# START™

THE ST QUARTERLY

PREMIERE ISSUE
Disk Enclosed

U.S.A. $14.95 CANADA $19.95
Summer 1986 Volume 1, Number 1

**FEATURES:**

*Tom Hudson's*
**ST Printer Driver**

*Tim Oren*
**on the Elusive GDOS**

**Voodoo Computing**

**Capturing Music on the ST**

**Sophisticated GEM Text Handling**

**Iconographics**
*The Digital Magnet*
*Plotting Magnetic Field Lines*

**Chips**
*Of Diagnostics and Debugging Procedures for the 68000*

Come to

XANTH Computer Systems Inc.

PIONEER BUILDING
600 FIRST AVENUE
SEATTLE, WA 98104
Information & Showroom
1-206-624-9292
orders only
1-800-647-7741

## The One Stop Shop

Let us help you select among the finest quality Systems & Software

At one convenient store

for one reasonable price.

Shop one of the largest Atari dealers on the West Coast

**Featuring the 520 ST and 1040 ST specials**

**Your system purchase includes one Free choice from the following quality items:**

- TDI-Modula II
- Regent Word II
- ST-Copy
- Softlogik-Clock Card
- Softworks Basic
- Versasoft dBMAN
- Meg Upgrade
- ST Station

Call now for details and take advantage of this special package.

**Introducing Softworks Basic Compiler . . .**
A "turbocharged" basic that experienced programmers will appreciate . . . . . . . . . . . . . .79.95

## 520 Station™ Integrate Your ATARI ST

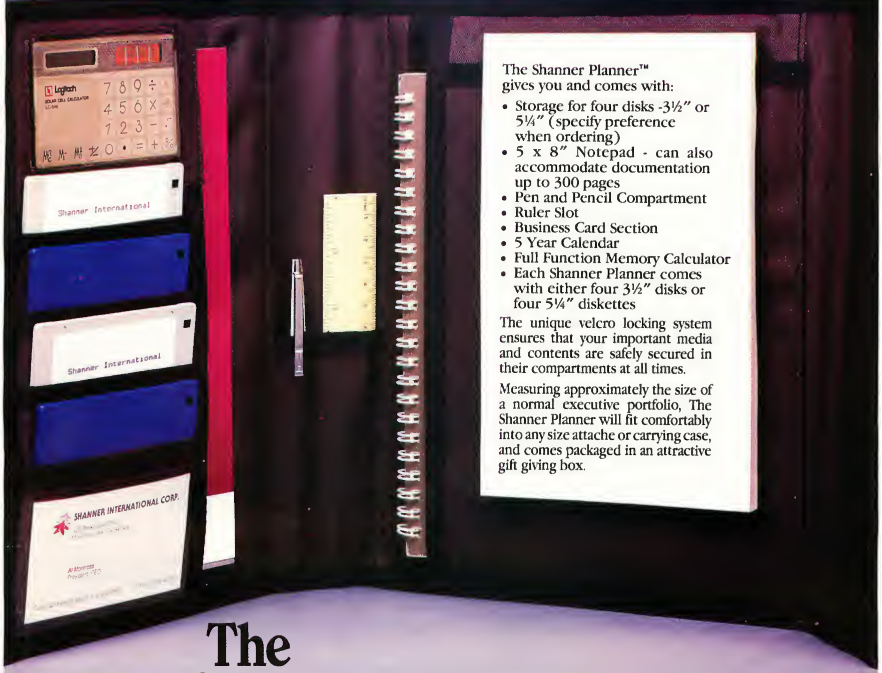
- An integrated workstation for homes and offices.
- One power switch.*
- Save valuable desk-top space.
- Protects disk drives and monitor from magnetic fields.
- 520 Station is vented to keep 520ST cool.
- Holds 2 floppy disk drives.
- Increased portability.
- The 520ST slides in and out easily from under the chassis, with adequate space for a mouse & joystick connector.

*with optional surge surpressor

ATARI & 520ST are registered trademarks of ATARI Corp.

Miller Computer Products
14640 N.E. 32nd, Suite D4
Bellevue, WA 98007-3649
(206) 624-9292
1-800-647-774

# It speaks for itself.



The Shanner Planner™
gives you and comes with:

- Storage for four disks -3½" or 5¼" (specify preference when ordering)
- 5 x 8" Notepad - can also accommodate documentation up to 300 pages
- Pen and Pencil Compartment
- Ruler Slot
- Business Card Section
- 5 Year Calendar
- Full Function Memory Calculator
- Each Shanner Planner comes with either four 3½" disks or four 5¼" diskettes

The unique velcro locking system ensures that your important media and contents are safely secured in their compartments at all times.

Measuring approximately the size of a normal executive portfolio, The Shanner Planner will fit comfortably into any size attache or carrying case, and comes packaged in an attractive gift giving box.

## The Shanner Planner™

**$39.95**

Dealer Inquiries Accepted
Call 800/828-6637

Remember the expression "A picture is worth a thousand words?" Take a second look. The Shanner Planner™ is the first ever full system portfolio created specifically for the computer user. Designed to address the individual needs of the computer industry, The Shanner Planner is tastefully constructed of long lasting, durable, textured nylon *(it's the in thing today)* to maintain its smart appearance over time.

**SHANNER INTERNATIONAL CORP.**
The Shanner Planner™ is a creation of, marketed worldwide by, and is a Trademark of SHANNER INTERNATIONAL CORP.

See the local authorized dealer nearest you or use this *SPEEDY* order form.

Please send me ____________ Shanner Planner(s) at $39.95 each, plus $6.00 postage & handling. (CA residents add $2.60 per Planner for sales tax.) Please indicate method of payment and media size required:

☐ Check ☐ Money Order ☐ VISA ☐ Mastercharge — Media Size ☐ 3½" ☐ 5¼"

Orders may be sent to:
Shanner International Corp.
453 Ravendale Drive, Mountain View, CA 94043

Account No. ____________________
Name ____________________
Address ____________________
City ____________ State ________ Zip ________

Prices subject to change without notice. ST/7/86
For direct orders:
Call our **HOTLINE 1-800-423-8673.** In CA **1-800-626-9273.**

# *Personal* Pascal for Atari ST Computers

## Just as Craftsmen Need Precision Tools... Programmers Demand Precision Software

Now **OSS** is proud to present **Personal Pascal**, the first Pascal for Atari ST computers that is destined to be the best!

Trust **OSS** to start you off in this efficient and popular language with a **complete** programming system—everything you need to start writing programs today! Editor, Compiler, Linker, GEM Libraries, and more.

Enjoy a language that starts with the international (ISO) standard and then adds dozens of expert features. Compatible string handling, powerful debugging options, special code optimizer—and of course, the famous **OSS support!**

Why wait? Purchase a copy of **Personal Pascal** and maximize not only the potential of your ST computer, but yours as well.

Also available from **OSS**:

**Personal DiskKit**—A disk utility complete with full **source code**.

**Personal Prolog**—Discover the world of logic programming with this language of the future.

For more information on **OSS**'s complete line of ST software, call or write for a free brochure.

*Atari ST Computers are a trademark of Atari Corporation. Personal Pascal™ is a trademark of O.S.S., Inc.*

OSS™

**Optimized Systems Software, Inc.**

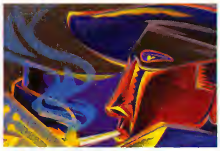
1221B Kentwood Avenue, San Jose, California 95129 (408) 446-3099

SUMMER 1986

THE ST QUARTERLY
START™

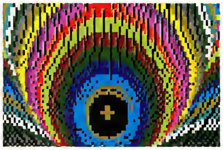
VOLUME 1, NUMBER 1

Your Atari ST may be missing part of its operating system. Find out what, and how to fix it, on page 36.



page 20

page 13



page 33

▪ *These articles contain a program listing.*

START THE ST QUARTERLY

**PUBLISHER**
James Capparell

**EDITORIAL**
DeWitt Robbeloth, Executive Editor
Michael Ciraolo, Editor
Jack Powell, Associate Editor
Dan Matejka, Contributing Technical Editor

**CREATIVE SERVICES**
Marni Tapscott, Art Director
Tim Barrett, Assistant Art Director/
Production Manager
Deborah Onodera, Ad Production Coordinator
Mark Gottlieb, Cover Photography
Contributing Artists: Matthew Foster, James Endicott, John Hersey, Barney La Hayre, Maciek Albrecht, Tom Hudson, Kat Guevara.
Jaciow Design, Consultants

**ADVERTISING SALES**
John Taggart, Director
(Please see Advertising Index, page 81.)

**CIRCULATION**
Les Tarak, Director
Cathy Sulak, Subscription Coordinator
Daniel Barrett and Steven Kulin, Dealer Sales

**MARKETING**
Jon Loveless, Vice President Gary Yost, Director Charles Cherry, Product Manager Lisa Wehrer, Product Distribution Manager Rebecca Hale, Customer Relations Eric Gupton, Technical Support Supervisor Scot Tumlia, Technical Support

General Offices & Catalog Customer Service:
(415) 957-0886
Subscription Customer Service:
(614) 383-3141
START, 524 Second Street,
San Francisco, CA 94107
Credit Card Subscriptions & Catalog Orders:
(800) 443-0100 ext. 133
(Continental U.S. & Hawaii)

**SUBMISSION INFORMATION**
START welcomes submissions. Please include both hard copy printouts of articles and program listings as well as disk files on ST compatible disks. Media will be returned if self-addressed, stamped mailer is supplied. START assumes no responsibility for unsolicited editorial material.

EDITORIAL

## READY, SET, *START!*

We at Antic Publishing are serious about the new Atari ST computers, and we know you are too. That's why I'm publishing START, the ST Quarterly. ST owners need a powerful publication, and that's what START is.

Let me tell you who we think you are, then you can look through this issue of START and see if it meets your needs.

You are probably well experienced with computers—used them in school or at your office—maybe you have owned one before. The important thing is that you have carefully considered the ST computer and concluded that it is the best buy for you. This is the machine you've been waiting for, and now you've got it!

Because you are experienced, you know you need the support of an active community—retailers, services and product manufacturers—as well as technical information and creative applications. You know there is a lot to learn—about the machine and how to use it. You know there is a lot to share—proud conquests and baffling problems. For this you need a publication that has the right stuff in it.

We think you want high-end information that requires a good grounding on your part both in computer fundamentals and practice. We think you want substantial articles and programs that will repay your investment of time and money in the ST and in START.

Is that you? If it is, take a look at our lineup this time: tough, strong pieces by the best ST minds around, backed up by six programs on the bound-in disk—no listings to type in, we think you're beyond that. (Note: some copies of START are sold without disk as a convenience to the trade and others without ST computers. If this copy does not contain a disk, you can order it for $10.95, plus $2.00 for postage and handling, from START DISK, 524 Second St., San Francisco, CA 94107.)

Because START is quarterly, we have the time to get, edit and present the best material for you, and you don't have to spend a fortune for complete ST coverage! Here's the deal: you can buy START on the newstand for $14.95 per issue (with disk), or subscribe to START for $59.95 (4 issues) and also receive 12 issues of Antic, that include the ST Resource. That means an additional monthly infusion of 30-40 pages of ST editorial material and programs in Antic.

Finally, we invite you masters of the 68000 to bend your brains in our direction. Information on article and program submissions can be found in The Dialog Box in this issue. The best ST articles and programs will always find a privileged place in START.

James Capparell
Publisher

## DIALOG BOX

### START SUBMISSIONS

*Programmers! Developers, hobbyists, students, gamesters, hackers: share the wealth.... of information. We have a new machine to explore, a lot to puzzle out. START wants to hear from you, be it in a letter, article or program. This is your forum, so take advantage of it. We're looking for those new discoveries, technical tips, tutorials and bugs.*

*Submissions and queries are very welcome. Please send correspondence to the address below. Submissions should be on both printed hard copy and on 3 1/2-inch, single-sided ST disk. (We prefer double-line spacing for articles; very long programs need not be placed on hard copy.) A cover letter, briefly describing your program or article, is suggested. If you wish your material returned, please include a self-addressed, stamped mailer. START is looking for original information and will be unable to publish articles or programs which have appeared in user group newsletters or on bulletin board systems.*

*Our address is:*

*START Editors*
*c/o Antic Publishing*
*524 Second Street*
*San Francisco, CA 94107*

### ST LANGUAGES

I am very interested in learning C but do not know which compiler would be best for me. I want to write good entertainment and/or role-playing, Ultima-type games. Okay, I may be 12-years old, but I am serious. Which compiler should I get?
Tim Gallo
Via CompuServe

*Many languages are available for the ST, including a few esoteric rarities such as Modula-2 and Personal Prolog. START considers C the most flexible language for the ST, but not necessarily the easiest to use. Alcyon C, included in the developer's toolkit from Atari, is difficult to use and slow to compile. Unfortunately, Alcyon was the first and has therefore established certain standards. Of the several other C's available on the ST, MegaMax is the easiest to use, but it is also the most expensive (for more details, see "Practical Software for the Non-Developer," in this issue). GST-C is more affordable and is nearly as friendly as MegaMax, though not as fast. But GST-C has no floating point and its link system is incompatible with Alcyon C's. Other versions offer greater incompatibility with Alcyon, or greater difficulty of use.*

*If you are creating from scratch, incompatiblity is not a problem. But if you are dealing with a program written for Alcyon C, such as a magazine listing, you are sure to run into trouble. At START we have decided to make our C listings compatible with Alcyon C because a standard of some kind must be established. In most cases, our listings will also work with MegaMax C. Some changes may be needed for GST-C or Lattice (usually for the linkage). Haba Hippo C will have many problems.*

*If you don't insist upon C, we recommend O.S.S.'s Personal Pascal. Pascal is very close to C and Personal Pascal is the only language currently available with full, understandable documentation on the GEM AES and VDI calls. With any other language (excepting Alcyon C) you will need to buy the Abacus ST books, numbers 2, 3, and 4.* ■

ILLUSTRATION BY JOHN HERSEY

## AUTHORS

**Christopher Chabris** is studying computer science in the Division of Applied Sciences at Harvard University. His fields of interest include artificial intelligence, analysis of algorithms, and parallel processing. Christopher is a regular contributor to Antic magazine and other Atari-oriented publications. He has extensive experience with 68000 systems programming, and his "Introducing 68000 Assembly Language" appeared in the November 1985 issue of Antic.

An internationally-ranked chess player, Christopher has represented the United States in competitions abroad and is among the top 50 players in the United States under 21 years of age. Currently, he is developing ST software and writing a book on artificial intelligence programming in Pascal.

Twenty-two-year-old **Joe Chiazzese**, together with his partner Alan Page, is putting the finishing touches on Flash, an ST terminal program. He was born and raised in Montreal, Quebec. After completing high school, Joe attended Dawson College with every intention of becoming a nuclear physicist. But, during his first year, he purchased an Atari 800 and quickly decided he preferred programming. He moved to Toronto where he enrolled at DeVry Institute of Technology.

Joe wrote many small public domain programs with the 8-bit Atari, but nothing of commercial value. When the 520 ST was announced, he immediately bought one. At the end of September 1985, he met Alan Page; Flash is their first major project.

**Corey Cole** joined the "micro revolution" shortly after the introduction of the Apple II. He has been programming professionally since 1975, and creating word processing and typesetting software since 1981. When Atari announced the 520ST in January of 1985, Corey saw a new revolution beginning, and decided to join the cause.

Corey is the president of Visionary Systems, which is currently developing state-of-the-art word processing and personal publishing products for the ST. Like the Atari ST, Visionary's products will feature "power without the price." Corey shares his San Jose, CA home with his wife Lori, two dogs, and an assortment of computers and musical instruments. Not content with traditional software marketing practices, Corey and Lori expect to have their own small computer user in late June.

**Jim Dunion** worked for the old Atari, first in the Software Development Support Group and later in Alan Kay's research group. While there, he authored DDT—Dunion's Debugging Tool, probably the best known debugger for the 8-bit Atari.

Jim has worked in the computer field since the early days of microprocessors and was a founder of a small retail computer store in Atlanta, GA, that eventually became Peachtree Software. At present, he is at The System Works in Redmond, WA, where he is working on the user interface portion of a maintenance planning and control system.

**Tom Hudson** is the creator of DEGAS, the popular ST paint program. Currently a freelance software developer, Tom was head of programming for ANALOG Computing magazine from 1982 to 1985. He first worked with computers on an IBM 1620 during his high school years. While earning an Associate of Science degree in data processing, Tom taught non-technical people how to program microcomputers, and from 1978-1982 he worked as a programmer/operator at a savings and loan association while earning his Bachelor of Science degree in data processing.

After leaving ANALOG in the summer of 1985, Tom wrote DEGAS for Batteries Included. He lives in Mission, Kansas where he is hot at work completing CAD-3D, a three-dimensional graphics system for the ST. If you want to talk to Tom, you can often find him on-line on the 16-bit section of CompuServe's SIG*Atari, where he recently became a SYSOP.

**Tom Jeffries** has been a professional musician for more than 15 years. He has played first trumpet with the St. Paul Chamber Orchestra and the San Jose Symphony; he has recorded with Dave Brubeck and numerous TV and radio shows. Though members of his family have been involved in computers since the 1950's, Tom managed to avoid serious work with them until 1984, when he became interested in the Commodore 64's SID chip.

Tom currently heads a company called Singing Electrons, developing and translating soundtracks and music-related software for microcomputers. He has written programs for most available micros, including the Atari ST, the entire Atari 8-bit line, the Amiga, the Apple II, the Commodore 64, and the IBM PC. In addition to the ST version of MIDImagic, mentioned in his article, Tom is currently completing an ST version of CZ PATCH, for Dr. T's Music Software.

**Daniel Matejka** has been programming professionally, on and off, for seven years. Three years ago, in Colorado Springs, he stumbled upon the crowd that makes and markets DB Master, a best-selling database on the Apple II which has been incarnated on several other machines as well. Dan was partially responsible for the IBM PC and Atari ST versions of that program.

Dan now lives in Fairfax, CA. He has since branched out and is writing programs independently. Antic's Disk Doctor is his, and Antic's forthcoming game, Red Alert, is the result of a collaboration with Stanley Crane, one of the original DB Master programmers.

**Tim Oren** is a familiar name among the community of ST developers. He is the author of ST PROFESSIONAL GEM, a biweekly GEM programming column available on Antic's Online CompuServe edition, and was a member of the original, Digital Research GEM development team.

At Digital Research, Tim designed and implemented the GEM Resource Construction Set, and worked on parts of the AES and the Desktop. Since leaving DRI, he has been the GEM interface designer for KnowledgeSet (formerly Activenture), which is planning to release a CD ROM-based encyclopedia for the Atari ST. Tim has a master's degree from Michigan State University. In his free time, he enjoys hiking throughout the glorious Monterey countryside, where he currently resides.

**David Small** was a longtime contributor to the now defunct Creative Computing magazine and a frequent contributor to Antic. His last piece demonstrated how to read and write IBM disks from the Atari ST (ST Uses IBM Disk Files, Antic, November 1985). David has published three books and written over a hundred magazine articles. He and his wife, Sandy, both have Computer Science degrees from Colorado State University. They are co-authors of "Guidebook for Winning Adventures" (Baen Enterprises, NY, NY), and their fourth book is due to be published January, 1987.

Though their two children purportedly keep them busy changing diapers, the Smalls have, nontheless, found time to complete their latest project: tricking the Atari ST into thinking it is a Macintosh. David has worked for several computer companies, and today is a consultant, freelance writer, and diaper changer. He briefly existed in San Jose, spent some time in exile in Austin, Texas, and now lives in Denver, Colorado.

**Russ Wetmore** attended Morehead State University with the idea of receiving a degree in music composition, but somehow ended up in the computer field. Maybe it was the influence of his father, who designs flight simulators for the military. In any case, Russ started in the late 1970's as a programmer with Adventures International, the Scott Adams company famous for its puzzling text adventures. While there, Russ wrote his first big hit, Preppie, memorable as one of the first Atari programs to use the vertical blank interrupt to support continual background music.

Russ then went off to form his own company, Star Systems. There, he wrote the integrated software package HomePak, which was snapped up by Batteries Included. He has since been kept busy adapting HomePak to most available microcomputer brands. The ST version should be available very soon. Russ is also a SYSOP on CompuServe's SIG Atari, and author of the terminal software for the 8-bit Atari's new XM301 modem. ■

# BE A HERO TO WINNIE THE POOH

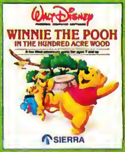
WALT DISNEY PERSONAL COMPUTER SOFTWARE

Last night the blustery wind came up and mixed up everything in the Hundred Acre Wood. If you can find the lost articles, like Pooh's honey pot and Eeyore's tail, and return them to their rightful owners, the characters in the Hundred Acre Wood will give a Hero Party in your honor!

Children can play this adventure game over and over again, and never play the same game twice. The "Save Game" feature allows players to work on one game in more than one sitting.

For ages 6 and up.
Reading comprehension, logical thinking, map reading skills.
For Apple II+/IIe/IIc, IBM, Atari ST, Commodore and Amiga.
At software dealers everywhere...or call toll free
800-423-5290 *outside CA,* 800-452-6000 *inside CA.*

*"...delightful graphics and an intriguing challenge."*
- Compute! magazine



SIERRA

## LOOK FOR WINNIE THE POOH AT THESE FINE DEALERS:

ARIZONA
**DSD Enterprises**
Masa
602-833-0303

CALIFORNIA
**Softwere World**
Bakersfield
805-322-3500
**Computertime**
Citrus Heights
916-989-4111
**Computer Insights**
Fremont
415-490-0685
**Boots Camera**
Fresno
209-432-0446
**Computer Place**
Downtown Mall
Redding
916-246-3282
**Home Computing Centers**
Tanforan Park Mall
San Bruno
415-588-1201
**Affordable Computer Systems**
San Jose
408-251-0102
**Home Computing Centers**
Bay Feir Mall
San Leandro
415-278-8881
**B & C Computervisions**
Santa Clara
408-749-1003
**Affordeble Computer Systems**
Santa Cruz
408-458-1644
**Computer Haven**
Upland
714-985-3278
**ECX Computer Co.**
Walnut Creek
416-944-9277

FLORIDA
**Computer Logic**
St. Petersburg
813-546-3137

GEORGIA
**Computer World**
Rome
404-232-7003

HAWAII
**Data 1**
Honolulu
808-946-1733
**PC Pricebusters**
Honolulu
808-523-1641
**Maui Microcomputers**
Kahului
808-877-7522
**Data 1**
Pearl City
808-487-5411

IDAHO
**Stokes Bros.**
Boise
208-323-1666

ILLINOIS
**General Computer Store**
St. Charles
312-584-6838

INDIANA
**World Central Computer**
Richmond
317-982-0995

KENTUCKY
**Softwere Source**
Louisville
502-456-4486

LOUISIANA
**ECO Data Centers**
Monroe
800-826-0606

MAINE
**Mementoe**
Caribou
207-492-1581
**Hends On Computer**
Westbrook
207-854-1155

MARYLAND
**Home Computers Company**
Towson
301-337-2733

MASSACHUSETTS
**Nichols Electronics**
Pittsfield
413-443-2568
**The Bit Bucket**
West Newton
617-964-3080

MICHIGAN
**Centon Computers**
Canton
313-459-4340
**Femily Computer Center**
Berkley
313-543-0520
**Renterteinment**
East Detroit
313-445-2983
**Software Plus**
Breton Village
Grand Rapids
616-942-7780
**Compusoft**
Kalamazoo
616-344-1184
**Castle Communications**
Lansing
517-371-4321
**Computer Depot**
Mt. Clemens
313-792-1717
**Commend Computer Systems**
Warren
313-573-8130
**Basic Bits & Bytes**
Westland
313-595-3171

MINNESOTA
**Wizard's Work**
New Hope
612-545-2136

MISSOURI
**Southland Electronics**
Aurora
417-878-4623
**Basic Computer Services, Inc.**
Florissant
314-837-4495

NEBRASKA
**Hobby Town**
Lincoln
402-464-2858

NEW JERSEY
**Computere Plus**
Bloomfield
201-429-8523
**Gemini Enterprisee**
Cedar Knolls
201-287-0988
**Family Computer Centres**
Fairfield
201-882-8370
**Softwere Station**
Morristown
201-455-7858
**Software Spectrum**
North Plainfield
201-581-8777
**Family Computer Centres**
Ridgewood
201-445-7503
**Femily Computer Centres**
South Orange
201-782-6661

NEW MEXICO
**Valliant Co.**
Albuquerque
505-247-4175

NEW YORK
**Buffalo Femily Computer Center**
Buffelo
716-835-0648
**Little Computer Shop**
Centereach
516-467-4352
**Senece Computer Systems**
Geneva
315-789-0734
**Computer Dome**
Johnson City
607-729-9222
**Software City**
Mt. Kisco
914-666-6038
**47th Street Photo**
New York
212-260-4417
**Omnis Computer Supplies**
Schenectedy
518-372-4491
**Software City**
Syracuse
315-445-2577
**Thruwey Computer Center**
Thruway Shopping Center Plaza
Walden
914-778-3535
**J.J. Stocker Co.**
Watertown
315-782-3780

NORTH CAROLINA
**Selective Softwere**
Arden
704-684-9997
**TDS Computers**
Carrboro
919-929-4595
**Softwere City**
Charlotte
704-366-5218
**TDS Computers**
Durham
919-288-3775
**Specialty Cash Register Co.**
Hickory
704-322-4884
**Software City**
Raleigh
919-833-1707

OHIO
**Magic One Computer**
Barberton
216-753-0431
**Futuretronics**
Elyria
216-322-2831
**Micro Weve Megic**
Fairfield
513-874-6560

OREGON
**Creative Computers**
Beaverton
603-644-1160
**Ruth Kahn Co.**
Bend
503-388-4807

PENNSYLVANIA
**Software Hut**
Philadelphia
215-462-2268

TEXAS
**Welk Office Systems**
Brownsville
512-548-0161
**Computer Discoveries**
Dallas
214-484-9104

VIRGINIA
**Fairfex Computer Products**
Feirfax
703-691-1930

WASHINGTON
**Butler's Computer Service**
Federal Way
206-941-9098
**Take-A-Byte, Inc.**
Kennewick
509-582-3311

WISCONSIN
**Starting Computers**
Brookfield
414-781-4444
**The Software Shop**
Fond Du Lec
414-921-8448
**Comp "U" Serv**
Milwaukee
414-649-8326

***SHIFT INTO...***

# SupraDrive
## Atari ST Hard Disk

ONLY **$549***
(10 MegaByte System)

"The necessary Peripheral for any serious Atari ST user."

Tom Hudson
Author of Batteries Included's
Best Selling **DEGAS** program.

- Improves the ST's overall performance (3-20X improvement on disk transfers)
- Atari™ compatible with software and other DMA Buss Devices
- Includes Hard Disk format and partition utilities
- High-Performance 20, 30, and 60MB SupraDrives are also available

*The 10MB SupraDrive is available only directly from Supra Corp. Higher capacity Drives are available from your local dealer.

**Supra Corporation**
1133 Commercial Way / Albany, OR 97321 / (503)967-9075

# CAPTURING MUSIC ON THE ST

## A MIDI SEQUENCER

by Tom Jeffries

Take advantage of your ST's built-in MIDI interface. This Simple Sequencer lets you record, save to disk, and playback music created on your synthesizer. Creates music files compatible with MIDIMagic, from Q-R-S Music Rolls. Pertinent files may be found on your START disk within the folder labeled MIDISEQR the basic principles of a sequencer. It is compatible with at least one commercial program that will provide better graphics, or you can add your own graphics routines. More on all that later.

## SEQUENCERS

Many kinds of information can be sent over MIDI, but the most important is note data: a steady stream of bytes specifying which keys are being pressed and released. A MIDI instrument can also receive such data and behave as though the data's source is the keyboard. In other words, it plays the notes being sent to it over MIDI just as though someone was playing the notes on the keyboard.

A sequencer can either save the data stream as it comes from the synthesizer (adding some kind of time code so that it knows note lengths) or send a previously saved stream of data to a synth to be played. You can buy a stand-alone dedicated sequencer with both hardware and software built-in, or you can use software to turn your micro into a sequencer.

In some ways a sequencer is like a tape recorder. You can play into it and save the result, or you can play back previously created music. However, there are some important differences. A tape recorder (at least a conventional analog tape recorder) reproduces in analog (infinitely varying) format any sounds that are sent to it. A MIDI sequencer can only record certain limited kinds of data: note on, note off, volume, etc. It cannot store a voice, for example, because vocal chords do not send MIDI information. Pitches are stored as note numbers with a range of 0 to 127.

There are, however, some things you can do with a MIDI sequencer that you cannot do with a tape recorder. For example, with a tape recorder you cannot change the speed of a piece without changing the pitch. A good professional sequencer can change the speed without changing the pitch—or change the pitch (transpose) without changing the speed. You can also change the sound on playback. Maybe you like the keyboard on one synthesizer but the sounds on another. With a sequencer you can record a sequence (a song or part of a song) using the keyboard you like, then play it back through the synthesizer that has the sounds you want. The only real limitation is that both synths must be MIDI-equipped.

Sequencers for professional use have some incredible features. Some of them take months to learn how to use, and took years to write. The Simple Sequencer I have written does not have lots of fancy features, but it can provide a lot of entertainment, and you can add your own features to suit your needs. With this sequencer you can play music on your MIDI-equipped synthesizer, "record" your playing in the memory of your ST, and, if you wish, save it on disk. You can play back anything you have previously entered, and change the tempo during playback.

## COMPATIBILITY WITH MIDIMAGIC

Our modern digital sequencers are not the first method of storing note data for later playback. The player piano was enormously popular near the beginning of this century. Composers like Scott Joplin or George Gershwin could cut a piano roll of one of their compositions and everyone with a player piano could buy a copy of the roll and listen to the music at home.

Q-R-S (Buffalo, NY) is the company that owns the rights to the old piano rolls. Many of these old, paper rolls are now being translated to MIDI-compatible disk files and distributed by a company called Micro-W (Butler, NJ), along with a program called MIDIMagic, which displays the songs on a graphic screen similar to that of the old player pianos.

I wrote the ST version of MIDIMagic, and since I knew I wouldn't have time to add fancy graphics to the Simple Sequencer, I made the files that the Simple Sequencer produces compatible with the files that MIDIMagic reads. If you own MIDIMagic, you can play the music you create with the Simple Sequencer with the MIDIMagic program and see what a piano roll created from your playing would look like. The piano roll offers an all-too-good display of keyboard technique, and, if you hold on to a note a little too long, you will be able to see it as well as hear it.

(Editor's note: *Your START disk contains two MIDIMagic songs, compliments of Micro-W, which may be played on the Simple Sequencer. Although the songs themselves are public domain, the actual performances are copyrighted and, therefore, may not be reproduced or resold. Any songs you create with Simple Sequencer are yours to do with as you like. You may not sell, copy, or distribute disks of your songs that use the MIDIMagic graphic driver program, which is copyrighted by Micro-W. However, Micro-W informs us that it would be interested in hearing any unusual compositions you create, for possible marketing.*)

## THE PROGRAM

To use the Simple Sequencer, first make sure your synthesizer is plugged in properly (one cable from the OUT socket of your synth to the IN socket of your ST, and a second cable from the IN socket of your synth to the OUT socket of your ST). Make sure your audio connections are all set and your synthesizer is on.

Alert boxes should guide you through the program fairly easily. Recording will not start until you press a key on your synth, so you can wait as long as you want before starting.

The program is a fairly standard

GEM application and has lots of comments, so I will not bore you with a line-by-line analysis. There are five main sections: Program Control (initialization, Main Loop, User I/O), MIDI Input, MIDI Output, Disk I/O, and Screen Displays.

## SCREEN DISPLAYS AND PROGRAM CONTROL

If you have been following Antic magazine's articles on the ST the screen display routines will not have any surprises for you. Note that, either by design or by accident, the screen resolution (as represented by the number returned by the **Getrez()** call early in the program) can be used to make automatic adjustments in the Y values for **v_gtext** calls as long as you are in either medium or high resolution. If you are using this program entirely on a monochrome monitor you may want to change to larger letters and reposition the text appropriately.

There is one point to note about the **Init()** function. I have arbitrarily set the maximum song file size to 37,000 bytes, which happens, for reasons unrelated to the ST, to be the maximum size for MIDIMagic song files. You can create larger files: just change the number #defined near the beginning of the program as **MAXSONG**. **Malloc(-1L)** will return the maximum amount of memory available. To maintain compatibility with different compilers (more on that subject later) I have used a temporary variable and a cast operation to set the pointer **notebuffer**. Alcyon C should allow you to eliminate the extra step; Megamax is stricter about data types and will only accept the syntax as is, since all bios and xbios functions are defined as returning a long word.

## MIDI INPUT

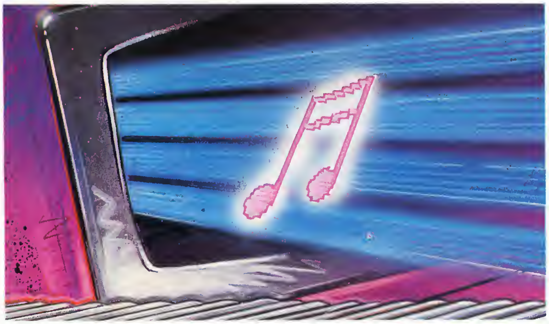
The MIDI input and output routines are the real meat of this program. **In_loop()** sets up the beginning of the file in MID- ▶

FIGURE 1.

| Note | Octave | MIDI# | Note | Octave | MIDI# | Note | Octave | MIDI# | Note | Octave | MIDI# |
|---|---|---|---|---|---|---|---|---|---|---|---|
| c | 0 | 0 | b | 2 | 35 | a# | 5 | 70 | a | 8 | 105 |
| c# | 0 | 1 | c | 3 | 36 | b | 5 | 71 | a# | 8 | 106 |
| d | 0 | 2 | c# | 3 | 37 | c | 6 | 72 | b | 8 | 107 |
| d# | 0 | 3 | d | 3 | 38 | c# | 6 | 73 | c | 9 | 108 |
| e | 0 | 4 | d# | 3 | 39 | d | 6 | 74 | c# | 9 | 109 |
| f | 0 | 5 | e | 3 | 40 | d# | 6 | 75 | d | 9 | 110 |
| f# | 0 | 6 | f | 3 | 41 | e | 6 | 76 | d# | 9 | 111 |
| g | 0 | 7 | f# | 3 | 42 | f | 6 | 77 | e | 9 | 112 |
| g# | 0 | 8 | g | 3 | 43 | f# | 6 | 78 | f | 9 | 113 |
| a | 0 | 9 | g# | 3 | 44 | g | 6 | 79 | f# | 9 | 114 |
| a# | 0 | 10 | a | 3 | 45 | g# | 6 | 80 | g | 9 | 115 |
| b | 0 | 11 | a# | 3 | 46 | a | 6 | 81 | g# | 9 | 116 |
| c | 1 | 12 | b | 3 | 47 | a# | 6 | 82 | a | 9 | 117 |
| c# | 1 | 13 | c | 4 | 48 | b | 6 | 83 | a# | 9 | 118 |
| d | 1 | 14 | c# | 4 | 49 | c | 7 | 84 | b | 9 | 119 |
| d# | 1 | 15 | d | 4 | 50 | c# | 7 | 85 | c | 10 | 120 |
| e | 1 | 16 | d# | 4 | 51 | d | 7 | 86 | c# | 10 | 121 |
| f | 1 | 17 | e | 4 | 52 | d# | 7 | 87 | d | 10 | 122 |
| f# | 1 | 18 | f | 4 | 53 | e | 7 | 88 | d# | 10 | 123 |
| g | 1 | 19 | f# | 4 | 54 | f | 7 | 89 | e | 10 | 124 |
| g# | 1 | 20 | g | 4 | 55 | f# | 7 | 90 | f | 10 | 125 |
| a | 1 | 21 | g# | 4 | 56 | g | 7 | 91 | f# | 10 | 126 |
| a# | 1 | 22 | a | 4 | 57 | g# | 7 | 92 | g | 10 | 127 |
| b | 1 | 23 | a# | 4 | 58 | a | 7 | 93 | | | |
| c | 2 | 24 | b | 4 | 59 | a# | 7 | 94 | | | |
| c# | 2 | 25 | *c | 5 | 60 | b | 7 | 95 | | | |
| d | 2 | 26 | c# | 5 | 61 | c | 8 | 96 | | | |
| d# | 2 | 27 | d | 5 | 62 | c# | 8 | 97 | | | |
| e | 2 | 28 | d# | 5 | 63 | d | 8 | 98 | | | |
| f | 2 | 29 | e | 5 | 64 | d# | 8 | 99 | | | |
| f# | 2 | 30 | f | 5 | 65 | e | 8 | 100 | | | |
| g | 2 | 31 | f# | 5 | 66 | f | 8 | 101 | | | |
| g# | 2 | 32 | g | 5 | 67 | f# | 8 | 102 | | | |
| a | 2 | 33 | g# | 5 | 68 | g | 8 | 103 | | | |
| a# | 2 | 34 | a | 5 | 69 | g# | 8 | 104 | | | |

Highest allowable MIDI note number.

* "middle" c on the piano



IMagic compatible format- tempo indication at byte 56, and note and time data starting at byte 64. It puts a short rest at the beginning (it sounds better that way), then calls a routine that reads the 200Hz timer to get a beginning time; all note timing data uses this as a reference point. The timer routine does not return to **In_loop()** until something comes in on the MIDI channel.

**In_loop()** then gets all available MIDI data and then does a little parsing to make sure that the contents of the file will work properly with the MIDIMagic display routines. Then it calls the timer again and waits for the next MIDI data. The note length is derived from the length of time between the first note data's beginning and the second note data. $FF signals timing data, unless it is followed by another $FF, signaling the piece's end. Because of the sign-extending which many C compilers perform on 8-bit numbers, the highest number that can be used for a note length is $7F. If the note is longer, the program will keep repeating $FF 7F until the variable **notelength** is below $7F.

When you press [Esc] to signal your recording's end, the timer routine tells **In_loop()** you are done by returning a 0. Three $FFs are added to the file's end to signal the end, and control is passed back to the **Loop()** in the Program Control section.

You will notice something rather odd in the way that the timer function reads location $04BA. $04BA, where the 200Hz click is stored, is in protected memory so you cannot read it from the 68000 chip's user mode. **(*pter)()** has to be defined as a pointer to a function so that the XBIOS function **Supexec** can be used to go into supervisor mode to read the time.

$04BA, where the 200Hz click is stored, is in protected memory so you can't read it from the 68000 chips's user mode. The XBIOS function **Supexec** allows you to go into supervisor mode, execute the function designated by your call to **Supexec**, and return to user mode. **(*pter)()** is therefore defined as a pointer to the time-reading function to be passed to **Supexec**.

**In_timer()** is a weak link in this program since the timings returned are not as accurate as a full scale professional sequencer would require. The solution, I think, would be to use one of the 68901 timers while shutting off as many of the other interrupts as possible.

I do not recommend using the **evnt_timer()** function in GEM: it seems to fail fairly often, especially when there is lots of I/O going on.

## MIDI OUTPUT

The output routine reverses the input routine. In this case I used simple nested counting loops for timing. Note that I keep checking for -1 for a timing byte or the end of the song. Many compilers will turn $FF into $FFFFFFFF or something equally obnoxious, so -1 avoids lots of problems. **Allnotesoff()** prevents notes from staying on when you quit in mid-song, by sending a MIDI "note off" signal to every possible note. MIDI does provide a special code for all notes off, but not every synthesizer implements it.

## DISK I/O

If you find **fsel_input()** or many of the other GEM gems confusing, I highly recommend studying DOODLE, which was written by Tom Rolander and Tim Oren and is the property of Digital Research, Inc. I borrowed **Setpath()** from DOODLE; you can do the same since they explicitly grant permission to do so as long as proper credit is given.

I also included two functions that can be found in the standard C libraries of most compilers: **Strlen()** and **Strcat()**. I did so in consideration for Alcyon C users, who, in order to use these two short functions, would otherwise have to link in 20,480 bytes of LIBF.

I want to point out one "undocumented feature" (some people call them bugs) of **fsel_input()**, since it caused me an inordinate amount of grief. **fsel_input()** sets the clipping rectangle for its own purposes and doesn't reset it on exit. If you are writing a program without windows and therefore don't expect to have to worry about clipping, this "feature" can cause a bug that is difficult to track down.

I have tried to make this program as compatible as possible with different compilers. If you are using Megamax, delete or comment out the line #define of **Supexec**.

## POSSIBLE ADDITIONS TO THE SIMPLE SEQUENCER

One of the nice things about having a program published in START is that you, the reader, can enhance the program. The Simple Sequencer could use several additions, some mentioned above. For example, fancy graphics would be nice. There are several small routines I did not put in; an available memory indicator, for example, or some code allowing the user to preset the playback tempo instead of having to wait until the song starts. (Of course you can use a disk editor to change the tempo byte before loading the song, but that's not exactly elegant.) A file mixing program could take several tracks recorded with the Sequencer and mix them down to a single track for playback.

For those of us who are real klutzes at the keyboard, a good step editor—permitting composition one note at a time—would be nice so that we could compose a song on the computer keyboard at our leisure and have it sound like we had performed it on the synthesizer.

If you really want to write some music without playing it on your sequencer and cannot wait to get (or write) a step editor, you can use any program that allows you to construct a file one byte at a time (like the SID.PRG that comes with the developer's kit). Make sure you allow the proper header space and put the tempo byte in the right place (byte 56 with the first byte being byte 0), and start your music data at byte 64, preferably with a short rest ($FF $14 works well). Each note requires three bytes to start, $90, the note number (see *figure* 1), and a key velocity (unless you know what you are doing use $40). There are several ways to turn off a note, but for compatibility with MIDIMagic use $90, the note number, and 0. Good luck.

## MORE ABOUT MIDI

MIDI is a complex and sometimes problematic tool. Each instrument maker seems to have implemented the MIDI standard a little differently, so things that work on one synthesizer may not work on another. By the time you see this, the Simple Sequencer will have been tested on a variety of synths, but if you run into problems please let me know through START Magazine.

One further note: do not try to send program change messages or Channel Pressure messages to the ST without changing the parsing routine in **In_loop()** appropriately. These status bytes are used so rarely I decided not to include the extra code, if you use them you'll have to make the necessary adjustments.

I hope that the Simple Sequencer is as much fun for you to use as it was for me to write. I'm really looking forward to seeing what people come up with as enhancements. ■

## Reference:

- ANTIC magazine, *Play it Again, Atari*, June 1985; *Midi Driver*, March 1986.
- Digital Research, Inc., 60 Garden Court, P.O. Box DRI, Monterey, CA 93942, (408) 649-3896.
- The International MIDI Association, 11857 Hartsook St., North Hollywood, CA 91607. (818) 505-8964.
- *KEYBOARD* magazine, 20085 Stevens Creek Boulevard, Cupertino, CA 95014-9967, (408) 446-1105.
- Micro-W Distributing, 1342B Route 23, Butler, NJ 07405, (201) 838-9027.
- Q-R-S, 1026 Niagra Street, Buffalo, NY 14213, (716) 885-4600.

**COPY II ST™**

**BACKUP PROTECTED SOFTWARE FAST.**

From the team who brought you COPY II PLUS (Apple), COPY II PC (IBM) and COPY II MAC (Macintosh) comes a revolutionary new copy program for the Atari 520 and 1040 ST computers.

- Copies many protected programs—automatically. (We update COPY II ST regularly to handle new protections; you as a registered owner may update at any time for $15 plus $3 s/h.)
- Supports single and double sided drives.
- Includes both a fast sector-based copier and a true bit copy mode for protected disks.

Requires an Atari 520 or 1040 ST computer with one or two drives.

**Call 503/244-5782**, M—F, 8—5:30 (West Coast time) with your VISA in hand. Or send a check for $39.95 U.S. plus $3 s/h, $8 overseas.

**$39.95**

Central Point Software, Inc.
9700 S.W. Capitol Hwy. #100
Portland, OR 97219

Central Point Software INCORPORATED

*Backup utilities also available for the IBM, Apple II, Macintosh and Commodore 64.*
*This product is provided for the purpose of enabling you to make archival copies only.*

# Finally, a magazine devoted entirely to the Atari 520 and 1040 ST:

THE ST QUARTERLY
START™

Look to *START* for comprehensive articles and programs covering applications, programming theory and techniques, languages, and products, written by the top developers in the Atari and 16-bit world for both new and experienced Atari users.

*Subscribe today and receive the most complete ST coverage available!*

- Get ST disk-based programs with every issue.
- Explore the GEM system and ST graphics.
- Get in-depth reviews of important products, and exclusive product news.
- Learn 68000 assembly programming, and more!

**4** issues of *START,* the ST Quarterly (with disk), plus 12 issues of Antic (ST Resource included). $59.95

Mail the attached card today!

ATARI

# The Price War Is Over

*We Will Beat Any Price. Anywhere.*

## Atari ST Hardware

| | |
|---|---|
| Atari 1040ST | Call |
| Atari 520ST RGB | Call |
| Atari 520ST Mono | Call |
| Atari 520ST (CPU) | Call |
| Atari SF354 | Call |
| Atari SF314 | Call |
| Atari SM124 | Call |
| Atari SC1224 | Call |
| ST Station | Call |
| Supra Hard Disk | Call |

## Printers

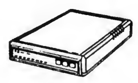
| | |
|---|---|
| Panasonic | Call |
| Epson | Call |
| Citizen | Call |

## Modems

| | |
|---|---|
| Avatex 1200 | $89.00 |
| QMI 1200ST | $129.00 |
| Hayes 1200 | Call |
| Atari XM301 | $37.95 |
| Supra/MPP 1000E | $37.50 |

*June Specials*

| | |
|---|---|
| Avatex 1200 PR Conn./Amodem | $149.00 |
| Atariwriter Plus | $24.95 |
| Avatex 1200 | $89.00 |
| Sakata SC100 | $128.95 |
| Teknika MJ-10 | $178.00 |
| Teknika MJ-22 | Call |



**Avatex 1200**

*$89.00*

*Hayes Compatible! Everyday Low Price!*

## Atari ST Software

| | |
|---|---|
| Borrowed Time | $31.00 |
| Chat | $13.00 |
| Final Word | $79.00 |
| Haba Software | Cheap |
| Hacker | $28.00 |
| H & D Base | Call |
| H & D Forth | Call |
| H & D Toolbox | Call |
| PC Intercomm | $69.00 |
| Personal Diskit | Call |
| Sundog | $22.00 |
| VIP Professional | Call |
| Silent Service | $24.00 |
| Flight Simulator | Call |
| Micro C Shell | Call |
| Mindshadow | $31.00 |
| Print Master | $24.95 |
| P. M. Art Gallery | $19.95 |
| Easy Draw | Call |
| Graphic Artist | Call |
| Music Studio | $37.50 |
| 2Key Accounting | $32.99 |
| Home Planetarium | $19.75 |
| Joust | $19.75 |
| CP/M Emulator | $32.99 |
| Apple ][ Emulator | Call |
| IBM Emulator | Call |
| Megamax C | Call |
| Metacomco Pascal | Call |
| Lattice C | Call |
| GST C | Call |
| Andra | Call |
| UCSD Pascal | Call |
| Regent Spell | Call |
| Regent Word | $35.00 |
| Regent Word II | $35.00 |
| Regent Base | Call |
| VIP Lite | Call |
| Micro C Shell | Call |
| Rubber Stamp | Call |
| Megafont II | Call |
| Meta 68000 Asm | Call |
| Brattacas | Call |
| Degas | $24.95 |
| Personal Pascal | $39.00 |
| ST Talk | $13.00 |
| Typesetter ST | $24.00 |
| Zoomracks | $48.99 |
| Modula-2 | Call |
| Personal Prolog | Call |
| Kings Quest II | Call |

## Atari 8-Bit Hardware

| | |
|---|---|
| Atari 130XE | $115.00 |
| Atari 65XE | Call |
| Atari 1050 | Call |
| Indus GT | $188.00 |
| Atari XC11 | Call |
| Atari 1027 | $98.00 |

## Accessories

| | |
|---|---|
| US Doubler | $42.00 |
| Rambo XL | $29.95 |
| R-Time Cart. | $43.00 |
| The Duplicator | Call |
| Indus Expander | Call |
| Bonus SS/DD 5.25 | $6.99 |
| Bonus DS/DD 5.25 | $7.50 |
| Flip & File All Sizes | Call |
| Paper 1000 Sheets | $11.00 |
| Paper 2500 Sheets | Call |
| Maxell 3.5 | $19.00 |
| Sony 3.5 | $21.00 |
| Fuji 3.5 | $19.00 |

## Interfaces

| | |
|---|---|
| P:R: Connection | $55.00 |
| Supra Microprint | $29.00 |
| Supra 1150 | $39.00 |
| Supra Micronet | $149.00 |
| Supra Microstuffer | $57.00 |
| U-Print | $48.00 |
| Apeface | Call |
| Atari 850 | Call |
| ST Printer Cable | $8.99 |

**Atari 130XE**

*$115.00*

*128K Of Memory! Lowest Price Anywhere!*

## Atari 8-Bit Software

| | |
|---|---|
| Acrojet | $19.50 |
| Action! | $42.00 |
| Basic XE | $42.00 |
| Basic XL | $33.95 |
| Battalion Command | $24.95 |
| Battle of Antietam | $32.00 |
| B/Graph | $24.95 |
| Broadsides | $24.95 |
| Computer Baseball | $24.95 |
| Computer QB | $24.95 |
| Conflict in Nam | $19.00 |
| Crusade in Europe | $24.95 |
| Silent Butler | $19.99 |
| Gemstone Warrior | $22.00 |
| War in Russia | $48.00 |
| Dec. In the Desert | $24.95 |
| Kennedy Approach | $19.50 |
| Solo Flight | $19.50 |
| Learning Phone | Call |
| Syncalc 130XE | $29.95 |
| Megafont II+ | $17.00 |
| Rubber Stamp | $18.00 |
| Paperclip | $35.99 |
| Mac/65 | $42.00 |
| Print Shop | $26.00 |
| Karateka | $18.50 |
| Lode Runner | $22.00 |
| Atariwriter Plus | $24.95 |
| Panzer Grenadier | $34.00 |
| Hacker | $19.50 |
| Space Shuttle | $15.75 |
| Mindshadow | $15.75 |
| Great Am. Rd. Rce | $15.75 |
| F-15 Strike Eagle | $24.95 |
| Silent Service | $19.50 |
| Gunship | $19.50 |
| Synfile 130XE | $29.95 |
| Page Designer | $18.00 |
| Typesetter | $21.00 |
| HomePak | $31.00 |
| OSS Toolkits | $17.99 |
| Printshop Lib. 1/2/3 | $15.95 |
| Chmp. Lode Run | $18.50 |
| Ultima IV | $39.00 |
| Sparta Dos CS | $24.95 |
| Home File Manager | $7.99 |
| Music Painter | $7.99 |
| Star Raiders | $7.99 |

*Toad Services*

**Black Patch Systems**

**TO ORDER (Visa, MasterCard): Call TOLL FREE 1-800-ATARI-02**

For technical information, order inquiries, or for MD orders call 301-757-1329, or write
Black Patch Systems, P.O. Box 501, Arnold, MD 21012

**Risk Free Policy:** In-stock items shipped within 24 hours of order. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the continental U.S. Volume discounts available. MD residents add sales tax. APO, FPO, and international orders and $5.00 plus 3% for priority mail service. Advertised prices show 4% discount for cash, add 4% for MasterCard or Visa. Personal checks require 4 weeks clearance before shipping. Ask about UPS Blue and Red label shipping. All merchandise carried under manufacturer's warranty. Free catalog with order. All items subject to change without notice. **Dealer Inquiries Invited**

# STEALING THE ST PRINTER DRIVER

## ST SCREEN DUMPS FOR ANY PRINTER

by Tom Hudson

*This Desk Accessory lets you install your own customized printer driver for either vertical or horizontal screen dumps at the press of the [Alternate] [Help] key combination. Printer drivers are included on the START disk and several can also be found on-line on CompuServe. TOS must be in ROM for this program to operate properly. Programs related to this article may be found in your START disk within the folder labeled PRNTDRVR*

If you don't own an Epson FX model printer, or one of its compatibles, you may find your ST screen dumps leave a lot to be desired. That masterful work of art may look like it was filtered through venetian blinds—or worse, it may be just a jumble of nonsense letters. Or, how about an infinite form-feed?

One of the features most often mentioned in "wish lists" for the Atari ST is the ability to load screen printer drivers for various printers other than those built into the ST. There are several reasons one might want to print the computer screen—aside from impressing friends with printer artwork. Printers such as the Okimate 20, for example, can be a perfect source of low-cost overhead transparencies by transferring their see-through, plastic-like pigment to acetate sheets. Or, if a bug appears in a commercial program, you may document the problem by dumping the screen to the printer and forwarding it to the company.

Fortunately, when Atari created the ROM version of TOS, they created a vector which is called to perform the graphic screen dump process.

This article will describe the screen-dump process and how to alter the screen-dump vector to point to a user-installed printer driver routine. Since there are a good number of printer drivers already written for the DEGAS paint program, the screen-dump routine installer presented here will follow the DEGAS printer driver standard. (DEGAS printer drivers can be found in DL4 of the Atari16 section of CompuServe's SIG*Atari, as well as on the DEGAS disk. We have also include some on the START disk with the extender .PRT.)

If you wish to use the installer immediately, skip to the section USING THE INSTALLER, and have fun! ▶

## STEALING . . .

### THE BUILT-IN ROUTINES

Each ST computer comes equipped with a simple screen-dump routine in TOS designed primarily for Epson-compatible printers. It may be started by pressing [Alternate] and [Help] simultaneously, or by calling the Atari BIOS trap #14 exception processor with a parameter of 20 decimal (Scrdmp). The GEM Desktop also has a convenient drop-down menu selection which starts the screen-dump routine.

For the [Alternate] [Help] version of the screen dump, the ACIA Receiver Buffer Full Interrupt routine performs a series of tests to determine the type of interrupt that occurred. If it is an intelligent keyboard (IKBD) interrupt, the key combination pressed is checked to see if it is [Alternate] [Help]. If so, the interrupt sets a flag telling the vertical-blank handler to start the screen dump. The flag that gets everything started is PRTCNT, which is located at $4EE. Normally, the value stored in PRTCNT is a -1 (word). When the Alt-Help keystroke is detected, the interrupt increments PRTCNT, making it zero.

The system vertical-blank (VBLANK) handler is what actually starts the screen dump. Each time the code is executed (50 or 70 times per second, depending on the type of monitor you are using), it tests the PRTCNT location to see if it is zero. If not, the VBLANK routine proceeds normally. However, if PRCNT contains zero, the VBLANK executes the SCRDMP routine to dump the screen to the printer. In the RAM-based TOS, this is a direct BSR (Branch to SubRoutine) instruction. In the ROM TOS, the SCRDMP routine is executed via an indirect JSR (Jump to SubRoutine) to the address contained at $502. This routine does all the set-up work and calls a routine called PRTBLK, which does the job of printing the specified portion of the screen (in the case of the Alt-Help or Scrdmp functions, the entire screen is dumped to the printer).

During the screen dump, the dump routine watches the PRTCNT location to see if it changes from its zero status. If so, the user has pressed Alt-Help again indicating he or she wants to abort the screen dump operation, and the dumper will exit. After the dump is complete, the PRTCNT location is reset to -1 and system operation returns to normal.

Interestingly, Atari has a vector to the screen dump routine, but none to the PRTBLK routine. Logically, PRTBLK should have had a vector pointing to it, since it does the actual printing. A pointer to a table containing all the printing parameters is passed to PRTBLK, but without a vector that we can alter, this information is useless.

The key to harnessing the screen-dump routine and having the system execute our own screen-dump code is the screen-dump vector at $502. We install our routine in a safe portion of memory and repoint the screen-dump vector to our code.

When a system trap instruction is used to start the screen dump, the trap handler simply calls a routine which sets PRTCNT to zero, simulating the action of the Alt-Help keystroke. It then calls the SCRDMP, either by a direct BSR (RAM TOS) or via the screen-dump vector at $502 (ROM TOS). After the screen-dump code returns, the PRTCNT flag is reset to -1, and the system processing returns to normal.

Since there are two ways of initiating a screen dump in the ST (Alt-Help and TRAP #14), the programmer must be sure that each calling method will properly call the installed dumper. In the ROM TOS, this is no problem—one vector change takes care of all the screen dump possibilities. In the RAM TOS, the programmer must install a pre-VBLANK processor *and* a pre-TRAP 14 handler which will intercept the Scrdmp call and execute the desired user-installed code.

For the sake of simplicity, this article assumes that the ROM TOS is installed. Atari says the ROM TOS is installed in a majority of machines at this point, making the restriction of ROM TOS a minor one.

### WHAT ABOUT THE CONFIGURATION?

The built-in screen-dump routine uses a special printer configuration word, PCONFIG, to determine what type of printer is being used. This word has bits that refer to the port (serial or parallel), the type of printer (color or black & white), the mode (draft or final), and so on. The PCONFIG word is set up by the "Install printer" desk accessory provided with the ST.

The location of PCONFIG has not been documented by Atari as of April 1986, but in the ROM TOS it is located at $E4A. I don't recommend using this location in any programs until Atari guarantees that it will not change. DEGAS drivers are typically set up for one configuration anyway, so they do not need to look at the PCONFIG word.

### THE PRINTER DRIVERS

When I was writing DEGAS, in June of 1985, the support of various printers, including color printers, was a major concern. The existing screen-dump routine in the ST was a simple, black & white driver for Epson printers that only printed images with four levels of gray-scale. I was not satisfied with the output, and decided that since no mechanism was then available to load screen-dump drivers into the system, I would create a standard driver format for DEGAS. This driver format, while not infinitely flexible, will allow screen dumps in color or black & white, to the parallel or serial port, to impact or laser printers, or even to plotters.

The drivers are not limited to simple screen dumpers, either. Since the driver is passed a set of parameters giving the color palette and screen address, a number of useful utility routines could be installed with DEGAS thinking they were printer drivers. Two that come to mind are a color rotation handler for color palette animation and a screen "clipping" routine which would allow the user to define an area of the screen to be held in a buffer for later use. The possibilities are endless.

The DEGAS printer drivers are designed to be a block of executable, position-independent 68000 object code 2000 bytes in length, which begin execution at the first byte in the block. The 2000-byte length was arbitrarily assigned, and is adequate for black & white and color drivers alike, with careful programming techniques.

The drivers are called with the C statement:

```
scrdump(command,resolution,
  screen,palette,workarea)
int command,resolution;
long screen,palette,workarea;
```

If the word **command** is a zero, the driver should initialize itself and the printer and return with a 1 in D0.W if the operation was successful, or a zero if there was an error. Typically, the initialization function sends the printer a command to set the linefeed and graphics mode accordingly, to check to see if the printer is connected and powered on. The printer driver is ALWAYS initialized before each screen dump.

If **command** is a 1, the driver is to perform the screen-dump function. During this process, the driver continually tests the keyboard to see if [Undo] has been pressed, and abort if it has. A successful screen dump returns a 1 in D0.W. An aborted screen dump returns with a 2 in D0.W. If any error occurs during the screen dump, the driver returns with a zero in D0.W.

The word **resolution** indicates the graphics mode that the driver is to use in dumping the screen to the printer. A zero in this value indicates the 16-color, 320-by-200 pixel low-resolution mode, a 1 indicates the 4-color, 640-by-200 pixel medium-resolution mode, and a 2 indicates the 640-by-400 pixel monochrome mode.

The longword (four-byte) **screen** is a pointer to the base address of the screen data that will be used for the screen-dump operation.

The longword **palette** is a pointer to an array of 16 word values that contain the colors used on the screen. These are in the standard form, in which the low three nibbles contain the red, green and blue settings of that color. These nibble values range from 0-7. For example, if a particular color palette entry contains $0456, the red level is set at 4, the green level is 5, and the blue level is 6. These entries can range from $0000 (black) to $0777 (white).

**When Atari created the ROM version of TOS, they created a vector . . . to perform the graphic screen dump.**

Longword **workarea** is a pointer to a 1280-byte (once again, an arbitrary length) portion of memory reserved for the driver's use. The driver can put whatever information it likes in this area, including print buffers and working variables. Since the driver code must be position-independent, this is the best place to put variables.

Commented source code for two DEGAS drivers (the Epson-compatible black & white driver and the Epson JX-80 color printer driver) are included on the START disk as examples so other programmers can see how one is written. Many printers are very similar and will only require the modification of various control codes to produce an operating driver. Others may require a bit more work, and I explain possible solutions below.

Note that the Epson black & white driver can provide screen dumps in two different formats, and uses the [Alternate] key to determine which format the user wants. If [Alternate] is not pressed, the driver prints a large image of the screen sideways on the page. If the key is pressed, the screen is printed in a vertical format, slightly smaller. If you write a driver for another black & white printer, it is a good idea to to maintain this "standard." Color printer drivers rarely have enough room to fit both horizontal and vertical routines in 2000 bytes, so they are written to be vertical-format only.

## THE EPSON BLACK & WHITE DRIVER

Take a look at the EPSON.S file on the START disk. The first thing you notice in this driver is that it saves all the 68000 processor registers in the work area, using A0 to point to the work area's start. The A0 register is maintained throughout the driver code, and always points to the work area. All working variables (**PHASE**, **KBSHIFT**, etc.) are set up as offsets from this address. **PHASE** is always referenced as **PHASE(A0)**. This is necessary because the driver code must be position-independent and absolute addresses cannot be used. If the 1280-byte work area is insufficient, the **Malloc** function can allocate needed memory. Because of a bug in the current GEMDOS **Malloc** call, ▶

this procedure is not recommended. So far, all printer drivers have been written using just the 1280-byte work area provided, with room to spare.

The driver uses a bit mask byte, **PHASE**, to mask off data to be added to the print data buffer. As each line of pixels on the computer screen is processed, **PHASE** is shifted right one or two bits, depending on the resolution. The bits that are on in **PHASE** determine which pins in the print head are used for that line of pixels. When **PHASE** is initialized to 192 decimal ($C0 or %11000000), the mask is set for two print head pins. When it is set to 128 decimal ($80 or %10000000), it is set for one pin. As the mask is shifted to the right, it finally becomes zero. At that point, a print line 8 dots high is ready to be printed, and the **PHASE** mask is reset.

An important part of the printer driver is the code labeled **printit:**. This is a useful subroutine which takes care of several concerns. First, it provides a single subroutine which prints any number of bytes to the Centronics parallel port. Second, it takes care of printer time-out handling just in case the user tries printing out a picture without a printer connected. If the driver makes 270,000 unsuccessful attempts to send a byte to the printer, it returns an error code. More than a quarter of a million tries may sound like a lot, but this takes about 30 seconds, a good figure for most printers—especially those which shut down when the print head reaches a certain temperature. The 30-second time-out value also allows certain printers to operate at their slowed-down speed without causing the screen dump to abort. If the operation was successful, **printit:** returns with a zero in D0.W. If it failed, **printit:** returns a -1.

Finally, **printit:** takes care of saving the 68000 registers D1-D2 and A0-A2, some of which may be altered by the TRAP instructions used to send data to the printer.

Both printer drivers included on the disk contain their own routines for getting pixel values from the screen. While these routines take up precious room in the 2000-byte driver file, they are faster than going through the Line A "get pixel" routine. If you find yourself needing room in the driver, you may want to use the Line A calls to save space, at the expense of some speed.

The horizontal-format screen dump prints an image 800 printer-dots wide, an even multiple of the 200 or 400-pixel high screens. This is an easy conversion for the driver.

The vertical-format dump is a little more complicated. This dump is 960 dots wide, which is a multiple of three for the low-resolution 320-pixel-wide mode, but only 1.5 for the medium and high-resolution modes. To take care of this, the driver outputs two printer dots for each pixel with an even X coordinate, and only one each for pixels with an odd X coordinate. Thus, vertical-format screen dumps may show strange effects on vertical lines.

Another important routine in the black & white driver is the **ppix:** subroutine, which converts the luminance level of a given color register value to a number ranging from 0-7. This gives standard DEGAS screen dumps 8 levels of gray-scale, a good range for most pictures. The number of gray-scale levels could be set to any value, depending on the programming of the driver. This gray-scale value is then used as an index into the gray-scale pixel table, which is built during the driver initialization at 1200(A0).

This driver assumes that the print head is configured with bit 0 at the head's bottom. If you want to modify this driver for a printer with bit 0 at the top, you'll have to flip the bits in the **PHASE** byte and reverse the directions of their shift operations.

If you are using a printer such as the Okidata Microline 193 which uses the character $03 to control graphics functions, you'll have to set up a special intercept routine in **printit:** which takes care of graphic data bytes with an $03 value.

## THE JX-80 COLOR DRIVER

One of the ST's big selling points is the ability to generate stunning color graphics, and as a result, many users will want to produce color printouts of their graphics masterpieces. Fitting a color printer driver into a 2000-byte block of code was a real challenge, but I think the solution used in the JX-80 color driver is effective. It produces fairly accurate representations of colors very efficiently.

The JX-80 color printer driver is on the START disk as JX80C.S Most of the basic routines used in the JX-80 driver will look familiar; they were lifted verbatim from the DEGAS Epson driver, and perform the same functions. There are several significant exceptions.

You will notice that there are now two tables built during the initialization phase. The first, **BLACK** (located at 200(A0)), is a gray-scale table in which each entry is a 16-bit table. Each of these table entries is a 4-bit by 4-bit mask (16 bits total) which is overlayed on the printer dot pattern. The table ranges from black (all bits on) to white (all bits off).

The second table, **WHITE** (located at 300(A0)), is a white-scale table similar to **BLACK**. Instead of adding black as the brightness of a pixel diminishes, the **WHITE** table adds white as the brightness of a pixel increases. We will see how this works in a moment.

The process of printing a color picture on most personal color printers is the same, regardless of the printer. There is usually a multicolor ribbon containing yellow, magenta and cyan portions. Most color impact printers also have a fourth color, black, for normal printing operations, such as program listings. Our color printer driver is simplified because it does not use the black portion of the ribbon. We get black by mixing the other three colors.

To make the various colors needed for each screen-dump line, the print head must make three separate passes over each print line, one in each color. It is best to make the three color passes starting with a yellow pass, then a red pass, then a cyan pass, to reduce color contamination problems on the ribbon. If, for example, the cyan pass was made first, followed by the yellow pass, the yellow ribbon would pick up cyan ink from the paper, making the yellow ribbon more of a green color. The same applies to the magenta ribbon, which is less contaminated by yellow than by cyan.

Since there are only three basic colors present on the ribbon, some sort of algorithm is needed to mix them on the paper to produce colors that approximate those on the computer screen. This is accomplished by the combination of a large table, **cvalues:**, which provides color information, and a simple algorithm for adjusting brightness.

**cvalues:** is at the end of the listing. It contains 512 one-byte entries (one byte per color possible on the ST) made up of two groups of three bits each (the high-order two bits are not used). Each three-bit group represents a printer color to be used for the corresponding screen color, and the two colors specified in the byte are mixed in a 50-50 combination on the paper. The color specified by the low-order three bits is used on even printer dots, and the color specified by the next three bits is used on odd printer dots. For a solid color, the two colors are set to the same value.

The three bits give a total of eight color combinations, of which white and black are not used, leaving six colors to be mixed. The total number of combinations of the various colors is 20. Twenty colors out of 512 would not be a very good proportion, so additional processing is done using the **BLACK** and **WHITE** tables.

Each pixel's color is analyzed according to the following steps:

1. The total brightness of the pixel's red, green and blue components is calculated, ranging from 0-21.

**The key to harnessing the screen-dump routine ... is the screen dump vector at $502.**

2. If the color's table entry is zero, the program goes to the **grayit:** routine, which calculates a level of gray-scale for the pixel, similar to the process used in the black & white driver. This is a special case for all colors which have the same red, green and blue levels, and produces a good gray.
3. If the pixel's brightness is less than 7, black dots are added from the **BLACK** table (see the **blakck:** routine).
4. If the pixel's brightness is equal to 7, the color is left as is (no adjustment of the black & white content).
5. If the brightness is greater than 7, white pixels are added from the **WHITE** table (see the **whitck:** routine).

With the addition of varying levels of black and white to the original color mix, the driver can create a total of 20 (basic colors) * 20 (6 black addition levels + normal + 13 white addition levels) or 400 colors, as well as eight levels of gray scale. While the colors may not match the screen image exactly, the color approximations are quite acceptable for general-purpose use.

## PRT FILES

A special procedure is required to transform driver source code to .PRT files. Included on the START disk is SAVER.C, the C source code for a program which will save assembled printer drivers to disk in the standard DEGAS .PRT format. To use SAVER.C, compile it to an object file using your C compiler. Next, assemble the assembly source for the printer driver you want to save into an object file. Now, link these files together using LINK68 or a comparable 68000 linker program to produce an executable .PRG program file. (See SAVER.BAT on your START disk for an example of using the EPSON.S driver.)

Remember that to successfully link the SAVER.C and the printer driver, the printer driver's executable code MUST start with the label **s_dumper**, so that SAVER.C knows where the driver code starts.

Once you have linked SAVER.C and the printer driver into an executable program (such as SAVER.PRG), you must execute the SAVER.PRG program to create the final printer-driver file. The program displays the file selector box. Enter the name of the printer driver file (be sure to use an extension of .PRT on the filename). The program will report the success or failure of the write operation and return to the desktop. If the file write was successful, the printer-driver file is ready to use with DEGAS or the printer driver installer program. ▶

## STEALING . . .

### THE INSTALLER

Also on the START disk are the files INSTALL.ACC, INSTALL.C and INSTAL.S. INSTALL.ACC is a GEM Desk Accessory which performs the installation of a DEGAS printer driver. INSTALL.C is the C source code for the accessory, and INSTAL.S is the AL source code for the assembly language portion of the accessory. These two source files must be compiled and linked with ACSTART.O in order to produce an executable desk accessory. (See INSTALL.BAT on your START disk.)

The C source code is fairly straight forward, except for a couple of interesting points.

When started by the GEM desktop code, the installer displays a two-button alert box which gives the user the choice of a VERTICAL or HORIZONTAL printout. This option is very important. You'll recall from the above discussion on printer drivers that the DEGAS drivers examine [Alternate] when initialized to see if it is being pressed. If it is, the driver shifts to a vertical print format; if not, the screen dump is printed sideways. Unfortunately, when the screen dump is initiated by the Alt-Help keystroke, [Alternate] will ALWAYS be pressed as the driver is initializing, and therefore all Alt-Help dumps would be printed vertically!

This was an undesirable result, but fortunately the ST BIOS has a way to get around the problem. There is a TRAP #13 call included in the OSBIND.H file called **Getshift** which allows you to get or set the status of the keyboard's Control, Alternate and Shift keys so that the next call to **Getshift** will think certain keys are being pressed. Some early versions of the OSBIND.H file contained an error in the way the Getshift call was #defined. Take a look at your OSBIND.H file and be sure the definition reads:

```
#define Getshift(a)   bios(11,a)
```

The earlier definitions left out the parameter specifier (a).

The first alert box allows the user to tell the installer which type of output is desired before loading the printer driver. Furthermore, if the user wants to change the output format at a later time, he or she simply selects the "Printer Driver" accessory and clicks on the appropriate button to change the output format. The result of the alert is determined and the variable **prtmode** is set to either 0 (no shift keys pressed) or 8 (Alternate key pressed). Use this variable to set the shift key state when the screen dump is called.

**Once a printer driver is installed, you can print the screen to whatever printer the driver is compatible with.**

The program then calls the GEM file selector dialog and accepts a printer-driver filename. The file is opened and read into the address of the **scrdmp()** function. This function is defined in the INSTALL.S file, and is nothing more than 2000 bytes of reserved memory. If the file read into memory was less than 2000 bytes in length, an alert box appears informing the user, and the installer aborts without installing a driver.

If the file is the correct length (no check is made to see if the file is over 2000 bytes, because some drivers downloaded with terminal software may have extra bytes tacked onto the end of the file), the driver is installed by calling the **install()** function. This simply replaces the screen dump vector at $502 with the address of our screen dump control routine, **dumpctrl()**.

Whenever the system is asked to perform a screen dump from this point on, the VBLANK code will call **dumpctrl()** to do the dump. **dumpctrl()** requests the system screen resolution, the physical base address of the screen and the color palette, then sets the keyboard shift bits according to the contents of **prtmode** and initializes the printer driver (command = 0).

The printer driver returns a success code of 0 or 1. If the value is 1, the initialization was a success, and the printer driver is called again with a command word of 1, telling it to dump the screen. When complete, control returns to the system.

### USING THE INSTALLER

First, you must have TOS in ROM in order to use this desk accessory. To get the printer driver installer up and running on your ST, just copy the INSTALL.ACC file from the START disk to the disk you use to boot your system. You should also place an appropriate printer driver file on this disk. If you can't find the right one on your START disk, several have been placed on CompuServe. Now, boot your ST with the disk containing INSTALL.ACC and the Desk drop-down menu will have a new entry called "Printer Driver."

To install a particular printer driver on your computer, select the "Printer Driver" menu item. A dialog box will appear giving the credits for the installer accessory. At this point, you will have two options: Horizontal or Vertical. A horizontal screen dump is printed sideways on the printer paper in a fairly large image size. A vertical screen dump is printed right-side-up on the paper, slightly smaller. Choose your format. You can always change it later.

The computer will display the file selector dialog, showing all the printer driver files (.PRT) on the current disk. Pick your printer driver and click on "OK" (if you already have a driver installed and just wanted to change the format from horizontal to vertical, you can click on "Cancel" to avoid reloading the file).

You will be notified of problems in installing the printer driver. Be sure you always have a successful load before trying to perform a screen dump. Failure to do so could lock up your computer.

Once a printer driver is installed, you can print the screen to whatever printer the driver is compatible with, either in color or black & white. Just press Alt-Help like you normally would or use the GEM desktop's "Print screen" option.

To abort a screen dump using the DEGAS drivers, press [Undo]. This is the only difference between the installable DEGAS drivers and the built-in driver, which aborts when Alt-Help is pressed a second time.

It is *not* necessary to set the printer configuration by using the "Install Printer" desk accessory when using the DEGAS printer drivers. The DEGAS drivers are specifically written for one configuration and do not use the information in the "Install Printer" dialog.

## FINAL WORDS

As more and more printer drivers become available for the ST series, the machine will become more useful to a larger group of people. I hope this information and the printer driver installer will inspire more programmers to write DEGAS-format printer drivers—They're not just for DEGAS any more! ■

## REFERENCE:

- *The Motorola MC68000 Microprocessor Family: Assembly Language, Interface Design, and System Design*, by Thomas L. Harman and Barbara Lawson. Prentice-Hall, Inc., Englewood Cliffs, NJ

- *A Hitchhiker's Guide to the BIOS*, Atari Corp.

Regent

*REGENT BASE*
*A FULL FUNCTION RELATIONAL DATABASE*
*Regent Base's procedural language make it a natural for handling any of your small business needs. Modules are available for Invoicing, Accounts Receivable, Checkbook Balancing, General Ledger, etc.*
*Regent Base is a relational database written specifically for the Atari ST. Don't settle for simple clones of IBM products. Regent Base is easy to use and state-of-the-art!*

REGENT SOFTWARE
7131 Owensmouth, Suite 45A
Canoga Park, CA 91303
(818) 883-0951

ATARI ST

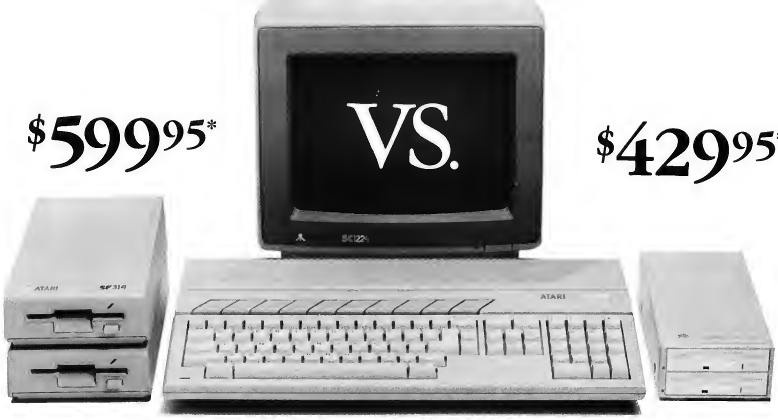
ST PROTECTION NOW AVAILABLE!

Disclone Service Won't Make You Nervous

Technical support, personal service, competitive prices.
Disclone full service quality tested diskette duplication, packaging, documentation production and processing ensures precise duplication, thorough quality control and expedient response to your requirements.
NOclone state of the art hardware based copy protection is true piracy protection for authorized allotments only. Each application is uniquely encrypted. Install routines are coded for nontransferrable hard disk allotments.
Disclone offers a choice of diskettes. Commitment dates are guaranteed. Fast turnover
- up to 1000 in 24 hours, any format.
- up to 10,000 in one week, any format.

disclone

DISCLONE SOFTWARE PRODUCTION SERVICES
1050 North Fifth Street, San Jose, California 95112
(408) 947-1161 OUTSIDE CA: 1-800-826-4296



# Advanced Technology for LESS!

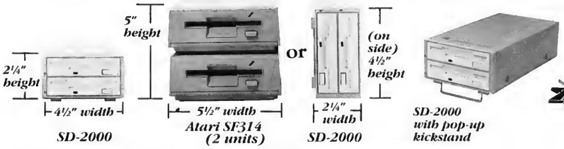
**The SD-2000 Disk Drive System is this year's most exciting new peripheral for the ATARI ST™, and it's the only one of its kind for any computer anywhere in the world.**

**Fast, full featured and compact,** the SD-2000 is smaller and more space efficient than one single stand-alone Atari drive. Complete with A + B drive identification, slip-proof rubber feet, pop-up elevating kickstand, this versatile drive system measures only 4½" W x 8½" D x 2¼" H and includes two 3.5" 1 MB, double sided, double density drives, and is finished in a sleek contoured, color-coordinated gray metal case.

**And there's more!** Through the advanced engineering of the Shanner System, there's now no longer the need for that extra power supply and cumbersome cable. Both Shanner drives operate off a single power supply through its unique "LOW POWER DESIGN" capability.

The SD-2000 comes complete with power supply, I/O Cable, registration card and manufacturer's warranty.

## You be the judge.



**The Shanner SD-2000 Disk Drive**

**$429.95***

**SHANNER INTERNATIONAL CORP.**
The SD-2000 is a creation of, and marketed worldwide by Shanner International Corp.

™Atari and Atari ST are trademarks of Atari Corp.
*Manufacturer's Suggested Retail Prices.

See the local authorized dealer nearest you or use this *SPEEDY* order form.

Please send me ________ SD-2000 Disk Drive(s) at $429.95 each, plus $15.00 for postage & handling (CA residents add $27.95 for sales tax)
Please indicate method of payment:

☐ Check ☐ Money Order ☐ VISA ☐ Mastercharge

Orders may be sent to:

Shanner International Corp.
453 Ravendale Drive
Mountain View, CA 94043

Account No. ________________ Exp. Date ________

Name ________________

Address ________________

City ________________

State ________________ Zip ________

Prices subject to change without notice.

Dealer Inquiries Accepted

S7/86

For direct orders, call our **HOTLINE 1-800-423-8673.** In CA **1-800-626-9273.**

# Moving 16-color Objects

## AL Routines for C.O.L.R. Object Editor

BY JOE CHIAZZESE

*Using AL routines within a C structure, this article demonstrates how to move 16-color objects (created with the C.O.L.R. Object Editor) swiftly around the screen. All related program files may be found within the COLRMOVR folder on your START disk.*

While playing the popular 8-bit Atari game Karateka, a visually sophisticated Karate cartoon, I was inspired to duplicate the main character's actions on the Atari ST. Two day's work convinced me that I'm a terrible artist, but did at least produce a set of assembler routines to manipulate images created with Antic's C.O.L.R. Object Editor.

These routines should serve as a great learning tool for many people. To show how to use them from C, I've written a demonstration program which allows any size image (within reason) to follow the mouse around on the screen.

The complete program has three parts: the assembler routines to handle the images, the C routines to make the program logic easier to understand and the image itself. Creating the final program requires four steps: 1) drawing the image and converting it to source code, 2) assembling the image-handling routines, 3) compiling the main C program, and 4) linking all the files together. ▶

## Objects . . .

### *CREATING THE IMAGE*

Though you may follow this procedure using your own display image, I originally tried it with the karate character and sample files on the START disk include this data. After drawing the karate character's image, C.O.L.R. Object Editor's Create Source option was used to save the image as assembler source code statements with a label. This was not quite enough: in order to link the image's label to the C code, the label must be global. Do this by adding the **.globl label** directive to the image code data using any standard programming text editor.

The **.globl** directive can be put anywhere in the source code created by C.O.L.R. Object Editor. I suggest the top so it can be found easily. Since Alcyon C adds the underscore character in front of all its external variables and routines, the image's label must begin with an underscore so the image label matches that defined in the C program, as must the names of the routines in the assembler source code. For example, the routine I call **save_screen** in C must be **_save_screen** in assembler to match it.

**I wanted generic assembler routines which would work with any size object.**

### *C/ASSEMBLER INTERFACE*

I wanted generic assembler routines which would work with any size object, and not just the specific karate character example. The routines designed, therefore, had to accept parameters from an outside source. When the subroutine is called, the return address is pushed onto the stack. Since C places all parameters on the stack in reverse order, the parameters are accessed at the current stack position plus four bytes (the return address is 32 bits long requiring 4 bytes to hold it). For example: **put(device,char);** pushes **char** on the stack first followed by **device** with the stack growing downward.

### *THE ASSEMBLER ROUTINES*

The assembler portion of the demonstration consists of five routines, each with a very specific function as discussed below.

**_save_screen** copies a block of specified width and height from the screen to a buffer. The buffer must be allocated by the C program and it is the programmer's responsibility to insure there is enough room. The buffer can either be an array or a block of memory allocated from the operating system, in which case a pointer is needed to tell the routine the buffer's location. All the routines expect the object's width to be specified in words, *not* pixels. The height is in lines. (Both of these numbers are displayed when creating source with the C.O.L.R. Object Editor.) All buffers and screen positions are passed as absolute addresses.

The C portion of the program is responsible for converting x and y coordinates into absolute screen addresses. The address is calculated and then passed to the assembler routines which, in themselves, have no provision for such conversions.

**_restore_screen** reverses the effect of **_save_screen** by replacing the previously saved block of screen memory.

**_shift_image** is a little more complex. Given the address and size of the original object, it shifts the image to the right by the specified offset and stores the result where told. The result is an image one word wider than the original. This new image is used by all the other routines; the original only serves to create another image which can be shifted to the right by up to 15 pixels. Even if the offset is zero, the object should be put through this routine because it creates an image which is one word wider, and the assembler routines actually expect to be working on an object which is a word wider than the width specified. If the given width is two, for example, the routines manipulate an object of width three.

The shifted images are necessary because screen memory is arranged in word-sized, 16-bit chunks. Each shifted image corresponds to an image that begins on different bits, or pixels, of a word of screen memory.

To have the background show through the empty portions of the object, you need a mask for the object. Rather than calculate the mask for every object, the **_make_mask** routine automatically creates one. All it needs is the size and address of the object and a place to put the mask. Each mask uses half the memory space of its image.

Finally, **_draw_image_** takes the image, the mask, and the saved background, mixes them all together, and puts the final picture on the screen with background showing through. First, the background is loaded, ANDed with the mask, then ORed with the image. This causes any blank sections of the image to appear transparent.

I chose to mix in the background from the saved buffer and not the screen. Although this means the background has to be saved first for the routine to function properly, there's no need to restore and save the screen again if the picture is drawn in the same screen block. So, if the picture is drawn at point 'A', it can be moved up to 15 pixels to the right and still be drawn on the same block using only an image shifted by a different offset.

## *THE C ROUTINES*

The main routine is simple: it opens the workstation, initializes and if successful, calls **follow_mouse()** (which is really the main routine). It then cleans up and closes the workstation.

**open_workstation()** and **close_workstation()** are the standard GEM routines used in most programs. I make it a habit to put them in every program in case I need them later. **finish_up** frees the memory allocated by **initialize()**.

**initialize()** does several things: it checks the resolution and returns you to the Desktop if the resolution is anything but low. It saves the screen's base address in **scrbase**, then allocates memory for all the images and masks. You will also find yourself back at the Desktop if sufficient memory is not available.

The default palette is saved so that it can be restored at program's end. A new palette is set and the background is filled with a wall pattern. The image is then shifted through all 16 different positions within a word and each one is stored for later use. At the same time a mask is generated for each of the images. Finally, to initialize the main loop, a screen block is saved once and **oldpos** is set to point to it.

On entry of **follow_mouse()**, we encounter a **while** loop which continues until a key is pressed. To avoid repeatedly redrawing the image at the same place, a **DO** loop has been placed within the **while** loop which will exit if the mouse is moved or a key is pressed. The x and y coordinates of the mouse are used to calculate the absolute screen position. We now wait for the next vertical blank to occur to minimize flicker. (A certain amount of flicker will still be noticeable when the image is at the top of the screen. To eliminate this calls for a complicated technique of multiple-screen image switching which is beyond the scope of this article.) If the object is in the same screen block, it is simply redrawn; otherwise we restore the background where the old image was, save the new block, point the old position to the latest one and then draw the image.

## *CUSTOMIZING*

If you want to change the program so that your own picture gets dragged around the screen, go to the top of the C file and change the **#define** statements to reflect the size etc. of your own picture. **OBJW** is the width in words, **OBJH** is the height in lines and **COEOBJECT** is the label of your picture. Link it in and that's it. (Don't forget to make your object label a global!) ■

**Software Discounters of America**

S.D. of A

**Orders Outside PA—1-800-225-7638**
**PA Orders — 1-800-223-7784**
**Customer Service 412-361-5291**

**Open Saturday**

- **Free Shipping on orders over $100 in continental USA**
- **No surcharge for VISA/MasterCard**
- **Your card is not charged until we ship**

**Your ST Software Connection!**

| | |
|---|---|
| **ABACUS BOOKS** | |
| ST Gem Prog. Ret. | |
| ST Graphics & Sound | |
| ST Internals | |
| ST Logo | |
| ST Machine Language | |
| ST Peeks & Pokes | |
| ST Tricks & Tips | |
| *Call For Low Prices!* | |
| *Optional Program Disks are available for most books! | Call |
| **ACADEMY** | |
| Typing Tutor | $23 |
| **ACCESS** | |
| Leader Board Golt | Call |
| **ACTIVISION** | |
| Borrowed Time | $33 |
| Hacker | $29 |
| Mindshadow | $33 |
| Music Studio | $39 |
| **ADVENTURE INT'L.** | |
| Fantastic Four | $16 |
| Spiderman | $16 |
| **ARTWORX** | |
| Bridge | $19 |
| Compubridge | $19 |
| Mail List | $14 |
| Strip Poker | $25 |
| **BATTERIES INCLUDED** | |
| D.E.G.A S | $25 |
| Home Pak | Call |
| Paperclip Elite | Call |
| **ELECTRONIC ARTS** | |
| Financial Cookbook | $33 |
| **EPYX** | |
| Rogue | $25 |
| Temple Apshai Trilogy | $25 |
| Winter Games | Call |
| **FIREBIRD** | |
| The Pawn | $29 |
| **FTL/SOFTWARE HEAVEN** | |
| Sundog | $25 |
| **HABA** | |
| Check Minder | $39 |
| Phone Book | $39 |
| Haba View | $39 |
| **HIPPOPOTAMUS** | |
| Hippo Art 1 | $25 |
| Hippo Backgammon | $25 |
| Hippo Clean | $19 |
| Hippo Computer Almanac | $23 |
| Hippo Concept | $59 |
| Hippo Disk Utilities | $33 |
| Hippo Jokes & Ouotes (not for kids) | $23 |
| Hippo Pixel | $25 |
| Hippo Ram Disk | $23 |
| Hippo Simple | $33 |
| Hippo Word | $59 |
| **INFOCOM** | |
| Ballyhoo | $25 |
| Cutthroats | $25 |
| Deadline | $33 |
| Enchanter | $25 |
| Hitchhikers Guide to the Galaxy | $25 |
| Infidel | $29 |
| Seastalker | $25 |
| Spellbreaker | $33 |
| Starcross | $33 |
| Suspect | $29 |
| Suspended | $33 |
| Wishbringer | $25 |
| Witness | $25 |
| Zork 1 | $25 |
| Zork 2 or 3 | $29 |
| **MARK OF THE UNICORN** | |
| Hex | $25 |
| PC Intercomm | $79 |
| Final Word | $95 |
| **MICHTRON** | |
| Bulletin Board System | $33 |
| Calendar | $19 |
| DOS Shell | $25 |
| Flip Side | $25 |
| Gold Runner | $25 |
| Kissed | $25 |
| M-Disk | $25 |
| Mi-Dupe | $19 |
| Mi-Term | $33 |
| Mudpies | $25 |
| Soft Spool | $25 |
| **MICROPROSE** | |
| Silent Service | $26 |
| **MI-GRAPH** | |
| Easy Draw | $95 |
| **MINDSCAPE** | |
| Brataccus | $33 |
| **MIRAGE** | |
| Tool Box Vol. 1 | $19 |
| Forth | $33 |
| H & D Base | $59 |
| **OSS** | |
| Personal Disk Kit | $25 |
| Personal Pascal | $49 |
| Personal Prologue | Call |
| **OMNITREND** | |
| Universe 2 | $49 |
| **ORIGIN** | |
| Ultima 3 | $39 |
| **PENGUIN/POLARWARE** | |
| Crimson Crown | $25 |
| Oo-Topos | $25 |
| Transylvania | $25 |
| **PRYORITY** | |
| Forbidden Quest | $23 |
| **OMI** | |
| St Talk | $14 |
| **OUICK VIEW** | |
| Zoom Racks | $49 |
| **REGENT** | |
| Regent Base | Call |
| Regent Word | $33 |
| Regent Spell | $33 |
| **SIERRA ON LINE** | |
| Black Cauldron | $25 |
| Hint Books | Call |
| Kings Ouest 1 | $33 |
| Kings Ouest 2 | $33 |
| Ultima 2 | $39 |
| Winnie the Pooh | $19 |
| **SSI** | |
| Phantasie | Call |
| **SUBLOGIC** | |
| Flight Simulator | Call |
| Jet | Call |
| **SYNAPSE** | |
| Mindwheel | $29 |
| **TELARIUM** | |
| Amazon | $33 |
| Fahrenheit 451 | $33 |
| Nine Princes in Amber | $33 |
| Perry Mason: Case of the Mandarin Murder | $33 |
| **TIMEWORKS** | |
| Data Manager | Call |
| Swiftcalc | Call |
| Sylvia Porter's Personal Financial Planner | Call |
| Word Writer | Call |
| **UNISON WORLD** | |
| Print Master | $26 |
| Art Gallery 1 | $19 |
| **VIP TECHNOLOGIES** | |
| VIP Professional | Call |
| VIP Professional Lite | $65 |
| **XLENT** | |
| Rubber Stamp | $23 |
| Typesetter | $26 |
| **ACCESSORIES** | |
| Bulk Disks 3½ | Call |
| Compuserve Starter Kit | $19 |
| Disk Case (Holds 30) | $9 |
| Dow Jones Starter Kit (5 hr) | $14 |
| Dust Covers | Call |
| Kraft Joystick | $9 |
| Panasonic 1080 cps dot matrix printer | Great Deal |
| Panasonic 3131 letter quality daisy wheel printer | Call |
| QMI 300/1200 Baud Modem w/ST Talk | $147 |
| Supra 10 Meg Hard Disk Drive | $599 |
| Supra 20 Meg Hard Disk Drive | Call |
| Supra 300 Baud Modem w/Omega Terminal | $59 |
| Supra 300/1200 Baud Modem w/Omega Terminal | $147 |
| Wico Boss | $12 |
| Wico Bat Handle | $17 |

**Have you seen our on-line catalog of 800 software titles for Commodore, Atari, Apple & IBM? It's on Compuserve's Electronic Mall—Just type GO SDA and shopping for software will never be the same again! P.S. If you don't own a modem or subscribe to Compuserve, call us, do we have a deal for you!**

**P.O. BOX 111327—DEPT. ST—BLAWNOX, PA 15238**

*__Ordering and Terms:__ Orders with cashier check or money order shipped immediately. Personal/company checks, allow 3 weeks clearance. No C.O.D.'s. **Shipping: Continental U.S.A.**—Orders under $100 add $3, **free shipping on orders over $100.** PA residents add 6% sales tax. **AK, HI, APO, FPO, PR**—add $5 on all orders. Sorry—no international orders. Defective merchandise will be replaced with same merchandise. Other returns subject to a 15% restocking charge—NO CREDITS! Return must have authorization number (412) 361-5291. Prices subject to change without notice. **Summer Hours: Mon.-Fri. 9A.M.-5:30 P.M. EDT • Sat. 10 A.M.-5 P.M.**

# The Digital Magnet

## Plotting Magnetic Field Lines

BY DAVID SMALL

*This spectacularly colorful graphics program simulates magnetic field line generation. If you are more interested in detail than color, you can also run it on a monochrome system; it works in any resolution. Related files may be found within the MAGNETS folder on your START disk.*

I guess magnets have always fascinated me. As a kid, I loved to play with iron filings and magnets, electromagnets, and whatnot.

One day in freshman physics we were discussing formulas governing magnets. I woke from my usual haze just enough to note the formulas and think, "This would make a neat program." At our local computer center, there were two unused Tektronix 4013 graphics terminals hooked up to a Cyber mainframe, so I spent some time on them and wrote the original Magplot, a magnetic field lines plotter.

The program wasn't a waste of time. After all, it helped me pass my freshman physics class. It also earned a reputation for pulling computer time out of the mainframe; when I ran the program, I could hear the chatter of teletypes in the next room slow down, sometimes stop, a result of all the floating point arithmetic going on.

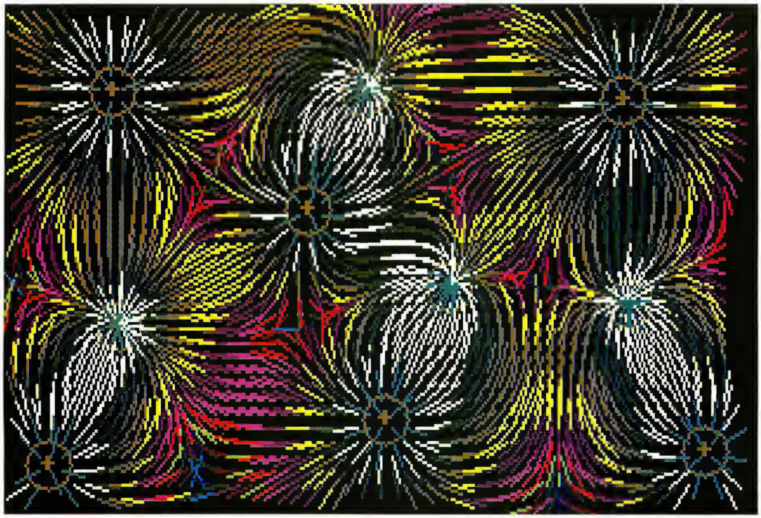
Well, enough history. Let's look at the program.

### THEORY

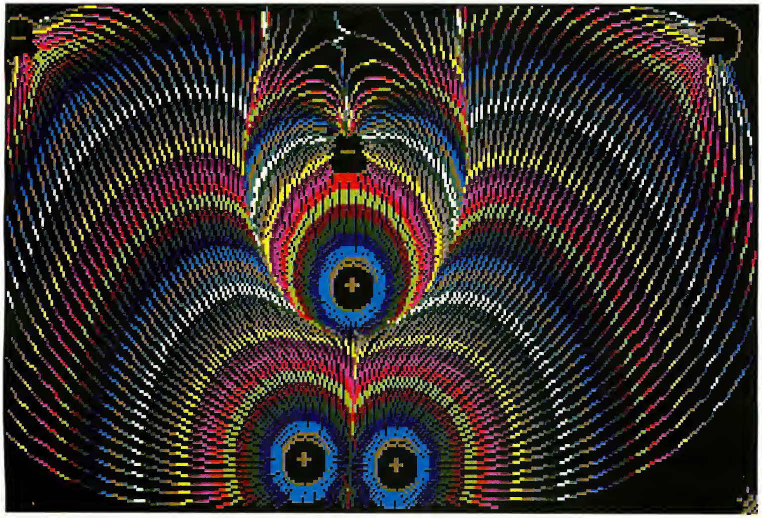
The formula for attraction between two charged particles is:

(CHARGE1 * CHARGE2) / (DISTANCE SQUARED)

▶


Plotting magnetic field lines in GRID mode, or . . .

. . . in POINT-POINT mode.



Or, the attraction/repulsion between two objects is dependent on their distance squared. You'll find this same relationship in other popular formulas, such as for gravity.

With multiple particles, you must calculate the force exerted at any point by summing up the charges exerted on each point by each particle. The easiest way to do this is to calculate each force as a vector, then add up the vectors. You end up with a vector which, at a given point, represents the sum total of the attraction/repulsion being exerted on it at that point.

This is similar to figuring out the direction a satellite will go by summing up the relevant gravitional pulls from all the bodies (the sun, planets).

If you then draw a small line in the direction the vector points, recalculate the attraction/repulsion, and so on, you will obtain a picture of the field lines around magnets. If you've ever done the experiment with iron filings and a magnet, the plots done by this program will look remarkably familiar.

Two like forces will repel each other; you'll see the field lines reflect this, as the " + " and " + " points repel one another. Opposite forces attract, again, you'll see the field lines traveling from " + " to " – ".

There's also a physics theorem that field lines plotted this way will never cross one another. This is the case in the program.

## *PROGRAM*

The program was originally written in Cyber BASIC, and over the years has ended up on an amazing variety of graphics-oriented computers. The ST and C language are the latest target, and one of the faster implementations; the 68000 and C seem to be pretty adept at floating point arithmetic, which this program spends a vast amount of time doing. You will see a certain amount of BASIC within the code in this incarnation of the program, including the entirely global variables. Please feel free to optimize it. Quite honestly, the original program took me so long to get working properly under BASIC, I have a mental block about playing with it too much.

The program has three main sections. The first involves initialization and setup of controlling variables. Since the ST's processor is not infinitely fast, there are some practical limits to make the plot occur reasonably quickly. I've set these up as easily tweaked variables in a procedure called **initglobals**; feel free to play around with them. Here's the rundown:

A) **linelength**: This is how far a line is drawn in a given vector's direction. Too long, and the smooth curves of the field lines will become jagged; too short, and the CPU will spend too much time calculating between points.

B) **clipit**: This variable, if found to be True, "clips" field lines to the size of the screen; in other words, if you're drawing a field line and it goes offscreen, the line is terminated. The problem here is that offscreen lines are pure CPU calculation and the wait on them can get pretty tedious; on the other hand, most plots look far more complete if you include the offscreen line. If you select no clipping, then the line is terminated only if it goes wildly out-of-bounds and has little chance of returning on to the screen area.

C) **degreeinc**: The starting point for each line is on a circle around positively charged points. The line then travels to a negative point, by whatever route, or gets terminated offscreen. **degreeinc** determines the degree increment between starting lines. The lesser this variable, the more lines are plotted off each positive point; on the other hand, it takes longer to generate a given plot. If you make this variable very low, you'll have quite a dense and good looking plot, but it will take awhile to generate.

D) **k2**: This is a miscellaneous force constant that determines how much strength a magnetic field has. If you've ever wanted to play god and turn off magnetism, here's your chance. The value is more or less kludged to make good plots.

E) **radius**: This value determines the "black hole" radius around a negative point and also determines the starting radius of each positive point.

The second section of the program lets you input the charge points using the mouse. This is done primarily within **inputpoints()**. Admittedly, it is a bit kludgy, but it does not require resource files or calls to the AES. This shortens and simplifies the program, particularly if you don't have the Resource Construction Set.

When the mouse button is pressed during point input, the point's X and Y locations are recorded, and a "menu" for positive/negative is displayed. Click on either POS or NEG to select that point's charge. I store the X,Y, and charge of the point in three arrays, **xcoord**, **ycoord**, and (creative name) **charge**.

The "menu" is then erased.

When you press [Shift] or [Alternate], you exit the **while** loop that inputs the points, and proceed to the program's third main section which deals with plotting the picture.

## *PLOTIT()*

There are two ways of creating the picture. One gives a fairly accurate picture of the field lines by plotting continuous lines, the other a good-looking plot by plotting one line on a grid overlaying the picture area. I include both; you select which one you'd like with the **gridstyle** option. If you want the grid approach, exit the point-input process with [Alternate]; if you'd like the point-point approach, use [Shift].

In grid mode, we start at all X's and Y's forming a grid on the surface, calculate the line direction at that particular point, draw the line, and move to the next point. The line's color is drawn based on the amount of force present at that particular point (the vector summation), which makes for a spectacular display in low or medium-resolution.

In point-point mode, we run several nested loops. In order, they are: 1) Cycle through all positive points; 2) Start a line at N points around each positive point; 3) Sum charges at each point; draw the line.

The charge summer is set up as a separate call to make grid plotting easier. The CPU spends most of its time here, as the variables are double floating point, required because of the fractions involved. You'll find the distance between points being calculated as **sqrt ( (x1-x2) squared + (y1-y2) squared** ). You'll also find the basic force equation here.

The charge summer inputs the point as **x1**, **y1** (old BASIC variable names) and outputs the next point on the line, scaled to **linelength**, in the same variables.

VDI's **v_pline (polyline draw)** then draws the line and proceeeds to the next point.

### *CONSIDERATIONS*

There are some interesting practical considerations that I ran into when designing this program.

- How should the lines start in order to create an even plot? This was solved by using polar coordinates around each positive point. (You can't begin a line at a negative point because it travels towards the nearest negative point—itself.)
- How do you terminate the line? This was not trivial. It is easy if the line goes offscreen, but, when the line nears a negative point, there is a "black hole effect"; the distance is so short that the negative point becomes the only relevant force. The line is drawn one **linelength** past the negative point, which causes it to overshoot the negative point; it then turns around, overshoots, and so forth, creating a ping-pong effect.

  The solution here is to terminate a line whenever it becomes **radius** close to any negative point. This stops the ping-pong effect. Regrettably, it involves calculating the distance to every negative point on each line draw, which involves extra CPU time; perhaps a simpler solution would be a subtraction against each negative point's coordinates.
- What if the line goes offscreen? There is special code to handle this case, if you decide to let the line wander offscreen and return. The variable **newline** "initializes" the line for subsequent **v_plines** so you don't get a strange skipping effect.

There are some miscellaneous routines to make the plot work on any style monitor. The high-res monitor, of course, gives the cleanest picture, but there are only two colors, black and white. The two color modes allow some interesting possibilities for color and animation. I cycle through the color register numbers (either 0-3 or 0-15) while drawing the force lines, and the result is a pleasing multi-color force line; if the color registers are then rotated "underneath" the plot, so to speak, the force lines become animated, travelling from positive to negative points. You'll see the rotation code in the line drawing routine.

As the code notes, the GEM color number seldom corresponds to the extended BIOS call hardware color number. This caused me some trouble, and I had to include a fair amount of code just to deal with this problem.

You might find the "POS/NEG" code to be useful in writing resolution independent codes. I calculate all the X and Y constants on the basis of character width and screen size returned from the VDI "open workstation" call. ■

## HOW TO RUN MAGPLOT

***1.*** **Double click on the program's file icon, MAG21.PRG. Once the program is loaded, the top screen will read: Exit: (SHIFT=pt-pt; ALT=grid). At this point, pressing [Shift], [Alternate], or [Control] will exit you to the Desktop.**

***2.*** **Click the mouse anywhere on the screen where you would like a point, then click on either POS or NEG to select that point's charge. Continue inputting points like this until you have as many as you like. A quick demo is two points—one positive and one negative—a distance from each other on the screen.**

***3.*** **Having chosen your points, press [Alternate] to plot in GRID mode, [Shift] to plot in point-point mode. Try both to see the difference. During the actual plotting—or when finished, you may restart by pressing [Shift], or exit the program by pressing [Control].**

**MAG21 works in all resolutions but is flashiest in lowres. On color monitors, the program will color-cycle at the end of point-point mode.**

# TRACKING THE ELUSIVE GDOS

## THOSE MISSING METAFILES

**by Tim Oren**

*A valuable examination of GDOS and metafiles by the programmer who helped write the GEM system. Discover why Atari's "color Mac" is currently incapable of using multiple fonts. Never heard of GDOS? Read on.*

An important chunk of GEM was left out of the Atari ST when TOS was placed in ROM. It's called GDOS, and its absence is the reason your ST does not have multiple, Macintosh-like fonts. But GDOS (Graphics Device Operating System) *can* be added to your system, and with its associated "metafiles," you may output identical images to such diverse peripherals as the screen, printers, cameras, and more. GDOS was first used for the ST in Easy Draw, from Migraph. An official version should be available from Atari soon.

In theory, a fully implemented GEM system utilizing metafiles permits, among other things, graphic output to various peripherals at the highest resolution of which the peripheral is capable. For example, a metafile, creating a picture image on a medium-resolution screen, will take advantage of the highest matrix density of a graphics printer. The same picture may also be output to a plotter or camera. Again, much of this is "in theory." Digital Research has yet to complete full implementation of all capabilities of GDOS—and its related OUTPUT.PRG application. But some of the abilities are currently available for the ST. Let's take a closer look.

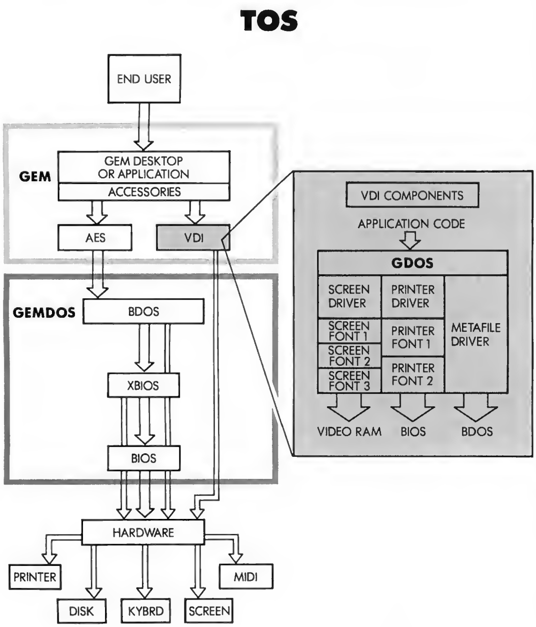
### TOS STRUCTURE

To understand the role of GDOS on the Atari ST, let's look at the system software (TOS) which already exists in ROM. (See *figure 1.*) There are two sections: GEMDOS, which handles disk and keyboard input and output, and GEM, which manages the mouse and on-screen graphics and windowing.

GEMDOS has three parts. The BIOS▶

ILLUSTRATION BY JAMES ENDICOTT

FIGURE 1.



(Basic Input/Output System) handles hardware level interaction, such as getting a single character from the keyboard or reading a sector from a floppy disk. The XBIOS (Extended BIOS) contains hardware level calls peculiar to the ST, such as setting palette registers and configuring the sound chip. The BDOS (Basic Disk Operating System) handles disk interactions at the logical level, such as creating, writing, and reading files.

The GEM portion of TOS, with which we will be the most concerned, is also composed of three major parts. The first is the VDI(Virtual Device Interface), which handles graphic I/O functions such as drawing lines and performing bit-blit operations. (Bit-blit stands for Bit Block Transfer, a technique of moving—or copying—a block of pixels from one place to another.) The AES (Application Environment Services) performs logical level screen actions, such as window handling, drawing and processing dialogs, and animating the menus. Finally, the GEM Desktop application provides the user's window and icon interface to the file system.

The terms "virtual" and "device" in the definition of VDI mean that it, in effect, creates a new graphics machine which you can program instead of writing directly to the graphics screen yourself. The "VDI machine" is able to create lines and circles, write graphics text, blit portions of the screen, and so on. An application which uses only VDI calls to create graphics should work on any machine which uses GEM, or on any device attached to the ST for which a VDI driver has been written. By accepting this restriction, a developer gains portability for his or her program, and leverage for his or her efforts.

## ST'S ABSENT GDOS

On most GEM machines, the VDI consists of three parts: fonts, drivers, and GDOS. Fonts define various styles and sizes of characters which can be written on an output device. A driver is the piece of code which implements the VDI machine for a particular output device. The GDOS is the device-independent part of the VDI, which (among other things) sends an application's VDI calls to the correct output device.

However, there is currently a problem in using this method with the Atari ST. The VDI as implemented in the ST's ROMs includes only a single device driver and set of fonts. The driver includes code for all three resolutions of the ST's graphic screen, and there are two font sizes defined for each resolution. The parts missing from the ROMs are the GDOS and VDI drivers for other devices, such as printers.

Because the GDOS is missing, a GEM program on the ST loses the device independent capabilities of the VDI. Specifically, it is unable to load new drivers or fonts, or to use the NDC (Normalized Device Coordinate) scaling system to draw pictures. The lack of a GDOS leads to several problems for ST developers and users. (We can point the finger two ways here: Digital Research did not deliver the GDOS code to Atari soon enough, and, in any case, Atari did not have enough room in their TOS ROM to include it.)

Since VDI graphics cannot be written directly to a printer, programs are forced to do bit-by-bit printer dumps of screen images, using specially written driver code. Thus, a paint program like DEGAS has to include a different driver for each printer which might be hooked to the ST. Worse, no other program can make use of these drivers. Because there is no standardized GDOS to define the driver interface, each developer ends up wasting a good deal of time.

A second problem is the loss of ability to load fonts. Although there is a way to force the ST screen driver to accept a different font (see sidebar), the standard GDOS load and unload font commands are not available. Also, since Digital Research purchased its original fonts from an outside source, the GEM developer's software has never included a Font Editor. These two problems have so far prevented the creation of a wide variety of fonts on the ST, such as has occured on the Macintosh.

Fortunately, a GDOS has been written for the ST by Digital Research, and should soon become available to developers. As of April 1986, Atari is testing a final version of the GDOS software. Already one product which incorporates the GDOS, Easy-Draw from MiGraph, has been released. (Developers should be warned, however, that the GDOS shipped by MiGraph is a preliminary version which may contain bugs in functions not actually used by Easy-Draw).

The GDOS for the ST is shipped as a loadable program, GDOS.PRG, which is placed in the AUTO folder to be run at boot time. The GDOS is a terminate-and-stay-resident program which means that once it is loaded it returns control (but not consumed memory) to the OS. When it is run, it places a pointer to itself in the 68000's TRAP 2 vector, saving the previous value, which is the vector to the VDI and AES. When an application is running, all TRAP 2 calls are screened by the GDOS, which processes functions that it recognizes and passes others on to the AES and ROM-resident VDI code.

## ASSIGN.SYS

When it is first run, GDOS also reads a file named ASSIGN.SYS (see *figure 2*). This file tells the GDOS which drivers and fonts should be loaded into RAM for all programs to use. (This loading subtracts from the RAM available for your applications to run.) The exact format of ASSIGN.SYS is described in the VDI manual. For our purposes it suffices that it associates numeric device IDs with their driver filenames and font files.

It is useful to know a few of the standard device IDs. The value one (1) is always associated with the screen. Device ID 21 is the printer, if one is installed. Device ID 31 is for a metafile driver, which will be described later. A device ID must used with the open workstation **v_opnwk** VDI call, so that the GDOS can tell which physical device you want to address.

Notice, by the way, that you do not need to open the screen as a workstation if you are using the AES. The AES automatically initializes the screen workstation at boot time, and returns its handle as a result of the **graf_handle** call. Your application then uses this handle as an input to the **v_opnvwk** VDI call, which opens a VIRTUAL workstation for your use.

This is a good point to define the difference between a physical workstation and a virtual workstation. The physical VDI workstation is associated directly with the device itself, so only one physical station can be open for any one device at a time. A virtual workstation is a software construct which is associated with the physical workstation, but saves an independent set of VDI parameters such as line thickness, writing mode, text style, and so on. Using the virtual workstation method, your application, the desk accessories, and the AES itself can all write to the display screen via its physical workstation, but without destroying each other's VDI parameter settings. ▶

FIGURE 2.

**SAMPLE ASSIGN.SYS FILE**

```
01p SCREEN.SYS          ; ROM-resident screen driver
IBMLSS10.FNT
IBMLSS14.FNT
IBMLSS18.FNT
IBMLSS36.FNT
21 FX80.SYS             ; Epson printer driver
EPSHSS10.FNT
EPSHSS14.FNT
EPSHSS20.FNT
EPSHSS36.FNT
31 META.SYS             ; Metafile driver
```

Returning to the ASSIGN.SYS file, the driver filenames all end with a SYS extention. For example, FX80.SYS is a driver for an Epson FX-80 printer, and META.SYS is the metafile driver. The special name SCREEN.SYS refers to the screen drivers which are already present in the ST's ROMs.

Font files always have names ending in FNT. One or more font files will be associated with each driver. Fonts come in a variety of styles and sizes, and the file name usually describes the device, style, and size of the characters it contains. For instance, a file named EPSHSS10.FNT contains a 10-point type face for use with the Epson printer's high resolution mode.

Note that an application is not restricted to only those fonts named in the ASSIGN.SYS file. When the GDOS is present, the application can also use the **vst_load_fonts** VDI call to bring additional font files into RAM from the disk. It can then use **set_font** to switch to the alternate typeface. However, fonts loaded in this manner are only available to the application which called them; they will disappear when the program terminates.

Using the GDOS and alternate drivers, your program can write to devices such as the printer using the same graphics commands that work on the screen: lines, circles, text, and so on. However, bit-blit operations only work with the screen and internal RAM of the ST. To create a bit image on the printer, you must first store the image in a file and then use the VDI's Output Bit Image File **v_bit_image** command to write the file to the output device.

In many cases, the device driver builds a bit-by-bit image of the printed page in memory, and you will need to use the update workstation call, **v_updwk**, to force output to actually begin, and then the clear workstation function, **v_clrwk**, to advance to a new page. The output window function, **v_output_window**, is also available if you need to force only a part of the picture to the printer.

## METAFILES

It is the metafile driver, however, which has the most interesting implications for ST developers. To understand the definition of a metafile, recall the notion that the VDI creates a virtual graphics engine, which is driven by commands sent by the application via the VDI bindings. In these terms, a metafile is simply a recording of the commands which created a given picture. By "replaying" these commands with a suitable program, you can generate a copy of the picture on any VDI output device.

Such a program is provided with the GDOS distributed by MiGraph and Digital Research. It is a full-fledged GEM application called OUTPUT.PRG, which you may invoke from the Desktop. If your application simply needs a method to create graphics which may be replayed for any device, this stock program should suffice. Unfortunately, due to bugs in the **shell_write** call, there is currently no reliable way to invoke OUTPUT.PRG from an application.

A GEM application generates a metafile by opening a physical workstation with the metafile device ID 31. By default, the metafile will be named GEMFILE.GEM, but you can change this by executing a **vm_filename** command immediately after opening the metafile workstation. From this point on, you can use most Control, Output and Attribute VDI functions with the workstation, (see *figure* 3). Note that, like a printer, a metafile can only accept bit image data if it is provided in a file.

The metafile commands are accumulated in a buffer and written to the disk when the buffer is filled. Closing the metafile workstation forces the contents of the buffer to the disk and closes the metafile.

## IMPLICATIONS

Useful though the "playback" ability of the metafile may be, it has greater implications for those writing applications which do on-screen graphics editing. Consider a standard "Paint" type application, such as DEGAS or Neochrome. When such a program generates a square or circle on the screen, it becomes a mere collection of bits. There is no way, short of erasing and redrawing, to change the size or shape of the object once it has been drawn.

Let's extend the notion of metafile to solve this problem. If the metafile is a copy of the instructions which drive the VDI engine to create a particular picture, then you can equally well store this sequence within the program's memory. This allows you to edit the commands, rather than the bits generated. This technique is the origin of "Draw" type graphics programs, which allow the figures on the screen to be manipulated as objects rather than images.

In such a program, the objects on screen are typically equipped with "handles" which may be moved using the mouse to redefine the location and size of the figure. You can also edit such attributes as color, line type and size, and fill pattern. In each case, the user's action results in a modification of the stored VDI command sequence, which is then replayed in whole or part to regenerate the picture on the screen. In most of these programs, you can also group sets of drawing objects, which may then be scaled and edited together.

Beyond simple screen editing, many other types of programs are made possible with the concept of graphical objects. For instance, objects can be linked up to fields or records of a database. The result is a point-and-click information retrieval system. This approach is best

## VDI METAFILE COMMANDS

FIGURE 3.

| OPCODE | SUB-OPCODE | VDI CALL | FUNCTION |
|---|---|---|---|
| 3 | – | v_clrwk | Clear Workstation |
| 4 | – | v_updwk | Update Workstation |
| 5 | 2 | v_exit_cur | Exit Alpha Mode Escape |
| 5 | 3 | v_enter_cur | Enter Alpha Mode Escape |
| 5 | 20 | v_form_adv | Advance Form |
| 5 | 21 | v_output_window | Output Window |
| 5 | 22 | v_clear_disp_list | Clear Display List |
| 5 | 23 | v_bit_image | Output Bit Image File |
| 6 | – | v_pline | Polyline |
| 7 | – | v_pmarker | Polymarker |
| 8 | – | v_gtext | Graphic Text |
| 9 | – | v_fillarea | Fill Area |
| 11 | 1 | v_bar | Bar |
| 11 | 2 | v_arc | Arc |
| 11 | 3 | v_pieslice | Pie |
| 11 | 4 | v_circle | Circle |
| 11 | 5 | v_ellipse | Ellipse |
| 11 | 6 | v_ellarc | Elliptical Arc |
| 11 | 7 | v_ellpie | Elliptical Pie |
| 11 | 8 | v_rbox | Rounded Rectangle |
| 11 | 9 | v_rfbox | Filled Rounded Rectangle |
| 11 | 10 | v_justified | Justified Graphic Text |
| 12 | – | vst_height | Set Absolute Character Height |
| 13 | – | vst_rotation | Set Character Baseline Vector |
| 14 | – | vs_color | Set Color Representation |
| 15 | – | vsl_type | Set Polyline Type |
| 16 | – | vsl_width | Set Polyline Line Width |
| 17 | – | vsl_color | Set Polyline Color Index |
| 18 | – | vsm_type | Set Polymarker Type |
| 19 | – | vsm_height | Set Polymarker Height |
| 20 | – | vsm_color | Set Polymarker Color Index |
| 21 | – | vst_font | Set Text Face |
| 22 | – | vst_color | Set Text Color Index |
| 23 | – | vsf_interior | Set Fill Interior Style |
| 24 | – | vsf_style | Set Fill Style Index |
| 25 | – | vsf_color | Set Fill Color Index |
| 32 | – | vswr_mode | Set Writing Mode |
| 39 | – | vst_alignment | Set Graphic Text Alignment |
| 104 | – | vsf_perimeter | Set Fill Perimeter Visibility |
| 106 | – | vst_effects | Set Graphic Text Special Effects |
| 107 | – | vst_point | Set Character Height, Points |
| 108 | – | vsl_ends | Set Polyline End Styles |
| 112 | – | vsf_updat | Set User-defined Fill Pattern |
| 113 | – | vsl_udsty | Set User-defined Line Style Pattern |
| 114 | – | vr_recfl | Fill Rectangle |
| 129 | – | vs_clip | Set Clip Rectangle |

exemplified by the Filevision package available for the Macintosh.

We have so far assumed that the location, size, and shape of the objects are under user control. If these values are instead determined by the program, we get another family of applications.

When the values are computed from underlying data, the result is a data graphing application. Such a grapher might also allow you to manipulate handles on the objects forming the graph, and cause a change in the related values.

You could also write a program which would relate each object's location, size, and shape to those of the other objects on the screen. For instance, you might force two lines to lie parallel, or to be of equal length. Such "constraint based" graphics editors are as old as Ivan Sutherland's original Sketchpad program, and are available in some CAD packages, but have yet to appear on machines of the ST's size and price.

As a final idea, consider a programming language which would include graphic objects in its command set and workspace. In such a language, you could write routines to compute objects' appearances as the result of a simulation. The results might range from simple animation sequences, such as a bouncing ball, to very complex situations like a rendezvous with a space station.

Whatever the application envisioned, a program which uses graphic objects must be able to read as well as write metafiles, because that is the format in which graphical commands will be ▶

## USING FONTS WITHOUT GDOS

*The following technique will let you load and use alternate fonts without using GDOS. Your font must be in the standard format (including header) as defined in the VDI manual. Either compile the font into your application, or load it from a file at run time.*

*Instead of doing the normal* **vst_load_fonts** *call, you will alter your application's control array directly. Control words 7 and 8 must be a 32-bit pointer to the font. Control words 10 and 11 are a 32-bit pointer to a scratch buffer which will be used by the VDI when doing special effects such as bold or italics. Control word 9 must be set to the length in words of this buffer. The buffer should be at least four times the size of the largest character in the font.*

*After setting up the control array, you should be able to select the font using the* **set_font** *function. When you are done with the font, or with your program, do a regular* **vst_unload_fonts** *call so the VDI doesn't continue to think the font is available.*

*(Thanks to John Feagens of Atari for this information.)*

stored. This requires a more detailed understanding of their format, because the VDI does not provide an input function. The definitive document on metafile format is currently the GEM VDI manual, but the following paragraphs should provide enough information to allow you to do some exploration.

### STRUCTURE

The metafile is composed of a header followed by an arbitrary number of commands. Each command consists of at least four words, and is generated by one of the functions given in the table in *figure 3*. The first word in the four is the command opcode. The second word is the number of vertices in the command. This field is used with the polyline and polymarker commands, for instance. The third word is the number of integer parameters required for the command. The fourth word is the sub-opcode, when it is required. If any of the last three fields are not used for the command, they are forced to zero.

Immediately following the required four words are the vertices, if any were specified. This is an image of the **ptsin[]** binding array associated with the VDI function which created the command. After the vertices, the integer parameters, if any exist, are written. They are an image of the **intin[]** binding array.

The close workstation command forces a final single word command to be written at the end of the metafile. It is a word of all ones (hexadecimal FFFF).

The standard metafile header consists of 14 words. Only the first four are required. The first word is a tag value, again hexadecimal FFFF. The next word is the length of the header in words. The third word is a version tag, and the next word shows whether the metafile workstation was opened for Raster Coordinates (RC), or Normalized Device Coordinates (NDC).

The remaining entries in the header provide information which will help OUTPUT.PRG when it attempts to regenerate the drawing on another device. You may set them using VDI calls after opening the metafile workstation. If they are not set, they are filled with zeros, and OUTPUT will use default values to generate the display.

The first four optional values define (in order) the minimum and maximum X and Y coordinates to be found in the metafile. This allows OUTPUT to bound the space in which graphics will appear. Your application may set these values with the **v_meta_extents** call.

The next two words in the header are the physical page size on which the graphic should be reproduced. The units are tens of millimeters, and the values are given in width-height order. While Appendix H of the VDI manual describes this call, there is no standard binding provided, so you will have to write your own if you wish to use it. If you do not set this value, OUTPUT will do its best to fit the drawing onto the target device, using the assumption that its pixel elements are square.

The final four words in the header may be used to define the coordinate space for the metafile. They contain, in order, the X and Y coordinates of the lower left corner of the drawing area, and the X and Y values of the upper right corner. These values are mapped onto the drawing area as defined in the last paragraph. Again, there is no standard binding for this function, but it is described in the VDI appendices. If the information is omitted, OUTPUT will use a default raster or normalized coordinate space, depending on how the metafile was opened.

As a last refinement, consider the **v_write_meta** VDI function. This works only with metafiles, and allows you to insert commands of your own definition into the metafile. This capability could be useful, for instance, in establishing the data/object relationships in the applications proposed earlier.

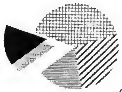
If you use this ability, you must avoid sub-opcodes 10, 11, 50, 51, and 81, which are used by Digital Research's GEM Draw program. These codes are given special handling by OUTPUT; their functions are defined in the VDI appendices should you wish to use them. With these exceptions, OUTPUT will ignore commands inserted with **v_write_meta**, since it has no idea how they should be interpreted. You may also encounter problems if the metafile is later edited using a graphics program which does not understand your special commands. In most cases, the information will be lost in the editing process.

In this article, I have tried to show how GDOS and metafiles fit into the framework of the ST's software, and how they can be used to avoid wasting effort on rewriting of device drivers. If more applications appear which use the GDOS, it should encourage manufacturers of printers and other peripherals who want to sell to the Atari market to write drivers compatible with GDOS. This can save us all time which can be better spent on programs that stretch the abilities of the ST. ■

## REFERENCE:

- *Smalltalk-80: The Language and its Implementation*, by Adele Goldberg and David Robsen, Addison-Wesley Publishing Company, Menlo Park, CA
- *Sketchpad: A Man-Machine Graphical Communication System*, by Ivan E. Sutherland, Proceedings - Spring Joint Computer Conference, 1963, pp. 329-346.
- *The Programming Language Aspects of Thinglab, a Constraint-Oriented Simulation Laboratory*, by Alan Borning, ACM Transactions on Programming Languages and Systems, 3:4, October 1981, pp. 353-387.

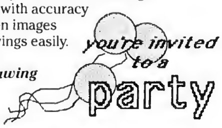
**EASY-DRAW®**
**IT'S A TOOL, NOT A TOY.**

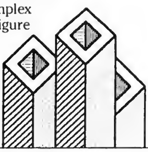


**There is** a difference between paint and draw programs. Paint programs are recreational packages that allow freeform painting on a dot-by-dot basis. With each new stroke you obliterate everything you cover. And erasing permanently removes everything you've created.

An object-oriented drawing program like Easy-Draw is a versatile, powerful tool you use to create business graphics, presentation materials, line drawings complex illustrations on a figure-by-figure basis. It lets you: lay down solid or transparent figures to build composite drawings • size, move and manipulate objects individually and collectively • use a grid system for controlled, precise scale drawings • produce print-outs with accuracy exceeding your screen images • create custom drawings easily.

***Ask your dealer for Easy-Draw, the drawing program for professionals.***





easy·draw®

*MIGRAPH™*
*720 S. 333rd St., Suite 201*
*Federal Way, WA 98003*
*(206) 838-4677*

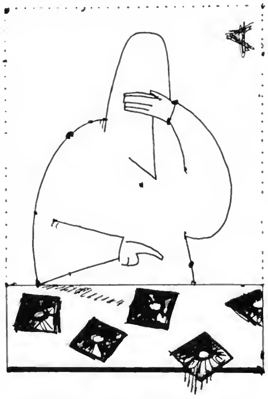
MIGRAPH

Easy-Draw is a registered trademark and Migraph is a trademark of Migraph, Inc.

# PRACTICAL SOFTWARE FOR THE NONDEVELOPER

## *JUST HOW USEFUL IS THE ST?*

**by Jack Powell**
ASSOCIATE EDITOR


ILLUSTRATIONS BY MACIEK ALBRECHT

When it comes to everyday applications—word processing, spreadsheets and databases—how practical is the ST? The 520ST has been available for almost a year and there is no longer any question of support from a growing list of software producers. But what programs actually transform this 68000-based bargain box into a viable, practical tool?

START examined practical ST products in three major categories: word processors, databases and spreadsheets. Synopses by category and by product are based on the opinions of START editors, plus those of several outside reviewers who tested these products for Antic magazine. ST programming languages are also examined as they are the tools by which "practical" products are created.

If a product was reviewed in Antic, we include the date of the review. Products marked "FINAL" are currently available. To indicate the future direction of practical ST software, we have included announced products, labeled "PRESS," for press release. Unless otherwise noted, all editorial commentary is made on products which are in final release state and have been submitted for review. If a product is copy protected, it is so noted. START welcomes comment and rebuttal from both users and manufacturers.

### AN OVERVIEW

ST word processors range in price from $145 to absolutely free. At the high end, Final Word, from Mark of the Unicorn, is the only really full-featured word processor currently available. But it is a throwback to the mainframe systems and unforgivably complicated on a machine designed for friendly interface. At the other price extreme, 1ST Word, from Atari, is more in the ST style. It is mouse driven, has plenty of GEM windows and bright graphics, but it doesn't support such elementary print commands as line spacing. Verdict: the Atari ST still lacks a powerful, up-to-date word and document processor that takes advantage of the machine's architecture.

Users of dBase II/III will be right at home on the ST. With both H & D Base, from Mirage Concepts, and dBMAN, from VersaSoft, on hand, many currently available dBase II/III programs may be ported directly to the Atari ST. Those who find dBase-type databases unnecessarily complex may prefer the simplicity of DB Master One, from Atari. It is not, however, relational and has very limited report capabilities. Again, the former programs are all-text, IBM clones, while the latter is simple and uses friendly GEM. ST owners are still waiting for such friendly powers as R:base 5000 or Framework.

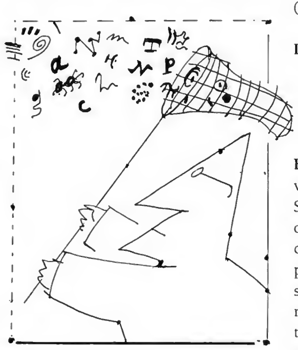
What? Only two spreadsheets? That's right. Early press releases from several companies touted Lotus 1-2-3 clones they were preparing for the ST. Only VIP came through with a product. The pattern continues here: the all-text, powerful VIP, and the simpler, GEM-based A-Calc.

Languages are included in this overview to show that tools exist to create new software. There are plenty of languages for the Atari ST. Unfortunately, all are primarily designed for developers. Currently, no languages are available for the hobbyist or casual user. ST LOGO and ST BASIC are hopelessly clumsy. The remaining languages are compilers, most with complicated linking procedures and numbing compile and link time. The ST desperately needs a language that is fun to use, like Turbo Pascal, or ACTION! on the 8-bit Atari's.

## WORD PROCESSORS

**1ST Word**
Atari Corp., 1196 Borregas Avenue,
Sunnyvale, CA 94086,
(408) 745-2000
Free
FINAL

This fully GEM-based word processor—which is bundled with the 1040ST and 520ST—is currently the best, entry-level, GEM-based word processor. It also



works nicely as a programmer's text editor. Limitations in print formatting features—including the inability to print in anything other than single-space—detract from this otherwise excellent product. Any company planning to charge money for a word processer has stiff competition here. Not copy protected. (Antic review 6/86)

**Final Word**
Mark of the Unicorn, 222 Third
Street, Cambridge, MA 02142,
(617) 576-2760
$145.00
FINAL

A thorough—but pricey—text-oriented word processor/document processor. Described by our reviewer as a clone of the IBM PC Perfect Writer, this is an entirely command-driven word processor with no GEM interface. The program, which uses virtual memory, is packed with features including multiple-buffers. It was found to be powerful, but difficult to learn. The documentation, as well as the program, reflects the fact that Final Word was a straight port from IBM and, in many cases, does not take advantage of the ST's individuality. Copy protected. (Antic review 4/86)

**Let's Write**
Mark Williams, 1430 W. Wrightwood,
Chicago, IL 60614, (312) 472-6659
$99.95
FINAL

Essentially, this is Micro EMACS with word-wrap—which is familiar to most ST developers since it was provided, in one form or another, with the ST developer's package. (EMACS was a popular programmer's text editor on mainframe systems.) Let's Write is an entirely command-based program that is obviously a throwback to the Unix mainframe systems. The package includes a spell checker, and the Kermit telecommunications module (which was also supplied with the developer's kit). Not copy protected.

**HabaWriter**
Haba Systems, 6711 Valjean Avenue,
Van Nuys, CA 94106, (800) HOT
HABA (US), (800) FOR-HABA (CA)
$74.95
FINAL

HabaWriter was the first GEM-based word processor for the ST. Ian Chadwick, who reviewed it for Antic, found version 1.0 to be buggy and mostly unusable. Since then, Haba has released version 1.1, which has not yet been evaluated. Copy protected. (Antic review 3/86) ▶

## SOFTWARE . . .

Express
Mirage Concepts, 4055 W. Shaw, #8, Fresno, CA 93711, (800) 641-1441
$49.95
FINAL

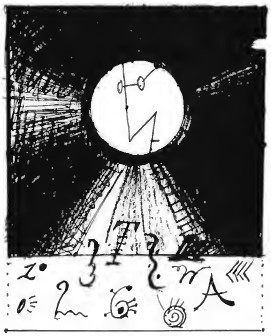
Express is a "letter processor," limited in word processing capabilities and designed primarily for simple letter output with mailmerge. This was the first word processor available for the ST and, as such, was roundly—and unfairly—condemned by the new ST community. Express is not really a word processor but is designed as an easy-to-use letter designer with simplified mailmerge system. As such, it works fine. Not copy protected. (Antic review 1/86)

HomePak
Batteries Included, 30 Mural Street, Richmond Hill, Ontario, CANADA L4B 1B5, (416) 881-9941
$69.95
PRESS

Integrated word processor, terminal program, and database fashioned after the popular 8-bit program of the same name by Russ Wetmore. Batteries Included plans to have this product on the market by June 1986.

ST Writer
Atari Corp., 1196 Borregas Avenue, Sunnyvale, CA 94086,
(408) 745-2000
Free
FINAL

John Feagans and a team of Atari programmers got fed up waiting for a word processor to come out for the early ST's. So they re-wrote the code of the 8-bit Atari Writer program and got it up and running on the ST (all in two weeks, it is rumored). The program can currently be found on CompuServe and in several user's group libraries. We use this program in-house, partly because it can adapt to most of the many incompatible text formats available on the ST.



### DATABASES

H & D Base
Mirage Concepts, 4055 W. Shaw, Fresno, CA 93711, (800) 641-1441
$99.95
FINAL

A dBase II clone (80% compatible). The first "serious" available ST database. Fashioned after dBase II, H & D Base can use dBase II command programs ported from other machines. Also, most of the many dBase II books found in stores will apply to this program. Interestingly, since H & D Base was written in Forth, the programmers have left in a command to "turn on" the Forth kernel so you can program in Forth as well as H & D Base. In April 1986, some bugs were being ironed out of the first releases, but customer support seems to be excellent. Not copy protected. (Antic review 7/86)

dBMAN
VersaSoft, 723 Seawood Way, San Jose, CA 95120, (408) 268-6033
$149.95
FINAL

dBMAN is currently the only available relational database for the ST which is compatible with both dBase II and dBase III. The final package arrived in our offices in April 1986, too late for review. An introductory price of $99.95 will be offered until July 1st. Not copy protected.

DB Master One
Atari Corp., 1196 Borregas Avenue, Sunnyvale, CA 94086,
(408) 745-2000
$49.95
FINAL

This easy to use, non-relational, GEM-based database was packaged with the ST during December, 1985. DB Master One is an excellent "simple" database. It is very fast and easy to use with limited reporting capabilities and field types. Stoneware plans on releasing packages that will increase the complexity of this database. In the meantime, this is a very good package for organizational uses. Not copy protected. (Antic review 6/86)

Zoomracks
Quickview Systems, 146 Main Street, Los Altos, CA 94022,
(415) 965-0327
$79.95
FINAL

Zoomracks is an oddity. Based on the structure of rack cards (such as "punch-in" time cards), this program is more a data organizer than a true database. Currently there are no mathematical capabilities. The card rack metaphor would seem an excellent use of GEM, but Zoomracks is another IBM PC port which does not use GEM. The screen has a cluttered appearance. The success or failure of this product will depend upon the willingness of ST owners to adapt to its unusual metaphor—assuming that metaphor works. Not copy protected. (Antic review 6/86)

**Hippo Simple**
Hippopotamus Software,
985 University Avenue, Suite 12,
Los Gatos, CA 95030, (408) 395-3190
$49.95
FINAL

An "easy to use" home database, found by our reviewer to be frustrating and confusing—mainly because of the poor documentation. Version 1.0 seemed to "have been rushed to market a little too quickly." Copy protected. (Antic review 3/86)

**The Manager**
BMB Compuscience Canada, 500 Steeles Avenue, Milton, Ontario L9T 3P7, Canada, (416) 876-4741
$169.95
PRESS

The Manager is allegedly the powerhouse relational database system ST owners are waiting for. Its firm describes it as a "paperless office". The implication here is that this product will eliminate the need for paper in your office. Well . . . we'll see. The product was first demonstrated at the November 1985 COMDEX in Las Vegas. It looked pretty impressive—but still no GEM.

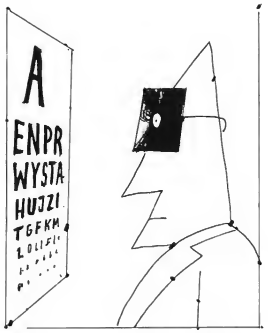
**db One**
Oxxi, 3428 Falcon Avenue, Long Beach, CA 90807, (800) 453-4900
$59.00
PRESS

A relational database which is promised to include mailing list, labeling, checkbook, and inventory.

## SPREADSHEETS

**VIP Professional**
VIP Technologies Corp., 132 Aero Camino, Santa Barbara, CA 93117, (805) 968-4045
$179.95
FINAL

VIP Professional is allegedly compatible with Lotus 1-2-3, and is currently the only full-featured, heavy-duty spreadsheet for the ST. This product tumbled through some rough PR waters during its release and is still suffering from a legal fight between parent company, VIP Technologies, and Shanner International, the marketing firm hired by VIP to launch their product. Originally promised—and advertised—as a GEM product with windows, the "mouse" version of VIP had not yet appeared in April



1986. The Antic reviewer found this to be a good product—if a little slow. Antic also recommended any buyer contact VIP and be sure of full customer support before purchase is made. Copy protected. (Antic review 5/86)

**A-Calc**
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$59.95
FINAL

A-Calc is a GEM-based spreadsheet created in Great Britain which holds 256 columns and 512 rows. This mouse-driven spreadsheet is easy to use, but not as complete as a Lotus 1-2-3 system. Not copy protected.

## LANGUAGES

**MegaMax C**
Megamax, Box 851521, Richardson, TX 75085, (214) 987-4931
$199.95
FINAL

Full, one-pass C compiler with editor and "smart" linker. Arguably the finest C compiler available for the ST, this package is currently crippled by a 32K array dimension limit and a 32K code limit. These limitations—which the company promises to repair—are imposed by the Macintosh OS, from whence this system was ported. Compile and link time is extremely fast and end code is relatively small. In most cases, MegaMax C is compatible with Alcyon C. The price, however, is pretty hefty. For another $100, you could get the developer's kit—and all the GEM documentation. Not copy protected.

**Metacomco Lattice C**
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$149.95
FINAL

A complete development C. Lattice C is the industry standard C and currently the only commercially available ST C which is at least as complete as Alcyon C in the developer's toolkit. Its linker, ▶

however, is not completely compatible with Alcyon's. Not copy protected.

**GST-C**
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$79.95
FINAL

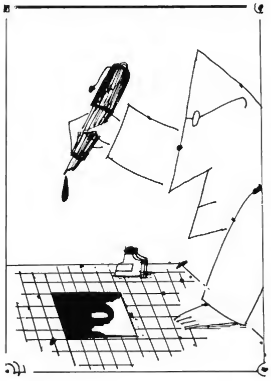
This C implementation is easy to use and includes a 1ST Word-type editor and GEM shell. An added bonus is the excellent "superstructure library" which simplifies many GEM commands much like OSS's Personal Pascal. Major shortcoming: no floating point and uses same link as Lattice C which, therefore, makes it incompatible with Alcyon C. Not copy protected.

**Haba Hippo-C**
Haba Systems, 6711 Valjean Avenue, Van Nuys, CA 91406, (800) HOT-HABA (US), (800) FOR-HABA (CA)
$59.95
FINAL

You get what you pay for when you buy Haba Hippo C, the least expensive of available ST C's . It does not support floating point arithmetic and contains a number of compatibility problems, many due to its UNIX-like shell called HOS. Few ST-specific C programs written in the developer's Alcyon C will compile properly in Hippo-C. Unfortunately, this was the first commercially available C for the ST market and a lot of owners snapped it up. Copy protected. (Antic review 2/86)

**Personal Pascal**
Optimized Systems Software, 1221-B Kentwood Avenue, San Jose, CA 95129, (408) 446-3099
$74.95
FINAL

Personal Pascal is the closest thing to a friendly language that currently exists for the ST. It is also one of the only languages to include detailed documentation on how to access and use the complex GEM VDI and AES commands. In another first, its editor actually includes an auto-indent feature, helpful in structured program formatting. From experience, we can add that any product purchased from this company includes excellent customer support. Not copy protected. (Antic review 5/86)



**Metacomco Pascal**
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$99.95
FINAL

A full, ISO 7185 standard which compiles to native code. Not copy protected.

**Modula-2/ST**
TDI Software Ltd., 1040 Markison Road, Dallas, TX 75238, (214) 340-4942
$149.00
FINAL

Created by Professor Niklaus Wirth, the inventor of Pascal, ST owners are truly lucky to have an opportunity to try this new language. Our reviewer found this to be a generally solid implementation of the language, though with weak documentation (particularly of the GEM system), and clumsy user interface. Not copy protected. (Antic review 5/86)

**H & D Forth**
Mirage Concepts, 4055 W. Shaw, #108, Fresno, CA 93711, (209) 227-8369
$39.95
FINAL

An 83-standard Forth with access to all GEM commands as well as BIOS and XBIOS commands. Holmes & Duckworth used this Forth for in-house development of such products as H & D Base. This is a "best buy" for ST Forth programmers. It deviates from 83-standard in that the stack is 32-bits wide, so there are no double words, and multiple dictionaries are not allowed. Otherwise, H & D Forth is very thorough and reasonable. Remarkably, no royalties need be paid for professional development with this product. Not copy protected. (Antic review 6/86)

**4xForth**
The Dragon Group, 148 Poca Fork Road, Elkview, WV 25071, (304) 965-5517
$99.95
FINAL

An 83-standard developer's Forth which includes limited multi-tasking capabilities. 4xForth was the first commercially available language for the ST (not counting developer's C and LOGO). Customer support seems to be excellent with this company. By April 1986, however, free

Forths are beginning to appear in public domain, and H & D Forth certainly seems a comparative value. Not copy protected. (Antic review 12/85)

**Cambridge LISP**
Metacomco, 26 Bristol Square, Bristol, United Kingdom BS2 8RZ
$199.95
PRESS

This is the only LISP currently planned for ST. After some delay, Metacomco still plans releasing this language for the ST. You can expect it late summer or early fall.

**DevPacST**
HiSoft, 180 High Street North, Dunstable Beds, United Kingdom LU6 1AT, (0582) 696421
$79.95
FINAL

A complete assembler-editor system. (See Christopher Chabris review, this issue of START.) Not copy protected.

**A-Seka**
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$39.95
FINAL

A high-speed, fully RAM-based assembler with debugger. (See Christopher Chabris review, this issue of START.) Not copy protected.

**GST-ASM**
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$59.95
FINAL

GEM-based assembler/editor. (See Christopher Chabris review, this issue of START.) Not copy protected. ■

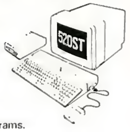
**DISCOUNT SOFTWARE**

ABBY'S

37 S. Broad St.
Fairborn, Ohio 45324

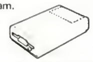
ORDERS ONLY PHONE: 1-(800)-282-0333
CUSTOMER SERVICE & OHIO RESIDENTS 1-(513)-879-9699

**ATARI 520 ST**
**PERSONAL COMPUTER SYSTEM**
**$659.95**



**$839.95 with Color Monitor**

**The Atari 520 ST Comes Complete With:**
**ST LOGO™ and ST BASIC™** to create programs.
**1ST WORD** is the perfect wordprocessing package for all letter and business writing
**NEOchrome Sampler** is a full-color painting program.



**2nd Drive For Only $69.95** Reg. List $199.95
**When you buy the complete 520 ST, SF 354 Drive & Monitor System**

**TOP HITS**

| | | | |
|---|---|---|---|
| VIP Professional | $109 | Silent Service | $32 |
| Haba Hippo C Compiler | $ 59 | Master Type (Ages 6+) | $32 |
| ACCESS Leader Board | $ 32 | Print Master | $32 |
| FIREBIRD The Pawn | $ 34 | Art Gallery I | $24 |
| FIREBIRD Starglider | $ 34 | Black Cauldron | $35 |
| Temple of Apshai Trilogy | $ 32 | Ballyhoo | $29 |
| EPYX Rogue | $ 32 | Mean 18 | CALL |
| Winter Games | $ 32 | Universe II | CALL |
| F 15 Strike Eagle | $ 32 | Flight Simulator II | CALL |

**SOFTWARE ONLY — Prepaid orders over $50 receive free shipping via UPS in continental U.S. Please add $2 orders under $50. HARDWARE and all orders requiring shipment via U.S. Post Office are subject to additional freight charges. Add $5 for COD orders. VISA/MasterCard orders add $2 service charge. Ohio residents add 5.5% sales tas. Personal checks require a three-week waiting period. No waiting when paid by credit card, certified check or money order. All items subject to availability and price change. WE CHECK FOR STOLEN VISA & MC. ALL RETURNS MUST BE ACCOMPANIED BY AN AUTHORIZATION NUMBER.**

**ProCopy**

**INTRODUCTORY OFFER**

**$24.95 ST BACKUP UTILITY**

Regular $34.95 Offer ends July 31, 1986

Software can easily be damaged by heat, humidity, magnetic fields, wear & tear or just dirt. The exposed surface of a disk can be affected by even a touch. The best way to preserve your software is to make backups. ProCopy provides the solution.

- Duplicates virtually any protected disk.
- Reasonable update policy available.
- Protects against accidental loss of expensive software.
- Not copy-protected for your convenience.
- ProCopy works on all Atari ST computers.

**ORDER TODAY AT THIS REDUCED RATE**

Send $24.95 (check or money order). VISA & MC accepted. Add $2.00 shipping & handling in U.S. & Canada. Overseas shipping & handling add $4.50.
Atari ST is a registered trademark of Atari Corp.

**Available Only From**

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

# DISK INSTRUCTIONS

Welcome to the START Disk Directory. All program listings referred to in this issue are on your START disk, which is provided in a special envelope bound into the magazine. We recommend you read articles related to the programs on the disk before attempting to run the program. If you just can't wait, jump on down to FAST START. (If you purchased the $4 version of START, without disk, you can still purchase the disk by sending $10.95 for disk, plus $2.00 for postage and handling, to START DISK, 524 Second St., San Francisco, CA 94107.)

Use a pair of scissors to open your disk envelope along the outside vertical edge. Remove your START disk, place it in your disk drive and click the disk icon to see its contents. (Please refer to your ST owners manual if you are uncertain of proper Desktop procedures.)

Your START disk contains six folders, each with programs, listings and data related to its particular START article. A runnable version of each program (usually recognized by its .PRG, or .TOS extender) is included, along with all related source code. Also, each folder contains a .BAT "batch" file specific to its program. Detailed information on the batch files and program compilation is contained in README.TXT. All programs were tested and compiled with developer's Alcyon C on a 520ST with TOS in ROM.

The START disk is single-sided and very full. We recommend transferring programs you wish to try to another disk. Desk Accessory programs with .ACC extenders must be transferred from their folders and placed on a "boot" disk before they will operate properly.

### *FAST START*

Write protect your START disk before trying any programs. Slide the disk tab until you see light through the little hole. To be really safe, make a back-up copy of your disk.

**COLRMOVR.STQ**: Joe Chiazzese's demonstration of 16-color icon movement works only in low-resolution. Open the folder and click on NEWMOUSE.PRG. Your screen will show a karate character which may be moved with the mouse. Exit the program by pressing [Esc]. NEWMOUSE.C is the C source code; COEUTIL.S, the AL source code; KARATE.S, the karate character bit image; and NEWMOUSE.BAT, the batch file for compilation.

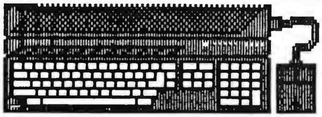
**MAGNETS.STQ**: David Small's magnetic field line program will work in any resolution. Click on MAG21.PRG and read the instructions in David's article. MAG21.C is the C source code of the program, and MAG21.BAT the batch file.

**MIDISEQR.STQ**: You'll need a MIDI-compatible synthesizer for this. Hook up both input and output lines between synthesizer and computer. Click on SEQUENCR.PRG to run the program, then read Tom Jeffries' article for instructions. Two demonstration songs, ARABESQU.SNG and ENTRTAIN.SNG, have been included, courtesy of Micro-W Distributing, Inc. Please print (c)NOTICE to screen or printer for important information. SEQUENCR.C is the main source code and SEQUENCR.BAT the batch file.

**PRNTDRVR.STQ**: Tom Hudson's program is a Desk Accessory which must be transferred to a boot disk. It will only work with TOS in ROM. Transfer INSTALL.ACC, and any of the .PRT files which match your printer. We have included four driver files: EPSON.PRT, for Epson printers; PROWTR.PRT, for NEC/Prowriters; CGP220/PTR, for the Radio Shack color printer; and OKI20C.PRT, for the Okimate-20 color printer. Please note, the Okimate-20 driver is for those Okimates configured with standard IBM interfaces. Many more drivers may be found on CompuServe (see article). Your local Atari user group may be helpful if you can't get on CompuServe. Once you have transferred the above files and booted the disk, you will find "Printer Driver" in the drop-down Desk menu. Complete instructions may be found in Tom's START article. INSTALL.C is the C source code; INSTAL.S is source for the AL routines; EPSON.S and JX80C.S are sample printer driver sources; SAVER.C is source for the printer driver generator; and INSTALL.BAT and SAVER.BAT are batch files.

**ROUTINES.STQ**: Dan Matejka's C and AL routines may be tested by clicking on PRTSHELL.PRG. This program will call FILE.ONE and FILE.TWO, so they must be on the same disk as PRTSHELL.PRG. PRINTOUT.C is the main C source code; MEMOPS.S, the source for the AL routines; PRTSHELL.C, is C source for the demo shell program; and PRTSHELL.BAT the batch file.

**GEMTEXT.STQ**: To run Corey Cole's GEM text demonstration, you will need both TEXTDEMO.PRG and TXDEMO.RSC on the same disk. Click on TEXTDEMO.PRG and follow the instructions in Corey's article. TEXTDEMO.C is the main C source code; TEXTDEMO.H, the defines for TEXTDEMO.C; TXDEMO.C and TXDEMO.DEF, are resource file sources; TXDEMO.H is the resource file header; TEXTDEMO.BAT, the batch file; and DEBUG.C is a generic debugging aid. ■



ST software from

P.O. Box 5228 Springfield, VA 22150

(703) 644-8881 ADD $2 PER TITLE FOR P&H

Tired of games, databases, spreadsheets, and writing letters on your new *520ST* or *1040ST*. What do you do with your NEOCHROME and DEGAS artwork? Welcome to XLENT Software's CREATILITIES for the ATARI ST.

**TYPESETTER ST** ≡ **RUBBER STAMP ST** ≡ **MEGAFONT ST** ≡ **ST MUSIC BOX**

**TYPESETTER ST** by DORFMAN & YOUNG

*This CREATIVE package is limited only by your own imagination.*

* Complete page designing flexibility * Use your DEGAS and NEOchrome art * HIGH resolution printouts * Use custom fonts * Supports both color and mono * Print to Epson w/Graftrax, NEC, Prowriter, Panasonic, Gemini, SG and compatibles. * $39.95 *

**RUBBER STAMP ST** by YOUNG & DORFMAN

Manipulate your artwork: Expand * Shrink * Rotate * Flip * Mirror * Add text * Fill * Overlay and more * Create ads logos, Typesetter ST icons.... Print half page, labels, index & Rolodex cards. * $39.95 *

**MEGAFONT ST** by DELLINGER & ROGNLIE

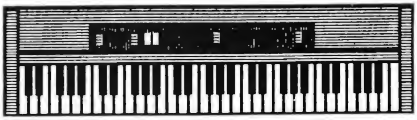
Print text files in different fonts, print graphics in multiple sizes, create fonts. $39.95

NEOCHROME is a trade mark of ATARI
DEGAS is a trade mark of BATTERIES INCLUDED

THIS AD WAS CREATED USING ***TYPESETTER***

**ST MUSIC BOX** by DORFMAN & YOUNG

* Mouse driven editor * Animated player *



* Compose in up to 8 voices * $49.95 *

# SOPHISTICATED TEXT HANDLING

## A Window On GEM Special Effects

by Corey Cole

How to manipulate text in GEM. A thorough investigation of sophisticated text handling. Technical tips—and pitfalls to avoid. Including a full-featured C program that lets you test all possible GEM text capabilities. Related files may be found within the GEMTEXT folder on your START disk.

ILLUSTRATION BY JOHN HERSEY

Before the advent of Apple's Macintosh, microcomputer programmers had an easier time using text in their programs. Computer screens generally could only display a single font. As computers became more sophisticated, users began to expect more from them: the Macintosh delivered reasonably priced proportional spacing and multiple type fonts.

Thanks to GEM, the Atari 520ST has many of the advanced text display features previously found only in expensive, dedicated publishing systems. Let's explore the use of some of these.

## USING GEM TEXT

GEM helps programmers by doing much of the hard work needed for sophisticated text display. You can ignore most of GEM's overwhelming set of operations for displaying and dealing with graphics text. If you are just trying to display plain text in a window, you only have to worry about where to put it. Beyond that, you can use as many of the more advanced features as you need.

Think of GEM's VDI calls as a toolkit; the basic set of tools for text consists of only seven VDI calls. The calls **v_gtext** and **v_justified** actually display the text. You can set text size in pixels with **vst_height**, or in points (a typographer's unit measuring 1/72 of an inch) with **vst_point**. The **vst_effects** call specifies characteristics such as boldface, italics, underlining, outlining, shadowing, and "light" (grey shading); **vst_alignment** positions and aligns the text. You can check the width of a string or of a single character with **vqt_extent** and **vqt_width**.

Depending on your application, you may need **vst_rotation** and **vst_color** (which set baseline rotation and foreground/background colors), and **vqt_attributes** (which returns the characteristics of the current typestyle). Once the final GDOS becomes available (see "GDOS & Metafiles" by Tim Oren, this issue), you may have use for **vst_load_fonts**, **vst_unload_fonts**, and **vst_font**—which allow you to deal with proportionally spaced and alternative typestyles. Finally, you may need **vswr_mode** and **v_bar** to deal with "reversed" (white on black) text or italic characters, and **vs_clip** to keep your text in the window.

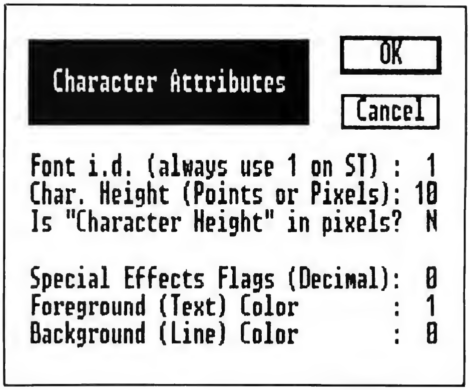
Unfortunately, there is a darker side to the ST's GEM. Several of the above features were omitted from the Atari version of GEM; others have bugs. A few features crucial to some applications never found their way into GEM at all. This makes developing text applications on the ST a real challenge—and tends to turn your life with GEM into a love/hate relationship.

At this point, let me recommend Tim Oren's article on GEM text—column #10—which may be found in the Professional GEM section in ANTIC OnLine on Compuserve. (Many Atari user groups have reprints of this article as well.) As Tim mentions, it is a good idea to start your text display on a byte boundary in memory (X-coordinate that is a multiple of eight). You should use the "Replace" writing mode and, whenever possible, try to do your own clipping (rather than relying on the **vs_clip** function). Each of these techniques improves text display speed.

## PUTTING IT TOGETHER

TEXTDEMO.PRG, on the START disk, demonstrates the use of most of GEM's text display features. It opens a window and displays lines of text according to your specifications. With TEXTDEMO, you can easily display several lines of text with different attributes, since TEXTDEMO will not erase the window until you give it an explicit "Erase Window" command.

FIGURE 1.



```
Font i.d. (always use 1 on ST) :  1
Char. Height (Points or Pixels): 10
Is "Character Height" in pixels?  N

Special Effects Flags (Decimal):  0
Foreground (Text) Color        :  1
Background (Line) Color        :  0
```

To use TEXTDEMO, pull down the "Display" menu, and select "Char Attributes." The Character Attributes dialogue is shown in *figure 1*. The first field is the "Font i.d." (as in the **vst_font** VDI call). Changing the Font i.d. has no effect currently; this option is included for use after GDOS is released (see the "Bug Box" sidebar).

Character Height may be specified in either pixels or points. Note that the ▶

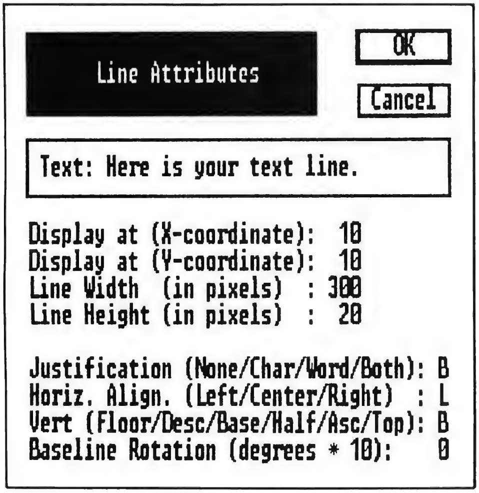
number of pixels applies only to the part of the character above its baseline.

The Special Effects value is the sum (in decimal) of the special effects bits you wish to be set. These bit values are: 1 = bold, 2 = light, 4 = italic, 8 = underline, 16 = outline, and 32 = shadowed. Try various combinations. The effects of the Foreground and Background Color values depend on your current resolution and color map, but 0 is generally white, while 1 is generally black. You can get some very interesting effects on a color monitor!

pixels relative to the display window's top left corner, and is the line's left baseline position.

A single letter specifies Justification. Character justification adds space between characters, while word justification adds space only between words.

Horizontal Alignment is also specified with a single letter, which determines the horizontal "anchor point" for text display (left edge, centered, or right edge). There is no visible difference between the various horizontal alignments when justification is in effect.



```
Text: Here is your text line.

Display at (X-coordinate):  10
Display at (Y-coordinate):  10
Line Width  (in pixels)  : 300
Line Height (in pixels)  :  20

Justification (None/Char/Word/Both): B
Horiz. Align. (Left/Center/Right)  : L
Vert (Floor/Desc/Base/Half/Asc/Top): B
Baseline Rotation (degrees * 10):    0
```

FIGURE 2.

After clicking on the OK button, select "Display Line" from the Display menu. (For convenience, pressing any key on the keyboard is equivalent to selecting "Display Line".)

The Line Attributes dialogue (*figure* 2) allows you to specify a line of text to display (up to 30 characters), and its positioning and alignment. Position is in

Vertical Alignment determines the position of each character relative to the specified Y-coordinate. That coordinate will fall on the top line, ascent line, half line, base line, descent line, or bottom line of the character cell. Since "Bottom" and "Base" begin with the same letter, we use "F" (Floor) for bottom alignment. The results of changing vertical alignment value will probably surprise you; specifying "Floor" moves text up with respect to the drawn "Baseline," while specifying "Top" moves the text down.

Finally, the Baseline Rotation value specifies a vector along which to display the text's baseline. Even though this is specified in tenths of a degree (0 is normal, 900 is straight up, etc.), there are only four meaningful ranges on the ST screen—normal, backward, up, and down.

Accepting the line attribute dialogue (pressing [Return] or clicking on "OK") displays the text line. If you are just changing character attributes, and want an immediate display, an easy trick is to press any keyboard key, then press [Return]; this brings up the line attribute dialogue, then accepts it without changing any line attribute values.

TEXTDEMO draws a rectangle around your nominal text line in color 2, and draws a dotted line along the baseline in color 3 (both are black on the monochrome monitor). If you are using baseline alignment, the text will be vertically centered in the line rectangle.

If your text window becomes too cluttered, you can use "Erase Window" at any time. "Quit" in the Exit menu returns you to the GEM desktop (or your shell). There are several ways to crash the system by specifying unusual combinations of the dialogue parameters. This is intentional—it allows you to find out in advance where GEM is most likely to choke on your own text applications.

## HOW IT WORKS

You should be able to learn quite a bit about how the various GEM text functions interact by first trying several minor variations of character and line attributes and then displaying the resulting strings above or next to each other. Start by experimenting with the program in this fashion, before spending significant time with the code and "how it's done." The source code for TEXTDEMO.PRG is in several files—TEXTDEMO.C and TEXTDEMO.H contain the main code, while DEBUG.C contains procedures for formatting numeric strings (used in TEXTDEMO's dialogue-handling routines). In addition, TXDEMO.RSC, TXDEMO.DEF, and TXDEMO.H contain TEXTDEMO's resources (the menu and dialogue boxes). DEBUG.C is generic—you can use it to debug your own GEM applications.

The general flow of control in TEXTDEMO is similar to that of Digital Research's "doodle/demo" application, from which several pieces were borrowed. TEXTDEMO begins (in **gemInit**) by initializing the desktop and opening a single window which completely fills the desktop's work area. The initial text line is set to fill most of the window's width, and to be vertically centered in it. The initialization routine also allocates and clears a buffer large enough to cover the entire screen. This is the "save buffer," and lets TEXTDEMO easily redraw the text window after it has been covered (as by the attribute dialogues).

After initialization, control passes to **mainLoop**, which consists entirely of an **evnt_multi** in a loop. The **evnt_multi** looks only for messages and keyboard events (treating any of the latter as equivalent to selecting the "Display Line" menu item). The only messages considered are "menu item selected" and "redraw" (the latter happens on beginning the program, and whenever a window leaves the desktop, as after a dialogue is displayed). Redraw is accomplished by copying the save buffer back into the text window.

Selecting the "Char Attributes" menu item invokes **charDial**. This procedure initializes the fields of the Character Attribute dialogue, then calls GEM's form handler, and finally (if the dialogue was not cancelled) stores the user-entered values back into the global **textLine** structure. The initialization and saves are performed by **initChDial** and **saveChDial**. Four general-purpose procedure set and retrieve values from the dialogue fields: **getName**, **getNum**, **setName**, and **setNum**. You may find this approach useful for editing complex dialogues in other applications. The save routine also uses a simple "character decoder" (**decode()**) to convert an entered character into the corresponding numeric value—this is easy, since GEM's attributes are all set with consecutive low numbers.

**lineDial**, invoked when a key is pressed, or when the "Display Line" menu item is chosen, is only slightly more complicated. **lineDial** uses the same approach as **charDial** to initialize and read the dialogue. When the dialogue is OK'd, **lineDial** restores the window—this is unnecessary for the other dialogues, since dismissing them causes GEM to issue a Redraw message to the application. Here, however, we want to display another line, then save the modified work area, which means we need to clean out the window first.

As is typical in GEM (and in other graphics-oriented environments), something like 90 percent of our code is devoted to handling the user interface. We finally come to the application's heart in **drawLine**. This routine looks up the values stored in the global **textLine** structure, and attempts to display a line of text based on them.

As mentioned, it is faster for an application to do its own clipping when displaying text. This is only significant if the text will in fact be clipped, and drawing off the screen without clipping is likely to crash the system (or at least GEM), so we ignore that advice and clip to the text window.

We then call **textStyle**, which calls GEM's attribute functions with the specified character and line attributes. This "environment" approach for defining display attributes may seem odd if you have used systems in which the attributes are specified in the display call. That approach is unusable in GEM, since there are so many different attributes. GEM's technique also lets you set the attributes once, then use them several times (as in displaying several lines of text with common attributes).

Pay close attention to the verbose comment about **vst_height**, in the **textStyle** procedure—the correct values for the default system font sizes are probably not what you expect them to be: size is specified in height above baseline, not total character cell height.

## JUSTIFIED TEXT

Displaying justified text under GEM is trivial—you simply call **v_justified** instead of **v_gtext**. It wasn't all that long ago that displaying or printing justified text was one of programmer's more ▶

START
THE ST QUARTERLY

*START* is a magazine with its programs on disk. Normally you will find the disk bound into the magazine and for sale on the newsstands at a combined price of $14.95.

But we know that some of you ST enthusiasts want to read *START* first, without paying $14.95, so we have provided a limited number of copies without disk for $4.00 each.

If this is your situation, you can complete your copy of this collector's issue of *START* by ordering the companion disk direct from us, for $10.95 plus $2.00 shipping and handling. See the handy order form.

cumbersome tasks. Of course, you still need to put reasonable amounts of text on each line, so that the spaces aren't too large or too small (line-breaking is a subject deserving its own article).

You will find the problems with GEM's justification feature (and with any display involving proportionally spaced characters) when you try to mix character styles in a line, or if your text is user-editable. Since you must specify X and Y coordinates for each text display operation, you need to determine where to start each "piece" of the text line (when mixing styles).

When using a monospace font and not justifying, this is easy: determine the character width in each style, and multiply by the number of characters displayed. To get the width of each sub-string, increment the X-coordinate by this amount each time.

The task is just a little harder with non-justified proportional type. GEM provides the **vqt_extent** call, which returns a box describing the exact size of a string in a given style. Use this to find the width of each sub-string, and increment as above (but see the Bug Box if your string is italic).

Justifying text makes things much tougher—where the substrings will go depends on how many spaces there are in each of them, as well as on the nominal string widths. Your best bet is to avoid the situation. If you can't, you have to pre-scan the string, counting spaces and characters-of-width, etc.

Much of GEM is based on the SmallTalk model for user interface. SmallTalk recommends a single procedure, "scanLine." With several functions—depending on a flag, it will display a string, measure the string's width (including the effects of justification on it), find the X-offset corresponding to a given index within the string, or return the index of the character closest to a specified X-offset.

## A GEMTEXT GLOSSARY

**Alignment:** *The relative positioning of text, compared to the coordinates specified in a* **vq_text** *or* **v_justified** *VDI call. The most commonly used vertical alignments are "baseline" (recommended if you plan to mix fonts, since the baselines of all text on a given line must be the same) and "top". You are familiar with top alignment from non-graphics text; it simplifies the issue of determining clipping and erasure boundaries. Horizontal alignment determines whether the text will be displayed to the right of, centered on, or to the left of, your specified position. Both horizontal and vertical alignment are set using the* **vst_alignment** *call (VDI opcode #39).*

**Alpha Text:** *A GEM term for the text normally used by a computer system (what non-GEM applications would display, the system font).*

**Baseline:** *An imaginary line on which the capital letters in a line of text rest. When mixing type fonts on a line, the rule is to use the same baseline for all of the fonts.*

**BitBlt:** *Bit Block Transfer, a term borrowed from SmallTalk, refers to the copying of a rectangular block of pixels (screen bit locations) from one place to another, possibly performing a transformation on the way. In GEM, characters are displayed by copying a bit image of the character from memory to the screen, a process made possible by the the speed of the 68000 chip.*

**Cell Width and Height:** *In GEM, the size of a rectangle of sufficient size to display a given character with an appropriate amount of white space on all sides. In a monospace font, all characters have identical cell width and height.*

**Font (or Typefont):** *A set of characters having the same general appearance, style, and size. The 8x8 screen font on the color ST monitor is an example; the italic version of this screen font is another "font."*

**GDOS:** *The "Graphics Device Operating System." GDOS is a portion of GEM which allows GEM to deal with certain device-dependent features, such as multiple "workstations" (screen and printer, for instance), and multiple type fonts. GDOS was not released with the initial RAM or ROM versions of TOS, but should be available shortly. It is an overlay to the main part of GEM, and so can be provided after the fact (i.e. GDOS would not have been part of the ROM code in any case). Until GDOS is released, many functions of GEM are unavailable (see the "Bug Box" sidebar).*

**Graphics Text:** *A GEM term for text displayed within GEM by plotting individual characters as graphics images (see BitBlt).*

**Justification:** *Adjusting character and/or word spacing in a line to make the string fit a specified width. The term is also used ("Left-Justified," "Center-Justified," etc.) to refer to horizontal alignment (see Alignment).*

**Leading:** *Amount of space between lines (distance from the baseline of one line to the baseline of the next). A good rule of thumb is to use ap-*

*proximately 20 percent extra lead with proportional space type; 10 Point type looks best with 12 Points total line leading. TEXTDEMO.PRG refers to it as "Line Height" to avoid confusion.*

**Monospace Typefont:** *A typefont in which all characters have the same width. All of the standard ST screen fonts are monospaced. Monospaced characters are easily manipulated from a computer program, since no special width calculations need be made.*

**Pitch:** *The normal way of measuring monospace fonts. Pitch is the number of characters which fit in an inch, horizontally (Elite type is 12 Pitch, or 12 characters per inch, while Pica type is 10 characters per inch). Note that GEM never deals with pitch—monospace character sizes are instead set by character height above the baseline, in pixels!*

**Point Size:** *The most common way of specifying the size of proportional space fonts. A point is 0.0138 inch. All characters in a font are nominally guaranteed to fit in an area with a height equal to the point size. Thus, consecutive lines will not overlap if the leading is at least equal to the point size. GEM allows programs to specify Point Size in full points (half-points would be better), and will display with the font most closely matching the specified point size.*

**Proportionally Spaced Typefont:** *A typefont in which different characters may have different widths (for instance, the letter "i" will probably be significantly narrower than the "W"). Nearly all books, magazines, and newspapers use proportionally spaced type—it uses much less (30 percent–50 percent) space, reads quickly and is visually more pleasing. GEM provides support for proportional fonts, but these features are incomplete on the ST at this time.*

**Raster vs. NDC Coordinates:** *GEM allows positions to be specified either in terms of absolute screen locations (Raster Coordinates) or relative locations in terms of an idealized screen (Normalized Device Coordinates, or NDC). Since typefonts are available only in certain pre-specified sizes, any program that needs to use text is well advised to stick with Raster Coordinates, which are significantly faster than Normalized Coordinates.*

**Style:** *A generic term for the attributes of a text character. These include the character's type family, and "special effects" such as bold, italics, underlining, etc. Modest use of special effects enhances text readability and appearance.*

**Typeface:** *Collection of all supported sizes of a given font, but staying with a single style.*

**Type Family:** *A collection of related typefaces. Popular type families include Helvetica and Times Roman. The entire collection of monospaced screen fonts for the various ST resolutions could also be said to comprise a "family" of type.*

Why put all that utility in a single routine? There are many ways to justify text or to display certain sequences—kerning, ligatures, and so on. (Editor's note: *for a better treatment of these advanced topics, see resources listed at end of article.*) The same algorithms must be used for any of the four operations mentioned, since inconsistent sizes or positions could otherwise result. Unfortunately, GEM does not provide such a scanning routine.

This means there is no straightforward way to accomplish mouse-oriented editing in justified or proportional text. You can try to second-guess GEM's justification algorithm (and hope that it will be consistent on other GEM systems to which you later port your application), and "scan" accordingly, or you will have to do your own justification. Once again, your best bet is to not allow direct editing of justified or proportional text—if you can survive without it.

## ITALIC CHARACTERS

Using italic characters under GEM also appears to be quite simple—initially. Simply call **vst_effects**, specifying the "Skewed" special effect, then do either a **vq_text** or a **v_justified** call to display the text. If your entire string is italic, this will work fine, as long as you leave a little extra room on the string's right.

The bad news comes when you mix italic and non-italic characters. It seems that skewed characters in GEM really *are* skewed—GEM displays them approximately half a character position to the right of where they should go! This means that, if you are using Replace Mode (recommended for smooth display), the next non-skewed character you display will erase the right edge of the last skewed character. (If you have 1ST Word, you will be able to demonstrate this phenomenon. Type "GEM" in italics, followed immediately by a non-italic space.) In addition, unless you take ▶

## THE BUG BOX

*Yes, GEM does have a few bugs. Here are several specifically related to text handling. START welcomes comments, additions, and—especially—fixes.*

*1 Italic characters are offset from their ideal "cell" location, so the right halves of characters are clipped when italic characters abut non-italic characters.*

*2 The* **vqt_extent** *VDI call (opcode 116) returns an incorrect value (much too large) for italic text.*

*3 GDOS is not included with TOS. Therefore, only the monospaced system fonts are available. (VDI opcode #119,* **vst_load_fonts**, *will crash the system. So will attempting to open a workstation other than the screen, such as a printer.) Application packages using GDOS are beginning to appear on the market, and Atari is evaluating a version of GDOS for release.*

*4 There is no "scan" function corresponding to justified graphics text display, making* **v_justified** *(opcode #11-10) useless for text editing applications (see article).*

*5 The* **v_justified** *call produces strange (and apparently unpredictable) results when both the word-spacing and character-spacing flags are False (no justification desired). This case should be the same as using* **v_gtext**, *but is not. Use* **v_gtext** *for non-justified text.*

*6 The rightmost pixel of bold characters (when using any of the system fonts) is clipped. This is not, strictly speaking, a bug—it is a "feature" which ensures that the width of characters in the system font remains constant even if bold characters are used.*

*7 Text display slows down significantly under some conditions. In particular, characters which do not fall on an even eight-pixel boundary take much longer to display than those which do, and text display becomes much slower if clipping is needed. Also, display of proportionally-spaced characters using version 1.2 of GEM on the IBM PC is incredibly slow; we can hope that the GDOS Atari releases will be faster (the "Blitter" chip can also be expected to improve all of the speed problems significantly).*

*8 GEM allows text to be rotated in tenths of a degree, but does not support fractional point sizes! (Rotations other than at 45 degree angles are rare, whereas certain half-point sizes, such as 5.5 Point type, are quite common.)*

special precautions (as 1ST Word does), you may end up with "mouse droppings" (extra displayed pixels) at the beginning of the italic string.

If this isn't enough, GEM also returns incorrect values if you do a **vqt_extent** (VDI #116) on an italic string. You have to get the width of a string of italic characters the hard way—call **vqt_width** (opcode 117) for each character, ignoring the left and right "alignment delta" values returned.

There is no such easy work-around for the position problem; how well you solve it depends on how much work you're willing to do. The simplest "solution" is not to mix italic and non-italic text at all. If you can get away with this, you probably don't even have to worry about the half-character positioning error—no one will notice.

## CONCLUSION

GEM has given us an excellent set of tools for displaying clear and "interesting" text. Although these tools contain a number of bugs and "blind spots," with a little care they can be used to greatly enhance the appearance of ST screen text. Using GEM for leverage, you can write text-oriented programs with much more sophistication, and in significantly less time, than on non-GEM systems. Use GEM's text features—your application and its users will benefit! ■

## REFERENCE:

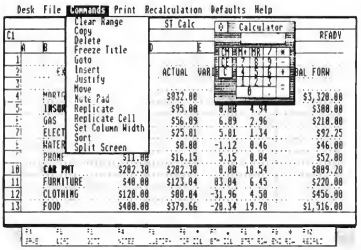
- *Phototypography: A Guide to Typesetting and Design*, by Allan Haley, Charles Scribner's Sons, New York, NY
- *Publication Design, Second Edition*, by Roy Paul Nelson, Wm. Brown Company Publishers, Dubuque, IA
- *Smalltalk-80: The Language and its Implementation*, by Adele Goldberg and David Robsen, Addison-Wesley Publishing Company, Menlo Park, CA

# ST CALC™

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

ONLY **$49.95**

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



```
C1
   A  B                    D         E
 1
 2       EX            ACTUAL    VARI          BAL FORW
 3
 4    MORTG           $832.00                  $3,320.00
 5    INSUR            $95.00    0.00   4.94     $380.00
 6    GAS              $56.09    6.09   2.96     $218.00
 7    ELECT            $25.01    5.01   1.34      $92.25
 8    WATER             $0.00   -1.12   0.46      $46.00
 9    PHONE    $11.00  $16.15    5.15   0.04      $52.00
10    CAR PMT $202.30 $202.30    0.00  10.54     $809.20
11    FURNITURE $40.00 $123.04  83.04   6.45     $220.00
12    CLOTHING $120.00  $88.04 -31.96   4.50     $456.00
13    FOOD    $400.00 $379.66  -20.34  19.70   $1,516.00
```

## FEATURES

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

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

## MOUSE CONTROL

Extensive use of the **GEM™** windows make **ST CALC™** a fast, extremely easy-to- use spreadsheet. With over 50 commands available from the mouse, the ease of use is unsurpassed. Imagine being able to move or copy an entire column of figures with a simple mouse control.

## CALCULATOR

**ST CALC™** includes an easy to use 10 key calculator that can be pulled down at anytime and operated either by mouse or keyboard. With the point of a mouse, the results of the calculation can then be transferred to the cell of your choice.

## NOTE PAD

**ST CALC™** lets you attach a personal note of up to 4 lines to any cell. The cell is then highlighted to remind you there is a note attached. For example, you could attach a note to the insurance cell of your personal finance spreadsheet reminding you that the cell applied only to car and home insurance. The note pad can be pulled down at any time.

## Help Calc ST™ Only 24.95

- 11 preprogrammed templates for use with **ST CALC™** or **VIP Professional™**
- **load-and-go** and these templates will take the work out of tedious spreadsheet setup.

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

VIP PROFESSIONAL is a Trademark of VIP TECHNOLOGIES.

## IF YOU DON'T HAVE OUR CATALOG . . . . . .YOU'RE MISSING OUT ! ! !

Over 1500 items for your Atari

Become a preferred customer and save $ $ $ $

ORDER YOURS TODAY!

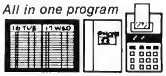
- SPECIALS
- CLOSE-OUTS
- BARGAINS
- REVIEWS
- WHAT'S NEW
- PLUS MUCH MORE!

**SEND $2.00** to get one year subscription to our catalog

## HELPMATE ST™ Only $39.95

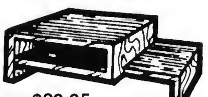
**HELPMATE ST™ INCLUDES:**
- 10 Key calculator
- Appointment calendar with alarm functions
- Telephone/name index

*All in one program*



The best part is that HelpMate ST stays "hidden" in memory until needed, and then can be called up for use, even while another program is running. The pull down menus can be used with most ST programs or by themselves.

## ST STAND



ONLY **$29.95**

*+ $5.00 Min. Shipping & Handling*

Custom made just for the ST, beautifully finished stand to hold your ST monitor, 2 disk drives, a modem, disk files, ETC...

## DELUXE DUST COVERS

*Deluxe Leather Grain*

**PROTECT YOUR INVESTMENT**

Custom filled, attractive leather brown color:

| | |
|---|---|
| 7.95 | • **KEY PAD**—ATARI CX85 • **PRINTER**—ATARI 1020 • **RECORDER**—ATARI 1010 |
| 8.95 | • **COMPUTERS**—400/800 600/800/1200XL 65/130XE • **DISK DRIVES**—ASTRA 1620 ATARI 1050 ST DRIVES ST HARD DRIVE INDUS GT PERCOM 88S1/88SPD RANA 1000 TRAK • **PRINTERS**—ATARI 1025/1027 AXIOM SLP/GLP OKIMATE 10 • **RECORDER**—ATARI 410 • **MODEM**—ATARI 1030 |
| 9.95 | • **COMPUTER**—ATARI 520/1040ST • **PRINTERS**—ATARI 825 AXIOM 550 CITOH/PROWRITER 8510 CPA 80/EP-150 EPSON MX/RX/LX/FX80 W/ & W/O TRACTOR FEED GEMINI/STAR SG 10/X PANASONIC 1091/1092 RITEMAN II/PLUS |
| 12.95 | • **MONITORS**—B/W/RGB MONITOR TEKNIKA MJ-10 • **PRINTERS**—EPSON FX 100/185 |

*Coming Soon!!*

## INVENTORY MASTER ST™ Only $179.95

**INVENTORY MASTER ST™** is a powerful, Inventory control and Report generation program. It will do more than just keep track business inventory, such as: detailed report generation, fast and easy data retrieval, versatile data entry, takes the work out of decision making, plus much more.

**COMPUTER PALACE**

MasterCard VISA AMERICAN EXPRESS

OPEN M-F, 9-6 Sat 10-4 (Pacific Time)
2160 W 11th Avenue Eugene, Oregon 97402

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

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

• 2 Day Air Shipping **AVAILABLE** •

# Voodoo Computing

## The Pragmatic Art of Defensive Programming

BY DAVID SMALL

*Being a rational contemplation of the vagaries surrounding computers, programming, and the precariousness of heavenly bodies and telephones in juxtaposition to floppy disks. Mr. Small expounds upon mayonnaise, cosmic rays and his personal, violent hatred of structured programming.*

To the outsider, computers are clean, utterly rational and cut-and-dried. They appear to be the same machines I was taught about at college: number crunchers governed by strict physical laws.

When I graduated and entered the real world, I learned it just ain't so. And over the past ten years, since my initiation into Hackerdom, I have formulated a valuable principle, which I will share with you.

But first, let us examine for a moment a typical computer, to get an idea how uncertain and fragile a medium it is.

### DIGITAL AMNESIA

We'll begin with memory—a frighteningly transient thing. Most personal computers today use "dynamic RAMs," a Faustian bargain to pack a lot of memory into a small area. The "dynamic" means that if you don't access that memory at least 100 times a second, it will fade away.

Now, you say, you've got to be kidding. One hundred times a second? That's pretty fast. But every computer working today is actively struggling just to keep its marbles. Turn on an Atari 8-bit computer; just sitting there, displaying "Ready," it is continually accessing and "refreshing" its memory.

Now let's talk about cosmic rays. Cosmic rays have a truly witty effect on dynamic RAMs. They make memory cells flip randomly.

Used to be, with the older RAMs, memory wasn't so sensitive. Not anymore. In Denver, a personal computer owner can reasonably expect one memory "soft hit," or failure, per week, during normal solar activity; this gets worse during solar flares.

Dynamic RAM chips are also the most skittish and easily zapped chips known. They have a nasty habit of "ringing" and "undershooting," which means they generate negative voltages which they feed to other chips, destroying all the chips involved. Also, any brief, transient power "spike," (like your next door neighbor's washing machine turning on), can flip a few bits.

Count on this: For every system crash caused by RAM failure, there are ten unseen events where RAM bits get flipped. They just happen to get flipped somewhere where it doesn't matter at the moment.

Think of dynamic RAM as 65,536 dominoes stacked on top of each other. One small push, and down they go.

And then, there are diskettes.

## *TRUST NOTHING*

Data is stored on diskettes with tiny, tiny, tiny magnetic fields, which are both remarkably durable and very sensitive. For instance, my daughter Jennifer once smeared mayonnaise across a 3 1/2-inch diskette (opening the little sliding door to do it, too). The diskette survived. However, I once had a phone ring next to a diskette; the diskette was completely erased. (Now I treat the phone as if it had the plague and keep it across the room.)

And how about keyboards? When you press a key, you bring two contacts together. What you're not told is that those contacts bounce against each other before coming to a rest, something like dropping a bouncy rubber ball. Whenever the computer looks for a key, it must deliberately wait for awhile before looking for another key, because the first key is still rattling from being pressed.

I could go on about rise times, about marginal clocks, about how even trying to view a crystal with an oscilloscope will cause the crystal to quit working. But the end point remains the same: Computers are a *very* "iffy" item.

You are working in a medium that is lucky just to keep running, that often fails through no fault of your own. Add to this the fact that human beings are not perfect and computers demand perfection in their programming, and you get the ultimate question of the Frustrated Programmer: Is this problem in my program or in the computer?

Something wonderful, childlike, and naive dies in a programmer the first time a computer genuinely screws up. Up to that point, you are *sure* the problem is yours, somehow, somewhere. Afterwards, you are never the same. It's like learning the truth about Santa.

So what philosophy does a sane person adopt?

Working with computers is like performing sorcery. Treat programming as you would treat casting spells. You'll go far.

Just as sorcerers drew pentagrams to control and bind the demons they summoned, I find I must adopt certain defensive measures to prevent computers from causing a lot of trouble. My philosophy is contained in ten maxims.

## *MAXIM NUMBER ONE:*

When you're having a bad day, *stop working*.

Every programmer has had days like this. Compilers mysteriously quit working. Disks stop loading. The hard disk flakes out. Nothing works right. You sit and think of alternate approaches, try them, and find your way blocked. The next day, of course, everything works right.

The thing to do here is to quit. Consider yourself bad luck that day. Anything you touch is going to fail. Remember those cosmic rays massaging your memory chips.

So lock your backup disks in a vault, and go do something useless, like creating a structured design for your next program.

## *MAXIM NUMBER TWO:*

Comment your code to death.

Dark Ages spellcasters wrote down every detail of their spellcasting in their spellbooks, or "grimoires," because they wanted to be able to retrace their steps in case of failure. Accept that your memory is not perfect. You will probably have to return to your code one day and fix it, or use it in another program. So comment it. It only takes a minute more, and it pays for itself many times over.

## *MAXIM NUMBER THREE:*

"Programming is an art best learned by apprenticing to a master." Or "Steal from the best." (Quote attributed to Russell Smith.)

Medieval sorcerers had no problem with borrowing each other's spells. Working out a spell, and keeping the demons from snatching you away to the inferno, was no easy task, right up there with debugging assembly language. Once you got something working right, it was most valuable. ▶

Good programmers, the true Hacker brethren, know this instinctively. This is why they are usually happy to share code. And looking at someone else's code is often the only way to pick up clever tricks.

Chris Crawford taught me all about ensuring that the decimal flag is cleared before doing interrupt processing; otherwise, arithmetic you do on the 6502 during interrupts can mysteriously go bad, unpredictably and untraceably.

John Harris found sections of his sound generation code (from "Frogger") being used in other games. Rather than being insulted, he understood the implied compliment: His code was worth copying.

So, let other people look at your code. Don't be embarrassed; asking for comments is the sincerest way to flatter someone.

Find out who your local hackers are and join their group. Hackers are basically people who believe in freedom of information exchange. If you become part of the loose, informal hacker group around you, you will have access to code, tools, and amazingly creative people.

### *MAXIM NUMBER FOUR:*

Use the best tools, and be willing to pay for them as necessary. Your time is valuable, and it is a pleasure to use good tools.

Use quality assemblers, compilers, interpreters. The time spent tracking down just *one* bug in your tools is worth the cost of using good tools to begin with. After all, you wouldn't expect sorcerers to skimp on the quality of newt eyeballs if they thought their spells might backfire.

Use *fast* tools. You'll be assembling, compiling and interpreting many times on any project. Even saving a few seconds per assembly is worth it. This also makes hard disks and RAMdisks worthwhile. The time you save is your own; nothing is more stultifying to the senses than watching a computer do something for the eightieth time.

Fast tools include a fast printer. You need to make lots of printouts as part of the game. If you stick with something that prints slowly, you'll be waiting a long time before you can make progress.

On my most recent project, making an Atari ST think it was a Macintosh, the program ended up being 7000 lines long (which ought to tell you something about how loosely my head is screwed on; it's all assembly language, too). Anyway, the printout is 120 pages long and takes awhile to print. Not a good job for a slow printer.

The time you save pays for itself. Your work gets done faster, which in the case of a freelancer, means you are available for other work.

### *MAXIM NUMBER FIVE:*

Keep a copy of everything you do—disks and printouts. Put it somewhere, file it away, but *keep it*. You will always come back to it.

Corollary: Reformat a disk you will "never use again" and you'll need it tomorrow.

Among the very few bright things I did at college was to keep most of the programs I wrote, just one extra printout. It only took a minute back then. And now I have a library of debugged routines and software.

This is especially important to freelancers. You haven't time to redevelop the wheel, and it is impossible to remember everything about a project you did awhile back. But if you have a printout of, for instance, how to set up a player-missile table, and better yet, if you can import that code off an old diskette directly into the editor, you've saved a lot of time.

Remember the John Harris syndrome: He kept all his floppy diskettes, including the master to the superb "Frogger," in one notebook. Once, he went to a computer show and demonstrated that disk. Someone stole the notebook and he lost all his disks and an incredible amount of development time. This is the real story of why a pirated Frogger showed up a year before the real Frogger; John was busy rewriting all his old code.

### *MAXIM NUMBER SIX:*

Backup your backups. Keep three of everything.

Computers typically fail as you back up the master of your program. This takes out not only the master, but the backup, since both are in the computer at the same time.

The solution is a third diskette. Just write a copy of your current program out to it every so often, then take it out and file it.

Don't be hasty to use your backups. Whatever killed your master disk might take the backup, too. Make sure you've fixed the problem. Test first.

Recently, I had the charming experience of trashing an Atari ST hard disk. I closed the current directory, and opened up the backup disk to repair the hard disk. The backup's directory promptly and permanently disappeared, leaving me the fun job of retyping the whole program I had been working on.

### *MAXIM NUMBER SEVEN:*

Frame this; hang it over your desk: *Don't be clever.*

There are so many clever tricks to use in a computer, and so little reason to use them. Unless you *really* need the speed and code size of a clever hack, avoid it. There are always side effects.

Some sorcerers used Latin and took other steps to make their spells unintelligible. They used lots of incense (which reminds me of structured programming: It makes your eyes hurt, and confuses the whole ceremony, but does make you feel you have accomplished *something*).

Avoid Utterly Clever Constructs. The worst offender in this department is C, which I have to come to see as an abbreviation for "Clever." C is a wonderful language to write code to show off with; "Gee, Fred, I never thought of doing it *that* way."

Fortunately, it is possible to write legible C code, that actually can be understood when reread. (This is contrasted to APL, a language once accurately described as "Write Only".)

Anyway, odds are, in a month, you'll forget just how that witty piece of code works, so why go to all the bother?

## MAXIM NUMBER EIGHT:

If it works, don't fix it.

The computer doesn't care how it looks. So it's a kludge? An utter pile of trash? *So what?* If it works, leave it alone and do something else.

Skip the indenting of your code; don't sweat the trailing comments and "pretty printing." Don't be a neat-freak; neat-freaks accomplish little. Leave it and go do something else. You owe it to yourself.

Consider code development as you would consider crossing an unstable, dangerous bridge. Sure, maybe you have to do it once. But who wants to do it twice? Look at Chris Crawford's Eastern Front code. There are sections that are not clean, elegant, or structured. It works. It is art. Who cares what the code looks like?

## MAXIM NUMBER NINE:

Always give your code the maximum chance to work. Or: It'll always think of something you don't.

When summoning demons in the old days, you wrote detailed contracts specifying exact terms of employment and payment. You tried to think of every condition—because demons were notorious for wriggling through loopholes. Really smart sorcerers added a "catch all" to the contract to cover what they didn't think of.

Do the same. Write defensive code.

Using a variable? Initialize it *just before you use it*. Don't depend on its value staying the same from the start of the program; you might use it for something else, or, (horrors!) the memory location it occupies might get clobbered accidentally.

Don't end a loop just by checking for a value equalling another value. Check for greater than or equal, or less than or equal; you might miss the equaled value. Give your program a chance to recover gracefully.

## MAXIM NUMBER TEN:

Structured programming is useless in the real world.

You don't need to program in a structured way. Give yourself some credit. You're neither a moron nor a menace to society. Don't use a language that forces structure on you.

Corollary: If a GOTO statement is so terrible, how come every microprocessor has one? And uses it frequently?

Pascal, to quote Chris Crawford, is the fascism of computer languages. And Pascal is The Great Language of Structured Programming. You can only do things one way—and there's a lot of mindless quibbling you have to go through to get it done.

Pascal also produces notably inefficient code, for the simple reason that computers don't use structured code, people do. Putting it mildly, something is lost in the translation. Personal computers are not yet at the point where you can afford to throw away a great deal of processor time letting the computer trundle through your "structure."

It is fine to draw up a general idea of what you're going to do, but carry it too far, and it's like working inside a straitjacket. You *cannot* think of everything at once, and that includes the structure of your program. Try to lay everything out in the beginning and you will throw out all the perfectly good ideas which occur during the coding process.

So use an anarchist's language such as C and obey just enough rules to get by. Don't take any extreme position, be it writing a thousand lines of uncommented assembler or pre-structuring your program. Remember, there are enough things working against you that you must retain maximum flexibility to get things done.

## CONCLUSION

Thus ends my Pragmatic Programmer's Philosophy. But I have more ideas. Many more. Should I be given the opportunity to speak again, I could tell you about Avoiding Burnout: The Career Programmer's Plight, or Fooling the Boss with Massive Printout. If the START editors will only let me.

(Editor's note: Perspectives is *a forum for commentary on the programmer's art by its diverse practitioners.* START *welcomes reader response to the ideas and opinions expressed in this department.*) ■

# AL and C Routines . . .

## Fast Memory Manipulations and More

BY DAN MATEJKA

*These two AL routines, plus a full, C "Help" text window module will speed your programs and make them more professional. Both AL routines—very fast implementations of a standard memory move and memory initialization—are accessed as functions from your main C program. The C text window module provides a standardized method of displaying text to screen or printer with auto-word-wrap. All programs may be found on the START disk within the folder labelled ROUTINES.*

When Stanley Crane and I were writing DB Master One, the database bundled with the ST last Christmas, we found frequent need for wholesale memory moves and initializations. DB Master One is a RAM-based system which loads the entire file from disk once each session, does its work all in memory, then writes it all back to disk. Necessary—but boring—operations like memory moves should be completed as quickly as possible. Straight C wasn't speedy enough, so we wrote the code ourselves to handle these situations.

In developing Disk Doctor for Antic publishing, I needed a friendly method of displaying "help" information on the screen. Since this would be a useful—and reusable—routine, I wrote a stand-alone C module for the purpose.

This article describes two assembly language routines—accessed as linked C functions—which provide a fast block memory move and an equally fast memory initialization. Following this is an analysis and description of the C printout module, PRINTOUT, which reads standard text files from disk and displays them to either screen or printer, left-justified and with word wrap. PRINTOUT will also provide an example of practical usage of the two AL routines.

## *MEMORY ALTERATIONS, WHILE YOU WAIT*

The two memory operations we're going to talk about are **setmem()** and **movmem()**. They can be found as 68000 assembly language source code in the file MEMOPS.S, on your START disk.

**setmem()** initializes memory to a chosen byte and **movmem()** grabs memory and copies it to some other location. Our version of these two functions is five times faster than their Alcyon C counterparts. This can make all the difference in the world to the person sitting in front of the terminal watching, for example, a scrolling screen. Screen scrolling is most painlessly accomplished by grabbing memory corresponding to each line of interest and putting it somewhere else.

Both functions were designed to be called from a C host—specifically a C host compiled by Digital Research's C compiler (the Alcyon compiler included in the developer's toolkit from Atari). This compiler allows unrestricted use of registers a0-a2 and d0-d2 within any procedure. **movmem()** and **setmem()** use no other registers, so no registers are pushed onto the stack for later retrieval. This compiler also uses two-byte integers. Pascal programmers will note the parameters are not cleaned off the stack at procedure's end, either.

If you are using the Alcyon compiler, all you need do to use these procedures is assemble the included file and link it with its host. Other compilers will doubtless have different linking and register usage conventions.

The complete C protocol for using these procedures is:

```
movmem(source,destination,count)
     char *source,*destination;
     int count;

setmem(destination,count,value)
     char *destination;
     int count,value;
```

For **movmem()**, **count** bytes are moved from **source** to **destination**. For **setmem()**, **count** repetitions of **value** are placed at **destination**. Note that **value** is a byte, not a word, value. For example, given a declaration like:

```
char stuff[100];
```

the call **setmem(stuff,sizeof(stuff),'a')** sets every element of **stuff** to 'a's. The call **movmem(&stuff[10],&stuff[6],20)** takes elements 10 through 29 of **stuff** and copies them to elements 6 through 25.

More or less equivalent C source code looks like this:

```
/* initialize memory */
setmem(d,c,v)
   register char *d;
   register int c,v; {

   while (c--)
      *d++ = v;
} /* end setmem */

/* move memory around */
movmem(s,d,c)
   register char *s,*d;
   register int c; {

   if (s > d)
      while (c--)
         *d++ = *s++;

   else {
      s += c;
      d += c;
      while (c--)
         *--d; = *--s;

   }
} /* end movmem */
```

Note that bad things happen if you send a negative value for count.

**setmem()** is straightforward. Starting where it's told, it marches through memory, setting each consecutive location to the value it's told to use, until it's told to stop.

**movmem()** works similarly, but a twist is encountered if the source and destination areas overlap. For example, given the following call, in which we move from a low address to a high address:

```
movmem( &stuff[6], &stuff[10], 20 )
```

If you start with **stuff[6]**, by the time you get to **stuff[10]** it has already been polluted by **stuff[6]**. Effectively, you will have copied the first four bytes five times. ▶

## Routines . . .

To avoid this problem, when working with overlapping memory, you must manipulate the memory elements in reverse order of the move. In the above example, start with **stuff[25]** and finish with **stuff[6]**. **movmem()** is smart enough to handle this for you, so you do not have to think about it.

### *CLOCKING IT*

Some time spent with the actual machine instructions generated from the above C code quickly shows the speed advantage of assembly language. The heart of the **movmem()** procedure,

```
while (c--)
    *d++ = *s++;
```

is compiled by DRI's package into assembly code which looks like this:

```
* (line 7) - a5 is s, a4 is d, d7 is c

moveloop:  move.b    (a5)+,(a4)+    * *d++ = *s++
           move      d7,d0          * copy count
           sub.w     #1,d7          * decrement
                                    * count
           tst.w     d0             * finished?
           bne       moveloop       * nope
```

Because compilers are not as quick to grasp the big picture as people are, some extra steps are included. This fragment would be better written something like this:

```
           bra       movecheck      * check for
                                    * zero count
moveloop:  move.b    (a5)+,(a4)+    * *d++ = *s++
movecheck: dbf       d7,moveloop    * decrement
                                    * count, continue
```

or, when possible, by longwords instead of bytes, like this:

```
           bra       movecheck
moveloop:  move.l    (a5)+,(a4)+
movecheck: dbf       d7,moveloop
```

A longword (four bytes) is the largest piece of memory the ST's processor can handle at once. Thus, it is the ST's most efficient chunk size for doing things like moving memory around. The longword moves in the above code fragments will move their four-byte chunks nearly three times faster than their byte-by-byte siblings.

The speed difference becomes telling when very large chunks of memory must be moved around. Using the above code fragments, the Atari 520ST moves a chunk of memory the size of its own screen at 150, 96 and 33 milliseconds. Those figures are not a straight 8 MHz multiplication of the number of clock cycles the MC68000 takes to perform the steps listed. Overhead slows the ST down to an apparent speed of about 7.3 MHz.

### *AND NOT ONLY THAT*

Just a few more points before we plunge into our "printout" module.

Whenever possible, everything is done with longwords instead of the bytes the procedures are defined as using. "Whenever possible" means two conditions must be met: the move must be longer than four bytes, and it must involve even addresses. The MC68000 is incapable of doing longword memory I/O at an odd address. If the situation can't meet both conditions, or can't be rearranged to meet them, the moves must be done by the byte.

Thus, these procedures really do not represent the fastest way to do the job. They spend too much time setting themselves up and deciding the best way to go about their business. But they are completely general and work no matter what parameters they are given to work with—unless, of course, they are sent bogus addresses or negative counts or similarly impolite things. Once they do decide on the best way to go about their job, though, they do it as quickly as possible and are an excellent compromise between generality and performance. For an example of their usage, examine the code in PRINTOUT.C, which we talk about next.

### *CONFESSIONS OF A FILE PRINTER*

PRINTOUT.C, on your START disk, is the source code for a self-contained, (nearly) stand-alone C module that reads standard text files from a disk and displays them to either a printer or the screen. The resulting display will be left-justified and word-wrapped on the right margin. It is completely independent of text size and Atari ST screen configuration. On the printer, PRINTOUT paginates when the page is full. On the screen, it begins at the top of its own window and continues to the bottom. When that bottom is reached, the window is scrolled and printing continues smoothly. The patient human watching all of this can pause or stop the display at any time.

PRINTOUT is a slightly modified version of a module in Antic's Disk Doctor. All printing in Disk Doctor is routed through this unit, but the modified version here is slanted toward displaying help files. All that need be done to read and show a help file is call **showfile()**, sending it the name of the help file it should show.

Again, PRINTOUT was developed using DRI's C Compiler. If you use some other compiler, beware that the Alcyon creature has some idiosyncrasies. Paramount among these is that it does not exactly follow the C convention of interchangeable pointers and integers; integers being only two-byte entities.

PRINTOUT needs six things it does not define itself. These are: **int schandle**, the work screen handle returned by **v_ opnvwk()** at program initialization; **int cellwidth**, the cell width of the current text font and size; **int cellheight**, its cell height (see GEM's **vst_height()** for details on these); and **ptsin[]**, a temporary int array which must be defined for any GEM application, anyway, so why not use it?

The fifth and sixth items are initializations, not variables. You must be sure the fill color is set to the background color, and text alignment must be set to bottom.

To provide an example of hooking PRINTOUT to an application, there is a simple shell program called PRTSHELL.C on your START disk. PRTSHELL.C will open a GEM virtual workstation, take over the screen, arrange things the way PRINTOUT wants them, display a menu bar, then wait for menu messages. Two menus are included for your enjoyment. One is the standard Desk Menu, the other is a Help Menu.

**Picking up the screen at any arbitrary point on a line and moving it somewhere else presents a fairly ugly problem.**

The first item under the Help Menu is "Quit". The remaining two items are psuedo Help items titled "About Something" and "About Something Else". They both call **showfile()** from PRINTOUT and display text from one of two files on the START disk (labeled FILE.ONE and FILE.TWO).

### *PRINTOUT FROM INSIDE OUT*

With a handy listing in front of you, notice that **gemdefs.h** is included for its window manager #define's, and **osbind.h** for its library bindings. Note that **PAGELENGTH** and **PAGEWIDTH** are also #define's. They can just as easily be variables carried over from the main unit. They are used only when printing, not when displaying to the screen.

The first procedures of real interest are in the section headed "Printing Routines." The first of these is **scroll()**, which moves the entire display window up one line. It is the only abstruse algorithm in the entire module, and many allowances are made for it elsewhere. It is best explained by first examining the structure of the Atari ST's screen.

Complication number one is that three different screen configurations exist: low, medium and high-resolution. High-resolution screens, capable of only two colors, are mapped in memory very simply: bit by bit, a 0 in the screen memory means that pixel of the screen is background color. A 1 means it isn't background. At 640 pixels wide, each line of the screen is 640/8 or 80 bytes wide. Each consecutive line is stored consecutively in memory. It's all easy enough, until you consider color screens.

Each pixel of a medium-resolution screen is capable of being one of four colors. In low-resolution, that's 16 colors. Strangely enough, that means each pixel of a medium-resolution screen is described by two bits of screen memory, and four bits in low-resolution. In fact, the screen is described in groups of 16 pixels. Each group of sixteen pixels is described by a group of two or four words of memory in medium of low-resolution. Specifically, the nth pixel is described by the nth bit of each word. In pictures, a memory map for a medium-resolution screen with pixels a through z looks like this:

```
---word one--- || ---word two--- || ---word three . . .
abcdefghijklmnopabcdefghijklmnopqrstuvwxyz . . .
```

Now, back to **scroll()**. Clearly, picking up the screen at any arbitrary point on a line and moving it somewhere else presents a fairly ugly problem. Picking it up at pixels which happen to fall on word boundaries is, however, fairly easy. For this reason, only a subset of the display window is actually scrolled. This is the part that begins at the leftmost part of the window that falls on a 16-pixel boundary and continues for the largest multiple of 16 pixels contained in the window. The scrolled portion of the window also omits a few lines on top and bottom for headers and messages and the like. Every screen line begins on a 16-pixel boundary, so vertical movement is not a problem. This is the origin of **scrollrect[]**, a subset of the work area of the display window (**windrect[]**). It was an intentional design consideration to use a subset of the window, rather than define the window itself to sit on a word boundary, because some border area was desired.

As mentioned before, a single line on a high-resolution screen is 80 bytes, or 40 words, wide. It turns out that both color screens are lines 80 words wide. Enter **linesize**.

To have the window show up anywhere on the screen, you need only have it show up somewhere else in the **openwind()** procedure. Since the scrolling area can then start nearly anywhere, **scroll()** needs to know how far from the beginning of ▶

## Routines . . .

the line (left edge of the screen) **scrollrect[]** is located. Enter **linestart**.

The window is also variable size, and a third thing **scroll()** needs to know is how wide **scrollrect[]** is, in bytes. Enter **linewidth**.

Note that **linesize** and **linestart** are defined as a number of words, while **linewidth** is a number of bytes. This is because the memory location of the screen is kept track of by an int pointer, and so offsets from it are done by the word. The movement of memory itself is accomplished by the earlier described **movmem()**, which works by the byte—so **linewidth** follows suit.

To calculate **linewidth** and **linestart**, the appropriate measurement in pixels is first divided by eight or sixteen, which converts to bytes and words, respectively. Each is then multiplied by the number of planes the current screen has: one for high-resolution, two for medium and four for low. **scroll()** accomplishes this with bit shifts instead of actual multiplication and division, because it is more convenient and ten times faster.

Once all this preliminary stuff is knocked out of the way, **scroll()** begins one line (**dy** pixels) from the top of the scroll area and, using sheer brute force (**movmem()**), grabs each line of the window and moves it up **dy** pixels.

### *EVERYTHING ELSE*

The remainder of PRINTOUT is best understood by watching it print a hypothetical file. This is accomplished by calling **showfile()** which opens the file and then keeps track of its length. This is necessary because **read()** in some earlier versions of the operating system choked when asked to read more data than remained in a file. **startprint()** initializes all local variables that need initialization, and in general prepares the print session.

Our other previously described AL routine, **setmem()**, may be found in **centerhdr()**. This is a minor function which centers a string on the page. **movmem()** is used to move the string to the right. **setmem()** fills the leftmost—or first part of the string—with blanks.

Our file is now read sequentially and given to the mercies of **printstr()**, piece by piece. **printstr()** can alter the value of **oktocontinue**. Finally, **endprint()** releases all the memory that **startprint()** snatched and closes the window it opened. Other things are cleaned up, and the show is over.

**printstr()** does the printing. It takes a null-terminated string of arbitrary length and heritage and plasters it all over the printer or screen, as requested by **startprint()**.

**printstr()** itself just searches for word boundaries and decides what will fit on each line of the display. All actual printing is done, one character at a time, by **printchar()**.

**printchar()** keeps track of the current column and line numbers, which are the globals **curcol** and **curline**. When **curline** gets too big it causes pagination. **printchar()** lets **printstr()** handle line breaks when **curcol** gets too big. If the printing destination chosen is the printer, **printchar()** calls the appropriate bios trap that tells the printer about it. If printing to the screen, however, it stores each character in a one-line buffer called **prnbuffer**, at the index **pbindex**. The buffer is only printed when a carriage return character is encountered. **scpixel** is an indicator of vertical position, like **curline**, but refers to the screen position instead of the line number. When **scpixel** gets too big, scrolling happens. **scpixel** is really used only for screen output and **curline** is really used only for printer output, since screen output isn't paginated, but it could be, and their uses aren't parallel; they are two separate variables.

As each newline is encountered, **printchar()** calls **checkeydown()**, which polls the keyboard. If **checkeydown()** finds a keystroke waiting patiently for attention, it eats it and pauses, stops or ignores it entirely depending on the key.

Pausing and stopping are accomplished through **statusline()**, which prints an appropriate message and waits for additional keystrokes to eat.

### *CRITIQUE AND OVERVIEW*

PRINTOUT's most damning problem is that it is what mouse-and-menu programmers are fond of calling "modal." When printing, all control of the computer is taken away from the no doubt unappreciative human who paid for the thing. This is not so bad, but when the printing pauses, and the mouse returns from wherever it was and the computer is once more paying attention to the human, PRINTOUT is off in its own little world, waiting for one thing and one thing only: a keystroke telling it to continue. There is no accessing any menu or other control device on the screen.

The advantage corresponding to all this rampant modalism is that PRINTOUT is a self-contained module which works almost by itself with no supervision from the main one. Making PRINTOUT nonmodal would of course necessitate a very intimate intertwining of its code with the main module, and it could hardly be published in its current incarnation as generic printing code. ■

# Of Diagnostics & Debugging

*Procedures for the 68000*

BY JIM DUNION

***A detailed look at the art of debugging the 68000, plus a dream list of what the best-dressed 68000 debugger may be wearing this season.***

Working in the department of Neurosurgery at Emery University and Grady Memorial Hospital in Atlanta showed me the similarity between the medical diagnostic process and the process of debugging a computer program. I was struck by the importance physicians place on information they see with their own eyes, and indeed, many of the recent advances in medical diagnosis come from instruments that provide images of our internal bodily processes.

Similarly, it makes sense to improve debugging tools by improving our ability to "see" what's going on inside our programs at any instant. One reason debugging is so difficult is that bugs usually involve some assumption that proves to be untrue. We intend to do one thing, but in reality we do something else. This mindset makes it hard to locate bugs, because we're so sure that it couldn't possibly be in a certain part of the code. One way of countering this in debugging is to put as much information as possible on the screen and let our visual system recognize when things don't look right. You've probably noticed how much easier it is to recognize the right answer on a multiple choice test than it is to generate the right answer in a fill-in-the-blank test. ▶

## BUG, BUG, WHERE'S THE BUG?

You're writing a program, and when you test it, it doesn't work. What now? At the risk of seeming simplistic, there are two things to do: Determine the bug's symptoms, and discover where the bug occurs.

As a general rule, fixing a software bug is usually easy; finding it is the problem. Some of the basic tools we use to find it are:

- A breakpoint mechanism: A way of setting a point in memory so that, if the program tries to execute that particular instruction, control transfers to the debugging system.
- A register display routine: To figure out what's going wrong in a program, the first step is to examine the processor registers.
- A memory display routine: Next in importance to the processor state is the current state of memory. We obviously need an easy way to examine exactly what's out in memory.
- A single-step mechanism: We want to make the processor execute a single instruction at a time, returning control to us after each instruction. Then we can "watch" precisely what is happening in the program, and see when it goes wrong.
- A trace mechanism: An elaboration of the single-step mechanism. Here we want to run the program in an interpretive mode, automatically making certain tests, recording machine states, and so on.

So far everything I've said applies more or less equally to all computer systems and languages. The closer to assembly language you work, the more you have to know about the processor itself. In the Atari 520 (or 1040) ST, that means the 68000. For many of us, the 68000 is the second or third microprocessor we have worked with, so what we really need to know is what aspects of 68000 assembly language programming are different from that of other processors. When you stop and think about it, the areas where any processor differs from others can be grouped into several main classes:

1. Register-Set Architecture.
2. Addressing Modes.
3. Instruction Set.
4. Special Event Handling.

The next sections discuss features in each of these areas that are likely to be unique to the 68000.

## REGISTER-SET ARCHITECTURE

The first thing we notice in looking at the 68000's register-set architecture is how many registers there are: seventeen 32-bit registers in addition to a 32-bit program counter and a 16-bit status register. The data registers D0-D7 are the true general purpose registers of the 68000. The next group, address registers A0-A6, is used mainly for address operations. There are some restrictions governing which instructions work with the address registers. Also, some operations on these registers don't affect status register bits. All in all, even though the address registers seem at first to be just like the data registers, they are, in fact, quite different.

The status register is divided into two bytes, the system byte and the user byte. These two registers contain a total of 10 bits of status information. Of these 10 bits, two are especially meaningful for debugging purposes, the S bit and the T bit. The S bit determines which of two possible states the 68000 is in, either Supervisor or User. In Supervisor mode, there are several privileged instructions that can be executed, but they will generate an error in user mode. The S bit also determines which of two possible stack pointers will be used.

The T bit, or Trace bit, is very important to debugging. When this bit is on, an exception (i.e., internal interrupt) is generated after each instruction. In effect, this is a built-in, single-step mechanism.

## ADDRESSING MODES

The 68000 has 14 available addressing modes, most of which can be used with any instruction. The main classes of addressing modes include:

- Register Direct Addressing.
- Absolute Data Addressing.
- Program Counter Relative Addressing.
- Register Indirect Addressing.
- Immediate Data Addressing.
- Implied Addressing.

Most of these modes are familiar to you from other systems, but there are a few things to notice. For instance, the Postincrement Register Indirect and the Predecrement Register Indirect modes are specifically included to provide for stack processing. Remember, however, that postincrement changes the address register after using the contents, as opposed to before with the preincrement mode. Also, the size of the increment is determined by the size of the instruction (e.g., byte instructions increment by 1, word instructions by 2, and long word by 4). Take care in using either of these modes with register A7 (the stack pointer). Since the system uses this register (and the implied stack) for address storage in subroutine linkage, the system always expects this register to be aligned on an even address. So even for byte-sized operations, the value of A7 will be adjusted by two.

One of the 68000 addressing capability's features is the ability to write position independent code, i.e., object code that can be moved around and still execute without change. The 68000 allows this by providing an addressing mode that is relative to the program counter. Several other processors allow relative branches, but in the 68000 even JUMPS and subroutine calls can be made relative to the program counter.

This is not to say that there aren't some quirks in 68000 addressing. The 68000 designers also put in some features to encourage reentrant code. For instance, you can read data relative to the program counter, but you can't alter data. Obviously, this is to protect the program from overwriting itself. Self-modifying code is considered outmoded these days. Data can be referenced, relative to a base address set up in a register, to encourage separation of program data and code. Though most instructions work with most addressing modes, there are exceptions.

## *INSTRUCTION SET*

The 68000 instruction set has only 56 instructions. But the ability for these instructions to work in many of the addressing modes and on several different data sizes makes them seem more numerous. For instance, the CLR command (which is a quick way of setting a location to 0) can be written several ways:

| | |
|---|---|
| CLR (A3) | Clears the word pointed to by A3 |
| CLR.B (A3) | Clears the byte pointed to by A3 |
| CLR.W (A3) | Clears the word pointed to by A3 |
| CLR.L (A3) | Clears the long word pointed to by A3 |

Care must be taken in sizing the data field on which you wish to operate, especially when the destination operand is a register. If you are operating on less than the full-sized register, the remaining portion is unaffected. In practical terms, this means the register's upper bits might not contain what you think. There is a little bit of a "gotcha" here, in that byte operations work on the bottom 8 bits of a register, but on the top 8 bits of a memory location.

Another feature, sign extend, comes from having instructions that work on variable size data. In the CMPA instruction (CoMPare Address, works only with word and long sizes), if you choose the word size, the sign (i.e., bit 15) is sign-extended to the full 32 bits and then the comparison is made. This can lead to some non-intuitive situations. For instance,

```
CMPA.W #$FFFF,A1
```

would set the Z condition code if A1 contained -1 ($FFFFFFFF), and would not set Z if A1 contained $FFFF. It is always a good idea to use long versions of instructions when placing addresses into address registers.

Several instructions are specifically included for high-level language support. Some of these can aid the debugging process. When taking a "snapshot" of the processor state during a program execution, there is always the question of which registers to save before the snap-shot subroutine executes. The MOVEM copies a specified set of registers to memory or back again. This instruction is very flexible, and able to save or restore any arbitrary group of registers. The machine language instruction actually contains a 16-bit register mask where each bit set to "1" indicates that the corresponding register should be saved or restored. Be careful when using this instruction to move 16-bit words from memory to address registers; they will be sign extended to 32 bits.

The DBcc (Decrement and Branch on condition) is a useful loop control command. Remember that the condition specified is the one that makes the program exit the loop, rather than stay in the loop. If the condition is not met, the counter will be decremented and tested for -1. If your loops are off by 1, this would be a good place to check.

An instruction is provided for multiprocess communication. In time-sharing situations, there is a classic problem known as the "deadly embrace," which can occur when two processes that are interrupt-driven both try to exert control at the same time. If a section of code first tries to read a status value, then takes control based upon the value's state, there could be trouble. What happens, for instance, if an interrupt occurs after the reading of the value, but before it is set to a new value? It depends. To help avoid such pitfalls, the TAS (Test and Set) instruction allows you to read, test and set a value all in one instruction. This allows the code to set a semaphore.

If you've come to the 68000 from a less powerful processor, there will be numerous pleasant surprises for you. Examples are: The bit-testing and setting instructions (BCHG, BCLR and BTST), the multiply and divide instructions, the exchange register instruction, some of the conditional branch instructions, and the data movement instructions.

The 68000 contains conditional branching instructions that test individual status register bits. There are also conditional branch instructions that on other processors require several instructions, including BLT, BGE, BLS, BLE and BGT. The most complicated of these are the "Branch if Less than or Equal" (BLE) and "Branch if Greater Than" (BGT). BLE will jump if the Z bit is set, or if the N bit is set and the V bit is not set.

The Bit TeST instruction, BTST, is a weird little instruction used for testing if a specified bit is set. The weird thing is that the instruction's action depends upon whether the destination is a memory location or a data register. The low order bit is specified as bit 0, and the high order bit as bit 7. Numbers larger than 7 are regarded as modulo 8. Memory is addressed, then, by bytes. If, however, a data register is the destination, then bit numbers range from 0 to 31, allowing all the register's ▶

bits to be tested. If the number is larger than 31 it is considered to be modulo 32.

Take care with the other bit-oriented instructions also. The 68000 uses memory-mapped I/O, so it is tempting to want to use an instruction like BSET to set a bit in a peripheral status register. What really happens in a BSET instruction however, is a read-alter-write sequence. Some peripherals are set up to become active whenever their address appears on the address bus. Thus, some subtle bugs can occur when you try to activate individual bits. A better approach is to use a MOVE instruction to set the register all at once.

## *SPECIAL EVENT HANDLING*

In the 68000, interrupts and other special events are known as exceptions. Exceptions are caused by external events known as interrupts; those caused by internal events are traps. The 68000 designers included several features to aid detection of program bugs. Specific hardware traps detect the following conditions:

- Word Access with an Odd Address.
- Illegal Instructions.
- Unimplemented Instructions.
- Illegal Memory Access (Bus Error).
- Divide by Zero.
- Overflow Condition Code (Separate Instruction TRAPV).
- Register Out of Bounds (CHK Instruction).
- Spurious Interrupt.

Also, programmers may use the 16 TRAP instructions to provide applications-oriented error detection and correction routines.

Finally, the CHecK register against bounds (CHK) instruction checks array bounds by verifying that a data register contains a valid subscript. A trap occurs if the register contents are negative or greater than a limit.

## *BUILDING A BETTER DEBUGGER*

Having examined 68000 features that can affect the debugging process, we return to the problem of using Atari ST features to aid in debugging. Color graphics and animation come to mind immediately. One example is to switch screens between the normal program display screen and a special debugger screen, which works because the ST has registers that determine where the video RAM (i.e., what the screen displays) is located. Visuals can be done in monochrome or color. I'm partial to color, and sacrifice resolution in order to use color to signify special events (e.g., a data value changing). The real challenge is to find ways to use the machine's graphic capability to display what the processor is doing. Indeed, numerous instruments are available to do this including: logic analyzers, signature analyzers, and performance analyzers.

Let's look at what we might call a program execution space display. Suppose we create a graphic display where a vertical line represents the memory space available to your system. Test the program in "trace" mode and it could display a colored pixel along the memory space line to indicate where the program counter is set. These pixels may even be tracked horizontally, creating a histogram of how many times a particular instruction has been executed. Just by watching such a display, we could get an intuitive "feel" for where the program is spending its time.

Similarly, a series of icons could be created that stand for some of the routines the program might execute. The trace program could highlight the icon of each routine as the routine is entered, thereby displaying the program's rough flow. This might not tell us the details of what went wrong, but it could provide some clues of where we should look more closely.

## *PROCESSOR AND MEMORY*

Once we have built a better display mechanism we still need ways of setting up a particular machine state and controlling the processor. Every debugger available has instructions for setting processor registers to specific values, and for depositing values to memory. However, some are easier than others. A powerful addition in this area is to allow symbolic references to variables from the debugging tool. Typically this means that the debugger must have access to the symbol table used by the assembler or linker or have a provision for defining symbols interactively.

A second way of improving our processor and state control is to allow conditional breakpoints. Simple, unconditional breakpoints are helpful. But ones with which you can, for example, say "Break" if an instruction tries to write to location TEST, are much better. Other types of conditional breakpoints are ones that break on specific instructions, program branches to specific ranges, and data accesses within specific ranges.

A speed control mechanism for single-step mode is also quite useful. Sometimes we just want to watch the overall flow of the program, while at other times we want to watch the details of specific instructions as they are executed.

We should also take advantage of the function keys and Mr. Mouse. These controls can trigger special kinds of debugger displays or processor control. The special function keys can be used for:

- Returning to the program under test.
- Single stepping the program.
- Switching screens between the program and debugger. ▶

# A DEBUGGING GLOSSARY

**ASCII Chart** - *A table showing each keycode's ASCII value. Some debuggers provide this as a convenience for low-level I/O debugging.*

**Assembler** - *A program that translates mnemonics meaningful to a programmer to executable object code.*

**In-line Assembler** - *A provision allowing a debugger user to deposit values in memory by using mnemonics without having to return to the assembler. Sometimes called an immediate assembler.*

**Backtracking** - *The ability to "go backwards" in time and undo the effects of preceding instructions. Requires a buffer to store the preceding instructions and some processor or memory state information.*

**Breakpoint** - *A point where execution of the tested program stops and control returns to the debugger.*

**Conditional Breakpoints** - *The ability to specify a set of conditions that determine whether a breakpoint will be triggered.*

**Sticky Breakpoints** - *Breakpoints that remain in place even after they have been triggered. They must be explicitly cleared.*

**Calculator** - *A provision to do some degree of arithmetic or expression evaluation directly in the debugger (e.g. calculating an effective address).*

**Decimal Arithmetic** - *Arithmetic capabilities in base 10.*

**Hex Arithmetic** - *Hexadecimal (base 16) capability.*

**Compare Capability** - *The ability to compare a given memory range to another memory range. Done properly, such a feature will show at what point the comparison fails, if it does.*

**Communication Ports** - *The ability to use an external communication port for communicating with the debugger. This is usually important for debugging applications that run on systems where there is limited ability to control the screen memory.*

**Debugger Isolation** - *Provides some degree of isolation between the debugger and the system under test. This is usually seen in systems with additional memory cards or In-Circuit Emulators.*

**Disassembler (Unassembler)** - *A program that converts between object code and assembly language mnemonics. Most debuggers have a provision for displaying a portion of memory in disassembly format.*

**Fill Memory** - *The ability to set a range of memory to a specified value or pattern.*

**Firewalling** - *A preventative debugging technique where modules are isolated from each other and interact only by affecting a group of variables.*

**Hardware Assisted Debugger** - *A debugging system that is provided with some additional hardware. Examples might be extra RAM, a hardware switch that generates an interrupt, or an In-Circuit Emulator system.*

**Help Screens** - *Display screens internal to the debugger that explain its functions.*

**In Circuit Emulators (ICEs)** - *A hardware assisted debugger that is in fact a complete external computer. Usually includes a cable and integrated-circuit plug-in device that replaces the target processor. In effect, the external computer system "emulates" the microprocessor under test.*

**Interrupts** - *Various conditions, or exceptions (as they are called on the 68000), that cause a hardware interrupt to be generated. These cause jumps through a vector table to interrupt processing code.*

**Linkers** - *Programs that load object code modules and resolve external symbol references.*

**Logic Analyzers** - *An instrument that monitors the busses of microprocessor systems, as well as allowing probes of other locations inside the system. A display much like an oscilloscope is created, showing the logic state over time for the line or point being monitored.*

**Map Files** - *Intermediate files produced by some assemblers that detail local symbols, local routines, external symbols and routines that are referenced, etc.*

**Move Memory** - *The ability to move a block of memory from one location to another.*

**Non-Maskable Interrupt** - *(NMI) an interrupt that can't be ignored by the processor. In the 68000, this can be implemented by setting the interrupt priority to 7.*

**NMI Switch** - *A "breakout" type of switch that is wired to generate a Non-Maskable (Highest Priority) interrupt. This is usually intended to return control to* ▶

*the debugger after the program under test has bombed.*

**Overlays** - *Additional portions of executable code that are brought in and "overlay" the code currently in memory. This is one technique for creating programs that are bigger than the physical memory size.*

**Patching** - *The ability to make local temporary changes to object code for quick testing.*

**Code Insertion** - *The ability to patch a section of new code without affecting the existing code.*

**Performance Analyzer** - *A device or program that captures processor execution information. Typically used to determine where the processor is spending its time, how many times particular routines are called, or specific timing information about a code fragment.*

**Protected RAM** - *RAM that is provided with some debuggers where the debugging system can maintain information that is protected from the program under test.*

**Reduced Speed Execution** - *Ability to run the program under test in a reduced speed interpretive mode where the action of the processor is slowed down so the user can roughly follow what is happening.*

**Screen Toggle** - *Ability to switch back and forth between a user-program screen display and the debug display.*

**Search Capability** - *The ability to search through memory for a given value or pattern. Usually updates a display to show the next memory range where a match is found.*

**Signature Analyzers** - *A testing device that creates a visual display representing what the processor is doing. Particular programs turn out to have repeatable, easily recognizable patterns.*

**Single Step** - *Ability to cause the processor to execute a single instruction and then return control to the debugger.*

**Single Step Past Calls** - *Ability to place a breakpoint beyond a subroutine call so that the processor executes a subroutine and then returns to the debugger after returning from the subroutine.*

**Sleeping Debugging Instructions** - *Diagnostic instructions or routines that are normally inactive, and which "wake up" and execute when a predefined abnormal condition occurs.*

**Source Level Debuggers** - *Debuggers that have some provision for reading source level code files and correlating those with the object code currently being debugged. A display is usually provided that shows the source language statement that contributed the instructions currently being executed. More sophisticated source level debuggers allow for breakpoints to be set at the source level.*

**Snap Shots** - *A static representation of the processor and memory state at any instant. Gives the debugger user a picture of what's going on in the processor at that instant.*

**Symbols** - *A sequence of characters that stands for either a memory location or a data value. The ability to use symbols makes debugging much easier.*

**Public Symbols** - *Symbols that are defined in general function libraries and are available to all program users.*

**Symbolic Debugging** - *The ability of the debugger to refer to a symbol table to make disassembly formats closer to assembly or other source languages.*

**Trace** - *The ability to monitor the processor and/or memory state after each instruction is executed. This is used for single stepping, conditional breakpoints, creating log files, and backtracking.*

**Watchpoints** - *Conditional types of breakpoints where certain conditions are monitored, and if they are satisfied, then control returns to the debugger.*

**Windows** - *A portion of a screen display (usually rectangular) that can be set up to monitor particular memory ranges, symbols, or other types of data structures.*

**Wolf Fence Method** - *A debugging technique where a bug is located by successively fencing it in smaller and smaller areas of code.*

- Activating special data displays.
- Scrolling a symbol table display.
- Changing the representation of memory window displays.
- Saving the current state to a disk file.
- Activating trace mode.
- Turning on a buffer mechanism to store instructions that have been executed.

The mouse can quickly point to exactly which register to change, which symbol to monitor, the speed of execution, or where in memory we want to look. The combination of a high-quality graphic display and a pointing device that can quickly indicate any spot in that display is a powerful one indeed.

## *ONE STEP BEYOND*

All of the features discussed so far can be found on existing debuggers if you look hard enough (though no one debugger has all of them). So what's the next step? Use your imagination. Are there other ways to visually represent the processor's activity? You bet there are. And I haven't even mentioned the possibilities of adding sound. The 68000 is a very powerful processor and the Atari ST combines this processor with excellent graphic capability.

We've just begun to tap the possibilities of this combination in all areas of programming, including debugging. It still takes too long to go from an idea to a working program. Too much time is spent exorcising bugs. Don't you think it's time our programming tools were a little more sophisticated? It wouldn't surprise me a bit to see a whole new class of debugging programs emerge soon for the Atari ST. And not a minute too soon.

(Editor's note: *Jim Dunion is currently putting the finishing touches on STDDT, his debugger for the Atari ST. It will be interesting to see if Jim can fit all the features mentioned in this article into STDDT.*)

## *REFERENCE:*

- *M68000 16/32-bit Microprocessor Programmer's Reference Manual (fourth edition)*, by Motorola, Prentice-Hall, Englewood, NJ
- *Motorola Data Sheets*:
  68000—#AD1-814
  68008—#AD1-939
  68010—#AD1-942
- *68000 Assembly Language Programming*, by Kane, Hawkins, and Levanthal, Osborne/McGraw-Hill, Berkeley, CA
- *Programming the M68000*, by Tim King and Brian Knight, Addison-Wesley, Reading, MA
- *The Motorola MC68000 Microprocessor Family*, by Thomas L. Harmon and Barbara Lawson, Prentice-Hall, Englewood Cliffs, NJ
- *The 68000: Principles and Programming*, by Leo J. Scanlon, Howard W. Sams & Co., Indianapolis, IN ■

**Remember, START is not complete without its disk. To order, see handy order form.**

coast to coast
C O M P U T E R S

**WE SPECIALIZE IN ST SOFTWARE**

*JOIN OUR ST OWNER'S CLUB FOR FREQUENT SOFTWARE AND HARDWARE UPDATES.* **CALL NOW!**

**ATARI ST SOFTWARE**

| | | | |
|---|---|---|---|
| DEGAS | $ 25.00 | REGENT BASE | $59.95 |
| CHECKMINDER | 39.00 | REGENT SPELL | 35.95 |
| H & D BASE | 59.00 | REGENT WORD | 35.95 |
| HIPPO PIXEL | 27.95 | SILENT SERVICE | 23.95 |
| LATTICE 'C' | 109.95 | ST COPY (CCI) | 27.95 |
| M-DISK | 24.95 | THE PAWN | 29.95 |
| P/C INTERCOM | 69.95 | TYPESETTER ST | 24.95 |

**WE CARRY THE MOST EXTENSIVE AND COMPREHENDABLE LINE OF MIDI SOFTWARE, HARDWARE, AND ACCESSORRIES. CALL NOW FOR AVAILABILITY!**

**HARDWARE**

| | |
|---|---|
| CASIO CZ-101 MIDI SYNTHESIZER | $309.00 |
| CITIZEN MSP-10 DOT MATRIX PRINTER | 265.00 |
| HAYES 1200 BAUD MODEM | 375.00 |
| HIPPO EPROM BURNER | 109.00 |
| HIPPO ST SOUND DIGITIZER | 109.00 |
| HIPPO ST VIDEO DIGITIZER | 109.00 |
| QMI 1200 BAUD MODEM | 159.00 |

*We guarantee to beat any advertised price.* Call the Bozos, then call your ST EXPERTS!

**WE'RE FAST, WE'RE FRIENDLY**

**CALL OR WRITE TO BE PLACED ON MAILING LIST FOR NEW PRODUCT INFO AND MEMBERSHIP**
**P.O. Box 3025, Northridge, CA 91323**

VISA MasterCard

NAT'L 1(800) 423-3444
CALIF 1(800) 424-3444
LOCAL (818) 886-5486

YOUR ST Connection 'Try Us!'

# COMPUTER CREATIONS

## YOUR ATARI 520ST SOFTWARE SUPPORT CENTER

| | |
|---|---|
| **ACTIVISION** | |
| Mindshadow | 34 |
| Borrowed Time | 34 |
| Hecker | 30 |
| **ACADEMY** | |
| Typing Tutor | 24 |
| **ATARI** | |
| DB Master | 39 |
| Antic | Call for various programs available |
| **BATTERIES INCLUDED** | |
| Degas | 28 |
| Peperclip Elito | Call |
| Homepek | Call |
| **DRAGON GROUP** | |
| 4X Forth | 89 |
| **ELECTRIC SOFTWARE** | |
| GST Macro Assembler | Call |
| GST Linker | Call |
| GEM Screen Editor | Call |
| **ELECTRONIC ARTS** | |
| Merble Madness | Call |
| Financial Cookbook | 36 |
| **HABA** | |
| Business Letters | 29 |
| Wills | 29 |
| Hippo 'C' Compiler | 51 |
| Checkminder | 51 |
| HABA Writer | 51 |
| Phone Book | 43 |
| Habadisk (10 Meg) | 669 |
| Hebeview | 51 |
| Habe Calc & Graph | Call |
| Habe Mailroom | Call |
| Hebe Word | Call |
| Habamerge | Call |
| **HIPPOPOTAMUS SOFTWARE** | |
| Hippo Computer Almanac | 25 |
| Hippo Jokes & Quotes | 25 |
| Hippo ST Disk Utilities | 37 |
| Hippo ST Remdisk | 25 |
| Hippospell | 29 |
| Hipposimple | 37 |
| Hippoert I | 29 |
| Hippobackgammon | 29 |
| Hippoword | 29 |
| Hippoconcept | 89 |
| Hippopixel | 29 |
| Hippoclean | 23 |
| Hippo Fonts I | 29 |
| Hippo ST Sound Digitizer | 99 |
| Hippo Video Digitizer | 99 |
| Hippo Eprom Burner | 99 |

| | |
|---|---|
| **HOLMES & DUCKWORTH** | |
| Toolbox | 28 |
| Forth | 34 |
| H&D Base | 89 |
| **INFOCOM** | |
| Ballyhoo | 28 |
| Zork I | 28 |
| Zork II | 31 |
| Zork III | 31 |
| Cutthroats | 28 |
| Deedline | 34 |
| Enchenter | 28 |
| Hitchhiker's Guide | 22 |
| Seestelker | 28 |
| Sorcerer | 31 |
| Suspect | 31 |
| Witness | 28 |
| Wishbringer | 28 |
| Infidel | 31 |
| Mind Forever | 31 |
| Plenetfall | 28 |
| Sorceror | 31 |
| Spellbreaker | 34 |
| Stercross | 31 |
| Suspended | 34 |
| **MARK OF THE UNICORN** | |
| PC Intercom | 84 |
| Final Word | 99 |
| Hex | 28 |
| **METACOMCO** | |
| Macroessembler | 60 |
| Lettice C | 129 |
| Mete Pescal | 79 |
| **MICHTRON** | |
| Kissed | 28 |
| DO.S Shell | 28 |
| D.F.T. | 34 |
| The Animetor | 28 |
| Personel Money Mgr. | 34 |
| Utilities | 45 |
| M-Disk | 28 |
| Mudpies | 28 |
| Soft Spool | 28 |
| Flip Side | 28 |
| Calender | 22 |
| Mi-Term | 60 |
| Gold Runner | 28 |
| Time Bandit | 36 |
| BBS | 34 |
| Business Tools | Call |
| Cornerman | 28 |
| Easy Record | Cell |
| Major Motion | Call |
| Mi-Dupe | Call |
| Mighty Mail | Call |

| | |
|---|---|
| **MIGRAPH** | |
| Easy Drew | 105 |
| **MINDSCAPE** | |
| Brateccus | 34 |
| **MIRAGE** | |
| Express Word Proc. | 34 |
| **OASIS** | |
| Sundog | 28 |
| **OSS** | |
| Personel Diskit | Call |
| Personel Pescal | 50 |
| Personel Prolog | Call |
| **PHILON** | |
| Compiled Basic | 61 |
| Fortren | Call |
| Pascal | Call |
| **PRYORITY** | |
| Forbidden Ouest | 28 |
| **OMI** | |
| ST Telk | 14 |
| **QUICKVIEW SYSTEMS** | |
| Zoomrecks | 62 |
| **SIERRA** | |
| Kings Quest II | 34 |
| Ultime II | 41 |
| Winnie The Pooh | 19 |
| Bleck Cauldron | 29 |
| **SPINNAKER/TELLARIUM** | |
| Homework Helper Meth | 35 |
| Ferenheit 451 | 28 |
| Perry Mason | 28 |
| Treesure Island | 28 |
| **SST SYSTEMS** | |
| Chet | 17 |
| **SUBLOGIC** | |
| Flight Simulator | Call |
| Jet | Call |
| **TDI SOFTWARE** | |
| Module · 2/ST | 69 |
| Andre/ST | Call |
| UCSD Pascel | Call |
| UNISON WORLD | |
| Printmaster | 32 |
| Art Gallery | 19 |
| **VERSASOFT** | |
| DB Men | 66 |
| **XLENT** | |
| Typesetter | 28 |
| Rubber Stemp | Call |
| Megafont II+ | Call |
| The Music Box | Call |
| VIP TECHNOLOGIES | |
| VIP Professional | 89 |

Atari ST Basic Training
Atari ST Grephic & Sound
Atari ST Logo
Atari ST Peeks & Pokea
Atari ST for Beginnera

ST BOOKS

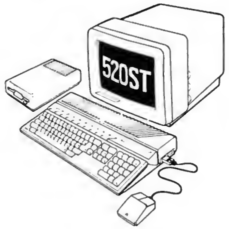
$ 18 ea.

Ateri ST GEM Reference
Ateri ST Internele
Ateri ST Mechine Language
Ateri ST Tips & Tricka
Atari ST from Basic to C

## FLOPPY DISKS AS LOW AS 59¢ EA.

| 5¼" | Black Generic Bulk | |
|---|---|---|
| Bulk | SS/DD | DS/DD |
| 20-69 | .69 ea. | .89 ea. |
| 70+ | .59 ea. | .79 ea. |

| 3½" | Verbatim | Sony | Sony |
|---|---|---|---|
| Bulk/Bx. (5) | SS/DD | SS/DD | DS/DD |
| 20-69/2-6 | 1.59 ea. | 11.90 bx. | 16.90 bx. |
| 70+/7+ | 1.49 ea. | 10.90 bx. | 15.90 bx. |



**ATARI 520ST**
**Personal Computer System**

## ATARI HARDWARE CALL FOR PRICES!!

**PACKAGE # 1**
Atari 520 STM Computer & SF 354 Disk Drive

**PACKAGE #2**
Atari 520 STM Computer, SF 314 Disk Drive and SC 124 Monochrome Monitor

**PACKAGE #3**
Atari 520 STM Computer, SF 354 Disk Drive and SC 124 Monochrome Monitor

**PACKAGE #4**
Atari 520 STM Computer, SF 354 Disk Drive and SC 1224 Color Monitor

**PACKAGE #5**
Atari 520 STM Computer, SF 314 Disk Drive and SC 1224 Color Monitor

**Call for Package Prices!**
*Let Us Design A System To Fit Your Needs!!*

ATARI PERIPHERAL PRODUCTS

| | |
|---|---|
| SF 354 SS/DD Disk Drive | **149** |
| SF 314 DS/DD Disk Drive | **209** |
| SM 124 Monochrome Monitor | **149** |
| SC 1224 Color Monitor | **329** |
| SHD 204 5¼" 20 MB Hard Disk | Call |
| TOS Operating System | 25 |
| Installed | 45 |
| One MEG Upgrade with purchase of System | 100 |

THIRD PARTY PRODUCTS

| | |
|---|---|
| Haba 5¼" Hard Disk | 669 |
| Shanner DS-2000 | 349 |
| 520 Station | 79 |

*To order call* **TOLL FREE**

# 1-800-824-7506

**ORDER LINE ONLY**

No extra charge for MasterCard or Visa

## COMPUTER CREATIONS, Inc.

P.O. BOX 493 - DAYTON, OHIO 45459

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

Order Lines Open 9:00 a.m. to 8:00 p.m. Mon.-Fri.; 10 e.m. to 4:00 p.m. Sat. (Eastern Stendard Time). Minimum $15 per order. C.O.D. (edd $3.00). Please specify computer system. Cell toll free number to verify prices and aveilebility ol product. Prices end aveilebility ere subject to change without notice. We ship C.O.D. to Continental U.S. eddresses only! Pleese include 4% shipping on all Hardware orders (min. $4.00). Software end accessories edd $3.00 shipping and handling in Continental U.S. Actuel freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico end APO. Ohio residenls add 6½% sales tax. Canadien orders add 5% shipping, (min. $5.00). All other foreign orders, pleese edd 15% shipping, (min. $10). For immediete delivery send cashier's check, money order or direct benk transfers. Personal end company checks ellow 3 weeks to cleer. School purchase orders welcome. Due to our low prices, all seles are final. NO CREDITS. All defective returns must heve a return authorization number. Pleese call (513) 435-8868 to obtain an RA# or your return will not be eccepted for replacement or repair. **FOR YOUR PROTECTION WE CHECK FOR CREDIT CARD FRAUD.** We do not bill until we ship.

# ST ASSEMBLERS

## A START COMPARISON

BY CHRISTOPHER F. CHABRIS

A year ago, the only way a programmer could do serious development work with the 520ST was with Atari's so-called "Developer's Kit" at $300.00. Now, there are several high-level language compilers and four macro assemblers. We'll compare those assemblers and the "standard" Digital Research AS68 program supplied in the developer's kit.

First, though, let's briefly review the function of an assembler in software development. An assembler is a program that translates a source text file composed of mnemonic commands into machine language for a particular microprocessor. On an 8-bit Atari computer, this is exactly what happens. With the ST, the process can be different. Usually the assembler produces something called "object code," an intermediate file consisting of machine language instructions that cannot be loaded and executed directly by the computer. Instead, it must first be processed by a program called a linker.

Why is this? Why not make the assembler simply produce executable files straightaway, like MAC/65 for the 8-bit machines? There are a couple reasons.

First, due to the MC68000's speed and rich instruction set, most commercial programs that run on a 68000-based computer are written in a high-level language like C or Pascal and compiled to run. Although the resulting program runs slower than its equivalent written originally in assembly language, the 68000's speed makes the difference virtually invisible to the user. Software houses prefer to use high-level languages because programs can be developed faster and are easier to port to other computers. Indeed, much of the early ST software was originally developed for other computers in C.

Second, because of the increasing complexity of applications software and the large size of the 68000 instructions themselves, executable program code is becoming larger. So, modular programming techniques are gaining popularity, and it is not uncommon to have several programmers working simultaneously on the same final product.

These reasons combine to produce the concept of separate compilation—dividing a large program into smaller modules and compiling them separately only when a change is made. We can create libraries of common procedures just once, instead of each time we compile the program that uses them. Compilers usually produce either an object file (as does Personal Pascal from O.S.S.) or an assembly language file (as does Alcyon C from the developer's kit). In the latter case, the code would have to be assembled into an object file. In either case, compiling a module results in an object code file output. This is where the linker comes in.

The linker can combine one or more object code files into an executable program under a given operating system. Since the object files are supposed to adhere to a standard format, the linker is even able to combine code from different languages into one program. This scheme makes it easy to write most of an application in a high-level language and code the high-performance sections in assembly language. A notable example is the Unix operating system, which is about 90 percent C and 10 percent assembly language.

For the assembly language programmer, this adds another ▶

step to the development cycle, but saves time in large projects. Atari supplies the standard DRI LINK68 program with the developer's kit. Since the linker component defines a development system's object code formats, assemblers and compilers that produce code acceptable to this linker can be used together in the same programming environment.

### *WHAT IS AVAILABLE?*

In addition to the Atari/DRI developer's kit tools, the following assembly language development products were available in April 1986:

- A-SEKA, Kuma Computers Ltd., version 020386 ($39.95)
- GST-ASM, GST Holdings Ltd., version 1.00 ($59.95)
- Macro Assembler, Metacomco, version 10.195 ($79.95)
- DevpacST, HiSoft, preliminary version 0.99 ($79.95)

These products were developed by British firms. Each includes a non-protected single-sided disk, and there the similarities end. Some include printed manuals, some use GEM features, and some have debugging facilities. I tested the four products on a 520ST with TOS in ROM, SC1224 color monitor, and two single-sided SF354 disk drives.

All four products, as well as DRI's AS68, use Motorola standard mnemonics and conventions. While they are for the most part correct, please do not get the impression that they are compatible with each other! I attempted to perform an informal benchmark to compare the five by assembling and linking an off-the-shelf 31K assembly language source file, (STerminal by Jeremy E. San, from the Antic public domain library). First I tried AS68 to set the performance standard for the others and everything worked fine (it took 3:19 minutes to assemble and a total of 4:21 minutes to create a runnable program file). However, none of the other programs completely accepted the source file, and all for different reasons! The moral is this: choose one assembler and stick with it. Otherwise, you will suffer the same headaches I did attempting to convert source code from one program's format to another. This shouldn't be a problem for developers who want to program from the ground up exclusively in assembly language. Needless to say, none of the four third-party assemblers can be directly substituted for AS68 when developing C programs with the developer's kit.

Interestingly, Metacomco packaged the GST linker with its Macro Assembler. The documentation covers both the GST and DRI linkers thoroughly, but only users with developer's kits will have access to the latter. GST-Link is incompatible with the DRI standard object file format, although Metacomco provides a utility to convert DRI format files to GST (but not vice versa). I'll look at the GST-Link program and the whole linker issue separately after the assemblers.

Atari provides the SID (Symbolic Interactive Debugger) utility of CP/M 68K fame with the developer's kit. I find it inadequate for all but the simplest tasks, compared even to the tools available on the 8-bit computers. Both A-SEKA and DevpacST include debuggers, while GST-ASM and Metacomco do not.

For program editing on the ST, many developers prefer to use old-fashioned, pre-GEM screen editors that offer speed and familiarity at the expense of ease of use. New programmers may prefer a GEM-based text editor that uses menus to issue commands. Atari has been supplying the Micro EMACS editor, a microcomputer version of a popular Unix tool.

### *A-SEKA*

A-SEKA is the least expensive of the assemblers for several good reasons. Its scant 34-page manual lacks an index and is provided only as a text file on the disk, so you must print it out from the GEM Desktop.

A-SEKA is by far the fastest assembler available for the ST; the manual claims a rate of 30,000 lines per minute, "even for large files," and this is believable. All source and object code is kept in memory until you explicitly write it out to the disk, a scheme that recalls MAC/65 on the 8-bit machines. In fact, the entire package is similar to such products. It has no screen editor, only a cumbersome, anachronistic line editing scheme.

The development process with A-SEKA looks like this: type in your source code line-by-line, correcting and revising with the Edit command; assemble it into memory; execute (and debug) the program using the Go command; repeat this process. Usually, code is assembled directly without linking. Some sort of built-in linker is provided, but the manual becomes foggy here. Apparently, the assembler can produce "linker code" (as opposed to executable code), which can later be combined with other such code into one executable program.

A-SEKA is outdated. While its fast response would be useful for learning 68000 assembly language, and perhaps for writing games, it currently cannot be used for serious development work. It demands revision, meaningful examples, and a completely new manual.

### *GST-ASM*

GST-ASM comes packaged in a green binder and slipcase with 170 pages of unindexed, detailed documentation. It does the best job of all four in providing a complete, self-contained development system for AL programmers. Unfortunately, it is missing a debugger and documentation of TOS/GEM routines. The latter is available in the Abacus book series, but there is no good standalone debugger available as of this writing.

GST has included GST-EDIT, an excellent windowing text editor reminiscent of its 1ST Word word processor, with its own 30-page manual. Here is an example of "overdocumentation," if such a thing is possible. If we could just give some of the extra to A-SEKA! GST-EDIT allows up to four simultaneously open files with a single cut/paste buffer between them. On-line help is available through a pull-down menu.

Although GST-EDIT is now my favorite text editor (replacing Micro EMACS), it has its problems. While scrolling up and down through a file is fast enough, jumping to the beginning or end is horrendously slow. More menu selections should have keyboard equivalents for faster operation. Ideally, the Help and Undo keys should also be supported. Most serious, however, is the omission of a Print command in the File menu. I had to write a program to print text files with pagination. This is a problem with all ST editors I have encountered.

The whole package is tied together by an "executive" shell program from which the components can be called, either separately or in batch operations (such as assembling a source file and chaining directly to the linker with default options). Everything makes good, clean use of GEM elements, and assembling and linking speed are comparable to the Atari developer's kit programs, if not better. The option of using the assembler module as a TOS program without GEM overhead is a nice touch, enabling the use of batch file processing. Initially, however, the shell will provide a convenient programming environment.

The assembler itself is complete and, like the rest of the package, well thought-out. Macros, conditional assembly, symbol table and cross-reference listings, and include files are all supported. Clearly this assembler was designed to support high-level languages—there is a COMMON directive that the manual connects to Fortran! The standard EQU, XREF, XDEF, DC, DCB, and DS pseudo-ops are available, as well as many others. The macro facility is extensive, including several "pseudo-functions" such as .LEN and .INSTR that can be used in string substitutions. GST thoughtfully includes a library of useful macros for conditionals, loops, stack handling, and subroutines. Assembler errors are all carefully documented, but TOS errors are not. (They are usually fatal anyway, the manual claims.)

GST has done an excellent job of making assembly language development accessible to ST users. Let's hope they're working on a debugger of similar quality!

## METACOMCO

Metacomco, an acclaimed system software developer in the 68000 world, has done a good job with its ST Macro Assembler package. It is the only package to provide an example source listing of a program that makes GEM calls, but like the others provides no real documentation.

ED is Metacomco's standard, simple, full-screen TOS-based editor, and is similar to Micro EMACS. It is adequate for working with assembly language source files, but takes some getting used to. It has some nice features, such as the ability to specify the text buffer's size and to continue executing a command (such as search and replace) until an error condition arises.

The assembler came closest of the four to functioning as a replacement for DRI's AS68. It has all the features of GST-ASM, plus local labels; but the macro facility is much weaker and no library of macros is provided. Strangely, two directives that do absolutely nothing are provided, yet there is no EVEN pseudo-op to make sure the following code begins at an even address. The manual documents the LINK68 linker, which is not supplied on the enclosed disk. However, the instructions for assembling the sample programs (provided in printed form only—aargh!) apply only to LINK68 and reference libraries not included on the disk. GST-LINK and its manual, identical to that in the GST-ASM package, are included. Metacomco's manual is typeset and carefully indexed, but does not accurately reflect the contents of the disk. This is a vanilla program—I suppose it lives up to its claim of being "full specification" but it is not as complete as GST-ASM. (Of course it beats A-SEKA!)

## HISOFT DEVPACST

HiSoft, a well-known European developer, was still working on their DevpacST in April 1986. I was furnished with a copy of version 0.99. (The final release is to be 1.00). Even so, the manual was bound and the packaging complete. DevpacST strikes a middle ground between the fast, loose, in-memory approach of A-SEKA and the formal edit-assemble-link paradigm of AS68, GST-ASM, and Metacomco's Macro Assembler. Its editor and debugger are both window-based, but it does not provide a linker.

EdST makes full use of GEM but provides only one text editing window. The amount of free memory and the current row and column are displayed at the window's top. On my 520ST with TOS in ROM, there were only about 60,000 free bytes—a small buffer when it is the only one you have to work with. For large programs the editor's restrictions could be severely limiting. EsST allows several methods of command entry including: drop-down menus, keyboard equivalents, and WordStar-like equivalents.

When you are ready to assemble your source file, you just choose the Assemble command (Alt-A) and a 2-pass assembly process occurs. The assembler supports the same standard features as the others, but the options for listing generation, mac-

▶

ros, and conditional assembly are even more limited than Metacomco's. There is not much to say here except that the assembler still had a couple of serious bugs in version 0.99. The assembler and editor are only available from the GenST shell, and no linker is employed.

The highlight of DevpacST is the MonST debugger. It uses four non-GEM windows: 68000 register value display, hexadecimal memory dump, disassembly (normally of instructions near the Program Counter value), and command entry/response. After loading MonST, you are prompted to enter the name of the program you want to work with, which is then loaded into memory. MonST preserves the program's screen and requires only about 12K of memory. MonST is automatically invoked whenever an exception occurs. You can set and clear breakpoints; single-step; view the program's screen; disassemble, search, examine, and modify memory; and modify the processor registers. This is a good first debugging tool.

**An assembler is not an intuitive program . . . you cannot usually "pick it up" as you go along.**

The short, 40-page manual is poorly written, even compared to A-SEKA's documentation. It needs expanding, rewritting, and proofreading. One nice touch is a list of known bugs in version 0.90. HiSoft promises various improvements in version 1.00 of DevpacST, including bug fixes, GEM AES and VDI constant and macro definitions, source to a complete application (this would be revolutionary!), external references, and the ability to produce linkable object files. I cannot recommend the current version, but 1.00 should compare favorably with GST-ASM and Metacomco if HiSoft delivers on its promises. Perhaps they will also index the manual . . .

### *GST-LINK*

GST-LINK, which is included with GST-ASM, Metacomco's Macro Assembler, and other language products from both companies, supports an object file format that is fully documented in the manual. It normally takes its input from a control file specifying which files are to be linked in what order to form the executable program. The control file language includes commands for setting the stack size, extracting modules manually and automatically from library files, and defining new labels (useful for correcting typographical differences between modules being linked). GST-LINK also produces executable programs directly without the annoyingly necessary RELMOD application used by LINK68.

GST-LINK is becoming popular in ST software development and is generally superior to LINK68, found in the developer's kit, but the incompatibility between the object file formats accepted by the two may cause problems.

For the programmer who is selecting one and only one language development environment, the question of linker compatibility is relatively unimportant. However, if you have any intention of combining languages, porting between development environments, or using standard libraries provided only as object files (for example, the LIBF Motorola fast floating-point library supplied by Atari in the developer's kit), the lack of a standard could complicate and retard ST development progress. I like GST-LINK (although LINK68 and the Personal Pascal linker have been adequate for all my needs so far), but a standard format is needed.

### *SUMMING IT UP*

I recommend GST-ASM as the most professional and complete package, and as the best value. Metacomco's Macro Assembler is comparable and desirable for use with its other language products. DevPacST in its current version is unacceptable, although its debugger is useful and version 1.00 has potential. A-SEKA is woefully inadequate unless you want to learn 68000 assembly language in an interactive environment. But if you already have Atari's developer's kit, you could safely stick with good old AS68 and LINK68 without falling behind the competition. I hope GST-EDIT is released separately for users who do not need the entire package.

Documentation is one of the most important aspects of a language environment. Before purchasing any of these packages, including the Atari developer's kit, carefully examine the documentation to make sure it provides you with enough information to effectively use the product. An assembler is not an intuitive program like a GEM-based word processor—you cannot usually "pick it up" as you go along. Unfortunately, none of the packages include sufficient examples or information on accessing system routines for a programmer to get started without other sources. C source code is much more plentiful than assembler in the ST world, so you should be prepared to do some work before you begin programming. The rewards of a successful GEM program in fast 68000 assembly language will be well worth the effort!

***LIST OF MANUFACTURERS***

Atari Corp., 1196 Borregas Avenue, P.O. Box 3427, Sunnyvale, CA 94088-3427, (408) 745-2000

Kuma Computers Ltd., 12 Horseshoe Park, Pangbourne, Berks RG8 7JW, United Kingdom, 07357 4335

GST Holdings Ltd., 91 Hight Street, Longstanton, Cambridge, England

Metacomco, 26 Bristol Square, Bristol BS2 8RZ, United Kingdom

HiSoft, 180 High Street North, Dunstable LU6 1AT, United Kingdom, 0582 696241

The Kuma, GST, and Metacomco assemblers are available in The Catalog in this issue.

***REFERENCE:***

- *Introducing 520ST Assembly Language* by Christopher Chabris, Antic magazine, December 1985
- *Atari ST Internals* by K. Gerits, L. Englisch, and R. Bruckmann, Abacus Software, Grand Rapids, MI
- *Atari ST GEM Programmer's Reference* by Norbert Szczepanowski and Bernd Gunther, Abacus Software, Grand Rapids, MI
- *Atari ST Machine Language* by B. Grohmann, P. Seidler, and H. Slibar, Abacus Software, Grand Rapids, MI
- *COMPUTE!s ST Programmer's Guide* by The Editors of COMPUTE!, COMPUTE! Books, Greensboro, NC
- *68000 Microprocessor Handbook, 2nd ed.*, by William Cramer and Gerry Kane, Osborne/McGraw-Hill, Berkeley, CA
- *68000, 60010, 68020 Primer* by Stan Kelly-Bootle and Bob Fowler, Howard W. Sams & Co., Indianapolis, IN
- *Mastering the 68000 Microprocessor* by Phillip R. Robinson, Tab Books, Ridge Summit, PA ■

START is a magazine with its programs on disk. Normally you will find the disk bound into the magazine and for sale on the newsstands at a combined price of $14.95.

But we know that some of you ST enthusiasts want to read START first, without paying $14.95, so we have provided a limited number of copies without disk for $4.00 each.

If this is your situation, you can complete your copy of this collector's issue of START by ordering the companion disk direct from us, for $10.95 plus $2.00 shipping and handling. See the handy order form.

**Software for the Atari 520ST®**

**H & D Base**

Relational Database Management Language for the Atari 520ST® Computer

***H&D Base*** is a Relational Database Management Language which allows novice and expert users alike to manipulate any type of data through the use of straight-forward, English-like commands.

For the novice, ***H&D Base*** is an interactive data storage and retrieval package suitable for maintaining phone directories, club rosters, genealogies and more.

For experienced users, it is a system development package containing its own programming language. A skilled programmer can use ***H&D Base*** to create a system for handling inventories, accounts payable and receivable, client lists, and other programs which would normally be written in more cumbersome languages.

Regardless of the level of user expertise, ***H&D Base*** is *the perfect tool* for data management.

**PROGRAM HIGHLIGHTS:**

- Comprehensive Manual: Introduction, Starting-up, Tutorial, Reference, Appendices & Index Sections plus Reference Card
- On-screen "Help" System
- Elementary Text Editor
- Powerful Report Generator with Saved Report Forms
- Sort on any Field to any Level
- Full Math Capability on any Field or Variable
- Disk Based System: Only Size Limitation is Disk Capacity
- Compatible with most "dBASEII"® Command Files
- Compatible with all "SDF" and "Delimited" Data Files
- Over 275 FORTH Commands (including "calls" to the GEM® Interface - *Atari "Developer's Kit" required*)
- Add Your Own Commands
- Create Auto-run Applications
- Includes Sample GEM Mailing List Program (written in ***H&D Base***)
- Run-time Developer's Package Available
- Source Level Maintenance
- Not Copy Protected

**SUGGESTED LIST PRICE:** $99.95

**ALSO AVAILABLE FOR THE ATARI 520ST® :**

***Toolbox - Volume One*** - $39.95 - Five Invaluable Utilities

***H&D Forth*** - $49.95 - FORTH Programming Language

For information: **(800) 641-1441**

In California, call: **(800) 641-1442**

Holmes & Duckworth

microeconomists

**MIRAGE CONCEPTS**

1341 Bulldog Lane, Suite B

Fresno, CA 93710

® "Atari 520ST" is a registered trademark of Atari Corp

® "dBASE II" is a registered trademark of Ashton-Tate, Inc.

® "GEM" is a registered trademark of Digital Research, Inc.

**POWER SYSTEMS** gives you the **POWER** with these products for the **ST** . . .

**POWEREDIT** is a powerful and easy-to-use text editor. Edit up to 20 files in from 1 to 8 windows. Move or copy data within a file or to a different file. Global change, search, auto save, and much, much more. Documentation on disk. 44.95

**POWERASSIST** detects and reports on abnormal termination of programs. Lists and/or prints register contents, memory contents, addresses, and a detailed error description. Errors may be corrected, and the program restarted or normally terminated. Documentation on disk. 29.95

**POWERVISION** provides NEOCHROME slide presentation development capability, with special effects, and variable display, i.e., corner to corner, top to bottom, side to side, etc. Data compression is performed, so 40 or more pictures may be stored on one disk. Documentation on disk. 24.95

**POWERHOUSE** is a home/business financial package which includes ammortization schedules, present and future values, annuities and sinking funds, budgeting, simple and compound interest, checking account tracking, investment tracking and Documentation on disk. 29.95

**POWERWRITER/MULTI-LINGUAL** is a word processing system with multi-lingual support. English, Russian, German, Hebrew, and more, including define your own. Documentation on disk. 49.95

*— 15% Discount for ordering 2 or more different products —*
Send check or M.O., or call collect for VISA/MC or COD orders.
— WA. State Res. add appl. sales tax —

**43328 S. E. 176th** **North Bend, WA 98045**
**(206) 888-2828**

***Now available for***
**ATARI ST, AMIGA, APPLE, IBM**
**ATARI 800/XLs/XEs and COMMODORE 64/128**

***THE MOST EFFECTIVE WAY TO LEARN TOUCH TYPING!***

**TYPING TUTOR + WORD INVADERS**

Two great programs in one package. Learn to use your keyboard quickly and properly. TYPING TUTOR starts with the 'home keys' and automatically evaluates your typing performance, introducing you to new keys in many gradual steps as your skills develop. WORD INVADERS puts real excitement into your touch typing practice while reinforcing proper typing techniques.

"This is the best typing tutor we have seen yet; ★★★★+"
**INFO-64**

"Best typing tutor I've seen—Better than Mastertype"
**Microcomputer Courseware Evaluation**

"WORD INVADERS is fantastic"
**Editors of Consumer Guide**

| | |
|---|---|
| ATARI ST<br>AMIGA<br>IBM PC, PCjr<br>Disk $34.95 | APPLE IIe, IIc<br>ATARI 800/XLs/XEs<br>COMMODORE 64/128<br>Disk $24.95 |

**ACADEMY SOFTWARE** VISA Shipping and handling $1.00 per order. CA residents add 6% tax
**P.O. Box 6277 San Rafael, CA 94903 (415) 499-0850**

## Mach1™ for the Atari ST

Mach1: multi-tasking Forth-83/GEM development system

With everything you need to develop stand-alone applications, including: integrated GEM editor, full GEM and DOS support, Motorola assembler, examples, and our 300 page manual.

Mach1 is interactive, so it allows you to experiment with the ST without going thru the compile-link-execute cycle. But when you do load in programs, look how we stack up:

| **Sieve** | **Compile** | **Link** | **Execute** |
|---|---|---|---|
| **Mach1** | **00.7** | **00.0** | **4.41** |
| **Megamax C** | **70** | **24** | **3.83** |
| **Hippo C** | **58.4** | **1:37** | **8.4** |

**C's w/o register variables**

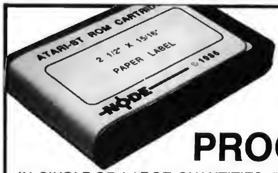
(That's three times the execution speed of other Forth's) Note the turnaround time. It simply takes less time to develop your programs or finished products with Mach1.

all for only
**$59.95**
plus $5 S/H
CA Res. add 6.5%
VISA/MC COD

Original Macintosh version (v1.2) available for $49.95

Palo Alto Shipping
PO Box 7430
Menlo Park, CA 94026
800/44-FORTH (Sales)
415/854-2749 (Dev. Support)

Available July 15, 1986



ATARI CARTRIDGE

**YOUR PROGRAM**

IN *SINGLE* OR LARGE QUANTITIES, PROVIDES:
*LOW COST, DISKLESS APPLICATIONS
*PROGRAM COPY PROTECTION
*DATA DURABILITY AND RELIABILITY

| | | 1 | 10 | 1000 | |
|---|---|---|---|---|---|
| TYPICAL | 64K | $69 | $49 | $29 | CALL FOR QUOTE AND |
| PRICING | 128K | $159 | $139 | $49 | PROGRAM INFO |

**ANSI VT100 CARTRIDGE**
FULL TERMINAL EMULATION WITH USER CONFIGURATION PLUG. DOES NOT REQUIRE DISK DRIVE.

**916-265-4668** VISA MasterCard

- 12 month warranty
- Quantity discounts
- Free UPS 2ND DAY AIR (Continental US) Hawaii add $3

**PAYMENT: VISA OR MASTERCARD, Checks held 14 days, C.O.D. add $2, California add appropriate tax.**

**NODE** COMPUTER SYSTEMS
408 BROAD ST.
NEVADA CITY, CA 95959

# UNIX FOR THE ST

## MICRO C-SHELL

BY RUSS WETMORE

Micro C-Shell
David Beckemeyer Development Tools
592 Jean St. #304
Oakland, CA 94610
(415) 658-5318
$49.95

The GEM environment is an excellent computer interface, especially for the novice user. However, sometimes an interface such as the GEM Desktop can be a hinderance. Programmers need more power than such a simple interface can support, and most types of utilities a developer needs are text oriented, not icon-based.

Micro C-Shell is a type of program commonly called a "CLI," for Command Line Interpreter and presents the user with a more traditional interface for disk files management. It isn't necessarily easy to use (though it is easy to learn) or pretty on the eyes, but it is fast and can perform several different tasks with one command line.

Micro C-Shell is based for the most part on a CLI written for Unix machines, called "csh," short for "C-Shell." It is called a shell because it acts like one, sitting over the normal operating system interface. It provides the user with a powerful, unified environment.

Csh was developed at the computer science department of the University of California at Berkeley. The Berkeley group has a long-standing reputation for its innovative additions to Unix's functionality. (Unix, by the way, is an operating system developed at Bell Labs, widely used on mini- and micro-computers.) Although part of csh's allure is it's ease of use in multi-user environments, its power makes it perfectly viable on single-user machines.

Beckemeyer's Micro C-Shell package also includes many useful utilities found on Unix systems. Non-programmers will find many of these utilities useful (such as the line/word/character counter and file string search facilities) but it is the serious developer who will appreciate the added power these programs provide.

Micro C-Shell doesn't use the GEM interface at all (except for an occasional error alert) and takes its input from the keyboard. You enter commands at the Micro C-Shell prompt. The commands may either be "built-in" to the shell, or be separate utilities on the disk (with a .PRG extender). Micro C-Shell is provided with the commands and utilities described in the accompanying sidebar.

Some of these commands are separate programs loaded in by the shell - you can add whatever commands you like to Micro C-Shell just by adding new programs. Those listed here are only the ones supplied in the currently sold package.

If all Micro C-Shell did was perform the commands listed above, it would be a great package. The more advanced features of the shell, however, are what make it especially valuable and powerful.

Typically, Unix systems have three assigned, default "devices" available to programs:

Standard Input
Standard Output
Standard Error

All three of these devices are normally set to the "console" device - the monitor screen for output and the keyboard for input. However, a program can *redirect* these to any other supported device. For example, you can optionally send output normally intended for the screen to the printer, to a ▶

disk file, or even to another program! (This is the subtle reason for the third device, the Standard Error device. This allows a program to send error or prompting messages to a device usually guaranteed to be the screen, even if the standard output is routed elsewhere. The standard error device can sometimes, however, be redirected just like the others - for example, if you'd like error messages sent to the printer instead.)

Unix uses the analogy of "pipes" in dealing with data. The "flow" of information can be redirected to another location, "piped" to another program, and even sent two different ways at once using a "tee".

Let's take an example. You want a directory listing sent to a disk file, instead of the screen, so that you can edit it into some documents. The standard directory command is:

```
% ls
```

(The "%" character is the standard Micro C-Shell prompt; you only would type the "ls" followed by a carriage return.) This prints a listing of all files and directories in the working directory to the standard output device, normally the monitor screen. In this case, you follow the command with a ">" character, followed by the file you'd like the standard output sent to, as in:

```
% ls >a:directry.txt
```

Now, instead of the directory being printed on the screen, it will be written to a file on drive A called DIRECTRY.TXT. You could, in turn, take the edited DIRECTRY.TXT file and print it to the printer using the "lpr" command.

When you send the standard output to another program, that program must be a "filter." This is a program which takes some input, manipulates it in some way, then outputs the result. Filters take their input either from a command line as a normal command, or from the standard input. Thus, when you send the standard output of one program to another, that data becomes the standard input for the next program. (Again, using the pipes analogy, this would resemble a filter in a pipe which removes dirt from a stream of water before passing the water on somewhere else.)

With "pipes" we can reduce this process to one command. If we were to type the following command:

```
% ls | lpr
```

the "|" character (ASCII 124) tells the shell that we want the standard output from the ls program sent (or "piped") as standard input to the lpr program.

Piping can get quite involved, and if used correctly can perform some powerful tasks. For example, the command sequence:

```
% grep -n register *.c | pr -h "found register:" | lpr
```

does the following:

1. finds each file in the working directory with a ".c" extender (typical of C source files).
2. in each file in turn, finds each line which contains the word "register" (including the line numbers where they were found).
3. formats the output into page size chunks with the phrase, "found register:" at the top of each page.
4. ships the whole kit'n'kaboodle to the printer.

A "tee" sends output two different directions at once, much

## MICRO C-SHELL COMMANDS

**alias** *Lets you define a sort of "macro" which replaces a given key word.*

**cat** *"Concatenates" (appends together) one or more disk files and prints them to the standard output device.*

**cc** *A short cut for programmers using the C language compiler sold in the developer's kit from Atari. It allows you to run a C source file through all of the various subprograms necessary to produce a linkable object file.*

**cd** *Changes to a new "working" directory or subdirectory. It's like opening a folder with the Desktop program.*

**chmod** *Allows you to change the "permissions" of files. You can make files "read only," "hidden" from directory searches, and designate "system" files.*

**cmp** *Compares two disk files and tells you if they differ, and if so, the line number and byte position of the first difference.*

**cp** *Makes a copy of one or more disk files.*

**date** *Allows you to change the real-time clock to a given date and time.*

**df** *Tells you how much free space you have on a given disk.*

**diff** *Compares two text files. If they differ, it can construct a script telling what lines must change in each file to make them equal. This script can be interpreted directly by the "sed" command, which will perform the necessary editing for you. This command is very useful for programmers - if you make a number of changes to a working source code file (and possibly forget everything you've changed) diff can tell you after the fact what those changes were.*

**echo** *Echoes the "tail" (the words following a* *command itself) expanding all shell variables and wildcard file designators.*

**fcp** *Does a track-by-track copy of an entire disk.*

**fmt** *Formats a disk.*

**gem** *Assuming there is enough RAM left, this command allows you to directly run a GEM-based program from the shell prompt. This keeps you from exiting to the Desktop to run a GEM-based program.*

**grep** *"grep" is shorthand for "Global Regular Expression Printer." It searches files for a string pattern. There are many different options, allowing you to search for a wide range of expressions. One common use for programmers is to find a label's occurance over a range of source code files.*

**head** *Prints a given number of lines at the "head" or start of one or more text files.*

**history** *A very powerful command, which tells you up to the last 16 commands that you have entered. These past commands can be referenced by number, or by name, and you can edit a past command. It lets you correct typing mistakes on long command lines, instead of typing the whole line again. It also lets you perform the same function over a range of files.*

**logout** *Exits to whatever program loaded Micro C-Shell (normally the Desktop program).*

**lpr** *This function is called a "filter" and prints one or more text files to the printer. You can specify page length and width, and make the printer pause between pages for single sheet entry.*

**ls** *List the contents of a directory. There are several options to this command, allowing for a wide range of directory formats.*

**mkdir** *Makes a new directory or sub-directory.*

**more** *Takes a text file and prints it out one screen at a time, pausing so that you can peruse the file at your leisure. You can also search for strings, backup to previous positions, and so on.*

**mv** *Moves (or renames) files. When you move a file from one directory to another, it deletes the original copy of the file so that only the destination copy exists. This is unlike the Desktop program, which only lets you copy a file to another drive or folder. When you "move" a file within the same directory, you are effectively renaming it.*

**pr** *Prints one or more files to the standard output device, with or without an optional header line.*

**pwd** *Tells you the present "working" directory (that is, the current default directory).*

**rm** *Deletes one or more files from the disk.*

**rmdir** *Deletes one or more directories from the disk, which must be emptied first.*

**sed** *The "Stream EDitor." This program copies a file to the standard output device, using a script file (see diff) which sed uses to make on the fly editing changes while it's printing. This is a short cut method of loading a file manually into a text editor and keying in the changes yourself. diff can create such scripts, and sed in turn rounds out an automated editing process.*

**set** *Allows you to set a shell variable. Some shell variables are used by the shell itself. You can create your own variables to reference in command lines.*

**tail** *Prints a given number of lines from the "tail" or end of one or more text files.*

**tee** *tee copies a file from the standard input device to both a named file or device and the standard output device.*

**wc** *Provides the number of words, lines, and/or characters in a text file.*

like its analogous "tee pipe fitting".

For example:

```
% ls -l | tee directry.txt
```

sends a long directory listing to both the screen and the disk file called DIRECTRY.TXT in the working directory. Adding one more pipe,

```
% ls -l | tee directry.txt | lpr
```

sends to both the disk file and the printer.

As mentioned above, you can create "aliases" for a string. This, in effect, extends the "language" of the shell. For example, let's say that you have your working assembler source files in the following subdirectory:

```
A:\ASSEMBLR\SOURCES\WORKING
```

In Desktop terminology, this would be the folder WORKING, inside the folder SOURCES, inside the folder ASSEMBLR, on drive A. If all of your source files have an ".ASM" extender, you could get a long directory—listing filenames, sizes and modification dates—using the following command:

```
% ls -l a:\assemblr\sources\working\*.asm
```

If you're going to be doing that a lot, though, it's a pain to type it every time. You could, instead, type:

```
% alias asmdir ls -l a:\assemblr\sources\working\*.asm
```

▶

which makes "asmdir" a synonym of that long command line. From then on, every time you enter:

```
% asmdir
```

at the shell prompt, its alias is substituted in its place and you get the directory you're looking for.

Micro C-Shell also maintains its own set of variables, which you can alter and add to. The following are defined by the system:

**$path** A collection of pathnames. Whenever you type in a command, if it needs to be loaded, the shell looks in every directory in the $path variable to find it. This helps keep you from having to know the location of every utility on every disk - the shell just keeps referring to $path until it finds what it's looking for or runs out of places to look.

**$prompt** The shell prompt. Normally a "%" character, you can change this to whatever you like.

**$include, $temp, $symb:** These three are used by the cc command for use with the Alcyon C compiler and the AS68 assembler, and describe the pathnames for C #include files, assembler temporary files, and where the AS68 symbol file can be found, respectively.

**$home** The "home" directory. If you jump around between directories a lot, but want to use one particularly as a "home base", you can easily set it by typing % cd $home

**$cwd** The current directory. This variable can't be changed, but can be referenced like any other.

Micro C-Shell always remembers the last 16 commands entered. The history command prints them out for you. You can refer to any of these commands, and execute them again, by typing "!" followed by the command's history number, as in:

```
% history
      1 ls -l
      2 cd tools
      3 history
% !1
```

executes the command "ls -l" again. You can also refer to previous commands by strings; that is:

```
% !ls
```

given the above history would do the same thing. (The shell looks backwards through the list until it finds one that begins with the string you've given, executing the first one it finds.) You can refer to the immediately preceding command by typing:

```
% !!
```

and by referring to a previous command by its offset as a negative value from the current history number. So, above, we're at history number 4 at the second prompt, and typing:

```
% !-3
```

would also execute the "ls -l" command.

Another powerful feature is the ability to edit previous commands. Let's say that you want a directory and type:

```
% ls -l a:\assemblr\sources\workjng\*.asm
```

Oops! The third subdirectory is supposed to be "working" instead of "workjng", and the shell tells you it can't find what you've asked. Instead of retyping the whole line, just to correct one error, you can type:

```
% ^workjng^working
```

which repeats the preceding word again.

There are other ways to edit previous commands, and you're not limited to just the most recently typed. For example:

```
% !10:gs/tmp/temp
```

executes the command with history number 10, substituting all occurances of "tmp" with "temp."

The Micro C-Shell disk comes with a new "standard I/O" library for Alcyon C programmers. This library fixes many of the problems with the libraries supplied by Atari, and is also faster and smaller.

The manual, though brief, is good. It is split into two parts: the first half is a short tutorial for newcomers, and the second half is a reference manual describing each of the commands in turn.

This package has its faults. It could definitely use more documentation. Some of the commands don't work as advertised in the manual, and indeed, some aren't documented at all.

In general, though, Micro C-Shell is worth every penny, especially if you're a developer. It has saved me hours of hassle, and I find more innovative ways to use it every day. I recommend it without hesitation.

## *REFERENCE:*

- *A User Guide to the Unix system* by R. Thomas and Jean Yates, OSBORNE/McGraw-Hill, Berkeley, CA.
- *Introducing the Unix system* by H. McGilton and Rachel Morgan, BYTE/McGraw-Hill, New York, NY.
- *The Unix Programming Environment* by B.W. Kernighan and Rob Pike, Prentice-Hall, Englewood Cliffs, NJ. ■

## C.C.S.

**Discount Prices** **Personalized Mail Order Service**

187 Crosby Ave., Kenmore, N.Y. 14217 • (716) 873-4349

Atari 520 ST Package
monochrome - **$659.95**, color - **$799.95**
1040 ST - **call**

**Disk Drive**
SF314 Double Side - **$219.95**
Atari 20MG Hard Drive - **$729.95**

**Printers**
Panasonic - 1080 - **$229.95**, 1091 - **$259.95**
Cannon P40 - **$259.95**

**Modems**
Avatex (300-1200 BD) - **$109.95**
Haba Modem (300-1200 BD) - **$189.95**
MPP-1200ST - **$199.95**
SIGNALMAN EXPRESS
(300-1200 BD) - **$279.95**

**Cables**
RS-232 Parallel - **$19.95**
RS-232-C Serial - **$14.95**

**SOFTWARE**

**Activision**
Borrowed Time - **$39.95**
Hacker - **$34.95**
Mindshadow - **$39.95**
Music Studio - **$49.95**

**Batteries Included**
D.E.G.A.S. (Graphic Art Sys.) - **$29.95**
Home Pak - **$39.95**
The Isgur Portfolio Sys. - **$199.95**

**Infocom**
Ballyhoo - **$32.95**
Spellbreaker - **$34.95**
Mind Forever Voyaging - **$32.95**

**Mark of the Unicorn**
P.C. Intercom Emulator for
Mainframe - **$99.95**
Final Word - **$99.95**
Hex - **$29.95**

**Migraph**
Easy Draw - **$119.95**

**Mirege Concepts**
H&D Tool Box - **$29.95**
H&D Forth - **$34.95**
H&D Base - **$79.95**

**O.S.S.**
Disk Editor - **$59.95**
Personal Pascal - **$59.95**
Personal Prolog - **$74.95**

**Polarware**
Crimson Crowns - **$29.95**
Transylvania - **$29.95**
OO-TOPOS - **$29.95**
The Coveted Mirror - **$29.95**

**Professional Software**
V.I.P. - **$129.95**

**Quick View Sys.**
Zoomracks - **$64.95**

**Regent Software**
Regent Word - **$39.95**
Regent Spell - **$39.95**
Regent Base - **$79.95**

**Sierra On Line**
King's Quest I or II - **$39.95**
Ultima II - **$44.95**
Black Cauldron - **$39.95**

**Spinmaker**
Winnie the Pooh - **$21.95**
Donald Duck - **$21.95**
Homework Helper Math or
Writing - **$39.95**

**Unison World**
Print Master - **$29.95**
Art Gallery I - **$24.95**

Write for catalog and list of printing ribbons. **Shipping**: Add $2 50 for software, $5 each for hardware. International orders pay actual charges. **Terms**: COD orders accepted. All orders will be shipped UPS or best way. No refunds or exchanges. Add 4% for VISA and MasterCharge.

## Atari ST Digitizer

**HippoVision B&W Video Digitizer . . . . . . $139.95**

Digitize video images from your TV, VCR or standard video camera into the ST and use NEOchrome or Degas to edit the picture. Have fun editing friends' faces, cartoons, movies, etc.

Hippopotamus Software, Inc.
985 University Ave. #12,
Los Gatos, CA 95030 408/395-3190

HIPPOPOTAMUS

## ATARI ST USERS!*
## ENTERTAINMENT JACKPOT
## 20 BIG PROGRAMS WRITTEN IN ST BASIC
## 72 PAGE MANUAL INCLUDED
## ALL FOR ONLY $34.95

**THE VISITOR**
Text adventure with graphics. Your smart but odd companion must rendezvous with its mother ship.

**BOMB SQUAD**
Text adventure with graphics. Find the terrorists' bombs in time.

**ADVENTURE CREATOR**
Write your own adventure games. Extensive how-to-do-it instructions. A "framework program" is provided so you can fill in the details of your own games. The program is analyzed in detail. A powerful, fast machine language parser routine is provided and explained. BASIC graphics subroutines included.

**THERAPIST**
"Talk to" your ST in natural English and it responds like a counselor. Similar to the famous ELIZA but "smarter".

**MANSION**
Text adventure. Find the second Mona Lisa.

**3-D TIC TAC TOE**
Challenging complexity

**THE WRITER**
Watch your ST write poetry and prose, personalize the vocabulary and characters for party entertainment.

**CASINO**
Lose your money at home.
includes: Roulette, Blackjack, Craps, Cards — Faro, Baccarat, Draw Poker, Slot Machine, Wheel of Fortune, Keno.

**OTHELLO**
Beat the computer.

**CHECKERS**
A classic.

**CRIBBAGE**
The popular card game.

**BACKGAMMON**
The ST is aggressive.

**MENTAL**
A great "psychic" illusion. The ST seems to possess amazing abilities.

**ANALYSIS**
Convincing "personality analyses" — just for fun. Mimics such things as "color analysis" machines and explains how they work.

**COLOR MONITOR REQUIRED**

**(Disk and Manual).**
**Only $34.95 ($43.95 Canadian); M.O., VISA, MASTERCARD (include expiry).**

*T.M. Atari Corp.

CHARGE CARD ORDERS ONLY
Ph. 800-628-2828 Ext. 635

**MARTIN CONSULTING**
**94 Macalester Bay**
**Winnipeg, Manitoba**
**Canada R3T 2X5**
**(204) 269-3234**

ATTN: PASCAL USERS

FOR ATARI 520ST

## MODULA-2
## the successor to Pascal

- FULL interface to GEM DOS, AES and VDI.
- 32-bit native code implementation with all standard modules.
- Full screen editor linked to compiler for rapid error detection.
- Smart Linker/Automated desktop.
- Installs on hard disk and RAMdisk.
- Supports transcendental functions and real numbers.
- CODE statement for assembly code.
- Modula-2 is NOT copy protected.

| RAMdisk Benchmarks | Compile | Link | Execute |
|---|---|---|---|
| Seive of Eratosthenes | 10 | 21 | 5.3 |
| Null Program | 7.1 | 5.5 | —— |

**Added features of Modula-2 not found in Pascal**

- CASE has an ELSE and may contain subranges
- Dynamic strings of any size
- Machine level interface
  Bit-wise operators
  Direct port and Memory access
  Absolute addressing
  interrupt structure
- Programs may be broken up into Modules for separate compilation
- Multi-tasking is supported
- Module version control
- Open array parameters (VAR r: ARRAY OF REALS:)
- Type transfer functions
- Definable scope of objects

Pascal and Modula-2 source code are nearly identical. Modula-2 should be thought of as an enhancement to Pascal (they were both designed by Professor Niklaus Wirth).

**Regular Version: $79.95** **Developer's Version: $149.95**

The developer's version supplies an extra diskette containing a symbol file decoder, link and load file disassemblers, a source file cross referencer, symbolic debugger, high level Windows Module, and the Resource Compiler.

TDI SOFTWARE, INC.
10410 Markison Road ■ Dallas, Texas 75238 ■ (214) 340-4942
Telex: 888442 Compuserve Number: 75026,1331

# RESOURCES

BY DEWITT ROBBELOTH
EXECUTIVE EDITOR

Books about the ST are essential for the serious programmer, and fortunately, considering the ST's newness, a number of them are available. Unfortunately, some were rushed to market and are not as complete or accurate as they should be. Nevertheless, several are respectable sources of technical information that will significantly shorten the time you spend getting up to speed on the ST.

Probably the best and most useful volume out so far is *Atari ST Internals,* $19.95, one of a series of ST books published in the U.S. by Abacus Software, P.O. Box 7219, Grand Rapids, MI 49510 (616/241-5510). *ST Internals* includes information about the 68000 processor and how it is used in the ST (including instruction set and addressing modes), the ST's custom chips, the disk controller, and many other I/O situations. It describes GEMDOS and has a complete listing of BIOS, the interface between GEMDOS and the hardware of the computer.

The ST found an early and avid following in Germany, and that's why *ST Internals,* like all of the Abacus ST books, was written in German (for Data Becker GmbH), then translated into English for the Abacus edition. The translation is very readable, but many typos have slipped in, raising some concern about the accuracy of technical information. But again, there is so much good information here you can't afford not to have it.

The next most important Abacus book is the *Atari GEM Programmer's Reference,* $19.95. As the title suggests, it is a reasonably complete guide to GEM, and the best thing you can get this side of the development documentation. Of course, if you want the full documentation, it costs $300 from Atari Corp. This includes 1,500 binder pages from Atari and Digital Research, Inc., designers of the GEM environment. Your $300 also buys six disks that contain the Alcyon C Compiler and Linker from DRI, a resource construction set, "DOODLE" (an application that explains and exercises GEM), plus tools and debuggers. The devoted programmer willing to pony up this much cash should order the ST Developer's Kit from Atari Corp., 1196 Borregas Avenue, Sunnyvale, CA 94088.

There are nine other Abacus titles for the ST, of varying interest and importance. *Atari ST Tricks and Tips,* $19.95, concentrates on utilities and cookbook effects, especially from ST Basic. *ST Graphics and Sound,* $19.95, and *ST Peeks and Pokes,* $16.95, also look very helpful. Several on the Abacus list do not merit our recommendation. *Presenting the Atari ST,* $16.95, is too superficial to be of value, especially if you have the better books of the series. *Atari ST for Beginners,* $16.95, is really for beginners. *Atari ST Logo,* $19.95, is a waste due to the relative inadequacy of ST Logo. There is an *ST Basic Training Guide,* $16.95, and an *ST Basic to C,* $19.95, which we have not seen.

One Abacus ST book, *Atari ST Machine Language,* $19.95, is an acceptable introduction to this important topic, and offers the reader instructive assembly listings for the ST. But any programmer serious about AL will want to have *68000 Assembly Language Programming,* $19.95, from Osborne/McGraw Hill. This book by Lance Leventhal, et al., is the bible for working with the 68000 chip and is widely available at technical book stores.

Another Osborne/McGraw Hill title is *The Atari ST User's Guide,* $15.95, by John Heilborn, who helped write Atari's ST documentation. His book is well illustrated for the beginner or others unfamiliar with the iconic desktop, but it focuses on Logo, which is not the language of choice for many ST users.

Two books by COMPUTE! Publications deserve mention. *The Elementary Atari ST,* $16.95, by William B. Sanders is an introduction to the machine, ST Basic, sound and graphics, files and I/O, Logo and Forth. Not nearly as detailed as *ST Internals,* it is still probably a better introduction than *Presenting the Atari ST. COMPUTE!'s ST Programmer's Guide,* $16.95, by the editors of COMPUTE! deals primarily with ST Basic and ST Logo, and for those languages it is a useful handbook.

Many other languages are available for the ST, and more are coming. Interest in the several implementations of C for the ST is growing, for which we recommend Prentice-Hall's book *The C Programming Language,* $24.95, by Kernighan and Ritchie, who developed the language. Other C books include *The C Primer,* $17.95, by Hancock and Krieger (a Byte Book), and *Learning to Program in C,* $25.00, by Thomas Plum, published by Plum Hall Inc., 1 Spruce Ave., Cardiff, NJ 08232.

Pascal is also quite popular, and in that area the *UCSD Pascal Handbook,* $18.95, is available from Prentice-Hall, Englewood Cliffs, NJ, or from Tecan Software Systems, 1410 39th St., Brooklyn, NY 11218. Tecan is also the source for a complete Pascal package for the ST, $79.95, which includes the language in software, plus the extensive and definitive p-System documentation supporting UCSD Pascal. ■

## ADVERTISERS INDEX

This list is provided as a courtesy to our advertisers and readers. *START* does not guarantee accuracy or comprehensiveness.

### ADVERTISING REPRESENTATIVES

JOHN TAGGART-ADVERTISING DIRECTOR
524 Second St.
San Francisco, CA 94107
(415) 957-0886

ROBERT JOHNS (Northwest)
524 Second St.
San Francisco, CA 94107
(415) 957-0886

THE PATTIS GROUP (Midwest)
LOUISE GRAUEL
JILL KROTICH
4761 W. Touhy Ave.
Lincolnwood, IL 60646
(312) 679-1100

GARLAND ASSOCIATES
PETER|HARDY (East Coast)
10 Industrial Park Rd.
Hingham, MA 02043
(617) 749-5852

CHARLES DURHAM & ASSOCIATES
CHARLES DURHAM (Southwest)
2082 SE Bristol St., Suite 216
Santa Ana, CA 92707
(714) 756-1984

**ELECTRONIC ONE***

**ATARI 520 ST**
**788⁰⁰**
- COLOR MONITOR
- 360 K DISK DRIVE
- BUILT IN TOS
- 512 K KEYBOARD
- 8 DIFFERENT SOFTWARE TITLES ST. SOFTWARE

| | |
|---|---|
| FLIGHT SIMULATOR | 29.99 |
| JET | 29.99 |
| MINEWHEEL | 24.99 |
| MUSIC STUDIO | 29.99 |
| STRIP POKER | 24.99 |
| BRIDGE 40 | 24.99 |
| FINANCIAL COOKBOOK | 39.99 |

**ST. HARDWARE**

| | |
|---|---|
| SF 354 DISK DRIVE | 159.99 |
| SF 314 DS DD | 219.99 |
| SHD 204 5¼" HARD DRIVE | 649.99 |
| SM124 12" HIGH RES MONITOR | 149.99 |
| SC1224 12" MED RES RGB MONITOR | 299.99 |

**ST. SOFTWARE**

| | |
|---|---|
| 1ST WORD | 32.99 |
| DB MASTER ONE | 32.99 |
| 2 KEY ACCOUNTING | 32.99 |
| NEOGRROME | 32.99 |
| HOME PLANETARIUM | 19.99 |
| JOUST | 19.99 |
| CP/M EMULATOR | 29.99 |
| BORROWED TIME | 29.99 |
| HACKER | 29.99 |
| DEGAS | 29.99 |
| SILENT SERVICE | 29.99 |
| UNIVERSE II | 47.99 |
| CRIMSON CROWN | 29.99 |
| FORBIDDEN QUEST | 29.99 |
| ZOOMRACKS | 49.99 |
| HOMEWORK HELPER | 19.99 |

HOW TO ORDER: CASHIER CHECK, MONEY ORDER, MASTERCARD* or VISA* (Add 4% for chorge cords)...NO PERSONAL CHECKS...NO C.O.D.'s...SHIPPED U.P.S. PRICES SUBJECT TO CHANGE.
SHIPPING: Prompt one doy shipping on in-stock merchondise. Ohio residents odd 5.5% soles tox. Add $3.00 an oll orders under $100.00...Add $5.00 on oll orders over $100.00.
INTERNATIONAL: Actuol freight chorge on oll orders outside the continentol United Stotes including A.P.O.
*CALL OR WRITE FOR FREE CATALOG*
CALL ELECTRONIC ONE **(614) 864-9995**

**Software Tools**
Unix™-like Utilities for the Atari ST

**MICRO C-Shell $49.95**
Unix-like C shell with many Unix-like Utilities included. Aliases, history, I/O redirection, pipes, shell variables, wildcards, C-like command scripts.

**MICRO Make $34.95**
Unix-like software maintenance utility.

***Multitasking Software!***

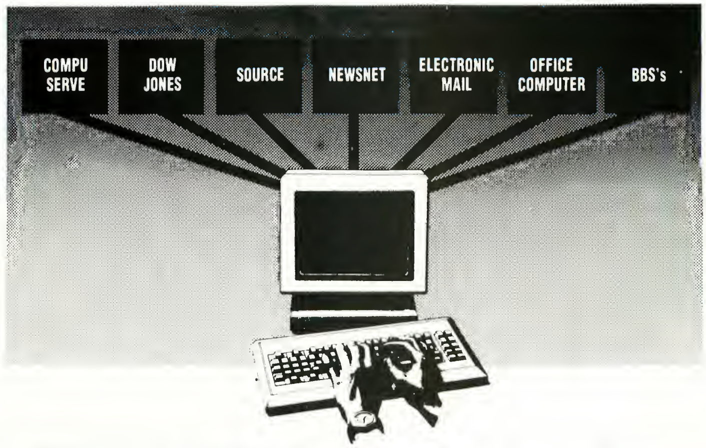
**MT C-Shell $129.95**
Fully Multitasking C-Shell with job control and builtin line printer spooler.

**MICRO RTX $69.95**
Real-time multitasking OS. TOS compatibility! Process control, communication, time-slicing.

**MT C-Shell $99.95 until August 31, 1986!**

***Beckemeyer Development Tools***
592 Jean Street #304
Oakland, CA 94610
Telephone: 415/658-5318

# INTELLICOM



## A COMPLETE Communications System

Now you can tap any of the countless telephone data based INFO sources, EXTRACT and FILE that data away for use later. Suddenly, your computer is the smartest terminal around because INTELLICOM emulates such terminals as:

- **Digital VT52/VT100**
- **TeleVideo 910** • **ADM3 A/5**
- **ADDS Viewpoint/25**
- **CompuServe Executive**
- **TeleVideo 925**

**Now you can transfer both binary and text** files, using simple ASCII, ASCII Capture, Xon/Xoff, XMODEM (both Checksum and CRC are supported), KERMIT or CompuServe A Protocols. Operate at any speed up to 9600 baud without ever being outdated as higher speed/lower priced modems are introduced.

Simple menu selections, with a built in **HELP** facility keep it easy forever.

**Autodial directories and auto login are supported along with simultaneous printing.**

**Additional features: Variable Buffer** Size reduces disk activity, while utilizing RAM capacity. Customizing utility program included to ease installation and use. **Persistence feature for autodialing and auto login.**

**Auto login script files** can be used to support almost any smart modem and/or automatically log into your favorite remote system. **User defined function keys. Color support. Remote access.**

Best of all, the entire versatile INTELLICOM package costs you only **$69.95**. FOR YOUR 520 ST!!

INTELLICOM is currently being used by ARMCO Steel, General Motors, Smithsonian Institute and countless other large corporations and government agencies worldwide. INTELLICOM is also available for use on IBM PC, Jr., AT and other true IBM compatibles as well as the Sanyo 550 series.

DON'T DELAY!
ORDER TODAY!
RETURN UPS SHIPMENT
GUARANTEED! HURRY!

INCLUDES FREE
SUBSCRIPTION TO
OFFICIAL AIRLINES
GUIDES
SAVE $50.00!

COMPUTER TOOLBOX inc.

MasterCard/VISA/COD orders accepted. Specify computer and DOS version when ordering. Connecticut residents add 7½% sales tax. **Add $5 shipping for all orders.**

1325 East Main Street, Waterbury, CT 06705
In Connecticut call: 597-0273

ORDER
1-800-321-DATA
TOLL FREE

# NEW for the ST . . . from MichTron

## CORNERMAN $49.95

What Sidekick did for the IBM, *Cornerman* does better for your Atari ST! This utility gives you a host of useful desk-top tools in one simple, neat package. With everything from a built in clock to a full function calculator, a cluttered desk is a thing of the past. And as a Desktop Accessory, *Cornerman* is a available nearly anywhere within GEM.

- REAL-TIME CLOCK in digital or analog form.
- NOTEPAD lets you write and store notes for handy future reference and reminders.
- ASCII TABLE shows the ST's symbols with their decimal & hex values. Great for programmers!
- PHONE BOOK stores all your important names & numbers, dials through your modem and even records elapsed calling time!
- 16 DIGIT CALCULATOR: works in binary, octal, decimal, and hex; 3 memory registers; math and logic functions; base conversions; "tape" printer output.
- DOS WINDOW for easy access to MichTron's Dos Shell program.
- 15-SQUARES GAME simply for entertainment.
- Doesn't interfere with other programs.

## MICHTRON UTILITIES $59.95

What would a utility be worth that could turn back time and end frustration? Before you find out the hard way that such a utility would be priceless, prepare yourself with *MichTron Utilities*. This utility searches for and retrieves deleted files and lost data. Change file contents, attributes, file and volume names, or any individual bytes on disk! Just type the new data or click on selector buttons. You can also format individual disk tracks and copy individual sectors to repair damaged disks. A new 10 sector format utility lets you add 80K of storage capacity to you disks.

## Personal Money Manager $49.95

This personal accountant will keep accurate totals for as many expenses, income sources and accounts as you need. Print checks and other kinds of reports for a true picture of your current financial status. You can form projected budgets for future expenses and income. Fast and powerful, the Money Manager uses the GEM system for simplicity.

## *UTILITIES:*

*THE ANIMATOR* ($39.95) - Animate Neo or Degas pictures for business presentations, or just for fun.
*BBS* ($49.95) - Complete Bulletin Board System
*BUSINESS TOOLS* ($49.95) - Over 200 attorney-prepared business forms, letters, and contracts.
*CALENDAR* ($29.95) - Desktop appointment calendar.
*D.F.T.* ($49.95) - Transfer files between the ST and IBM.
*DOS SHELL* ($39.95) - Mimics the MS-DOS command structure: "global" commands, batch files, and more.
*ECHO* ($39.95) - Uses X-10 modules for a wireless remote-control system for home or office
*Introduction To LOGO* ($49.95) - Easy tutorial lets you learn to program in Logo.
*KISSED* ($39.95) - Debugger features full-screen editing, miniassembler/disassembler, help function and more.
*M-DISK* ($39.95) - RAM-disk emulator gives you the equivalent power of an extra disk drive!
*MI-DUPE II* ($39.95) - Fast, easy file duplication.
*MI-TERM* ($49.95) - Advanced communication program lets your ST talk to virtually any other system.
*MIGHTY MAIL* ($49.95) - Complete mailing-list manager features multi-level sorts and conditional "flags" for specialized reports.
*SOFT-SPOOL* ($49.95) - Frees your ST to print & compute at the same time.

## *ARCADE GAMES:*

*GOLD RUNNER* ($39.95) - Infiltrate enemy gold mines. 63 screens test your skill and logic (color monitor).
*MAJOR MOTION* ($39.95) - Race down the highway, destroying enemy spies as you gain new weapons and defenses (color monitor).
*MISSION MOUSE* ($39.95) - Avoid the prowling cats as you climb to the next goal (monochrome monitor).
*TIME BANDIT* ($39.95) - Collect the Treasures of Time! Choose from a multitude of worlds: 13 unique arcade lands and 3 complete arcade/adventures (color monitor).

## *STRATEGY GAMES:*

*FLIP SIDE* ($39.95) - Play Reversi against live or computer foes.
*CARDS* ($39.95) - Blackjack, Cribbage, Klondike, Poker Squares, and Solitaire (medium or high res only).

*All reasonably priced, with more coming every day. Ask for our latest catalog!*

Dealer inquiries welcome • Visa and Mastercard accepted • Add $3.00 shipping and handling to each order.

**MichTron**

576 S. TELEGRAPH, PONTIAC, MI 48053
ORDERS AND INFORMATION (313) 334-5700

BATTERIES INCLUDED

Atari ST Software

Integral Solutions

**D·E·G·A·S**

DESIGN & ENTERTAINMENT GRAPHIC ARTS SYSTEM

*by Tom Hudson*

*The artistic standard for the ST! Beautiful graphics program for business and pleasure. All the key drawing/painting functions, text integration, and graphic design tools! Available: Now!*

**TIME LINK** *by Softechnics*

*Scheduling & Time-keeping tool for home and business. Your day, week, month, year at a glance. Many incredible uses! Available: Now!*

**THUNDER!** *by Mark Skapinker*

*Use this unique real time spelling checker desk accessory within* **any** *ST GEM application. 50,000 word real time spelling checker. Abbreviations function completes a word when you enter the first letters. Amazing speed. Available: 3rd Quarter 1986.*

**HomePak** *by Russ Wetmore*

*ST version of InfoWorld's Best Buy of the Year Award! 3 integrated programs on one disk. Telecommunications. Word-processor. Information manager. The easy answer to three key software needs. Available: 3rd Quarter 1986.*

**BTS THE SPREADSHEET**

*by Alan Porter, Martin Rezmer and Jason Loveman*

**Two in one!** *Sophisticated full featured spreadsheet program. All the key Math, Stats, Logical and Financial functions. 1000 x 1000 worksheet.* **Plus** *Desk accessory version on same disk! Available: 3rd Quarter 1986.*

**The Consultant** *by the B.I. Software Development Team*

*THE ultimate relational data base. Easy to learn. Unique new features. Power and sophistication you can use right away. For business or personal use. Available: 4th Quarter 1986.*

**THE ISGUR Portfolio SYSTEM** *by Lee Isgur and Mark Skapinker*

*Investment management program designed for private investors and professionals. On-line portfolio updating. Analytical functions for more profitable decisions. A PC Magazine editor's choice! Available: Now!*

**I·S TALK** *by Stephen Couchman*

*Full-scale telecommunications program. Easy to use and virtually error-proof. Includes 50,000 word spelling checker and three levels of macros. Available: Now!*

**PaperClip Elite** *by Dan Moore and Steve Ahlstrom*

*Next generation word-processor. All the high-productivity features plus a real-time spell checker, idea processing, integrated text/graphics, much more! Available: 3rd Quarter 1986.*

**D·E·G·A·S Elite**

DESIGN & ENTERTAINMENT GRAPHIC ARTS SYSTEM

*by Tom Hudson*

*Second-generation graphics program creates presentation-quality visuals. Full range of design/color functions. Multiple workscreens, new fonts, many other enhancements. The deluxe Degas Elite is totally compatible with all other Degas files! Integrate your Degas Elite pictures with PaperClip Elite text files. Available: 3rd Quarter 1986.*

**B/GRAPH™ Elite** *by Alan Page, Joe Chiazzese and Robert Wilson*

*Serious graphics/charting and statistics desk package. Pie charts, 2 and 3 dimensional bar charts, area graphs, much more. Change designs without re-entering data. Make beautiful presentations. Available: 4th Quarter 1986.*

**I·S TIME AND BILLING** *by Roy Miller*

*Office management program for professionals. Functions include Daily Records, Automatic Billing, Accounts Receivable, Billing Breakdowns and more! Available: 4th Quarter 1986.*

*I·S TALK, TIME & BILLING, PAPERCLIP ELITE, CONSULTANT, DEGAS ELITE, DEGAS, PORTFOLIO, BTS SPREADSHEET, HOMEPAK, THUNDER AND B/GRAPH ARE ALL FILE COMPATIBLE, OF COURSE!*

BATTERIES INCLUDED

WRITE TO US FOR FULL COLOUR CATALOG OF OUR PRODUCTS FOR COMMODORE, ATARI, APPLE AND IBM SYSTEMS. FOR TECHNICAL SUPPORT OR PRODUCT INFORMATION PLEASE PHONE (416) 881-9816. 30 MURAL STREET, RICHMOND HILL, ONTARIO L4B 1B5 CANADA, (416) 881-9941, TELEX: 06-986-266. 17875 SKY PARK NORTH, SUITE P, IRVINE, CALIFORNIA, USA 92714, (416) 881-9816, Telex: 509-139. © 1986 BATTERIES INCLUDED. APPLE, ATARI, COMMODORE AND IBM ARE REGISTERED TRADEMARKS RESPECTIVELY OF APPLE COMPUTERS INC., COMMODORE BUSINESS MACHINES INC., AND IBM BUSINESS MACHINES INC.

I·S Integral Solutions

THE

# CATALOG

SPRING

$2.00

We are actively seeking new software for worldwide distribution.

# YOUR SUPPORT AND OUR COMMITMENT TO GUARANTEED SATISFACTION, HAS MADE THIS THE MOST TALKED ABOUT CATALOG FOR ATARI USERS.

WE PUT YOU FIRST in every decision we make. You wanted to be the first to have up to the minute news, information and innovative software . . . We delivered a catalog with your Antic magazine. Antic made sure you were the firST to get ST software . . . with a special ST section in the magazine. We listened to you when you asked for more ways to be in contact with our customer service and technical team. Thanks to you we have been able to expand our ANTIC ON LINE through CompuServe and open our telephone lines five days a week 8:00 AM through 1:00 PM Pacific time.

At Antic our customers come first. To make sure that you get first rate customer service we have listed a few helpful guidelines:

**TO ORDER:** Call Toll Free (800) 443-0100 ext. 133. This number has been dedicated to order taking only. When ordering please refer to the product code (e.g. ST0202) listed with each product.

VISA and MasterCard accepted.

**CUSTOMER SERVICE:** Write or call:

Antic Customer Service
524 Second St.
San Francisco, CA 94107
(415) 957-0886 M-F 8AM-1PM Pacific time

Please include your name, address, daytime phone and a clear explanation of your inquiry. For technical questions be sure to include hardware configuration information.

Retain all receipts and record method of payment.

**FOREIGN AND ALASKA:** Please call or write our corporate headquarters listed above in the customer service section.

☐ Check shipping and handling charges on the order form.

**COMPUSERVE:** Log on to ANTIC ONLINE—type GO ANTIC

☐ New Product Information
☐ BBCS Sysop Corner
☐ Customer Service
☐ Ordering Information
☐ Antic Catalog Service

Thanks to you we're able to provide better service and deliver Atari XL/XE and 520 ST software at the best value possible. We're only a phone call away. Call us today.

## UNCONDITIONAL GUARANTEE OF COMPLETE SATISFACTION

**We unconditionally guarantee every product we sell to be free of defects and to operate properly. If you are not completely satisfied, or if any item is defective, just contact our customer service department by mail, or phone, within 30 days of receipt of merchandise to arrange for a prompt replacement. Only returns in new condition, with the original packaging materials will be accepted.**

## UPGRADE POLICY

**All Antic APX Classics programs are backed by an excellent upgrade policy. Just send in your current original program disk with proof of purchase and specify the revision you want.**

**We will copy the new version directly onto the original disk. Please include a $5.00 upgrade and handling fee and send it to Antic Catalog Upgrades—Customer Service Department.**

## PRODUCT WARRANTY

**Antic Publishing, Inc. warrants that the products sold in this catalog will operate properly and be free of defects for a period of 30 days. Should you require warranty service, assistance or information, contact:**

**Antic Customer Service
Antic Publishing
524 Second St.
San Francisco, CA 94107
(415) 957-0886**

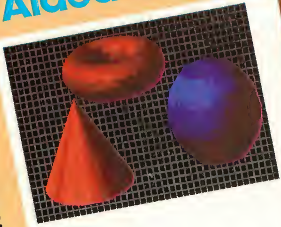
**NOTE: You must send your warranty card to Antic to be covered by this warranty.**

## *We proudly distribute world class* ***ST*** *products*

*U.S.A.*

The author of DEGAS™ enters a new dimension.

# Tom Hudson's CAD-3D™ (Computer Aided Design in 3 Dimensions)



**A full, three-dimensional modeling station for your ST. Solid modeling, with shading from three adjustable light sources, plus wireframe and hidden surface dimensional sculpting —*you name it.* All your 3-D graphics fantasies have been poured into the power and speed of the ST's 68000. So step on the gas and take off!**

From starships to solar systems, skyscrapers to your front stoop. Create and animate views of your own 3-D objects. And no keyboarding here—CAD-3D is entirely GEM-based. It's got to be seen to be believed.

DESIGNERS: Visualize ideas instantly—speeds up design process.

DOODLERS: Relax with true creativity. Save images for later use or alter them with any popular ST paint program.

CAD-3D features include:

☐ Works with monochrome or color monitor. Monochrome mode gives 16 levels of shading, color gives choice of 1 color/15 shades or 2 colors/7 shades each.

☐ View objects in see-thru or hidden wireframe form or solid form with true light shading. Change object form with one mouse click.

☐ Three independent user-defined light sources plus ambient lighting.

☐ Built-in 3D primitives (cubes, spheres, toroids).

☐ Assemble up to twenty 3D objects into a scene and view from any perspective.

☐ "Glue" objects together to create more complex objects.

☐ Create radially symmetrical objects or expand 2D shapes into 3D shapes with one simple operation.

☐ Animation capability—"record" view sequences for viewing with separate display program (included).

☐ Spin and Extrude functions allow even the novice 3D experimenter to create a wide variety of beautiful shapes.

☐ Independent scaling and rotation of objects or groups of objects.

☐ "Camera View" includes zoom lens and variable perspective.

☐ GEM user interface allows use of four views at once, or one large view. Four-view mode is user-definable.

☐ Saves completed images in DEGAS, Neochrome or COLR Object Editor format.

☐ Print your objects with a graphics printer.

☐ Detailed printed user manual.

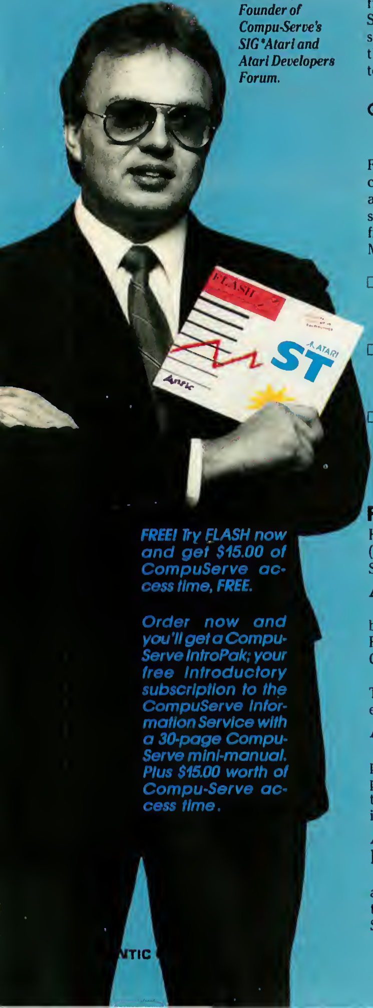
**Requires TOS in ROM**
**ST0214** **$49.95**

Available May 15.

*CANADA*

***"FLASH is the only telecommunications program I use."***

***— Ron Luks, Founder of Compu-Serve's SIG*Atari and Atari Developers Forum.***



***FREE! Try FLASH now and get $15.00 of CompuServe access time, FREE.***

***Order now and you'll get a CompuServe IntroPak; your free introductory subscription to the CompuServe Information Service with a 30-page CompuServe mini-manual. Plus $15.00 worth of Compu-Serve access time.***

# FLASH™

*by Joe Chiazzese and Alan Page*

Everybody knows that Antic is passionately involved in telecommunications. For the ST, we searched for the finest possible terminal program. The best we had seen in other fields was Crosstalk™ on the IBM and Smartcomm™ on the Mac. We wanted something better. It had to be something that would reduce the most complex telecom problems to one mouse-click.

**Good news. We found it.**
**. . . And it uses GEM.**

FLASH goes far beyond any communications software currently offered for any computer. Here is a sampling (a very small sampling) of what you can count on from this extraordinary—PROGRAMMABLE—terminal program.

☐ Hassle-free, GEM-based memo EDITOR. Use your mouse or cursor keys (features block move, undo, search, merge files, and more).
☐ FLASH allows you to scroll back and forth at high speed to review your session—edit it, print it, send it, or save it to disk.
☐ FLASH COmmand Language (FLASH COL) to automate log-ons, file transfers, and unattended operations.
☐ VT100 keypad editing emulation (full 24 line × 80 character display).
☐ CompuServe Vidtex high-resolution graphics terminal emulation. Save Vidtex graphics as DEGAS files and modify or print them out.
☐ Supports Xmodem (CRC) and ASCII TEXT protocols.
☐ Extensive DOS functions at your fingertips. Two clocks: Built-in real-time system clock and elapsed timer.
☐ 20 editable function keys. Chain them together using FLASHCOL, creating totally automated macros.
☐ Translation tables can independently filter any incoming or outgoing characters. Configure your ST to act like any other micro, terminal, or even mainframes. Plus, use filters to create your own secret codes and encrypt files.
☐ High-res flip flop between 24 and 48 lines in monochrome.
☐ Printed manual by Ian Chadwick.

**ST0220** **$39.95**

Crosstalk™ Microstuf
Smartcomm™ Hayes

## FREE OFFER!

Here's what you get on CompuServe (with no surcharge) if you own an Atari ST.

## ANTIC ONLINE

Get your technical questions answered by Tim Oren. Originally with Digital Research, Tim wrote the GEM Resource Construction Set.

Read two new chapters a month of Tim Oren's PRO*GEM tutorial. Available exclusively on-line.

## ATARI 16-BIT FORUM

All of the best ST public domain programs. Gossip with ST users and programmers from Great Britain to Australia. Even reach the authors of FLASH in Canada.

## ATARI DEVELOPERS FORUM

Developers drop in daily to find out about the latest tools and talk of the trade. Includes the ANTIC ON-LINE SOFTWARE SIG.

## VIDTEX ONLINE GRAPHICS

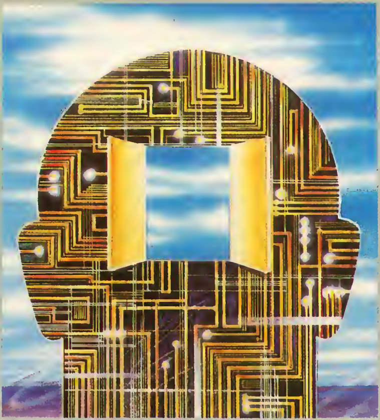
Digitized pictures of Hollywood stars, FBI 10 most wanted list, weather maps, and the new Antic On-Line Art Gallery. With Antic, on CompuServe, you can step into the future of telecommunications.

# ARTIFICIAL INTELLIGENCE

*FRANCE*

***An Expert Systems Package For Your ST***

**EXPERT OPINION™**



***ANTIC WANTS YOU! PUBLISHING OFFER.***

*A New Opportunity.*

*Antic is looking for people to create useful knowledge bases using EXPERT OPINION. If you're an expert on a subject (any subject), get EXPERT OPINION—write a knowledge base . . . And if it's very good, and has broad appeal, we'll publish it for you in this catalog. Write to me, Catalog Product Manager, for more details (my address is on the back page). Give me your expert opinions. I'll try to help you publish some of your knowledge bases on a commercial basis.*

EXPERT OPINION and your expertise can turn your computer into an expert on any subject.

EXPERT OPINION is a full-featured expert system for the Atari ST. (Note: Expert systems are powerful, intelligent programs. They use a reasoning process to develop quick, pragmatic answers for a wide range of complex problems currently requiring human expertise.) Currently, expert systems are used in the fields of medicine, economics, and geological surveying.

WITH THE MANUAL PROVIDED AND SUGGESTED SUPPLEMENTARY READING MATERIAL, YOU CAN BUILD YOUR OWN KNOWLEDGE BASE—EXPERT OPINION DOES THE REST.

Once you've built your knowledge base, EXPERT OPINION is easy-to-use because it's based on a powerful natural language interface, so you can give your input—and get your answers in plain English. EXPERT OPINION can explain how it arrived at a particular conclusion by listing the rules used. Plus, it is the only expert system presently available for micro-computers which can clearly explain why a particular question is being asked (it backtracks up to 10 levels).

**ORDER NOW! FREE BONUS!**
TWO QUICK-START EXAMPLES:
Finance Knowledge Base—30 rules
Palmistry Knowledge Base—150 rules.

**ST0219** **$99.95**

EXPERT OPINION offers more ways to solve a problem than any other expert system:

☐ **DEDUCTION**—Use this mode if you have some initial data about the problem (also known as Forward Chaining).
☐ **VERIFICATION OF A HYPOTHESIS**—Your computer asks you questions about your hypothesis (also known as Backward Chaining).
☐ **EXPERTISE** (totally new technique)—A combination of the above, for situations where you have no initial data about the problem, and no hypothesis (Mixed Chaining Mode).

EXPERT OPINION features:

☐ Dictionary linked to each knowledge base.
☐ Unlimited number of knowledge bases.
☐ Extensive printed manual includes: How To Use an Expert System, How to Construct a Knowledge Base, plus a "Theory" chapter, Glossary, Bibliography, and more.
☐ User-friendly command structure and [Help] functions for simple usage.

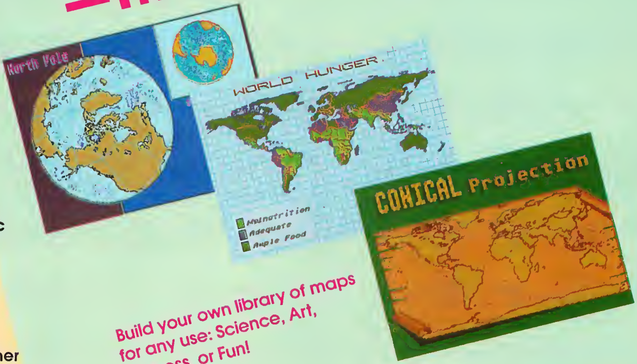
WARNING: This is a sophisticated computer science tool requiring study to use it effectively. We recommend the following books to help you use the program to its fullest:
THE COGNITIVE COMPUTER, Addison Wesley 1984
EXPERT SYSTEMS, Wiley Press 1985
INTRODUCTION TO ARTIFICIAL INTELLIGENCE, Addison Wesley 1985

**From Mind Soft, France**

TOLL FREE — Credit card holders, call toll-free, 24 hours-a-day — 800-443-0100, ext. 133 (Continental U.S. and Hawaii) — ORDERS ONLY!

Orthographic
Mollweide
Perspective
Foucault
Flamsteed
Conical
Mercator
Polyconic
Azimuthal Equidistant
Werner
Cylindrical

# MAPS AND LEGENDS™ —The Cartographer



Build your own library of maps for any use: Science, Art, Business, or Fun!

*Written for the Atari ST*
*by Harry Koons and David Chenette*

Put yourself anywhere in the world — even your own hometown. And now view that land mass from any altitude—in full perspective! Design a wide variety of maps with your Atari ST, then print them out on a graphics printer.

Using advanced CAD system algorithms, MAPS and LEGENDS takes less than a minute to do what used to take hours for computers—and weeks by hand. It features ultra-fast drawing of world maps from one of 11 perspectives, three resolutions, up to 16 colors.

## ENTIRELY GEM-BASED FEATURES INCLUDE:

☐ DRAWING TOOLS
- Multiple Patterns
- Variable Size Pen and Eraser
- Text (3 sizes, includes Antic's Font Accessory and over a dozen fonts)
- Fills (both solid and patterned)
- Make any size maps using window sliders and size button. Put up to 10 different maps and perspectives on one screen!

☐ OPTIONS MENU:
- TEXT Mode (place labels anywhere)
- LOCATE Mode: Reads geographic coordinates from map, and distance and bearing from any point on the Earth.
- STATE BOUNDARIES (for U.S. maps)
- GRATICULE Mode (variable-size grid)

☐ DEGAS and Neo compatible. Use your maps as basic elements in the most popular ST graphics packages.

☐ Works with color and monochrome systems.

## ☐ EDUCATIONAL AND PRACTICAL:

- For Educators from K thru 12, and university levels too. Teach your children to recognize land masses, oceans, other points of interest.
- Next time you go on a trip, make a map to go with your itinerary. (Very impressive).
- Complete documentation reference file (includes: glossary, bibliography, and technical notes).

**ST0202** **$34.95**

NOTE: When we first saw this program at Antic, we were impressed. It performs functions usually found on cartography systems that cost $700 and up (plus, it's GEM-based, so anyone can use it). After it was in the office for a few weeks, everyone began making maps. The editorial, art, circulation, marketing and accounting departments all found creative uses for it and everyone loved it because it was so much fun. So, we decided that it should be in the catalog, but we couldn't decide on the price. We settled on $34.95, with a huge documentation file on the disk, because everyone ought to be able to afford a copy.

*FINALLY! GEM DEVELOPMENT AND LEARNING TOOLS AT A SUPER VALUE!*

*UNITED KINGDOM*

# A-SEKA™ by Kuma (68000 Assembler, Editor, Debugger)

**When you want it NOW.**
**A-SEKA—For Speed.**

*By Andelos Systems/Kuma, UK*

**Sometimes you just need to get that code running faster. A high-level language application needs a burst of energy. Or maybe it's arcade action—high end stuff. A-SEKA does it *fast*, because it is all in RAM. All of it: The Assembler, Editor and Monitor/Debugger. Those who know how can create exciting codes mighty fast. And if you're learning Assembly, you won't ever have to wait for your latest attempt to go through the assemble and link process.**

A-SEKA assembles source codes at over 30,000 lines per minute! And since it can assemble and link simultaneously, you can *run your code instantly.* Of course, A-SEKA is also a macro assembler and uses standard Motorola mnemonics. But what really sets it apart is its powerful machine language monitor, disassembler and symbolic debugger.

## DEBUGGER FEATURES:

- Symbol table access.
- Arithmetic operations. Input in any base.
- Disassembles 16 lines at a time.
- Motorola mnemonics.
- Single step. Trace.
- Multiple breakpoints. Memory inspect and modify.
- Line assembler.
- Examine registers.

## AND MORE . . .

All this, for under $35! You're probably saying to yourself, "Sounds great, but what's the catch?" OK, here it is. RAM-based assemblers can only assemble programs which are small enough to fit into the edit and code buffers of random access memory at the same time. On the ST, that's quite large, but there will always be a limit (there is no size limit for our other assemblers).

**ST0216** **$34.95**

NOTE: In a recent product review in *Page 6*, the original British magazine for Atari users, the reviewer said, *"A-SEKA is most useful to the programmer interested in learning 68000 assembler. It provides everything you will need. . ."*

# A-RAM™ by Kuma

**Take a look at our RAMdisk. (Random Access Memory disk emulator) It's a remarkable value.**

*By Roddy Pratt, UK*

Can your RAMdisk partition any size disk emulator you want?
► A-RAM can.
Can it work with TOS in ROM?
► A-RAM can.
Can your RAMdisk accelerate your floppy write speed by turning off the verify mode?
► A-RAM can.
Can you have multiple RAMdisks present at the same time?
► You guessed it. A-RAM can.

A RAMdisk is an area of memory set aside as a buffer that responds to most of the available disk commands—only much faster. Everybody needs a great RAMdisk, and A-RAM is powerful, simple and flexible enough for *every* application.

**ST0215** **$19.95**

TOLL FREE — Credit card holders, call toll-free, 24 hours-a-day
**800-443-0100, ext. 133** (Continental U.S. and Hawaii) — ORDERS ONLY!

# GST C™ Compiler

*UNITED KINGDOM*

## Here is the compiler that 1ST WORD was written in.

**Do you want to write GEM-based programs for the Atari ST? With GSTC you can—without spending hundreds of dollars on expensive compilers. Now you can add windows, dialogs, and all the GEM forms to any program — it's *easy* with GSTC. Use your mouse and pull-down menus to write C programs within a desktop menu-driven "shell" environment. GSTC allows compile-assemble-link and assemble-link operations to be batched, avoiding tedious and error-prone command line entry. And it all fits on one single-sided disk. No excessive disk swaps.**

At the heart of the GSTC package is the remarkable *GEM Superstructure Library*. This enables the beginner to write GEM applications software at once, without the complex learning curve associated with GEM AES and VDI. Open a fully-functioning window with one call. GSTC is fast—providing compile and linkage turn-around times speedy enough for the most impatient hacker!

GSTC features include:
- GEM Text Editor
- Linker
- C Compiler
- GEM "shell"
- 68000 Assembler
- GEM Superstructure Library
- GEM bindings, (Standard Unix, GEM VDI, GEM AES, GEM XBIOS, TOS)
- Comprehensive printed user manual

**ST0217 $79.95**

NOTE: We searched all over the world to find the best introductory C compiler. When we discovered that GSTC was used to write 1ST WORD, we decided that it was just what we were looking for. This compiler is very powerful and remarkably easy to use. But at present, it doesn't have structures or a floating point library. If you're writing a program that uses very serious math, you may need to look at Lattice C. But if you're only writing a word processor, GSTC will do the job.

## GST-LINK

GST-LINK is supplied with GSTC and GST-ASM and enables separately compiled or assembled program modules to be linked together and to extract any run-time library routines from the GEM libraries. GST-LINK features include:
- Relocatable, compact, binary format
- Optional SID debugger symbols
- Automatic run-time relocation of modules by the TOS loader
- Comprehensive link map listing with optional symbol table
- Optional global symbol cross reference
- Link operations driven from a batch control file

GST-LINK is the linker that Metacomco chose to use with all their products.

**FREE! With every GSTC and GST-ASM!**

## GST-EDIT

(Universal GEM Screen Editor)
GST-EDIT is to programming, what 1ST WORD is to word processing. It's a GEM-based text editor which you can use for writing programs in any language that accepts ASCII files. If you know how to use 1ST WORD, you're already an expert with GST-EDIT. Its features include:
- Up to four simultaneous files in separate windows
- Block cut and paste between windows
- Comprehensive search and replace functions
- Cursor movement by mouse or keyboard
- Full on-screen help information

**FREE! With every GSTC and GST-AM!**

# GST-ASM™

## A high-level Macro Assembler with an unbeatable combination of price, performance, and features.

**GST-ASM is a Motorola-compatible 68000 macro assembler with advanced features — including high-level control flow instructions, very powerful macro facilities and extremely fast throughput. GST-ASM is designed for the professional who needs a sophisticated macro assembler to develop real-time software products. And since it uses the GST GEM interface, it's a joy for beginner and intermediate programmers as well.**

GST-ASM features include:
- 68000 macro assembler
- Linker
- GEM text editor
- GEM "shell"
- Unique, high-level instruction macro library (IF, WHILE, REPEAT, CASE, etc.)
- Generates relocatable code
- Produces object code compatible with Lattice C, Meta Pascal, and Meta Assembler.
- Comprehensive printed user manual

**ST0218 $59.95**

## LATTICE C™

**The standard for the 68000.**

*UNITED KINGDOM*

### COMPILER

- **Full Kernighan and Ritchie implementation**
- **Powerful data types (pointers, arrays, structures, unions)**
- **Separate compilation**
- **Conditional compilation**
- **Macros**
- **LATTICE design**
- **True native code compiler**
- **Comprehensive error handling, including warning messages**
- **Full floating point arithmetic**
- **Optimized to produce fast, compact code**
- **No runtime licenses required**
- **All C language features are supported, including:**

  **PRE-PROCESSOR COMMANDS:** #include, #define, #undef, #if, #ifdef, #ifndef, #else, #endif, #line.

  **STORAGE CLASSES:** extern, static, auto, register, typedef.

  **TYPE DECLARATORS:** int, char, short, unsigned, long, float, double, struct, union.

  **OBJECT MODIFIERS:** ", [ ], ( ). Declarations may be arbitrarily complex.

  **INITIALIZERS:** Full range of expressions accepted.

  **SCOPE RULES:** Identifiers may be redeclared at the beginning of any block, but all "extern" objects must be declared consistently within the same module.

  **STATEMENT TYPES:** All are supported, including labels and goto.

  **OPERATORS:** All are supported, in the standard precedence, including conditional and comma operators.
- Other features include comment nesting, variables up to 39 characters in length and separate name lists for each structure or union.
- Full Text Editor
- Complete Linker
- 270-PAGE MANUAL
- Use of the linker allows complete interface to GEM VDI and AES functions and to library of Unix and utility functions. Libraries are provided for complete interface to these functions, allowing all the features of the Atari ST—icons, windows, graphics, etc.—to be used. The graphics libraries are included in source code form to aid understanding and to allow the user to change the libraries (if required).
- Compatible with two linkers. The object code produced by the compiler is compatible with both the linker supplied with the kit and also with LINK68 from Digital Research.

**ST0207** **$149.95**

**Technical questions? Talk with a Metacomco System Software Engineer: Call (408) 438-7201.**

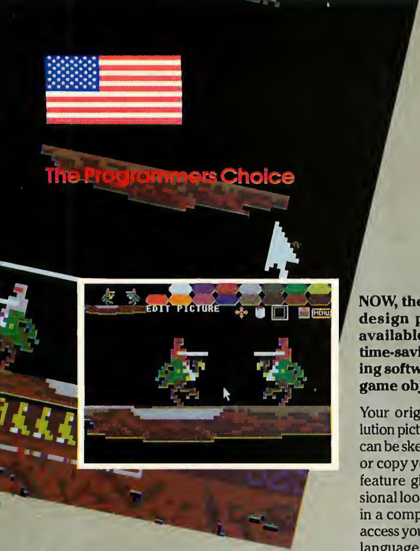
## ANNOUNCING—A*PLUG— THE ANTIC Programming Languages User Group

CompuServe has set aside a special part of the Atari Developers SIG (type GO ATARIDEV) for the users of ANTIC languages. A*PLUG has its own Data Library (DL1) and Message Base (Sec. 1). Here you can meet others using LATTICE C, GST-C, GST-ASM, A-SEKA, META PASCAL, META MACRO ASSEMBLER and the other fine development tools published by ANTIC.

A*PLUG is:
A User Group
A Programmer's Resource
A Programmer's Exchange
A Place to ask other programmers
A Place to answer other programmers

A*PLUG is not:
ANTIC Customer Support. As always that is in ANTIC ON-LINE or call (415) 957-0886

*FREE—ON COMPUSERVE! (no surcharge regular CompuServe rates apply)*



# C.O.L.R. Object Editor

*by The Rugby Circle*

**NOW, the first all purpose graphics design programming package is available for the ST. An essential time-saving tool developed for creating software sprites and bit-mapped game objects.**

Your original art or favorite low-resolution picture from any ST paint program can be sketched or modified. Invert, twist or copy your objects. The zoom editing feature gives your picture that professional look. Your object is stored on disk in a compatible format allowing you to access your work from the programming language of your choice.

Store up to 20 objects in memory at once and instantly access four rainbow paint palettes from RAM.

C.O.L.R. OBJECT EDITOR includes object motion examples. This is the only graphics programming tool of its kind for the professional and hobbyist programmer.

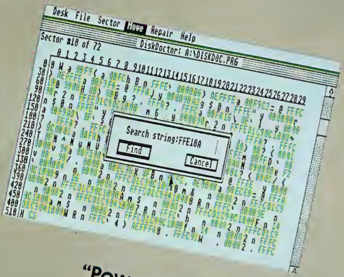
See the November issue of ANTIC for more information on C.O.L.R. OBJECT EDITOR.

*"An all-purpose graphic design utility package for ST program designers using 16-color mode."*

**ST0201** **$29.95**

# DISK DOCTOR

*by Dan Matejka*



"Powerful GEM-based disk utility package."

**Learn the secrets of ST disk structure, and if you're adventurous, try some "disk archeology." Recover deleted files *automatically* and repair damaged files and disks.** Make quick modifications to *any* ST single or double-sided floppy or hard disk. Use DISK DOCTOR to *customize* program menus and messages. Search for character strings or go to any sector instantly. Quickly edit full sectors on-screen using your mouse or cursor keys. Get directory history, file attributes, and more. DISK DOCTOR is perfect for the casual user who just wants to repair a file, OR the *serious programmer* who wants to discover what's really going on. Includes on-disk Help and thorough technical reference information.

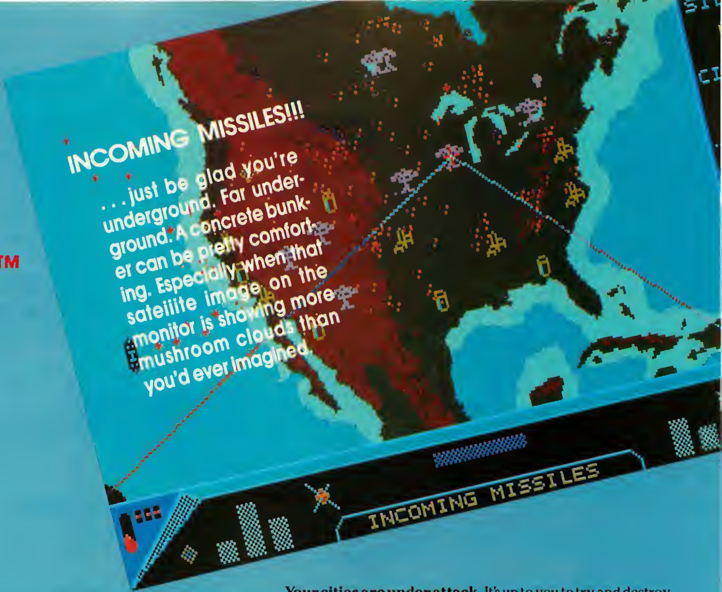
**ST0211** **$29.95**

TOLL FREE Credit card holders, call toll-free, 24 hours-a-day 800-443-0100, ext. 133 (Continental U.S. and Hawaii) ORDERS ONLY!

*U.S.A.*

## RED ALERT™

*Written by Stanley Crane and Daniel Matejka*


INCOMING MISSILES!!!

. . . just be glad you're underground. Far underground. A concrete bunker can be pretty comforting. Especially when that sateiiite image on the monitor is showing more mushroom clouds than you'd ever imagined.

But you have some options. A few. Strategically-placed Ground-Based Lasers and Antiballistic Missile Silos are the backbone of your defense. Your last space-based Particle Beam Weapon is available, but you must allocate your resources wisely to use it.

NOTE: RED ALERT is an addicting game, but it's also a thought-provoking version of an SDI (Strategic Defense Initiative) nightmare scenario. (RED ALERT was developed by the authors of DB Master.)

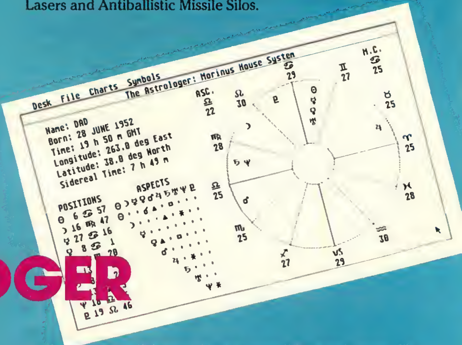
**ST0223** **$24.95**

**Your cities are under attack.** It's up to you to try and destroy the Russian ICBM's and MIRV's—plus the missiles launched from nuclear subs. RED ALERT features real-time animation, on-screen help, auto-demo mode, color/monochrome compatibility, three selectable levels of play, PLUS a unique construction set which lets you customize North America with the cities that you want to protect, and the positions of the Ground-Based Lasers and Antiballistic Missile Silos.

**Create Your Astrological Chart Instantly Using Multiple House Systems!**

## STAR STRUCK™ —THE ASTROLOGER

*By Harry Koons and David Chenette*



```
Name: DAD
Born: 28 JUNE 1952
Time: 19 h 50 m GMT
Longitude: 263.0 deg East
Latitude: 38.0 deg North
Sidereal Time: 7 h 49 m
```

Now your ST can *instantly* generate accurate natal charts for you, your family, and friends—the way an astrologer would do it. Entirely GEM-based, THE ASTROLOGER creates charts based on the most popular types of house systems used throughout history (Morinus, Equal Spaced, Porphory, and more). On-screen help windows show you exactly what all of the custom astrological icons mean.

Thorough on-line documentation is available for reference. All aspects are displayed in graphic form. U.S. and world maps are built into the program—locate your birthplace with the mouse or key-in the exact coordinates. Charts can be printed out or saved to disk in DEGAS format. Compatible with color and monochrome monitors.

**ST0222** **$24.95**

*ISRAEL*

If you haven't heard about MURRAY or MOM yet, you haven't read:
New York Times
London Times
Los Angeles Times
San Francisco Chronicle

## MOM and ME™

Just think. Your own computerized Jewish mother—on-screen in an overstuffed chair; knitting, cajoling, dispensing advice, and offering you encouragement—or making you feel guilty, of course. MOM is a full-color, computerized, cartoon character always ready to chat. Dressed in granny glasses and fluffy slippers, she speaks to you by name and brags about you to your friends.

MOM and ME uses extensive personality files for a wide variety of human-like responses, plus mouse-based response functions for easy interaction.

**ST0204** ~~$34.95~~
**$19.95 this catalog only (monochrome/color)**

**An Electronic Jewish mother created by a former Playboy cartoonist!**

**Written by Yakov Kirschen and Just For You, Inc.**

**On December 5, 1985 the NEW YORK TIMES chatted with MOM:**

| | |
|---|---|
| MOM: | Are you getting enough exercise, Tom? |
| NYT: | Yes Ma. |
| MOM: | By setting in front of the computer all day? |
| NYT: | No Comment. |
| MOM: | He's sulking again! Why don't you go out to a social function? Meet a nice girl. |
| NYT: | I'm married, remember? |
| MOM: | He's married, sorry, I keep forgetting. So how is what's-her-name? |
| NYT: | Change the subject, Ma. |
| MOM: | Mom was only trying to be polite, Tom. |
| NYT: | I know Ma, I know. |
| MOM: | Mom is not waiting for an apology. |
| NYT: | What are you talking about? |
| MOM: | Insult me as much as you want, Tom. Mom loves you. |
| NYT: | When did I insult you? |
| MOM: | I'm knitting a nice warm sweater for you, Tom. |
| NYT: | You're changing the subject. |
| MOM: | But you asked me to change the subject. |
| NYT: | That was before. |
| MOM: | You see how your Mom remembers? |
| NYT: | What are you talking about? |
| MOM: | The sweater, Tom, the sweater. You're so forgetful. You get it from your father's side. |
| NYT: | Cut it out, Ma! |
| MOM: | Mom has to go now, Tom. Be a good boy. |

## MURRAY AND ME™

**Some people don't want a Jewish mother. How 'bout a Jewish uncle?** Meet MURRAY. The LONDON TIMES called MURRAY and ME, "the first in a new generation of Biotoons"—Living computerized, interactive cartoon characters that use your Atari ST to create an artificial personality by using artificial intelligence programming methods. MURRAY is always ready to cheer you up when you're feeling blue (or take pot-shots at you when you're on top of the world).

NOTE: Here are the first two titles in a whole new generation of Artificial Intelligence (AI) programs. These use a behavioral interface with the computer to illustrate how to teach a machine to respond differently to differing stimuli. MOM and ME and MURRAY and ME are the first steps towards an understanding of software behavior engines and artificial intelligence programs.

**ST0203** ~~$34.95~~
**$19.95 this catalog only (monocrochome/color)**

**TOLL FREE** Credit card holders, call toll-free, 24 hours-a-day **800-443-0100, ext. 133** (Continental U.S. and Hawaii) **ORDERS ONLY!**

| | | # Progs/K | Mag No. | Disk No. |
|---|---|---|---|---|
| MAR | **ULTIMATE PRINTER GUIDE**<br>Feature Programs: KWIK DUMP (Best Graphics Dump), FONT-MAKER, CUSTOM PRINT (for Atari special char's. by Matt Ratcliff), LABELMAKER, KEYBOARD MACRO COMMANDER, TWO BIG GAMES, and Secrets of Atariwriter, Printer Guide, XL Parallel Bus Revealed (part III) . . . | 8/61K | AMS0385 | ADS0385 |
| APR | **COMPUTER FRONTIERS**<br>Feature Programs: S.A.M. SPEECH EDITOR, EIGHT QUEENS PROBLEM (Solution), PRICE'S PAINTER GETS FRIENDLIER, DOT MATRIX DIGITIZER, CRYPTOGRAPHY, MANEUVER, CRAZY EIGHTS!, and Welcome to ANTIC ON-LINE, Expert Systems, XL Parallel Bus Revealed (part IV), Profile: Nolan Bushnell . . . | 10/87K | AMS0485 | ADS0485 |
| MAY | **3RD ANNIVERSARY ISSUE**<br>Feature Programs: SON OF INFOBITS (Database Editor), TSCOPE AUTODIALER, ARENA RACER, BEER PARTY ATARI, MODE MIXER, FADER II, AMAZING (Action! game), and ST: Meet the 68000, GEM overview . . . | 10/86K | AMS0585 | ADS0585 |
| JUNE | **COMPUTER ARTS**<br>Feature Programs: VIEW 3-D, GRAPHICS UTILITY PACKAGE (For Atari BASIC), PICTURE ENHANCER, GUITAR TUTOR, THE MUSICIAN, HELICOPTER ROUND-UP, TURBO TYPO II, and MIDI overview, Logo music, GEM Seminar coverage, Profile: Ron Luks (SIG*Atari) . . . | 10/173K | AMS0686 | ADS0686 |
| JULY | **COMPUTER CHALLENGES**<br>Feature Programs: 3 Puzzles (CRYPTOQUOTES, SLIDE, NAME THAT SONG), 4 Arcade Games (STAR VENTURE, DARKSTAR, OVERFLOW, MINIATURE GOLF), and 130XE Bank Switching, Everything About Every DOS, Profile: Joel Billings (SSI) . . . | 8/145K | AMS0785 | ADS0785 |
| AUG | **TELECOMPUTERS**<br>Feature Programs: ATARI 'TOONS (BBS Cursor Art), WETMORE ON THE 1030 MODEM, 1030 PROTERM, DISPLAY MASTER (Special Effects), VALLIANT, and ST SECTION I (Kermit Transfers, Interior View, Desktop Intro), Profile: The Microbits Boys . . . | 9/117K | AMS0885 | ADS0885 |
| SEPT | **POWER PROGRAMMING**<br>Feature Programs: BASIC REVISION C CONVERTER, 16-BIT MUSIC, 130XE ONE-PASS COPIER, MIRRORED DISPLAY LISTS, SOUND EFFECTS LIBRARY, FINE SCROLLING WORLD (Andrews, part I), 8 QUEENS ACTION!, PAGE FLIPPING, CRICKETS, and ST SECTION II (1st Address Map, Using GEM Control Panel, Hi-res Art) . . . | 10/60K | AMS0985 | ADS0985 |
| OCT | **MIND TOOLS**<br>Feature Programs: GRAPH 3-D, YOGA BREATHING, BANJO PICKER, ALIEN ASYLUM, LEMONADE (APX hit economics simualation!), FINE SCROLLING WORLD (part II), and ST SECTION III (CD-ROM IN DEPTH, 1st ST Benchmark, GEM COLOR Program), S.A.T. Software Review . . . | 8/54K | AMS1085 | ADS1085 |
| NOV | **NEW COMMUNICATIONS**<br>Feature Programs: MORSE CODE RECEIVER/TRANSLATER, RAPID GRAPHICS MODE CONVERTER, 130XE MEMORY MANAGEMENT, MORE TYPO II ENHANCEMENTS, VAMPIRE RATS, FAST CURSOR MOVES, and Radio Modems and Software Overview, ST SECTION IV (ST Uses IBM Disk Files, ST Sound) . . . | 8/63K | AMS1185 | ADS1185 |
| DEC | **4TH ANN. SHOPPERS GUIDE**<br>Feature Programs: DISKIO PLUS (Enhancements Plus 2.5 Compatibility), BBS CRASHBUSTER (Ratcliff), BUILD YOUR OWN EPROM BURNER, BOX-IN (J.D. Casten), and 100 Best Atari Products, Profile: Lucasfilm Design Team, ST SECTION V (Intro To 520ST Assembly Lang., ST LOGO Exploration, ST Products Guide) . . . | 6/71K | AMS1285 | ADS1285 |
| '86 JAN | **ATARI PRODUCT REVIVAL**<br>Feature Programs: APPOINTMENT CALENDAR, DUNGEON MASTER'S APPRENTICE, BINGO CALLER, SYSOP SANTA CLAUS, WIREBALL, and Mapping the XL/XE (Ian Chadwick), ST SECTION VI (ST FONT LOADER, TOS Roadmap, UK Software) . . . | 7/69K | AMS0186 | ADS0186 |
| FEB | **PRINTER POWER**<br>Feature Programs: T-SHIRT MAKER, INSTANT-DOS, DOS 2.5 FOR AXLON, MULTI-COLORED PLAYERS, WARRIOR 3000, 1020 PLOTTER, BASIC ON/OFF SWITCHER. ST SEC VII (FORTH ESCAPES, ST Disk Secrets, Crash Clues) | 26/57K | AMS0286 | ADS0286 |
| MAR | **PRACTICAL APPLICATIONS**<br>Feature Programs: LIE DETECTOR, STICKWRITER, HOME HEARING TEST, LUNAR LANDER CONST. SET, and Atari at Work, New Owners Column. ST SEC VIII (MIDI DRIVER, LOGO MODERNE, HIPPO SOUND). | 25/49K | AMS0386 | ADS0386 |
| APR | **COMPUTER MATHEMATICS**<br>Feature Programs: FRACTAL ZOOM, GUESS THE ANIMAL, LIFE REVISITED, 3-D TIC TAC TOE, V(ERSION) SAVER, and New Users Column part 2. ST SEC IX (3-D FRACTALS, ST Cartridges, GEM and BASIC-VDI) | 17/46K | AMS0486 | ADS0486 |
| MAY | **4TH ANNIVERSARY ISSUE**<br>Feature Programs: MOLECULAR WEIGHT CALC, MYGARDEN, POSTERMAKER, JOYSTICK CURSOR, ROCKSLIDE, HEXCONVERTER, and New Users Column Part 3. ST SEC X (JoySTick, ST PONG, and more). | 17/68K | AMS0586 | ADS0586 |
| BONUS | **THE CASTEN GAME DISK**<br>The fabulous games of J.D. CASTEN, updated and improved including some never before published: ESCAPE FROM EPSILON+, RISKY RESCUE (regular and INDUSTRIAL versions), ADVENT X-5, BOX-IN, BIFFDROP (normal and NIGHTMARE versions), NEMESIS, CRAZY HAROLD'S ADROIT ADVENTURE. | 10/80K | | ADS0001 |
| | **ANTIC ST SECTION COMPENDIUM**<br>A packed 3.5" disk! Includes *object* and source code for all ST programs in the August, '85 through January, '86 issues. Featuring FONT LOADER Desk Accessory (and a baker's dozen of fonts), SOUND.C (sound chip demo), COSINE (graphics), plus LOGO programs and some in-house code that's never been seen before. | | | #SB0101 |
| | **ANTIC ST SECTION COMPENDIUM II**<br>Includes all programs from the ANTIC ST SECTION Feb 86 through May 86. Featuring FORTH ESCAPES, MIDI DRIVER, HIPPO SOUND, LOGO MODERNE, 3-D FRACTALS, GEM AND BASIC, VDI, JoySTick and ST PONG | | | #SB0102 |

**Back issues are $5.00 each. Disks are $12.95 each.** All Antic Archive programs are protected by international copyright laws and **are not** public domain.

TOLL FREE

**Credit card holders, call toll-free, 24 hours-a-day**

VISA **800-443-0100, ext. 133** (Continental U.S. and Hawaii)

ORDERS ONLY!

## ANTIC ST PUBLIC DOMAIN LIBRARY

### S/Terminal plus SOURCE!

Get on-line and transfer files with S/Terminal, a full-featured terminal program written in 68000 assembly language. S/Terminal features Xmodem, Xon/Xoff, 300/1200/2400 baud support (and more, up to 19.2K baud), and on-line help screen. ALL SOURCE CODE IS INCLUDED, in addition to object code. S/Terminal is designed for successful Xmodem transfers under difficult conditions and will work with Compuserve from foreign countries. This disk also includes several C source and object graphic examples, plus five LOGO demos.

PD0057 $12.00

### ST BASIC/LOGO SAMPLER NEW!

BASIC: Includes MIDIREC.BAS—a simple MIDI sequencer and sample song files, BG.BAS—backgammon, Fractals in BASIC, Biorythm's, and more. LOGO: Nearly a dozen useful routines including complex graphics. Plus two bonus desk accessories.

PD0078 $12.00

### ST DOODLE plus SOURCE

The perfect GEM learning tool. PD paint program written in "C", including object and source files for you to explore. Works in all three resolution modes. Demonstrates GEM drop-down menus, windows, scroll bars, color selection, fill algorithm, three brush sizes. Comes with NEOVERT—converts your pictures from NEO to DOODLE format. Learn how GEM and the ST work . . . without any typing!

PD0058 $12.00

### DEGAS COMPETITION HI-RES WINNERS NEW!

The top eight monochrome entries from Batteries Included's DEGAS art competition. Includes a slide-show viewing program—DEGAS is not required.

PD0076 $12.00

### DEGAS COMPETITION COLOR WINNERS NEW!

The top eight color entries from Batteries Included's DEGAS art competition. Includes a slide-show viewing program—DEGAS is not required.

PD0077 $12.00

"Learn about Fractals and GEM with "C" source code."

### ST FRACTALS plus SOURCE!

Features MANDLEZOOM by Harry Koons. Uses Mandelbrot algorithm to draw fractals in GEM windows in any resolution. Then zoom in with 2×, 4×, 8×, or 16× magnification. Change fractal iteration values and rescale fractals to enhance their color. Then save your fractal picture to disk. Includes all "C" source and object files. PLUS, a half dozen other fractal programs that use different algorithms and display techniques (some also with source code).

PD0068 $12.00

### SOLID SOURCE CODE NEW!

Features Jim Luczak's VDI SAMPLER and C PRIMER, which demonstrates C programming techniques and the use of VDI functions and their C BINDINGS. BICALC, a desk accessory Binary-Hexadecimal-Decimal calculator. Plus two very fast versions of LIFE, written in Assembler. All source and object code is included and is well commented. And more!

PD0079 $12.00

## YOU NEVER GET A SECOND CHANCE

**at making a first impression, so give your graphics, letters or reports color that's worth remembering.**

**Exciting things are happening in the world of Atari graphics and data output.**

Now you can add the power of color to your printouts. All of our paper is 9½" by 11" with "clean-edge" perforations that tear down to a smooth 8½" by 11". We have colors and selections to safisfy the most discriminating creative, professional and home user.

Get ready to unfold the power of color!

Antic® The ATARI Resource

Premium Sampler Pak contains 160 sheets of 60 lb. paper, 20 each of 8 brilliant colors: red, blue, green, white, pink, parchment, and yellow, 200 matching greeting card size envelopes, and 200 matching business size envelopes.

**CP0001 $19.95**

Professional Pak contains 375 pieces: 250 sheets of paper: pearl gray, aqua, buff, pewter gray, and ivory (50 of each color), and 125 business size envelopes (25 of each color).

**CP0002 $24.95**

The Pastel Sampler Pak contains 350 pieces: 200 sheets of paper (40 of each color: dusty rose, winter green, goldenrod, pastel yellow, and powder blue), 75 greeting size envelopes (15 of each color), and 75 business size envelopes (15 of each color).

**CP0003 $12.95**