations can be carried out with only the currently selected "ink" color, or using all colors. The screen's border color isn't transparent as in most paint programs, but opaque. Transparent areas can be defined only on brushes and only from the grid. Additional options make it possible to replace all pixels of a given color with another color, or even to have the program auto-outline images on the grid. Pixel-by-pixel fine-tuning is possible, but the magnify mode is a little unusual. The magnified area (64 x 64 pixels) is cut to the grid, edited and must be copied *back* to the workscreen by clicking on magnify again, or the changes aren't incorporated.

The program features a special map mode for designing games. Maps are made up of brush "blocks." Brushes are assigned ASCII characters, and the map proper consists of an ASCII table with characters defining which block goes where. A final map file is output as a .S file, which can be merged into a program's source code. The brushes are output as bit images.

Simple animation is supported by Art Studio. It saves and loads pictures in NEO-Chrome format, thus supporting NEO's color-cycling animation. Sprites can be designed and test-animated from within the program as well. Sprite images are placed in the brush store, in proper order, and can be viewed in movement on the control screen or in front of a workscreen (at an adjustable speed). Defining such a loop is easy, but the sprites *must* be in order, first to last. The sprite ani-



mation is not saved as part of the picture file. But, like the map mode's brushes, sprite data can be saved for inclusion in programs.

Art Studio does have its weaknesses. Brushes can only be 8x8, 16x16, 16x32, 32x16, 32x32, 32x64, 64x32 or 64x64 pixels, with no intermediate or greater sizes allowed. The "spray" painting tool is not adjustable, and there seem to be no modes for drawing ellipses, rays or continuous lines. The copy (cut and move) option only duplicates a block. You cannot use a cut block as a brush (though you can *stamp* it multiple times).

While the program is strongly mouse-dependent, it does not use the mouse for file selection. You have to type in path- and filenames. To worsen matters, the program will not recognize a drive beyond unit D. I could not get it to access partitions E and F of my hard disk.

Art Studio has separate *undo* options for both the screen and the grid, but the ST's undo key is not supported. To undo a mistake on a workscreen, you must return to the control screen to access the undo icon.

Finally, there is no provision for setting up printer drivers. The manual instructs you to use the Control Panel/Set Printer accessory. Unfortunately, that accessory doesn't let you set up a printer driver, and the program makes no provision to let you access the accessory or any such controls from within itself. If you need to alter the printer settings, you'll have to quit the program, make adjustments, then run Art Studio again.

With its map and sprite-designing features, The Advanced OCP Art Studio looks useful for game designers and programmers.

The program supports Eidersoft's Pro-Draw graphics tablet, and comes with a demo which shows how the program is used. Files containing a font brush and regular brush stores are included. The single-sided disk is not copy protected, but the program refers the user to its manual, asking you to type in a key word, in order to assure that you have a legitimate copy. The manual is good overall, but some descriptions are not very clear. It sometimes takes several tries to figure out just what you're supposed to do.

To sum up: With its map and sprite-designing features, The Advanced OCP Art Studio looks useful for game designers and programmers. However, with its weaknesses in many purely artistic regards, and not terribly intuitive user interface, I cannot recommend it to most users. If you desire to paint, you would be better off with the multitude of easy-to-use tools in Art Director or DEGAS Elite or with the color possibilities of Spectrum 512.

The author would like to thank Larry Elliott at Computer House in Sparks, Nevada, for providing the software for this review.

RE
VIEW

SOFTSYNTH

Digidesign
1360 Willow Road
Suite 101
Menlo Park, CA 94025
(415) 327-8811
$295

Reviewed by David Duberman

The Atari ST is by far the best hardware value in MIDI computing, thanks to its low price and built-in MIDI ports and programmability. Nonetheless, most of the interesting (i.e., experimental) MIDI software are available only for Apple's Macintosh, simply because it's been around longer. Thus it's extremely encouraging to note that one of the foremost publishers of Macintosh MIDI software, Digidesign of Menlo Park, California, has released its $295 *Softsynth* program for the Atari ST.

The name Softsynth is an abbreviation of the term Software Synthesizer. The program uses your ST's computing power to create complex new sounds to be played back on a sampler. Supported samplers are: E-mu Emax and Emulator; Ensoniq Mirage; Korg DSS-1; Sequential Prophet 2000; Roland S-10 and S-50; and Akai S900, X7000, X700 and S612. In case you're not familiar with the terminology, a sampler is a kind of synthesizer that can record and play back real sounds, rather than create them entirely from scratch as conventional synthesizers do. Samplers use digitized sound, just like CDs. And since a sampled sound is just a (long) string of numbers, there's no reason why its source can't be a computer instead of a microphone.

Softsynth uses the ST to perform two different kinds of synthesis—additive and FM—separately or in combination. Additive synthesis is not a simple topic, but Softsynth's manual provides a clear, thorough discussion of it. As a brief explanation to help you understand this review, every sound you hear can be broken down into several combined sound waves (much as a ray of light can be divided into a spectrum with a prism); a fundamental wave and any number of harmonic waves or partials, whose frequencies are multiples of the fundamental's. Additive synthesis uses multiple independent sine waves to "build" new sounds or recreate existing ones. Each sine wave in an additive sound has its own envelopes for frequency and amplitude. This means that during the time that you hear a sound, the loudness and/or pitch of each of the sound's elements vary in more-or-less subtle ways to produce what musicians called timbre events, or changes in the nature of the sound over time. Given sufficient control, additive synthesis gives you the power to recreate synthetically any sound in existence—or out of existence.

Softsynth's version of additive synthesis gives you up to 32 independent sound waves or oscillators. It's a fully GEM-based application that works in medium-resolution color or high-resolution monochrome. Although drop-down menus are available, the most commonly used commands are always accessible via on-screen icons and the keyboard, making the program quite easy to learn and use.

Softsynth's main screen shows an overall 3-D view in three-quarters perspective of the amplitude envelopes of the 32 sound waves, which you can rotate to four different angles. From here you set the fundamental frequency, the sample rate and the file length. Menu commands let you set your sampler, save and load parameter files, and save the program's current configuration. Arrayed across the bottom of the screen are 32 mouse-controlled sliders for setting each harmonic's overall loudness. When you click on any slider, its current numeric value is shown.

When you click on a harmonic's identifying number below a slider the main screen is temporarily replaced by the harmonic's edit screen. This shows the envelopes for amplitude and pitch, both of which you manipulate directly with the mouse and tools represented by icons.

You can preview the partial's sound at any time. From here you can select directly any other partial to edit, but you must return to the main screen to hear the full sound.

Softsynth lets you set as many as 40 (!!) stages in each partial's amplitude envelope and up to 15 in the frequency envelope. You can also set the partial's ratio to the fundamental pitch, after which Softsynth calculates the actual frequency for that partial. Finally, you can set a waveform type for the partial, selecting from Sine, Square, Triangle, Bandlimited Noise and White Noise. This ability to use waveforms other than the usual sine wave makes it easier to create interesting sounds more quickly, as well as to add special effects.

Softsynth also offers an alternative method of additive synthesis which gives you more control over the overall sound while sacrificing fine control. Time Slice Editing lets you control relative levels of the individual partials by means of timbre events, which are "snapshots" of the amplitude of each partial at a given moment in time. A single, overall amplitude envelope is shown, with any number of timbre events arrayed underneath. Click on a timbre event and the current levels of all partials at that point are shown on the slider display, ready for you to adjust if you like. You can add, delete and move timbre events. The manual provides a tutorial in use of the unusual Time Slice method.

The concept of FM synthesis in a nutshell involves different sound waves interacting as carriers (audible) and modulators (usually inaudible, modulating carriers). Softsynth's advanced FM implementation lets you set any partial to modulate any other partial (including itself), with modulators audible or not. You'll be surprised at the complexity of the sounds you can create with relatively few operators. That's the power of FM synthesis, while the drawback is loss of fine control over subtler aspects of the sound.

Finally, Softsynth's Smartsynth function is a wonderful way to experiment with sound. To quote the manual, "Smartsynth uses parameter ranges and a random number generator to create new sounds"—that is, it's an "auto-droid" function. You can set just about any combination of a screen full of parameters such as Harmonic Series, Range and Filter, Partial Detuning, Doubling, Frequency Movement and Attack, and Attack and Decay Rates. Then Softsynth throws a few dice and creates for your ears only a brand-new never-before-heard sound, based on your Smartsynth settings and any existing sound parameters. You can fine-tune any successful results with Softsynth's other sections.

Once you've created a new sound, the Preview command lets you hear an approximate rendition via your monitor speaker or with the ST Sound Digitizers (Navarone) or ST Replay (Michtron/Microdeal). Next the Synthesize command generates and saves to disk a sample file that can then be sent to the sampler. If you're using the program to experiment with, I recommend use of a RAMdisk for this step to save significant amounts of time, both in saving the file and in sending it to the sampler.

The process of sending a sample file to the sampler varies with different keyboards. Fortunately the manual includes a separate section with detailed instructions for each supported sampler. Softsynth supports each sampler's choice of methods of creating multi-samples in memory. For example, with the Korg DSS-1, you can replace any sample in any existing multisound, append a sound to a multisound or create a new multisound.

The program is copy-protected, but uses the key disk protection scheme so that you can run it from a hard disk, RAMdisk or a copy in drive B. But the original disk must be in drive A before you start—you're not prompted for it. You get a free backup when you send in your warranty card.

While Softsynth is near the top of the price range for Atari ST software, it's certainly one of the finest programs that I've used on any computer. It's well-designed, easy to use and fast and efficient in operation. Samples produced are clean and glitch-free. The program's professional caliber, combined with the variety of integrated ways of creating sounds, offers tremendous value even at $295. I recommend Softsynth to anyone with an ST and one of the samplers listed above, and in particular to musicians seeking an endless source of new and unusual sounds.

David Duberman has been involved in Atari computing for the past five years. He is primarily interested in the use of personal computers for enhancing human creative potential.

Computer Garden

Wilkes-Barre & Scranton's Favorite Computer Store

Accolade	
Hardball	$25
Pinball Wizard	$23
Test Drive	$25
Antic	
Spectrum 512	$56
Cyberstudio	$70
Atari	
520STFM	$Call
1040ST	$Call
Mega-2 ST	$Call
Mega-4 ST	$Call
Avant-Garde	
PC-Ditto	$Call
Avatex	
1200e modem	$79
2400 modem	$169
Modem cable	$15
Batt.Included	
DEGAS	$49
Digital Vision	
Computereyes	$179
Dr. T's	
KCS	$189
The Copyist	$189
MIDI Recording Studio	$26
Epyx	
Impos. Mission II	$25
FTL	
Dungeonmaster	$25
Sundog	$25
Future Sys.	
GTS-100 drive	$199
ICD	
F20A-ST drive	$669
Intersect	
Interlink	$25
LC Tech.	
Stereotek kit	$139
Megamax	
Laser C	$159
Michtron	
GFA Basic	$39
GFA Compiler	$39
GFA Companion	$33
GFA Basic book	$25
Microprose	
F-15 Strike Egl.	$25
Gunship	$33
Silent Service	$25
Microsoft	
Microsoft Write	$89
Panasonic	
1080i-II Printer	$175
1091i-II Printer	$195
Printer cover	$9
Printer cable	$15
Practical Per.	
Monitor Master	$44
Mouse Master	$35
SeymorRadix	
IMG-Scan	$79
Sierra	
Space Quest	$33
Space Quest II	$33
King's Quest	$33
King's Quest II	$33
King's Quest III	$33
Donald Duck	$16
Police Quest	$33
Liesure Suit	$25
Mother Goose	$20
Soft Logik	
Pub. Partner	$60
Pub. Prtnr Pro	$120
Font Disk 1or 2	$20
Springboard	
Certificate Mkr.	$33
SSI	
Wizard's Crown	$25
Rings of Zilfin	$25
Phantasie	$25
Phantasie II	$25
Phantasie III	$25
War Game Con.	$23
Star	
NX1000 Printer	$169
Color version	$229
Printer cable	$15
Sublogic	
Flight Simulator	$33
Scenery Disk 7	$15
Scenery Dsk 11	$15
Supra	
2400 modem	$139
Modem cable	$15
Timeworks	
Wordwriter	$49
Datamanager	$49
Swiftcalc	$49
Partner	$33
Desktop Publish	$79
Trio Eng.	
Digispec	$39
Unispec	$49
Unison World	
Printmaster +	$25
FantasyArtGlry	$20
Fonts&Borders	$23

Toll-free order line:

1-800-456-5689

For information call 1-717-823-4025

3% charge for VISA-MC-AMEX. Shipping, extra.
Computer Garden, 106 W. Carey St, Plains PA 18705

CIRCLE #109 ON READER SERVICE CARD

ALICE

The Personal Pascal

An integrated programming environment with 700 HELP screens, an editor that makes errors impossible, and the best GEM interface anywhere. Only $79.95.

"An excellent value." **– Antic**

"It is about as painless a method of learning Pascal as can be devised short of hypnosis. It works!" **– Computer Shopper**

"The product is all anyone could ask for. I would recommend this product to anyone who is considering learning PASCAL . . . or anyone who wishes to prototype small applications which deal closely with GEM." **– ST Informer**

Orders: 1-800-265-2782

Looking Glass Software

124 King St. N. Waterloo, ON. N2J 2X8 519/884-7473

CIRCLE #108 ON READER SERVICE CARD

REVIEW

STuff

Michtron
576 S. Telegraph
Pontiac, MI 48053
(313) 334-5700
Resolution varies with program used
$39.95

Reviewed by Andy Eddy

Without a doubt, the most difficult computer products to review are utilities. Generally, all the reviewer can do is comment on the adequacy of the user interfacing, whether the program works or not and to what extent. Graphics usually are not too much of a concern, so there's even less to discuss.

Now I've got my work cut out for me with a scattering of ST utilities—21 in all—grouped into one single package from Michtron called *STuff*. As you'll see, these programs give the user various powers not originally programmed into the ST; some you'll use, others you may not need. All are intended to be helpful to different users.

These utilities are broken down into various categories: AUTO folder programs, GEM programs, Desk accessories, .TOS programs and .TTP programs. Following is a list of the programs on the disk, with a small capsule review of each (if possible or necessary). I'll also mention how well the program works and/or its limitations. Hey, what else can I say? I told you it wouldn't be easy. . . .

Auto Programs

AUTODATE—For those without clock hardware to automatically plug the time and date into the ST, this program lets the user input those figures each session, whereupon they are saved on the disk. When you boot up, AUTODATE requests that you update the system time and date figures (it uses the last session's statistics as a starting point) using the keyboard. If you don't start entering those figures within a five-second period, the program passes up and continues with the boot process.

CAPSLOCK—Eliminating unwanted striking of the Caps Lock key, this program requires the Alternate-Caps Lock combination to toggle that feature on or off. This takes a while to get used to, but for sloppy typists like myself (I tend to rest the balls of my palms against the lower rail of the keyboard and frequently bump into the Caps Lock inadvertently), it can protect you from wasted time.

HARDAUTO—Not owning a hard disk at the time of this writing, I was unable to test this program. It claims to allow a user the power to boot the system up off of the hard-drive's AUTO folder. If there are any remaining .PRG files in the A: drive's AUTO folder, they will execute following the hard-drive boot.

HIGH—Placing this in an AUTO folder bumps the ST into medium resolution from its default, low. This'll be useful if any of the remaining AUTO programs need to be booted in medium, as the ST doesn't read the DESKTOP.INF file before activating programs inside the AUTO folder (see AUTOGEM for more insight on all this).

KEYCOMBO—Sets up certain key combinations to actuate some basic functions of the ST from the keyboard. Alternate-Undo sends a form-feed code to your printer; Alternate-Delete blanks the screen on your color monitor, presumably saving the phosphors of your display (this key pairing doesn't function in Monochrome); Control-Alternate-Delete is like hitting the reset button at the rear of the ST and warm starts the machine; Control-Alternate-Left Shift-Delete accomplishes a cold start, like cycling the machine with the power switch.

ONEHAND—Advertised as being written primarily for handicapped computer users, this program turns the Alternate, Control and both Shift keys into toggling switches, just like the Caps Lock key. While it's impossible to use them during some applications (like word processing, for example) it is handy for single-handed mouse manipulation (by toggling the Alternate key on and using the arrow keys to control the pointer).

RESET— This changes the function of the reset button, causing it to initiate a cold start instead of the usual warm start. This will clear out normally reset-proof programs, like 512K (see .TOS programs) and some RAMdisks, for instance.

STSELECT—Lets the user pick from the files in the AUTO folder and desk accessories to choose which ones from the boot disk should be run at boot-up. It will rename unwanted AUTO files to .PRX extenders and .ACC files to .ACX extenders so the ST won't load them in at boot-up. As with AUTODATE, STSELECT uses the arrow keys for specifying which programs are to be loaded, and if the user doesn't respond within five seconds, the boot-up process continues.

VERIFY—Shuts down the write verification function of the ST's disk drive to expedite writes to disk. In actual testing, by copying an approximately 200K file from RAMdisk to floppy disk, the ST took 48.0 seconds with normal verification as opposed to 27.5 seconds after running VERIFY.PRG.

Desk Accessories

AUTOGEM—The only desk accessory in the package, AUTOGEM ingeniously allows auto-booting of any GEM program. Formerly, .PRG programs couldn't be placed into AUTO folders because those programs required that GEM be loaded before they are run; unfortunately, GEM loads in after the AUTO folder boot.

AUTOGEM takes control of the mouse, activating windows and double-clicking on icons to initiate the booting of a specified program after the boot process is completed. This requires you to use the AUTOGEM editor to choose or change the program to be run, or disable the feature altogether. Moving the mouse away from the center of the screen before the auto-execution process starts, turns the option off.

My only complaint has to do with the automated mouse movement, which is incredibly slow during AUTOGEM. Granted, the first few times you see it do its magic it's a kick to watch the phantom pointer movement, but it's a novelty that soon wears out. The function of this program is to boot up the predetermined file, and if you're like me you want that to occur as soon as possible. Hopefully, they will pick up the pace of the mouse in the future.

GEM Programs

AUTOFOLD—With all the AUTO files provided on this disk, it would be helpful to prescribe the order of execution. For instance, there are cases where you will want certain programs to load before others so their features will provide the most benefit—having HIGH set up the resolution ahead of other programs' execution, for example. AUTOFOLD lets you, the user, choose the sequence of execution and re-orders the folder for that line-up.

FILELOCK—For security in your disk data, you can code them with this password system, cutting off unwanted snooping. Up to three passwords can be employed to provide maximum protection. Once a file is locked, there is no way to unlock it without the proper password entry.

On the negative side, FILELOCK will wreck your file if the improper password is used. This means that you must take special care to re-enter your passwords *verbatim*, and keep FILELOCK and your "locked" files out of unwanted hands; at the very least, you should make backups of all files you lock.

TOS Programs

512K—Why would someone write a utility that would take your one-Meg ST (or more) and emulate a 512K ST? Well, some programmers have been a bit shoddy in doing their chores, using some improper addresses in their code. The end result could prevent a one-Meg-plus user from running an application.

To 512K's credit, your machine will run like a 520 ST and bring the poorly-programmed file back to the land of the usable. Actually, the consumer of the badly written program should request that the developer update the software to make it compatible if the user upgrades the machine they use, and hopefully the company will comply. With that in mind, a program like 512K would become obsolete. Realistically, there will be situations when its availability will save someone a good deal of time and trouble waiting for the update, though as time goes on the loss of available RAM may become the resulting limitation to the user.

KEYCODE—This program returns to the screen the ASCII code and the keyboard scan code for a struck key. Both of these values are helpful to programmers, particularly with some "same" keys having different values. On the ST, both the "1" key on the top row of the main keyboard and the "1" on the numeric pad have the same ASCII value (which KEYCODE displays as 0x31), but different scan codes (0x02 and 0x6D, respectively). I can't help but think that this program would've been a little more efficient as a desk accessory, where the user—especially a programmer—wouldn't have to back out of an application to get a key's value.

TTP Programs

FC—Another programmer-focused file, this compares two different programs, determines the bytes that don't match between them and lists them to the screen.

FDEL—Here's another security-minded utility that will delete a file with no chance of recovery. The ST's normal procedure for deleting a file doesn't actually overwrite the file (a file's location is marked as being usable for future disk writes and, provided that those sectors haven't been written to, the data exists intact), and programs are available for "undeleting" files that haven't been written over. FDEL goes a step further by filling the file's previous location with garbage characters, ensuring that someone doesn't come along and spill its valuable contents out for all to see.

GREP—As with the Search command in a word processor, this file searches any other files for a particular character or string. It would've been a bit easier to use if examples of the various commands were provided, but that is the general situation with .TTP programs; they don't support mouse use, often require the user to use a cryptic command set and frequently must be played with on a trial-and-error basis before the user gets the best out of them.

HEADER—These next two are, again, programmer-oriented. HEADER takes the target file and tells you the length of the Text, Data and BSS segments of a file, as well as the symbol table length, if one is used. And....

HEX—Takes any file and very neatly displays it in hexidecimal.

TOUCH—This is an interesting application: It will "touch" the desired file, thereby stamping the file with the system time/date (or a user-specified time/date figure for those without a clock card), helping you to track the latest version of a file. Especially beneficial when used in conjunction with AUTODATE to set the system up.

UNHIDE—Lastly, this lets you get into the guts of the file directory. You can set or remove flags for hidden, read/read-write and/or system attributes on the disk.

The STuff package also contains a program called PATCHER, a utility that is used to insert bug fixes, commonly called "patches," during future upgrades. This will enable Michtron to reduce the cost of upgrading by not having to send an entirely new disk to each owner. Factory Programming (the company that created STuff and some other fine programs for Michtron) has put PATCHER into the public domain and claims they will include it with all future releases. That's a consistency that's good to hear of, very reassuring to owners of Michtron/Factory products when they consider support for their purchases.

For obvious reasons, Michtron has left the STuff disk unprotected so you can place any of the programs on boot disks, frequently used disks or—given that you own one—install them on a hard drive. This isn't an invitation to pass on the files to your friends. Please respect this action by Michtron, particularly with the low cost of the package; better, encourage others to pick up STuff if there is anything in it that strikes their fancy. I found a few programs that are useful on a day-to-day basis; others you'll call up once in a blue moon. Still, it's a minute price to pay for all that STuff.

RE
VIEW

Phantasie III: The Wrath of Nikademus

by Doug Wood
SSI
1046 N. Rengstorff Ave.
Mountain View, CA 94043
Medium Resolution
$39.94

Reviewed by Steve Panak

By the time most sequels reach their third installment, typically even the best series has been stripped of any surprise and originality it might have had in the first place. Even the third Star Wars was somewhat of a letdown. Such is not the case, though, with this new game. When I reviewed its two prequels, I declared them to be light-years beyond any previous fantasy games. What is truly surprising is that *Phantasie III* is light-years beyond its two landmark predecessors.

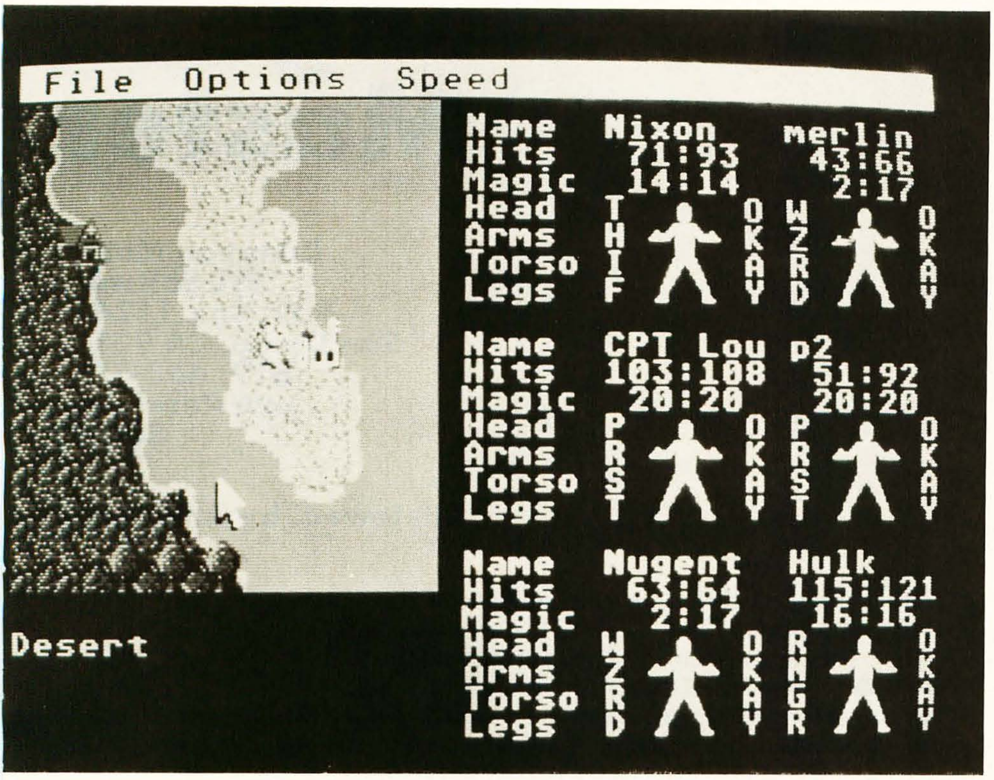
I hardly need to go into the specifics of play. Phantasie III continues the storyline begun in I & II, basically a rich blend of dungeons and dragons, swords and sorcery. Nikademus is still on the run, still spreading his own special brand of evil. And you are again pursuing him. Or, rather, your characters are. A number of characters can be created, each randomly assigned attribute values increasing in direct proportion to the warriors' strength, intelligence, charisma, luck, dexterity and constitution. Up to six are then banded into a group, armed and trained and put on the road to begin their (and your) adventure.

Moving out of the town of Pendragon, you cross the countryside, battling scores of creatures in the forests and dungeons of the land, stopping at inns and towns to rest and build strength. Successful quests are rewarded with gold and experience points; unsuccessful ones are rewarded with death. The kingdom you explore is vast and full of dark dungeons and dangerous monsters, as well as a number of hidden worlds not immediately discovered. But rest assured that if you manage to explore all the nooks and crannies filling the data disk, you will meet the evasive Nikademus and battle him on his own turf. Of course, this climax is days (both real-time days and game-time days) away. But it won't take any time at all to

learn the simple, intuitive commands you use to control your destiny.

The mouse is used for nearly all input, which is done on carefully designed and efficient displays, containing logical defaults to speed repetitive selections. For example, when creating a character it is assumed that you will continue to create characters of the same race and profession until you choose otherwise. Very useful, as you might discard scores of fighters until you get one with randomly assigned attribute values that satisfy you. Nearly all screen displays have been redesigned, adding more detail, color and action. The status of your band is displayed at all times, graphically, on the screen. Each member's body is displayed with wounds indicated as to body part and degree of injury. A red, broken left arm can render a character incapable of using his bow. A removed head can render a character incapable of even simpler tasks.

The list of improvements continues. Party members can be assigned to ranks in the group. Placing your weaker wizards at the back protects them from the brute force of the melees. You can save more than one dungeon map, and your party can accumulate up to



```
Name   Nixon      merlin
Hits    71:93      43:66
Magic   14:14       2:17
Head   T      O   W      O
Arms   H      K   Z      K
Torso  I      A   R      A
Legs   F      Y   D      Y

Name   CPT Lou    p2
Hits   103:108     51:92
Magic   20:20      20:20
Head   P      O   P      O
Arms   R      K   R      K
Torso  S      A   S      A
Legs   T      Y   T      Y

Name   Nugent     Hulk
Hits    63:64     115:121
Magic    2:17      16:16
Head   W      O   R      O
Arms   Z      K   N      K
Torso  R      A   G      A
Legs   D      Y   R      Y
Desert
```

Phantasie III skillfully mixes a great story line filled with plenty of action and adventure, a number of improvements and a user interface that is second to none. I hope we will get to experience more Phantasic games like this one.

a billion gold pieces. You are allowed to distribute items upon entry into a town, and trade items among party members, rectifying a clumsy fault in the two earlier Phantasies. A couple of more spells have been added, as well as the ability to target spells at specific ranks. Bows are added as a weapon, and you can selectively increase a character's nine skills by specific training as he rises in level. You have more control over the destiny of your characters.

The documentation is equally superb. The 25-page manual completely describes all aspects of game play and program operation. Tables and charts simplify determining the allowable spells and strengths of the various creatures inhabiting the Phantasie universe. Two handy appendices provide impatient players with quick start information and veterans with a list of the enhancements contained in III. An ST reference card contains machine-specific information, which I felt was a little brief, but adequate overall.

In fact, this game is so close to perfect, I have to dig really deep to find any fault. But here goes. While I liked the ability to copy both game and data disks, the required input of answers to questions on the manual's contents multiple times during a play session was distracting—even after we had memorized all the questions. In the prior two versions, upon casting a spell, menus listing the available spells for each character indicated those spells which were not available due to weakness. In III you must try to use the spell only to be told your character is too weak. Menu windows open up over the status display, making it difficult and time consuming to administer healing spells and magic potions to a weary group, as you often must exit the routine in order to check the status of your men. Finally, although you are able to transfer your characters from I & II to III, I was unable to do so with my loyal group, who had finished both of the previous adventures. The transfer program worked perfectly with other data disks, just not with the one I wanted to use. However, we simply recreated our warriors and played despite the annoyance.

So, if you can't tell by now, I'll make it known: I loved this game. Phantasie III skillfully mixes a great story line filled with plenty of action and adventure, a number of improvements and a user interface second to none. As a stand-alone game, or the climax to the series, it is a must for all adventurers. I can only hope that we will get to experience more Phantastic games like this one.

RE
VIEW

Dungeon Master

Dungeon Master
FTL Software
P.O. Box 112489
San Diego, CA 92111
(619) 453-5711
Low Resolution
$39.95

Reviewed by Andy Eddy

You are no doubt aware of the *Nightmare on Elm Street* movies and their popularity. As graphic and controversial as they are, there's one thing that keeps the public coming back to sequel after sequel: suspense. If you can keep people on the edge of their seats with "things that go bump in the night" and such, you'll have a hit.

Ladies and gentlemen, welcome to Nightmare-ware on ST Street . . .

Dungeon Master, a slick, new adventure from FTL Software, has taken the ST world by storm. "New" is a relative term as ST users may recall a demo FTL released—about two years ago—that first demonstrated what they were working on. It showed up in the databases of the ANALOG/Atari SIG on Delphi in June of '86 as the Firestaff Demo. Since then, they've switched to a different language, for added speed, and it's obvious they haven't wasted any of that time.

The final result is a suspenseful blend of strikingly colorful graphics and strong action, but by far the finest aspect of DM is that feeling of fear and apprehension you get wondering what's going to pop out from around the next bend. Countless DM enthusiasts—and my hand is raised here, also — have admitted to actually being startled by a surprise visit from one of the ghoulies lurking in the multitude of rooms in the castle.

The software takes quite a while to load, understandably due to the custom routines FTL has employed to get more than 16 colors on the screen at once, as well as the size of the castle you'll be rummaging through. The added hues are used to smooth out the shading of the surroundings, but, most importantly, to make the assaulting creatures look as life-like as possible. And they do!

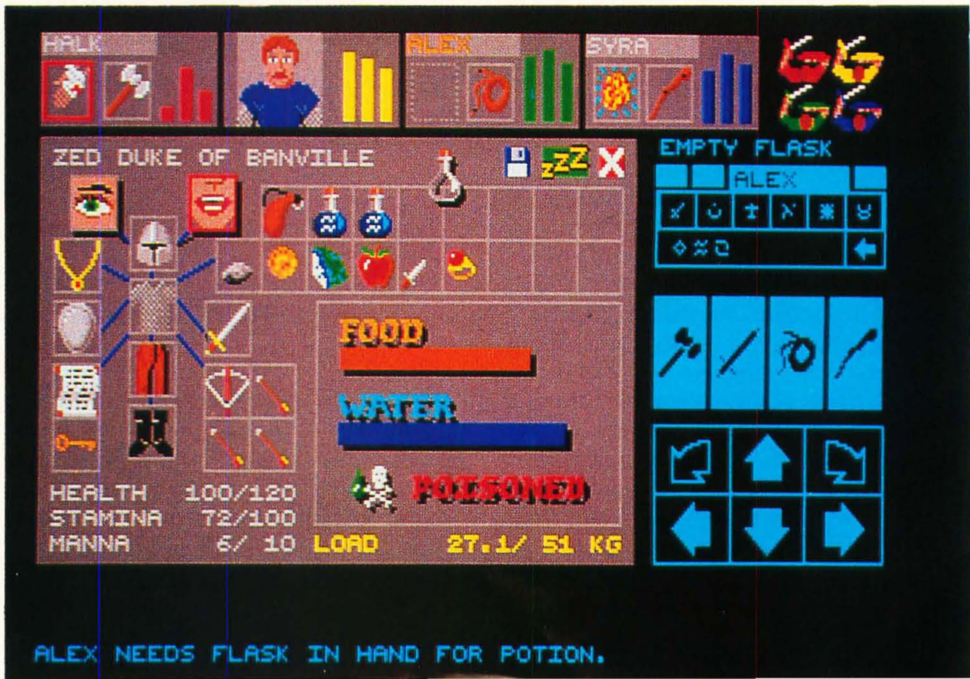
Couple that with the digitized sound effects, which accompany each offensive attack you or the monsters initiate and you get the spirit of being there. For instance, they've even added comical "gulps" and "ooofs", when you eat something or accidentally run into a wall with your party. Having one of your party killed off brings a heart-wrenching moan from the monitor speaker; an audible assist as you will probably be too busy slinging your weapons in combat to see that member's health indicator bars drop down. It's an involving experience, to say the least.

Wrassle in the Castle

At the onset of the contest, you wander through the Hall of Champions, a chamber of characters who have failed in their past attempts at defeating the evil Lord Chaos. As the story that accompanies the documentation attests to, you are a young apprentice to the Grey Lord. As you further discover, Theron's mission is to oversee four heroes in the quest to retrieve the Firestaff, a stepping stone to defeating Lord Chaos.

Each Champion has their portrait hung on the wall of the chamber and clicking on the picture brings a screen detailing their attributes and cache of weaponry. You can restore them as they are, or resurrect them under a different name, though at a lower ability level. As with most adventures of this genre, you gain experience points in certain areas, or actually avocations—such as Priest, Wizard, Ninja, etc.—as you successfully fend your way deeper in the maze of rooms and levels.

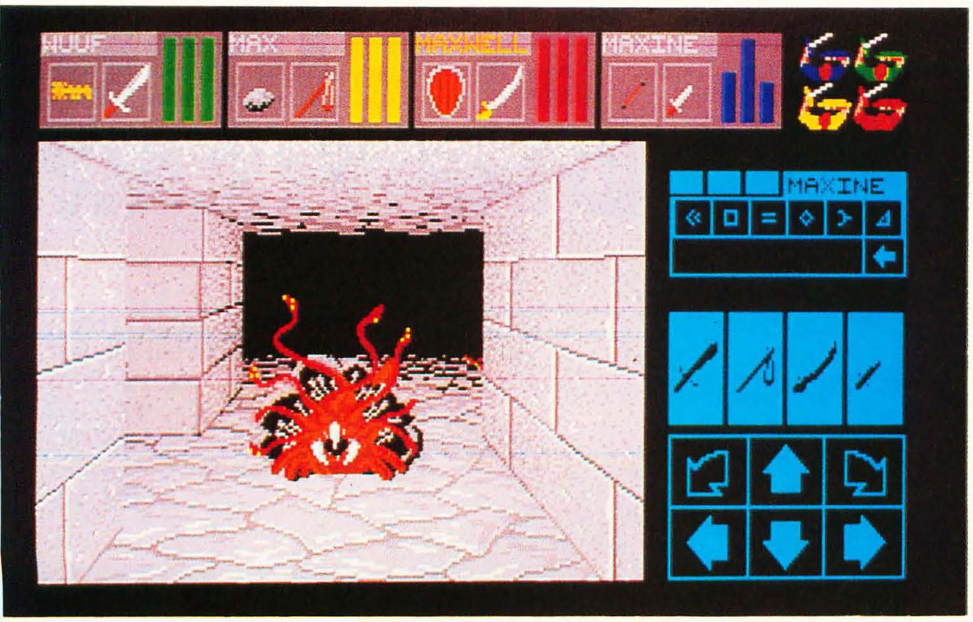
Also, as is common in this kind of gameware, you gather necessary com-



ZED DUKE OF BANVILLE

HEALTH 100/120
STAMINA 72/100
MANNA 6/ 10 LOAD 27.1/ 51 KG

ALEX NEEDS FLASK IN HAND FOR POTION.



ponents—be it a weapon, key, informative scroll or vital nourishment, among other items—to keep the party healthy and on top of the onslaught. So as not to discourage the novice adventurer, the first levels are the easiest—providing a nice, smooth learning curve for getting used to the surroundings of the castle and the software's controls—with each successive level adding more opposition in the way of harder-to-kill meanies and intricate mazes. I've victoriously battled my way into level 4, though I've been told I have a ways to go; there are in excess of ten levels, enough to keep a player busy for some time.

I think icon, I think icon . . . .

Playing DM is a treat in comfort; everything is icon-driven and mouse-controlled, so there's no call for any typing (even if you choose new names for your champions at the start). I've always found the burden of typing commands in text adventures to be a tedious exercise that generally accounted for my past apprehension over playing them.

Also unlike text adventures, picking up an object is as simple as moving the mouse pointer (in the shape of a hand) to the object on the screen and clicking; click again to drop it. You can even use the acquired item as a weapon by holding its icon in the center of the screen and clicking, which causes the character holding it to throw it.

The viewer accurately shows the perspective of how your party views the area around it; if items or creatures are down the hall a bit, you'll see them in their 3-D rendition. And those surroundings are nicely garnished with occasional embellishments, like a patch of algae or missing mortar on the walls and puddles on the floor for added realism.

If you need a bite to eat, just click with the right button on the icon of the selected champion at the top of the screen, click on the object you want in the backpack (which holds 17 objects per party member), carry it to the mouth icon and click. . .*glup*. . .and the player's food indicator bar jumps up to a more acceptable level; the same occurs to satisfy thirst.

You also have areas on-screen for the preparation of weapons or magical objects. There are two hand icons for holding objects, and placing an item in the right hand enables it for activity, like slashing with a sword or flinging a throwing star. As well, there are pockets for holding small objects (like a compass or rabbit's foot) and locations for clothing and armor (such as a helmet or chain mail).

Abracadabra

One other essential thing you have in your defensive favor is magic spells. At the far right of the display, there is a box with symbols in it and four buttons, each with the name of the party members in it. Clicking on one of the names and following that by selecting (also with the mouse) a series of symbols will have that person chanting matching spell syllables. Certain combinations of symbols effect different results: fireballs, poisonous missiles, a mystical torch for lighting your way, healing, etc. The higher the level of "mana" your characters have, the stronger and more frequently they can recite the incantations with better results.

I know all this sounds overwhelming, but keep in mind that learning to use any complex piece of software can take a little bit of getting used to—keep in mind, it's the program itself that's complex, not the actual use of it. As I alluded to before, FTL has provided ample room for getting the hang of the user interface, while risking very little damage to your party.

Dialog boxes for saving the game to disk lead your way easily to insure you don't waste your efforts. The only complaint I can make is that you can't reload a game on the fly; only at the start of a new session or when all of your party members have bitten the dust. There are times, if you know you are faring poorly, that you wish you could get a fresh start on the scenario, but for now it's not supported in DM.

That's too bad, because you'll find yourself reloading your game saves over and over, due to the ruthlessness of a terribly challenging variety of attackers. There are mummies, oversized mushroom creatures called Screamers, poisonous purple worms, and many more. All of this gets more intense as the contest rages on. And to reiterate, the graphics and animation of these beasties is fabulous: flailing arms, fang-filled mouths and slithering bodies slinking around the maze at every turn.

The foundation of the dungeon

According to Wayne Holder, President of FTL, the work that went into the DM software will result in a toolbox of sorts, giving them the ability to create other adventures in a multitude of venues. Changing the graphics data file, for example, can alter the arena of play from the DM castle to deep space. This configuration will permit them to come out with these games quicker, cleaner and cheaper than a product that starts from scratch. Holder stated, albeit optimistically, that their intentions are to bring out four to six games of this sort each year, with the first one coming after they work all the wrinkles out of this potentially powerful development system—possibly by the end of 1988.

Keep in mind that you can't possibly get the feel of how well the game looks or plays, or understand all the nuances of the product by just reading the rantings of this satisfied reviewer. My advice is to *run* to your nearest computer store and buy a copy of this gem to see for yourself. As I'm sure you'll agree, it's worth every penny.

In your haste though, don't forget one thing: keep an eye peeled for movement around each and every corner, just to be safe. . .you *never* know what's hiding there!

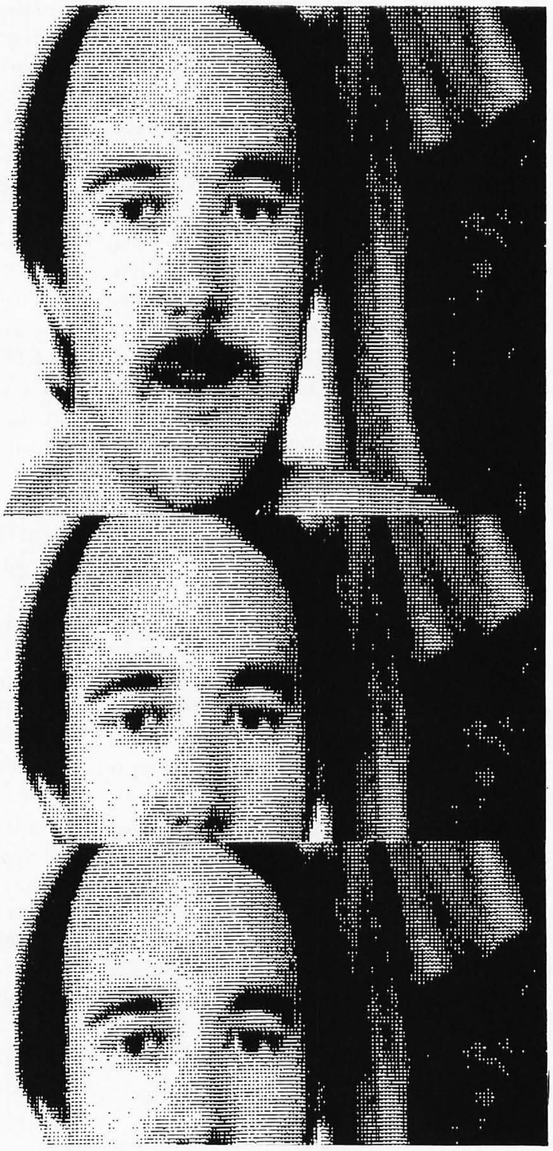
RE
VIEW

ST VIDEO DIGITIZER

Navarone Industries
1043 Stierlin Road
Suite 201
Mountain View, CA 94040
(415) 624-6545
Low or High Resolution
$139.95

Reviewed by Andy Eddy

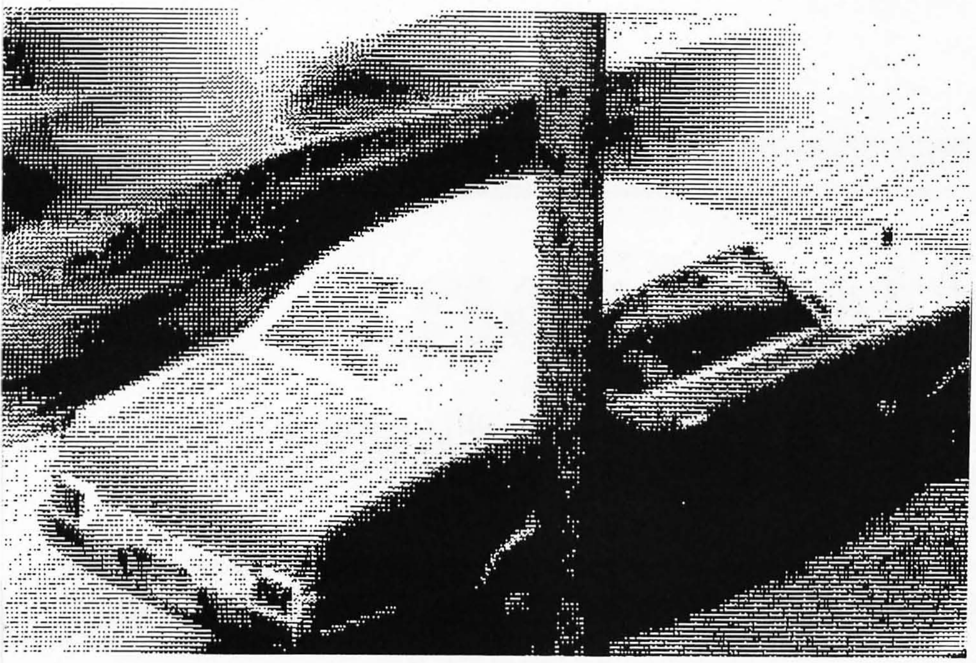
The product is pretty functional, but choices in the marketplace are ever increasing—better quality digitizers can be found.



Personal computer owners are showing an increased interest in graphics. A look at the SIGGRAPH show in California, a major computer-graphics show in which more and more small computer works are appearing, is an example of this occurring. The computers are going up in what they offer ability-wise while they descend in price. This means that more of the masses are able to partake in the artistry. Among the tools that are gradually making their way on the scene are digitizers like Hippo's and Digital Vision's Computer Eyes. Hippo's HippoVision digitizer has been picked up by Navarone and we'll see how the subsequent *ST Video Digitizer* now fares in comparison.

First a little history: Hippopotomus Software, among the many products it came up with since the introduction of the ST, introduced a digitizer that was to its benefit, the first to be released to the public. It had all the appearances of being rushed to market, as many buyers of the HippoVision complained of software crashes and poor picture quality.

Shortly thereafter, amidst many rumors of their shaky survival, Hippo phone lines were ominously vacant, and the company disappeared from the scene. Needless to say, this left many unhappy consumers in the wake. Navarone, previously known as the maker of the Timekeeper clock cart, announced that it was picking up the licensing of both the Hippo video and audio digitizers. Navarone took the tooling it had in place from the clock cart manufacturing to package the digitizers,



giving a similar appearance to its products—as well as saving some bucks by using an existing production system.

Specific to the video digitizer, it rewrote the software entirely in an attempt to eliminate the problems that came up with the original version. Unfortunately, those who already bought the device while it was being offered by Hippo are basically stuck with what they have; but Navarone will upgrade the software for a cost of $19.

I never had the opportunity to use the original HippoVision package, but I did hear enough from those who used it to get the idea of its quality. Though it's impossible for me to compare the two devices, I can say this: While changes had been made to the device and the software, the digitizer is still not up to the quality that's necessary for a graphics-dependent device like this one.

Setting Up

To get started, you take the cartridge (which accepts video inputs via the RCA socket connector) and slip it into the cartridge port on the ST. To the designer's credit, the unit doesn't require external powering taking the necessary voltages from the ST itself, thereby eliminating a further tangling of cords. Booting up the software after completing the hookup from a video source brings about automatic synchronization with a dialog box, appearing if there are any discrepancies.

Aside from the RCA socket, the only other thing that is exposed to the user is a thumb wheel for adjusting picture width. All of the pictures accompanying this review were taken with the thumb wheel fully counter-clockwise, which corresponds to the widest image. As we'll mention later, this didn't eliminate the image not taking up the entire screen or having what appears to be garbage on the edge in certain instances.

The disk that comes with the package—smartly unprotected so you can make a backup—contains various buttons on its desktop for picture saves/loads, specifying what format you want to save the pictures in (DEGAS or Neo), on-screen command help, a choice of Digitizing or Animating modes, full-screen display of the picture that currently resides in memory, as well as a few adjustments to the status of picture acquisition parameters.

The other tweaks you can enact concern Contrast, Brightness and Sync Level. The first two are easy to comprehend as they relate to similar controls on a television set; in fact, the sync control is quite the same as on a TV. It allows the user to correct any shimmering that may arise due to poor synchronization, though with the software synching at boot-up, it's very likely that this won't need any further adjustments. These three parameters can also be altered during digitizing using the ST's number pad, which lets you see the effect on the image directly.

The software operates in either low or high resolution. In both modes, you have a choice of two, four or eight levels of gray (the manual states that your actual breakdown for high resolution is three-, five- or ten-gray levels), though you can alter the low-resolution palette to whatever you choose from any of the ST's 512 colors. This is accomplished by picking the Palette button on the on-screen desktop and moving the sliders to vary the hue as you would from the ST's Control Panel. As an example of the annoyances I alluded to earlier, if you click your mouse too quickly in changing your palette, you'll find the software will crash, bombing you back to the main desktop. It's much easier and more efficient to employ a paint program like DEGAS, DEGAS Elite or Neochrome to tailor the palette to your liking.

The Navarone unit does have something in its favor with the speed of its hardware in grabbing images. They claim it can take in one gray level per frame (1/60th of a second), so if you have the programming set for eight-grey levels it will—according to their figures—take 2/15ths of a second to acquire the picture. It's quick, but I wouldn't say it was that quick: In a two-gray-level setting pictures pop onto the screen rapidly, but not too fast to prevent you from stopping the process to see a desired shot; set to eight gray levels, it will be a few seconds between image refreshing. From this speed, they've built in an Animation mode for getting a sequence of digitized shots.

To do this, you click on the Animate button on the program's control panel, and you'll get another dialog where you can set the delay between each acquisition, how many "frames" you want to grab and what you are going to call your "movie." From here, you can also start the animation/digitization process, view the sequence and save it to disk. The restriction you have with animation is that your image can only be made in a two-gray-level setting, which results in a not-too-detailed graphic.

The other major problems I came across while testing the unit out had to do with the actual digitized images themselves. When receiving pictures in eight gray levels, things generally seem to be okay. In the other modes, you seem to get a tinier version of the picture, not only on-screen but in actual picture-size also. Many times, the resulting shot has a stripe of garbage lines running down the right side of it that must be cropped off with a paint program to get a clean representation of the actual subject (see the four-level shot of the stuffed cat).

I'm not one to enjoy giving a bad review to a product, but in the case of Navarone's ST Video Digitizer it's unavoidable. The product is for the most part functional, but I find that the choices are increasing in the marketplace to allow the ST user an opportunity to find a better quality digitizer.

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 family of Atari computer owners.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. 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

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY $28

SAVE $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $105

SAVE TIME AND MONEY SUBSCRIBE TO ANALOG

SAVE $14 OFF THE COVER PRICE WITH THE CONVENIENCE OF HAVING ANALOG DELIVERED DIRECTLY TO YOUR DOOR *BEFORE* IT EVEN HITS THE NEWSSTANDS. GET THE MOST OUT OF YOUR COMPUTER.

SUBSCRIBE TO ANALOG TODAY

☐ 1 YEAR @ $28 — SAVE $14! MCHWY
FOREIGN — ADD $7 PER YEAR
☐ 1 YEAR WITH DISK @ $105 DCHWY
FOREIGN — ADD $15 PER YEAR

☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MC #________________

EXPIRATION DATE SIGNATURE

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box N. Hollywood, CA 91615. Offer expires OCTOBER 26,1988 . Your first issue will arrive in 6 TO 8 weeks.

WATCH FOR IT!

RE
VIEW

Spectrum 512

Spectrum 512
by Trio Engineering
The Catalog
524 Second Street
San Francisco, CA 94107
Orders: (800) 443-0100
Customer Service: (415) 957-0886
Low or Medium Resolution
$69.95

Reviewed by Andy Eddy

Everyone has a little *artiste* in them. For that reason, paint programs have been heavily in demand—perhaps the most popular category of software of ST users—since their introduction. First into users' disk drives was the NeoChrome program provided by Atari to the first ST buyers; that was followed by N-Vision/Paintworks, Tom Hudson's DEGAS and DEGAS Elite. Those programs have satisfactorily carried ST users—until now.

An engineering company from Massachusetts called Trio Engineering—made up primarily of three Russian emigrants—has taken the ST to limits of color display never attained before. To reach this programming pinnacle, they first sought to determine the limits of the ST's internals, through the use of various test gear. The resulting program, marketed under the moniker *Spectrum 512*, brings all 512 colors to your screen *simultaneously*, giving you the ability to make any picture you paint that much more detailed in shading, and, in turn, more realistic. In fact, you can place more colors on a scan line—48 of them—than was supposedly possible for the entire screen.

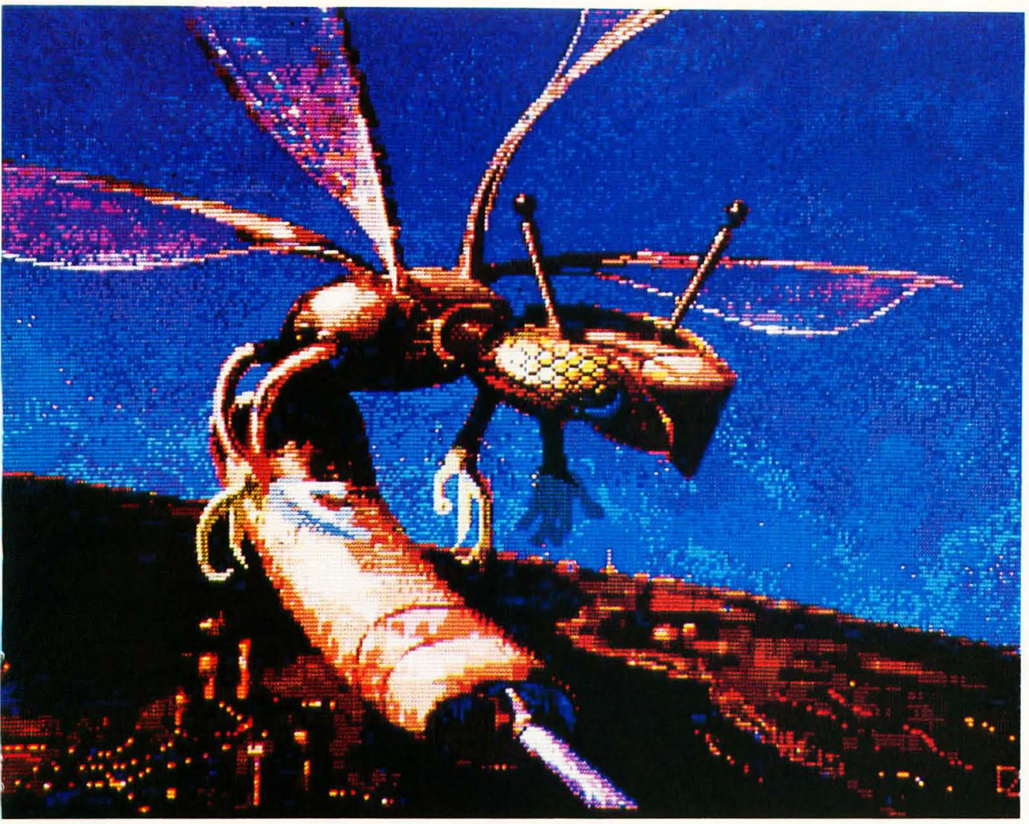
But to limit the description of the program to just the additional colors it gives you would be unfair. They have, by virtue of that engineering, expanded the abilities of a paint program to include smoothing of lines and curves—long acknowledged as a major limitation of computer graphics—through the use of a process called "anti-aliasing." Anti-aliasing takes what would normally be a jagged line and shades it with darker pixels of the same color to give it the appearance of being smoother. Circles look like circles; angular rays take on less of a staircased form.

Another effect of the expanded palette allows Spectrum to shrink down blocks of a graphic with very little change in resolution through what is called "pixel averaging." This lets you take a full-sized picture and scale it down without losing much of the original's quality. The process lets you work with larger shapes at first, then size them down to fit into a picture, or to place text or fill patterns around it. You can also change the original's shape (not unlike DEGAS' stretch command), which makes a change of perspective simple.

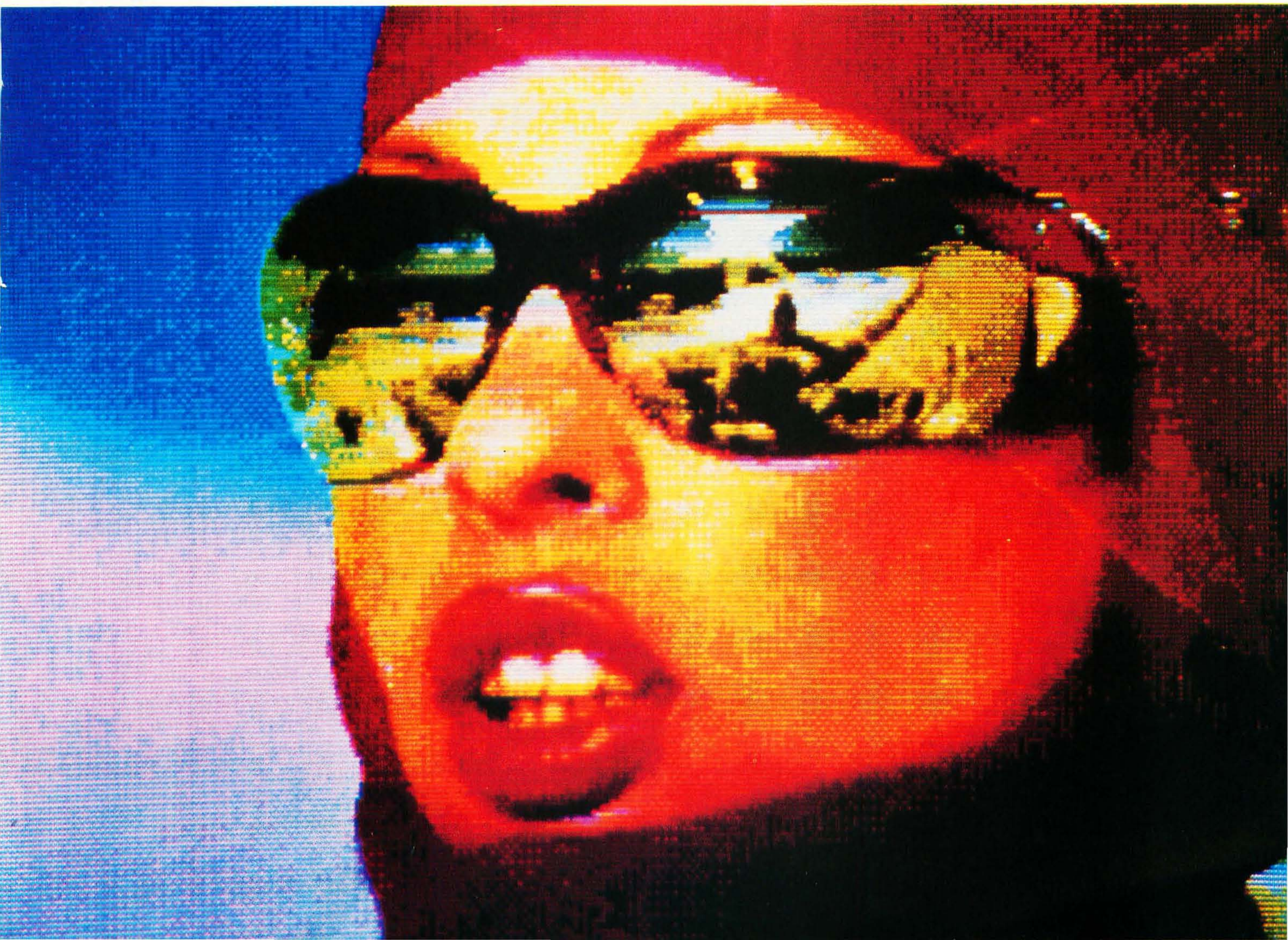
To top it off, with so many colors to play with, Trio's work has provided another powerful feature: the Gradient Fill. You pick a starting and finishing color, and Spectrum will fill (in a variety of forms) the chosen area with all shades between those two colors. This makes it easier to build a scene with gradual color alterations, such as you'd find in a sunset panorama.

The last few functions I mentioned just touch on the more innovative and diverse abilities that reside in Spectrum; it still offers the primary graphic processes: fills, magnification areas of the screen for touch-up, simple shape creation (line, polygon, circle and ellipse), and block copies and moves. With the oodles and oodles of features that Spectrum provides, the most difficult part is in learning how to get to each nook and cranny, some located in the depths of the program.

For example, a simple command like loading a picture actually has two options:



Trio Engineering has taken the ST to limits of color display never attained before.



If you click on the LOAD selection from the main menu with the left button of the mouse, the program will bring up a file selector to choose which file to display; clicking on LOAD with the right button brings up a dialog box prior to the file selector, prompting you for the type of file to load, as Spectrum can import DEGAS and Neo screens, as well as .IFF and HAM (Hold And Modify) formats from the Amiga. With the 4,096 colors available on that computer, Spectrum uses a dithering algorithm that tries to retain the original picture's quality. They claim it simulates 3,000 colors on the ST.

Another basic example of the intense user interface is in the primary use of the right button from the workscreen. If you hit the right button in the upper half of the screen, you bring up the main menu; hitting the right button on the lower half gives you access to the color matrix, which contains all 512 colors; and using the right button within ten pixels of either side of the screen (when a palette is present) changes the active color to the one next to the pointer. Starting to see what you're up against? It's certainly not the fault of the programmers—they had to do a lot to give the user the access to all of these wonderful features—but shows how much more you have to work to get the most out of this product.

Once you learn how to work the user interface, you can form masterpiece after masterpiece, as the programming offers so much to assist the user—often you lose the feeling of working on a computer.

The manual doesn't help much. I feel confident that the tutorial section (which runs you through the features by instructing you, step-by-step) will assist those who are just starting out with Spectrum; and experienced users will be able to go to the area in question by using the Table of Contents and the Reference Section (a run-through of the various commands, in order of how they are placed on the main menu, in addition to their many submenus). There are even two additional appendices to aid you: a discussion of the Gradient Fill technique by Boris Tsikanovsky, Spectrum's programmer; and a short tutorial by Darrel Anderson, Antic's favorite artist who created most of the first Spectrum art pieces that were seen by ST users before Spectrum's release. All that is well and good.

The problem is that the manual is too "chatty," a situation that too often had me hoping they would just get to the point. Also, there is a propensity for cutesy banter that I find too distracting. I suppose some folks would find this approach more friendly and comforting, but it only serves to lengthen an already involved learning process.

Finally, you'll find an inordinate amount of redundancy. Granted, many users need a bit of a push with a program of this caliber to absorb all of what is offered; yet, driving a point home repeatedly gets to be annoying, even insulting. Being told over and over that the undo key will cancel the effects of the last operation sinks in after just one or two tries, especially when running through the tutorial. My advice—and this goes for many software releases, not just this one—would be to include a reference card that could rest atop the computer. No one wants to have to dig up their binder every time they need to find the use of a command, particularly in the heated passion of artistic creation.

On the whole, Spectrum shines through these minuscule misgivings, as the program proves itself as a tight piece of code. Once you master the user interface, you can form masterpiece after masterpiece, as the programming offers so much to assist the user, such as having intermediate shades automatically appear on some palettes. You often lose the feeling of working on a computer.

What is most exciting in the making of Spectrum is that its programming breakthroughs can easily be used to enhance other graphic-based programs for the ST. There is talk of improving animation quality and design/engineering programs (such as CAD 3-D). So what started in the programmer's view as a "static display program" will likely have a far-reaching effect on the future of ST programming.

An example of how this has already happened is Trio's Digispec accessory ($34.95 from Trio Engineering, P.O. Box 332, Swampscott, MA, 01907; (617) 964-1673). It runs under the ComputerEyes system software and expands that digitizer's quality by a quantum leap.

There are very few products that I can consider a must-buy for the ST. I can safely say that if you have any intention of letting your artistic stylings run free on the phosphors of a monitor, Spectrum is the medium where your fingers should do that running.

REVOLUTIONARY NEW PRODUCT

SWITCH BACK

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**

WATCHCART ST

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

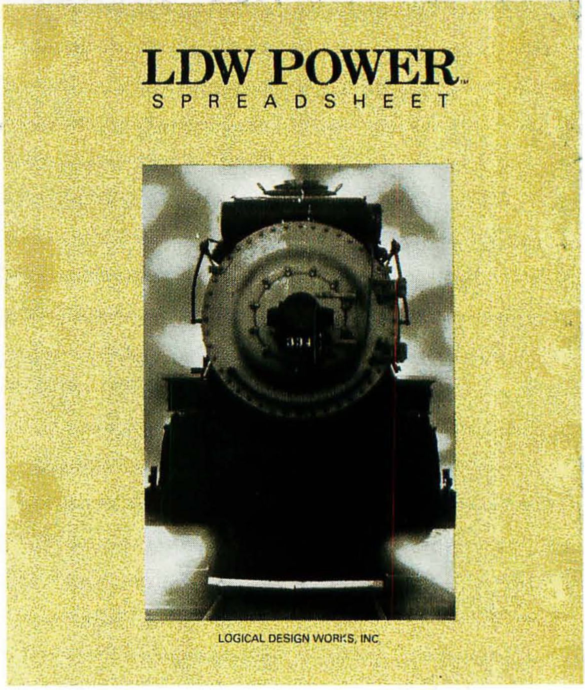
216-374-7469

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 #112 ON READER SERVICE CARD.

This Is All The Power You Will Ever Need In A Spreadsheet.



Isn't it about time that you bought a spreadsheet that takes advantage of all the POWER of the ATARI® ST™? That spreadsheet is here in LDW POWER.™

LDW POWER is a full function spreadsheet that boasts an extensive array of formats, commands, and functions along with sophisticated graphing abilities. Its Lotus® 1-2-3® compatible environment takes advantage of the GEM® interface allowing you to enter commands with the keyboard or mouse. There is also a convenient macro recorder which actually records keystrokes for future use.

Add to that its incredible speed in scrolling, recalculating, and copying, and you have one powerful tool.

Just look at these features.

GENERAL
- Lotus 1-2-3 Release 2 compatible
- GEM and Lotus interchangable interface
- Simultaneous use of up to four worksheet windows
- Condensed display allows 29 rows per window
- Ability to hide columns
- 18 display formats
- Customized monetary formats
- Alignment for labels and numbers
- Note feature for every cell
- Close to 300 commands
- More than 80 functions

GRAPHS
- Five graph types: XY, line, bar, stacked-bar, pie
- Automatic and manual scaling
- Optional grid, average/standard deviation lines
- Built-in printing procedure for graphs
- Metafiles and laser printer (GDOS) support

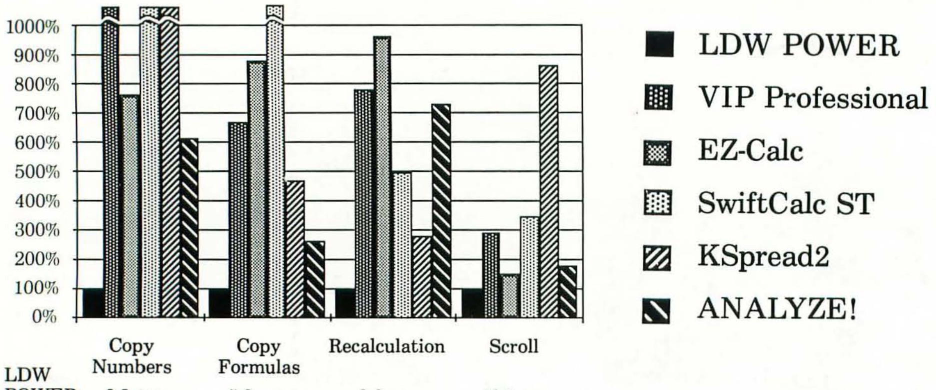
PRINTING
- Sideways printing program

FILE HANDLING
- Direct loading of Lotus and ASCII files
- Saving and retrieving of partial worksheets
- External file link function
- Password file protection
- Conversion of files into Lotus format
- Data Parse command

DATABASE
- Quick sort using one or two keys
- Sophisticated Data Query operations
- Criteria involving variables and logical operators
- Single and double variable tables
- Frequency analysis

MACROS
- Macro language compatible with Lotus 1-2-3
- Macro recorder
- Unlimited number of macros

TIMING COMPARISON



LDW POWER: 0.3 sec. 5.3 sec. 0.6 sec 42.1 sec

$149.95 (suggested retail price)
See your local software dealer to order or cal
(408) 435-1445 for more information.

©1988 Logical Design Works, Inc.
All rights reserved.

Logical Design Works, Inc.,
780 Montague Expwy., #403,
San Jose, California 95131,
Fax: (408) 435-7355.

LDW

All brand and product names are registered trademarks of their respective holders.

CIRCLE #113 ON READER SERVICE CARD.