SPECIAL MIDI·MUSIC ISSUE!

FDC 50076

ST LOG

THE ATARI ST MONTHLY MAGAZINE

JUNE 1989 ISSUE 33

U.S.A. $3.95
CANADA $4.95

NAMM:
ATARI AND MIDI

MIDI CAPTURE

THE ST AND THE JUNO-106

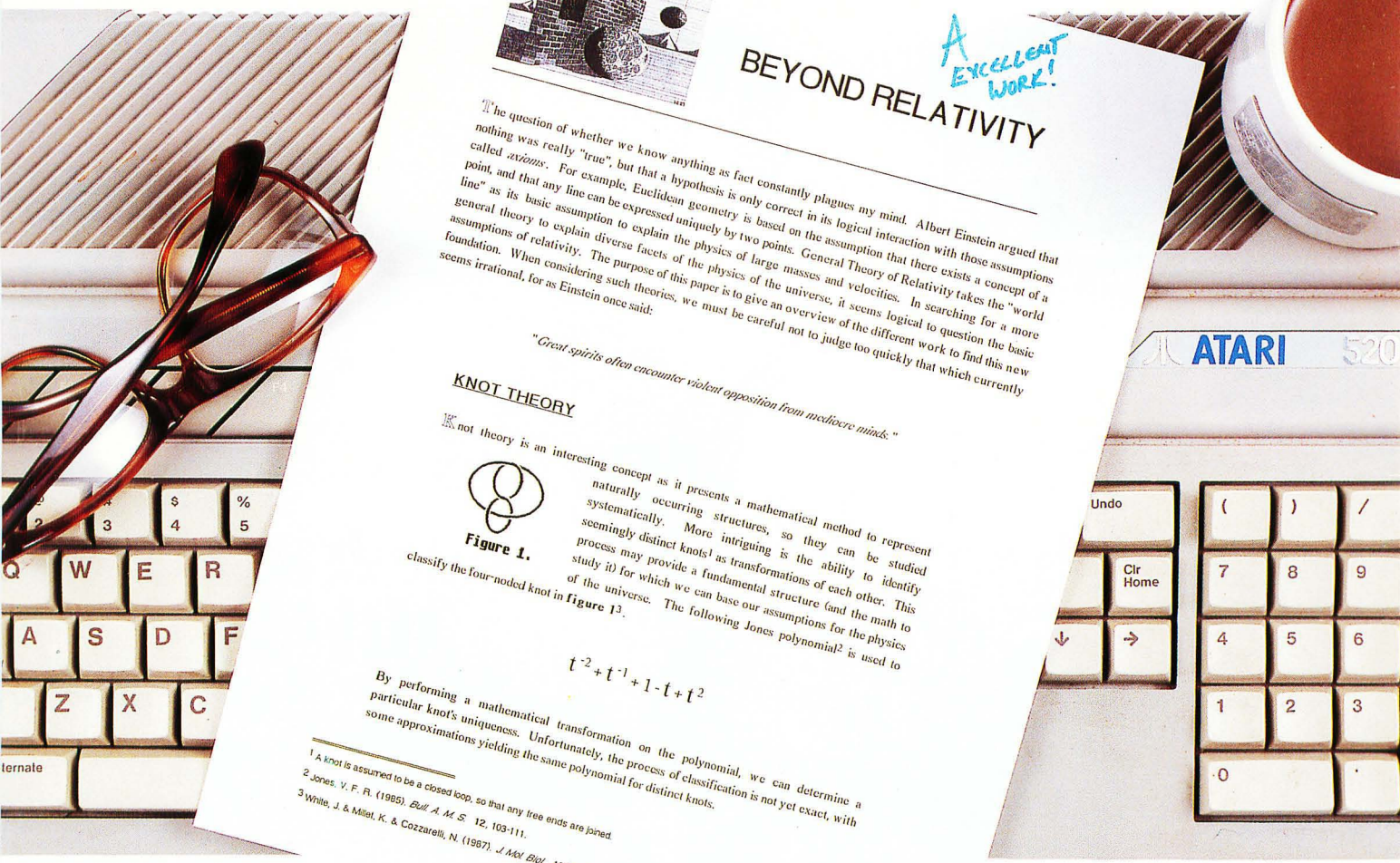
REVIEWS

Roland MT·32
Sound Designer
VDOS ProQueue
SkyChase
Out Run

06
0 74369 50076 1

WordUp...

NEW VERSION STILL $79.95



BEYOND RELATIVITY

A EXCELLENT WORK!

The question of whether we know anything as fact constantly plagues my mind. Albert Einstein argued that nothing was really "true", but that a hypothesis is only correct in its logical interaction with those assumptions called *axioms*. For example, Euclidean geometry is based on the assumption that there exists a concept of a point, and that any line can be expressed uniquely by two points. General Theory of Relativity takes the "world line" as its basic assumption to explain the physics of large masses and velocities. In searching for a more general theory to explain diverse facets of the physics of the universe, it seems logical to question the basic assumptions of relativity. The purpose of this paper is to give an overview of the different work to find this new foundation. When considering such theories, we must be careful not to judge too quickly that which currently seems irrational, for as Einstein once said:

"Great spirits often encounter violent opposition from mediocre minds."

KNOT THEORY

Knot theory is an interesting concept as it presents a mathematical method to represent naturally occurring structures, so they can be studied systematically. More intriguing is the ability to identify seemingly distinct knots[1] as transformations of each other. This process may provide a fundamental structure (and the math to study it) for which we can base our assumptions for the physics of the universe. The following Jones polynomial[2] is used to classify the four-noded knot in **figure 1**[3].

Figure 1.

$$t^{-2} + t^{-1} + 1 - t + t^{2}$$

By performing a mathematical transformation on the polynomial, we can determine a particular knot's uniqueness. Unfortunately, the process of classification is not yet exact, with some approximations yielding the same polynomial for distinct knots.

1 A knot is assumed to be a closed loop, so that any free ends are joined.

2 Jones, V. F. R. (1985). *Bull. A. M. S.* **12**, 103-111.

3 White, J. & Millet, K. & Cozzarelli, N. (1987). *J Mol. Biol.* **197**, 585-603.

...Impresses

WordUp is the only word processor for the Atari ST/MEGA™ that can integrate multiple fonts and pictures.

For the dealer nearest you
Call (805) 482-4446

CIRCLE #101 ON READER SERVICE CARD.

NEOCEPT, Inc.
547 Constitution #A • Camarillo • CA 93010

The language of the above sample document is not intended to be factual. Document was created with WordUp and printed with an Atari Laser. WordUp is a trademark of Neocept, Inc.

EDITORIAL

by Clayton Walnum

It's been almost a year since we've had an issue of ST-LOG dedicated to MIDI and music. That seems kind of silly when you consider that, if there is one niche in which the ST is clearly the leader, it's the MIDI market. Musicians everywhere, both amateur and professional, have responded with great enthusiasm to the ST, buying them by the thousands. The Atari/musical experience has even infected nonmusicians: Many ST owners with little or no musical experience now have MIDI instruments in their homes.

For that reason, we at ST-LOG have decided to expand our coverage of MIDI, making an attempt to keep you more up to date with the goings-on of the computer music industry. That doesn't mean, of course, that ST-LOG is going to change into a music magazine. It just means that we will dedicate more space than we have in the past to MIDI-compatible programs and equipment.

A lot has happened in the world of MIDI in the last year, and it would be impossible to cover it all, but you'll find that this issue is packed with important information, nonetheless. We've got articles for both MIDI novices and the more experienced among you. Whether you want to learn to program MIDI or would just like a little background in this interesting field, you'll find something that fits the bill in these pages. Ten of the articles in this issue are either dedicated to, or contain some information about, the MIDI/Atari combination.

Articles of particular interest include "NAMM, Atari and MIDI," a report on a recent musician's show in Anaheim, California, "MIDI Capture," a beginning tutorial on MIDI programming and *Step 1*: "Make Mine Music," in which Maurice Molyneaux explains, on a novice level, what MIDI is and how it works. Rounding out this MIDI extravaganza are several product reviews and some other interesting articles/programs.

We're willing to bet that those of you who have yet to be bitten by the MIDI bug will be itching to get started once you've finished reading this issue. As a matter of fact, I just went out and spent $800 on a Korg-707 synthesizer, an Alesis HR-16 drum machine and a Peavey KB-15 portable keyboard amplifier. These three pieces of equipment, along with my ST and a sequencer program, allow me to compose and play songs using eight different instrument sounds from the synthesizer and a complete percussion section from the drum machine—a one-man band! The results you can obtain for even this small investment are astounding.

But once you get started, be forewarned that it's tough to tear yourself away. My last "recording session" lasted all night.

The bug bites.

IN THIS ISSUE

FEATURES

BALLBUSTER 42

NAMM, ATARI AND MIDI 12

REVIEWS

COLUMNS

DEPARTMENTS

PROGRAM · LISTING · GUIDE

ST-LOG (ISSN 0890-9601) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

U.S. subscription $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscriptions: add $10 per subscription per year. Single copy $3.50 (add $1 for postage). Change of address: six weeks advance notice, and both old and new addresses are needed. POSTMASTER, send change of address to: **ST-LOG** Magazine, P.O. Box 16928, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

JUNE 1989
ISSUE 32

STAFF

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

ADVERTISING SALES

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

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ST-LOG,** P.O. Box 16928, North Hollywood, CA 91615 or call (818) 760-8983.

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

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

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

PERMISSIONS

SUBSCRIPTIONS

ST-LOG, P.O. Box 16928, North Hollywood, CA 91615 or call (818) 760-8983. Payable in U.S. funds only. U.S.: $28.00-1 year; $52.00-2 years; $76.00-3 years. Foreign: add $7.00 per year per subscription. For disk subscriptions, see the cards at the back of this issue.

AUTHORS

When submitting articles and programs, both program listings and text should be provided in printed *and* magnetic form, if possible. Typed or printed text copy is mandatory and should be in upper- and lowercase, with double spacing. If a submission is to be returned, please send a self-addressed, stamped envelope with your manuscript to **ST-LOG**, P.O. Box 1413-MO, Manchester, CT 06040-1413.

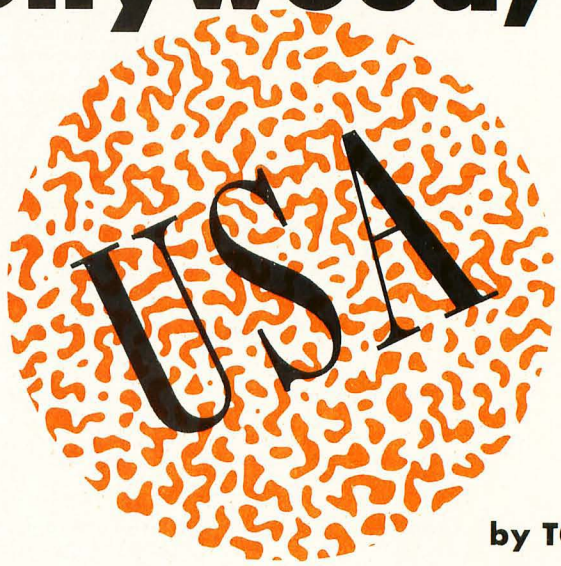
COVER PHOTOGRAPHY:
LADI VON JANSKY

COVER MODELS:
CHRIS DALO
MATT DISCIULLO

INTERIOR PHOTOGRAPHY:
STEVEN HUNT

ILLUSTRATORS:
ALAN HUNTER
JIM SALVATI

ST Gossip from Hollywood, USA



by TG

TOS II?

Remember TOS 1.4, the updated ST operating system? Ever since the first rumors of its existence surfaced, there has been much wailing and gnashing of teeth in the ST community. Many people are frustrated at waiting for its release. Others slam Atari for having taken so long to get it done in the first place.

Several disk-based Beta-test versions of this update have been circulating among developers over the past year. A lot of these developers have highly criticized the bugs and incompatibilities in TOS 1.4, while many users have taken to heart the "Too little too late" sermons of the more vocal ST owners. The final release has yet to be seen, and already TOS 1.4 has been tried and convicted in absentia.

Through all the hullabaloo, the programming staff at Atari has quietly pushed ahead, testing TOS 1.4 more extensively than any previous OS. And the word is out that the ROM version of TOS 1.4 should be available by the time you read this. (Cross your fingers and toes!)

But TOS 1.4 is old news. You're tired of hearing about it, right? Right. And you probably don't care about the TOS to come after 1.4 either. I'll just feed these notes on it to the cat.

What? You are interested? Whoops. Let me get my notes back. Come on, Fluffy, cough it up.

Rumor has it that while TOS 1.4 was being wrapped up at Atari, development continued on the OS beyond Version 1.4. Rather than waiting, Atari programmers are pressing ahead to further refine and enhance TOS, even as Version 1.4 prepares to step into STs everywhere.

If you were waiting for TOS 1.4, buy it. Don't wait for the next version, because, at this stage it could be a while before it shows its face. (Remember how long TOS 1.4 has been in test phase.) There is no direct word from Atari on this subject, but my sources say that the next TOS is supposed to address a lot more of the complaints ST owners have leveled against the current TOS and TOS 1.4.

What kinds of changes are being considered? How about breaking the 16-megabyte limit on hard-disk partitions? In current TOS versions, the maximum amount of data that can be on a hard-drive partition is 16 megabytes. TOS allows only Drives A-P, which are 16 devices, two of which are reserved for floppy drives. This means that the upper limit of hard-disk storage an ST can currently use is 224 megabytes. This may not seem like much of a limit, but to power users who maintain and manipulate massive amounts of data, it can be a serious problem.

Beyond that enhancement the trail gets harder to follow and the information less reliable. Still, it's been whispered that other changes may include the ability to address more than four megabytes of RAM (which is the ST's limit now). If this is true, some changes in the ST hardware must follow. The current MMU (Memory Management Unit) only uses 22 of the 68000's 24 address lines. Just plunking a new MMU into an ST won't work because the MMU's socket doesn't have all the address lines. This implies either a major hardware hack or (dare I say it) another new ST motherboard!

Laser compatibility

Beyond TOS changes there are other interesting developments at Atari Corp. The word about the Atari SLM804 Laser printer is the addition of the letters "PSC" to its name. PSC is supposed to stand for "PostScript Compatible." PostScript is a page-definition system devised and trademarked by Adobe Systems Inc., which is considered the standard in the desktop-publishing field. The current SLM804 printers use a Diablo-emulation system, not PostScript. To further complicate matters, the ST uses Epson printer control codes. In a way, Atari's laser printer is incompatible with the computer it was built for!

Though nothing is yet chiseled into

stone, it appears that the PostScript capabilities will be built into the hardware, giving us a new version of the SLM804 printer.

An Atari Mac?

Strange as it may sound, one rampant rumor is that Atari Corp. is planning to sell an Apple Macintosh clone. If you believe everything you hear, this wonder machine will use a 68030 microprocessor, the actual Mac ROMs, and run blazing circles around Apple's own machines. Some of this may be true, but it's hard to believe that Atari could get away with using the actual Mac ROMs. Apple Computer is quick with lawsuits when it feels stepped on (just ask Franklin Computer, Digital Research or Microsoft), so it's unlikely that Atari will use the actual Mac ROMs.

Aside from the ROM problem, it doesn't seem unreasonable or unlikely that Atari could release a Mac clone. Atari has already taken tentative steps into the clone market with its Atari PC computers (still not available in the U.S., but selling in Europe). The Macintosh, while enjoying a much smaller market share than the PC and its compatibles, still garners a respectable amount of sales. If Atari could produce a Mac clone at a reasonable cost without cutting corners, and at the same time get them into the proper distribution channels without incurring Apple's wrath, it might be worthwhile. If it hurts ST development to do so, that's another matter!

Holy MIDI! An ATARI keyboard!

It is possible that Atari will be unveiling its first entry into the MIDI market since the ST was introduced with built-in MIDI ports. The new product in question is an Atari MIDI keyboard. As usual, specifics are in short supply, but supposedly this product will be customizable into various configurations, and depending on the complexity and power you require, the price will vary from affordable to quite expensive.

It's been said that this keyboard was "completely designed" by Mick Fleetwood of Fleetwood Mac. No confirmation of this, but unless Mick is an electronic engineer, it's doubtful this statement is accurate. Most likely Fleetwood's design input was in the areas of features the keyboard should have and how it should work (from the user's point of view).

If the design is indeed flexible and modular, with the potential for expansion (unlike most other Atari hardware, which is "closed" and difficult to expand), and if Atari applies its "Power without the price" motto to this new entry, it could well be a considerable success. Musicians already know and appreciate Atari's equipment and are likely to give the Atari keyboard a fair shake. Atari is a leader in the field where computers and MIDI meet, and their continued support for this market can only be a good sign. ▪

After a long and relaxing stay at the Institute for the Potentially Nervous, TG has decided to give up his favorite vice, police chiefs' daughters, and his favorite sport, van dodging. Because he's found that fresh air stimulates his creativity, clears his complexion and prevents nosebleeds, he now writes this column while hang-gliding over the Pacific Ocean.

READER COMMENT

The disk-only program

My first contact with ST-LOG occurred this past Christmas. My children offered me two issues as a Christmas gift, and I was so pleased by the November and December issues that I now eagerly await for the new ones to appear on the newsstand.

I like the mixed contents of your magazine. Were I your only reader, I would insist on more technically advanced material, but thinking of the variety of ST users you must cater to, I find that you manage, in every issue, to maintain a harmonious balance in topics and computer "literacy." Overall, I find your magazine interesting and appropriate enough for my needs and interests. I particularly appreciate the type-in programs, with the variety of computer languages they represent and the programming ideas, projects and techniques they suggest.

Those were the flowers; but my view of ST-LOG is not all roses.

The sixth item on your reader survey directly addresses a frustration that is renewed with every issue. Yes, I strongly urge you to publish, in their entirety, all quality programs, no matter how long they are. A case in point is *Ultra-Graph* in the very first issue of ST-LOG I received. Not everybody can afford to buy the disk or log onto DELPHI; I compensate for that with elbow grease (or is that finger grease?). Indeed, I now choose your magazine over the others because I know that, with a little effort on my part, I will get to see and try some of the programs I read about, even though I can't afford them in their ready-made form.

However, if you don't publish the more elaborate (and possibly best) of them, I feel let down and left out—even more so because the magazine cover leads the buyer to believe that all the programs are within the magazine. If you are not going to publish a specific listing, at least have the honesty to say so on the cover.

I commend you for offering readers an opportunity to have their say in their favorite ST magazine and encourage you to keep up the quality work.

—Guy Le Bleu
Montreal, Quebec

The problem of how to present long ST programs has always been a controversial one. We really have only two choices: either supply those programs on disk or DELPHI—or don't supply them at all. We feel that not to offer these programs would be a great disservice to both our readers and the authors.

When readers ask us to print an entire listing, no matter what its size, we doubt they really know what they're asking. Listings for programs like* Ultra-Graph *and* Opus *would fill the entire magazine, leaving room for nothing else. Can you imagine typing something of that length? It would be akin to sitting down with your favorite novel and typing it into a word processor. Even if you had the motivation to take on such a task, the chances of your typing all that code correctly are slim indeed. After spending weeks of typing, you'd almost certainly have a program that, if it ran at all, would be riddled with bugs.

You do have a point, however, about it being difficult for readers to tell at a glance what programs are actually included in the issue. For that reason, starting with this issue, we will always state in the Table of Contents when a program listing is not included in the magazine. If you are buying the magazine for a particular program, first check that program's description to make sure you're getting what you want. We hope that you'll buy the magazine, whether or not a program is "disk only," but we certainly don't want you to think that we're being dishonest. ■

Keyboards for the clumsy

There is an ad on TV these days, showing a fellow waking up to a three-foot-tall alarm clock, dragging a five-foot-wide attache case and trying to talk into a six-foot-long telephone receiver. The ad is for a cold medicine, and at the end, the main character recommends the tablets to a secretary sitting behind an eight-foot-wide keyboard.

That ad came to mind when I read *Ian's Quest* in the January '89 issue and noticed the line "12 cm wide by 14 cm high, with a 6-cm gap between keys (almost 7 cm on the AT)."

Those keys are about 5 inches by 6 inches with a two-inch gap. Does the author have the keyboard from the commercial, or should it have been mm (millimeters) rather than cm (centimeters)?

As I come from Europe, I could not help but submit this piece of "constructive criticism."

—Will Faber
Boca Raton, FL

Well, the secret's out. The truth is that Ian's hands became incredibly enlarged when he had a run-in with a steamroller. Rather than give up computing, he had special keyboards made that would allow him to type with his now foot-long digits. We do thank you, though, for mentioning that TV advertisement. One of Ian's keyboards came up missing a while back, and now we know where it went. ■

Aw, shucks

I would like to compliment you on the high technical quality of your publication. It is refreshing to find an ST publication that does not assume that every ST owner is a computer neophyte. As long as you maintain a high level of technical content, including C and 68000 assembly columns, I will continue to purchase your magazine. Thank you for giving ST users the respect they deserve. —Robert Luneski
Woodland Park, CO

And thank you, Robert, for your kind praise. I'm sure that you'll be delighted to see that four different languages—C, 68000 assembly, Pascal and GFA BASIC—are represented within this issue's pages, along with or as part of several tutorials dedicated to programmers interested in MIDI. ■

A word from Spectrum HoloByte

When we talked to Atari users, we learned that many wanted to see our best-selling air-combat simulator, *Falcon*, converted to the Atari ST. They also wanted us to take full advantage of the machine's capability, rather than just doing a simple conversion from the Macintosh or IBM.

Many of our competitors warned us that within weeks *Falcon* would be up on bulletin boards, and our sales would fall to zero. We chose to disregard these comments and felt that the piracy problem only existed with a handful of users. However, within 30 days of releasing *Falcon ST*, pirates had put the product up in the bulletin boards—complete with diagrams for the code-wheel protection, keyboard layout and mission maps.

The real cost of such software piracy is not the lost $49.95 sale, but rather the lost industry support for Atari ST.

When development, marketing, advertising and production are included, it costs anywhere between $250,000 and $500,000 to introduce a new product. After retailers and distributors take their shares of the purchase price, the publisher receives in the range of $12 to $20 per each copy sold. In addition, publishers must support their products with updates and offer telephone and network support for users.

There is no clear-cut solution to the problem of piracy. All we can ask is that if you like a program, buy it. Think of it as an investment. The more invested, the more and better titles you'll see for the ST. Help us send a message to the rest of the industry that there really is an ST market willing to buy good software. Spectrum HoloByte will continue to monitor the ST market and keep a close eye on what happens with *Falcon ST*. It's a shame that a few users can hurt a market as badly as the ST pirates can and deprive thousands of good ST users of the product support that other machines receive.

—Gilman G. Louie
CEO/Chairman
Spectrum HoloByte

calamus®

Desktop Publishing

The NEXT Generation

"There is no question that this version of Calamus blows its Atari competitors out of the water..."
"In fact, Calamus, in my opinion, competes neck-and-neck with all page layout programs on the Mac and the IBM."

Personal Publishing, September 1988

At a suggested retail price of US $299.95, Calamus is setting a new price performance benchmark for the desktop publishing industry.

For more information or to place your order, call or write to:
ISD Marketing, Inc. 2651 John Street, Unit #3 Markham, Ontario, L3R-2W5, Canada.
Tel. 416 479-1880, Fax: 416 479-1882.

ST NEWS

More music software

Musicians interested in computer-generated music should take a look at Dr. T's *Tunesmith*. *Tunesmith*'s theme generator allows up to five themes with three variations on each to be generated and accessed at any time. Also, advanced rhythm algorithms permit the creation of rhythmically consistent themes.The melody parameters allow the creation of melodies that range from the mundane to the bizarre, and variation algorithms assist in the creation of alternate themes logically extrapolated from the original themes. Up to six voices of MIDI output are obtainable, with each voice independently adjustable for articulation, delay, MIDI channel, octave, dynamics and muting. Parallel harmonies, counter melodies, echos, arpeggios, straight rhythms, cross rhythms and accent doublings are some of the effects that can be created, while drum algorithms allow the creation of percussion that can be derived from, or completely independent of, the melody.

Tunesmith sells for $149.

Also available from Dr. T's is *Copyist DTP*, a desktop music publishing, score-editing and transcription program, which supports both mouse and keyboard control, along with pull-down menus and windows.

A complete set of musical symbols is provided, as is the ability to create your own. The program will transcribe treble, bass, alto and percussion clefs, and allows you to convert any of the supported formats to any other.

Complete editing functions simplify the transcription process. Quantizing is also available.

Sequencers that can be transcribed with *Copyist DTP* include *KCS*, *MIDI Recording Studio*, *Steinberg Pro 24*, *Texture*, *Sequencer Plus*, *Mastertracks* and *SMUS*. The program also reads standard MIDI files.

All Postscript printers are supported, as well as the Atari laser printer, HP Laserjet Plus, HP plotters, and Epson FX- and HP Inkjet-compatible dot-matrix printers. Maximum score length is 100 pages.

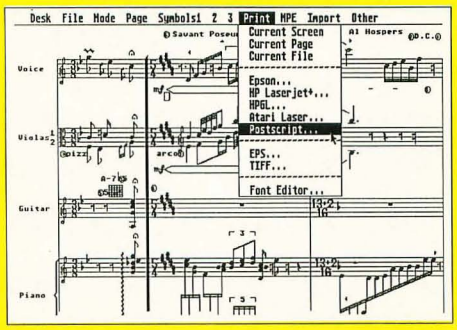
Copyist DTP retails for $399. Two simpler versions, *Copyist Apprentice* and *Copyist*, are also available for $99.99 and $249.00 respectively.

Dr. T's Music Software
220 Boylston Street
Chestnut Hill, MA 02167
(617) 244-6954

CIRCLE #130 ON READER SERVICE CARD.

TUNESMITH

COPYIST DTP



Two new games from Spinnaker

It's not unusual these days for American software publishers to import titles from Europe for distribution in the U.S. Spinnaker Software has now jumped on this lucrative bandwagon with the release of *Starray* and *Quadralien* for the Atari ST.

Starray is a simple shoot 'em up that, according to *Computer & Video Games*, a European game magazine, is "by far the best pure blaster." *Starray* offers fast-paced action in seven levels of horizontally scrolling screens.

Quadralien's action is set in the year 2057, where players must prevent the meltdown of Astra, Earth's cybernetic nuclear fission complex, and destroy the Quadralien forces. Not simply a shoot 'em up, *Quadralien* requires a little thought since players must choose the right combinations of droids, each of which has strong and weak points. The droids must also be properly maintained. Both games sell for $29.95.

Spinnaker Software
One Kendall Square
Cambridge, MA 02139
(617) 494-1200

CIRCLE #131 ON READER SERVICE CARD.

Rainy day flying

What do you do when the weather is bad and you have the urge to fly a radio-controlled (R/C) plane? Easy. Just pick up *R/C AeroChopper* from Ambrosia Microcomputer products and fly your heart out—on your ST.

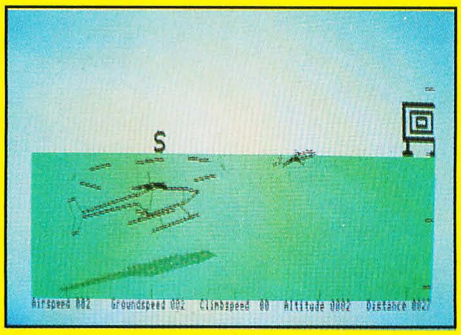
R/C AeroChopper is a hardware/software combination that allows you to control an on-screen plane in the same manner you would fly a real R/C plane. The package includes not only the flight simulator software, but an interface cartridge and professional, two-joystick transmitter (Futaba Conquest), as well.

The program boasts a screen refresh rate of up to 30 frames per second, which gives computer fliers realistic animation and control response. (The frame rates vary, of course, depending on aircraft image size, background complexity, etc.) Seven different aircraft (including a helicopter) are available, with 22 sets of flying parameters. The aircraft and flying conditions can be modified to suit the user through the use of menu parameters. *R/C AeroChopper* sells for $199.95.

Ambrosia Microcomputer Products, Inc.
98 West 63rd Street, Suite 371-H
Willowbrook, IL 60514
(312) 655-0610

CIRCLE #132 ON READER SERVICE CARD.

R/C AEROCHOPPER



New release from Magnetic Images

From the people that produced *Gold of the Realm* comes a new graphic adventure, *Lost Dutchman Mine.* Actually several games in one, *Lost Dutchman Mine* sends players back to the old West, where they can explore the deserts and mountains in search of the over 100 mines containing treasure and clues.

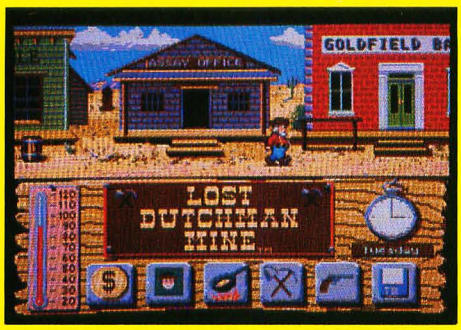
Players will find themselves fishing in the rivers, shopping in the towns and hunting down gold-robbing bandits with prices on their heads. To further enhance the adventure, players can play cards in the saloon, have wounds tended to at the doctor's office and read the local news.

Lost Dutchman Mine features digitized sound, MIDI-compatible music and animated sprites. The price is $49.95.

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

CIRCLE #133 ON READER SERVICE CARD.

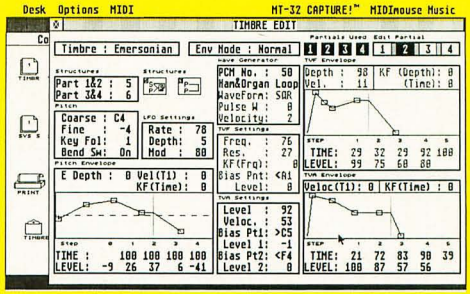
LOST DUTCHMAN MINE



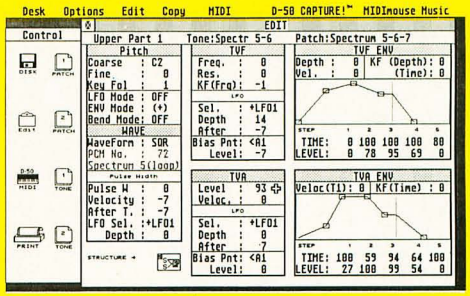
Music from MIDImouse

Under their Sonicflight label, MIDImouse Music has released two new patch editor/librarians, one for owners of the MT-32 synthesizer and one for owners of the D-50.

MT-32 Capture is a full GEM application that provides windows and icons for the display and control of the MIDI data being manipulated. As well as the usual editor/librarian features, *MT-32 Capture* provides a randomize feature that allows the generation of completely new sounds within user-definable parameters.



MT-32 CAPTURE



D-50 CAPTURE

Sound envelopes are displayed on-screen and may be edited from the keyboard or by using the mouse to "drag" the envelope to the required settings. Also included is complete printout capabilities and a D-50 to MT-32 conversion program. *MT-32 Capture* retails for $99.95

D-50 Capture is very similar to *MT-32 Capture*, giving owners of the D-50 synthesizer complete patch editing, programming and librarian functions. It also retails for $99.95.

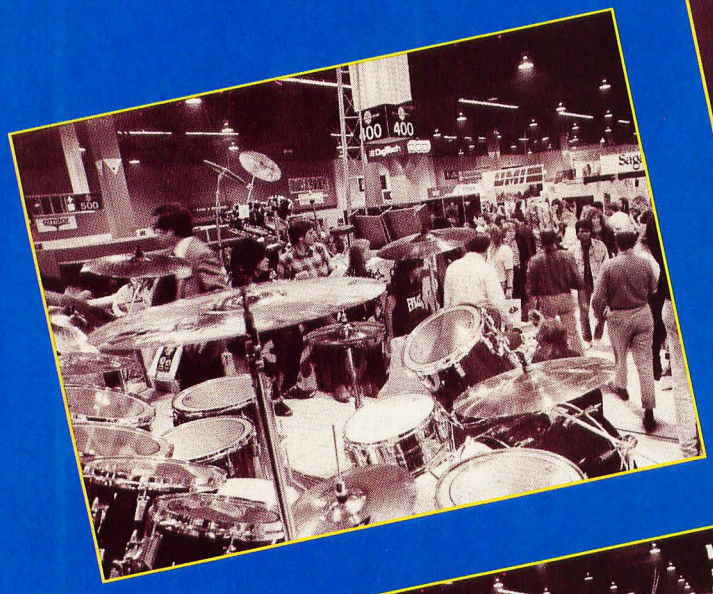
MIDImouse Music
Box 877
Welches, OR 97067
(503) 622-4034

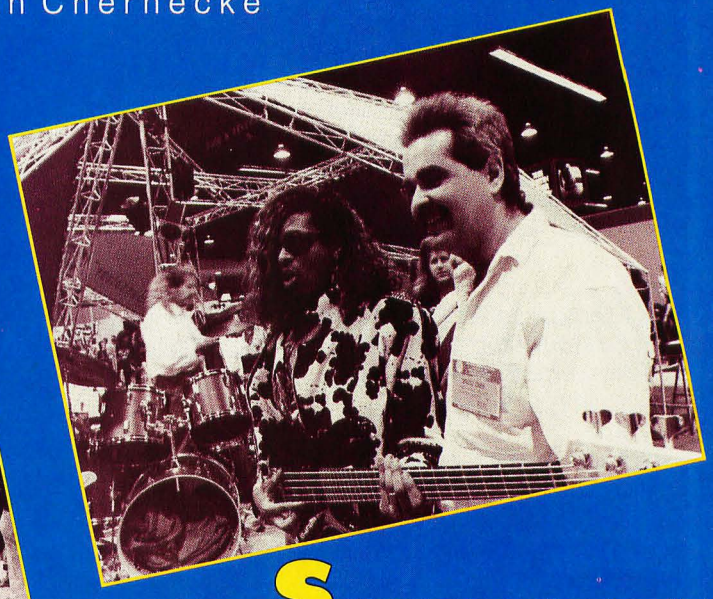
CIRCLE #134 ON READER SERVICE CARD.

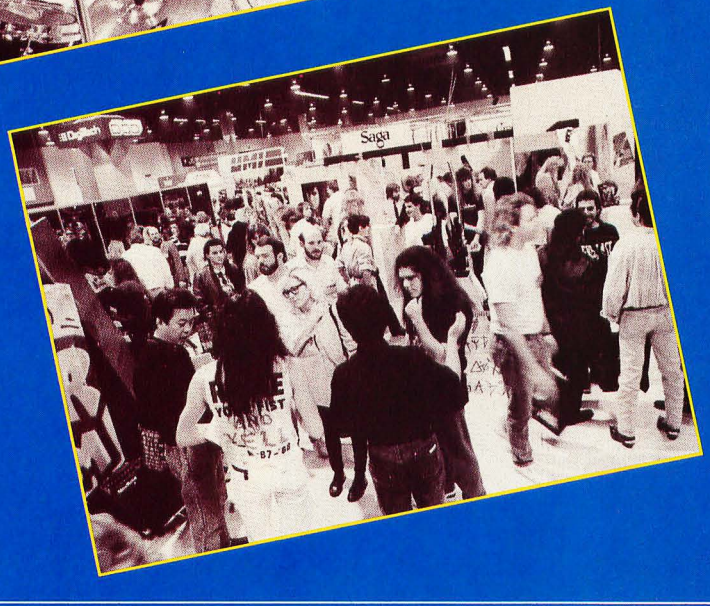
Winter NAMM INTERNATIONAL MUSIC MARKET

NAMM, ATARI AND MIDI

by E. Brian Chernecke







So there I was, standing about 15 feet from Mick Fleetwood of Fleetwood Mac, all that separated us were several beautiful women in skimpy attire. . .ahhh, the music business!

NAMM is an acronym for the National Association of Music Merchants. It is an organization composed of various segments of the music industry, from manufacturers of the small metal corners on speakers to performing musicians. In 1989, NAMM members and invitees will gather three times to display what's new in music. The show I attended was in Anaheim, California, on January 20, 21 and 22. The second was in Frankfurt, West Germany, from January 28 to February 1; and the third show will be in Chicago, Illinois, from June 16 to 18. The show in Anaheim was the first one of 1989 and traditionally important in showing annual trends.

The show was for the most part held in three large buildings with a total of over 300,000 square feet of exhibition space: nearly seven acres of new music gadgets! Upon entering the show mass confusion was the first impression. It was a literal carnival of light and sound, complete with performing sideshows. Some of the spectators were more interesting than the exhibits. People were dressed in everything from three-piece suits to leather webbing, both sexes in each.

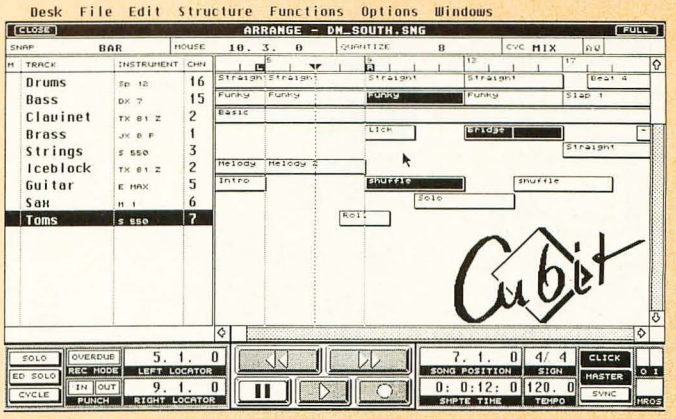
There were over 550 exhibitors at the winter NAMM show. Though computers play an increasing role in the music business, they still accounted for only a few dozen exhibitors, however, that number would be much larger if MIDI instruments are counted as peripherals. There was software for the IBM, Macintosh and Atari computers. Apple Macintosh had the physically largest display, with Atari a close second. I did not see a display area for the IBM PCs, an unusual situation in any other computer situation.

Though the Atari exhibit seemed less overwhelming than that of the Apple Macintosh, the Atari ST had the largest overall representation when the Atari booth was counted with all the software distributors who support Atari products. Space does not permit mentioning all the exhibitors or even all the software of each of the mentioned exhibitors. This is intended as a brief overview of a few products I found interesting.

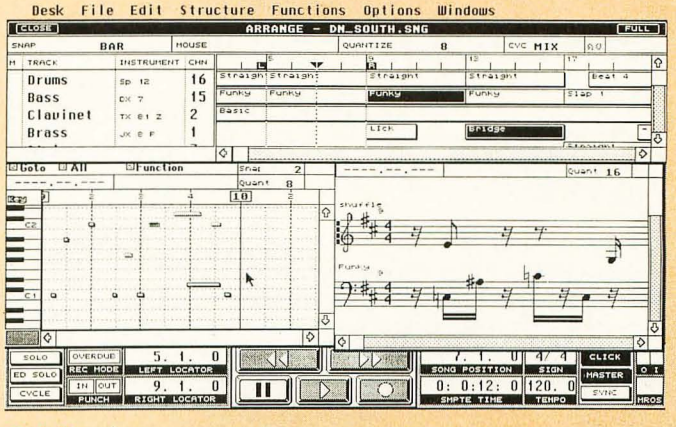
Dr. T's had a display booth of their own and a small station within the Atari area, as well. Such costly double exhibiting indicates great support for the Atari.

Dr. T's has excellent software for the ST. Their *KCS* sequencing program functions as a 48-track tape recorder with lots of standard editing features. I believe it retails for around $200. An added option is the Programmable Variations Generator (PVG), which allows previously recorded music parts to generate new ones with certain parameters such as pitch or rhythm varied. PVG also comes with a Master Editor, enabling very advanced editing features. Dr. T's low-end, entry-level sequencer is *Midi Recording Studio V1.1*, an eight-track sequencing program.

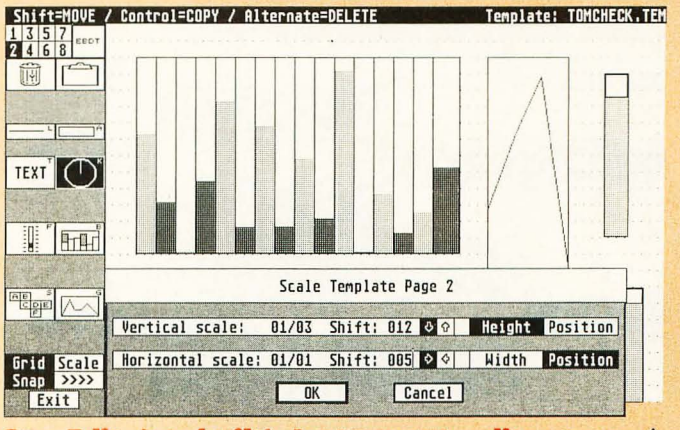
Hybrid Arts Inc. had a space away from other exhibitors at the main entrance to the show. They had an impressive demonstration of their products set up within a mini-sound studio. They have both low-end products for beginners (like a sequencer called *EZ Track* + that retails for $55) and high-end, film-synchronization programs and hardware.



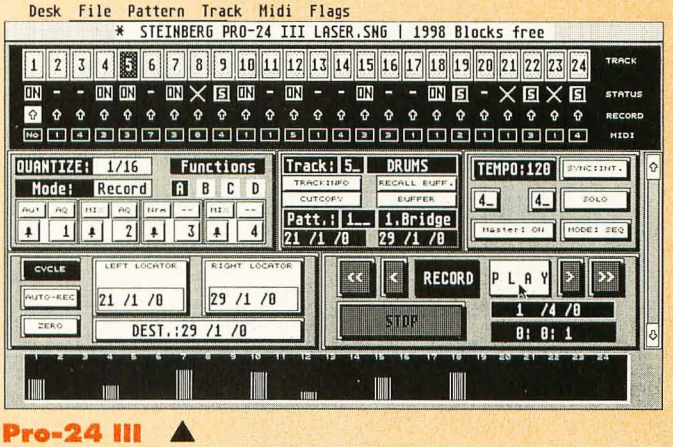
Cubit ▲



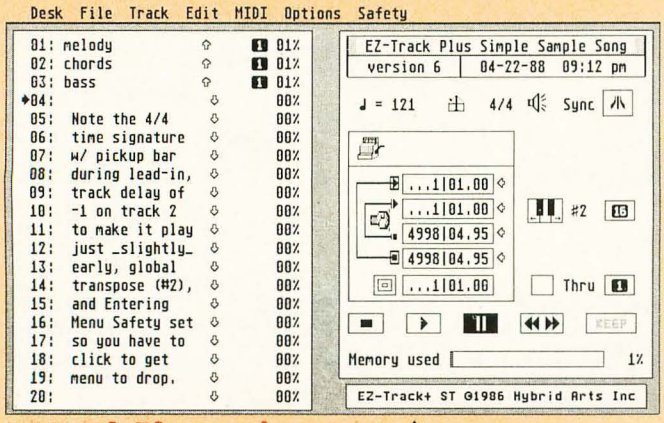
Cubit ▲



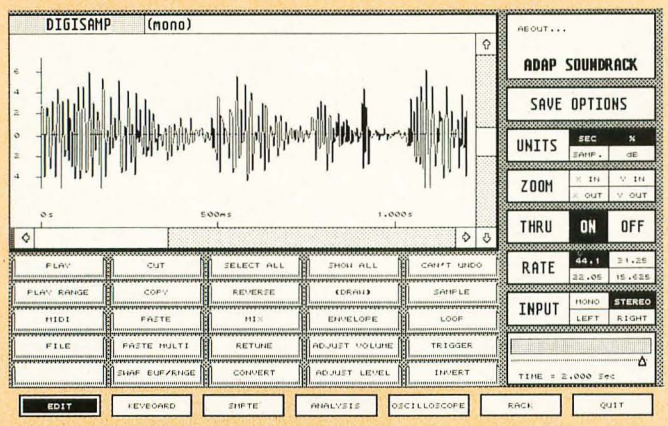
Gen Edit · (prebuilt) · Instrument · edit screen ▲



Pro-24 III ▲

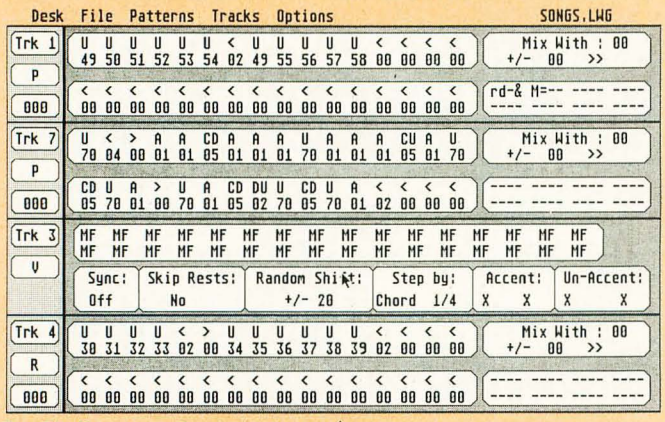


EZ-Track Plus · main screen ▲



ADAP I · edit screen ▲

ADAP II Workscreen · with Markers shown ▲



Ludwig · main screen ▲

MIDI does not record anything but MIDI instruments, so there is a market for devices which can digitize acoustic sounds and store them for later playback. These digitizing systems are hard-disk-based because of the immense memory requirements of such a system. Hybrid Arts was displaying these systems, *ADAP I* and *ADAP II*, but price ($1,995 and $3,495, respectively) will limit their use to only the most professional applications.

Hybrid Arts also markets a librarian/editor program called *Genedit* that allows the editing and storage of sounds in all MIDI devices. Many other librarian programs are specific to a particular keyboard or MIDI device. *Ludwig* is a music composition program using algorithms to generate music sequences with random elements. In other words it composes music for you. Add a few changes of your own and a new Bruce Springsteen is born!

The most noticeable change in music software was the catch phrase "Multiprocessing environment" or MPE. Though it is often touted as something else, it seems to be essentially a switching environment for STs. With MPE a person can load more than one program at a time and then switch back and forth during a session without rebooting. This has practical applications in the music field, as one might want several of the basic types of programs loaded at once.

Steinberg/Jones was showing an entry-level 12-track sequencer, *Twelve*, for $69 that has fairly impressive editing and display features, certainly worth a look if you want to get started. The beginning version is compatible with their top-of-the-line sequencer and should prove to be no transition problem later. Their 24-track sequencer, *Pro-24 III*, which is the next step up, is $350. Steinberg/Jones has a multiprocessing environment program called *Switcher* for $35, which they indicated would work with any programs.

If you want to multi-*task* with an ST, you need Steinberg/Jones' *Cubit*. *Cubit* utilizes Visual Song Processing (VSP), essentially a method of sequencing and editing music using windows and graphics. The exciting part is its use of M.ROS, MIDI Realtime Operating System. M.ROS is an operating system that allows true multitasking on the ST! The price of *Cubit* is "to be announced."

Another company present at NAMM was MIDImouse Music, showing a couple of truly innovative programs. *Mousterpiece*, written by David Carter, who was present to demonstrate it, is really a new musical instrument. Imagine a three-dimensional

ATARI®

AUTHORIZED SERVICE CENTER FOR ALL ATARI PRODUCTS

MICROTYME

A DIVISION OF MICRO PERIPHERALS, INC.

P.O. BOX 369 • KETTERING, OHIO 45409

VISA · MasterCard · DISCOVER

ATARI®

HARDWARE

ST'S. . .IN STOCK!!!

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

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

MODEMS

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

ATARI ST SCANNERS, SOUND & VIDEO DIGITIZERS In Stock!

!!!UNBELIEVEABLE!!!

HAYES COMPATIBLE 2400 Baud Modem - RS232

$125

$ $ $ SAVE $ $ $

HARD DISK DRIVES FOR ST'S

SUPRA 20 MB HARD DISK $569

SUPRA 30 MB HARD DISK $659

SUPRA 60 MB	CALL
ATARI SH 204	CALL

PRINTERS

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

ACCESSORIES

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

MIDI

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

★ ST SOFTWARE ★

10th Frame Bowling	26
221 B Baker Street	28
3D Breakthru	26
3D Helicopter Simulator	34
AB Zoo	21
Advanced OCP Art Studio	31
Air Ball	26
Air Ball Construction Set	17
Algebra 1, 2, 3	ea 14
Aliants	19
All About America	41
Alt	21
Alternate Reality-The City	32
Alternate Reality-The Dungeon	32
America Cooks Series	ea 9
Architectural Design	25
Arctic Fox	26
Art Gallery 1, 2, 3	ea 19
Assem Pro	39
Autoduel	34
Award Maker	27
Balance of Power	34
Bally Hoo	27
Barbarian	26
Bards Tale 1 or 2	ea 34
Base Two	45
Basketball (Two on Two)	26
Battle Droidz	25
Battlezone	19
Beyond Zork	34
Biology 1, 2, 3 or 4	ea 14
Bismarck	28
Black Lamp	17
Blockbuster	27
Boulderdash Construction Kit	17
Brataccas	15
Breach	27
Bridge 5.0	24
Bubble Ghost	24
Bureaucracy	11
Business Tools	26
Cad 3D	65
Captain Blood	33
Carrier Command	33
Certificate Maker	33
Championship Baseball	27
Championship Wrestling	26
Chartpak	34
Chess (Psion)	38
Chessmaster 2000	29
Circuit Maker	54
Clip Art 1, 2, 3, 4, 5, 6	ea 13
Club Backgammon	23
Colonial Conquest	27
Color Computer Eyes	179
Colorburst 3000	25
Compubridge	20
Copyist (DR T)	165
Cosmic Relief	26
Cracked	21
Crazy Cars	25
Cross Town Crazy 8	13
Cyber Control	45
Cyber Paint	58
Cyber VCR	49
Dark Castle	27
Data Manager ST	49
Datatrieve	33
DB Man	159
Death Sword	13
Deep Space	31
Defender of the Crown	33
Degas Elite	39
Desk Cart	69
Diamond Mike	13
Digi Drum	27
Dive Bomber	26
Dr. Drums (DR T)	19
Dr. Keys (DT T)	19
Drafix	129
Dungeon Master	26
Dyna Cadd	449
Easy Draw (Regular)	68
Easy Draw W/ Supercharger	99
Easy Tools	33
Empire	38
Expert Opinion	72
EZ Score Plus	99
EZ Track Plus	43
F15 Strike Eagle	26
Fast Basic	67
Fast Basic M Compiler	39
Fire and Forget	26
First Cadd	33
First Letters & Words	34
First Math	27
First Shapes	29
First Word Plus	63
Flash	23
Flash Cache	54
Flight Simulator 2	35
Scenery Disks	ea 18
Font Disks (Pub Part)	ea 20
Fonts and Borders	24
Fontz ST	23
Foundations Waste	26
Fracton Action	26
Frostbyte	17
Gateway	31
Gato	34
Gauntlet	33
Genesis (Molecular Modeler)	59
GFA Basic	39
GFA Basic Book	27

LARGEST SELECTION IN THE U.S.

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

★ ST SOFTWARE ★

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

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

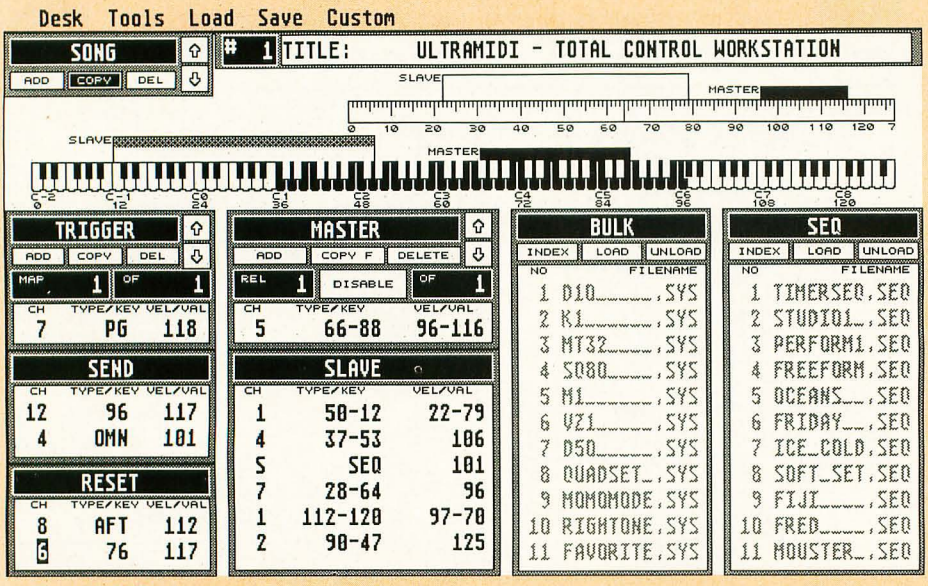
ALL 50 STATES CALL TOLL FREE 1-800-255-5835

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

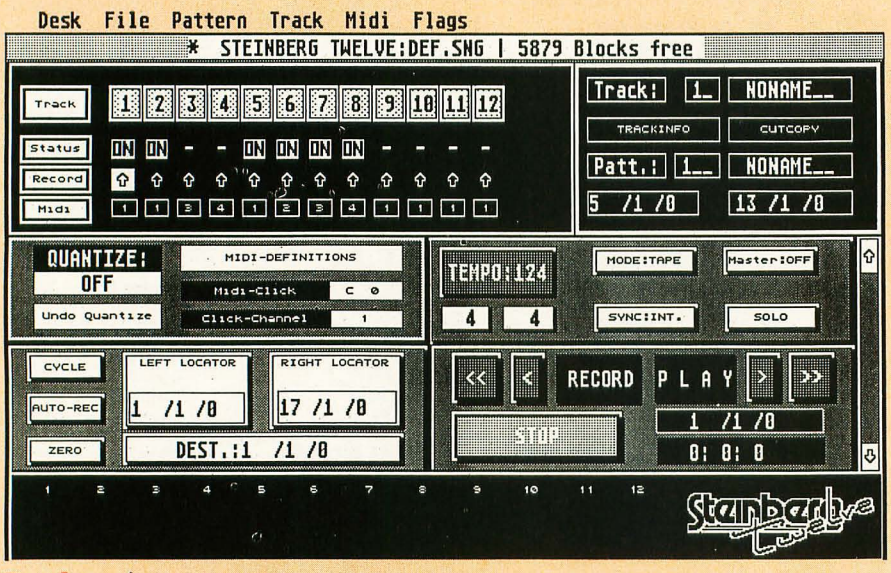
TERMS AND CONDITIONS

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

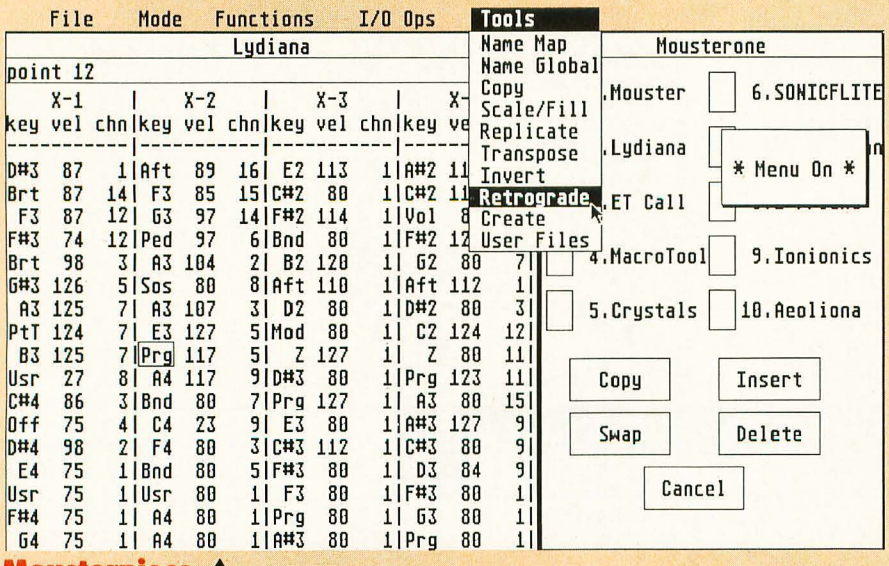
CIRCLE #104 ON READER SERVICE CARD.



UltraMidi ▲



Twelve ▲



Mousterpiece ▲

universe with a cursor centered within it. On each of the X, Y and Z axes there exists four tracks. The user can assign any music parameter, such as pitch or sustain, to any track on any axis. Up to twelve keyboard configurations can be played in three dimensions at once. In order to play the "instrument" created, one only has to move the mouse. As the cursor moves, it selects the music data corresponding to the location of the cursor in X, Y, Z coordinates. The sounds are amazing. *Mousterpiece* ($229) also includes two sequencers and various editing features.

The second attraction from MIDImouse Music was *UltraMIDI* ($229) by Darren Stevens. Again, the programmer was on hand to demonstrate the program. *UltraMIDI* allows a user to reconfigure their MIDI programs and instruments. Instead of using your computer keyboard to load a song or begin a sequence, you can assign that task to a key on your keyboard. *UltraMIDI* allows 255 maps per song, 255 masters per map and 255 slaves per master. It allows macros to erase your sequences from memory after they have been played and has a runtime module to allow the playing of sequences by communicating with the LCD screen on many MIDI instruments, thus eliminating the need for a monitor. The last feature should appeal to working musicians who don't want a monitor present on stage or don't want to drag one around.

Conclusion

The NAMM show is an absolute barometer of the music community. As such, it is clear that the Atari computer is a major force in that community and will remain so for some time to come.

Companies mentioned in this article:

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

Dr. T's Music Software
220 Boylston Street, Suite 206
Chestnut Hill, MA 02167
(617) 244-6954

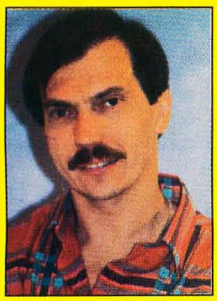
Hybrid Arts, Inc.
11920 W. Olympic Blvd.
Los Angeles, CA 90064
(213) 826-3777

International MIDI Association
5316 W. 57th Street
Los Angeles, CA 90056
(213) 649-6434

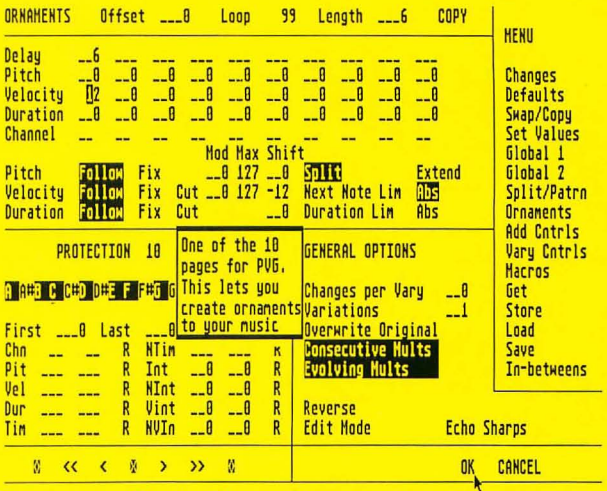
MIDImouse Music
Box 877
Welches, OR 97067
(503) 622-4034

National Association of Music Merchants
5140 Avenida Encinas
Carlsbad, CA 92008-4391
(619) 438-8001

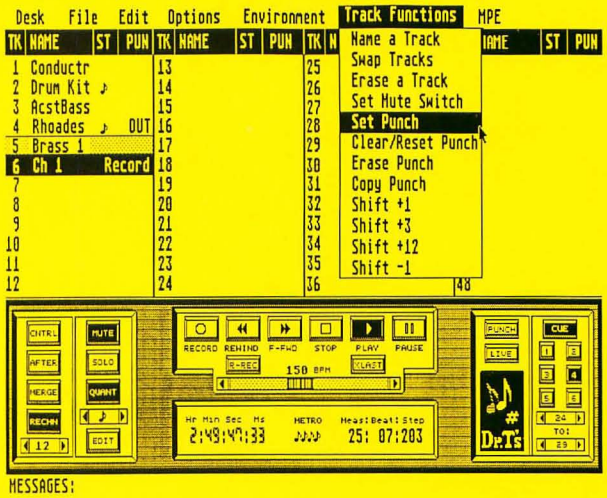
Steinberg/Jones
17700 Raymer Street, Suite 1001
Northridge, CA 91325
(818) 993-4091



E. Brian Chernecke boots his ST in Southern California, where his diverse business and personal interests keep him searching for more and more application programs. He has been involved in the music industry for over ten years.



Program Variations Generator ▲



KCS ▲

Some MIDI background

MIDI is an acronym for Musical Instrument Digital Interface, and is a universal standard for the transmission of musical data between electronic musical instruments. The MIDI standard was officially adopted at the 1983 summer NAMM Expo, although the standard has been revised a bit since then to iron out the bugs and add features. The standard specifications are available from the International MIDI Association for $35.

Though the standard was originally implemented for transmission of data for music purposes, it is apparent that the standard could be used to transmit data for many purposes. It is already being used to control other aspects of concerts such as lighting and visual effects.

As computer technology goes, the MIDI standard is well established. Thousands of musical devices exist that utilize MIDI transmission. They are available in every price range. Most MIDI instruments are designed to communicate directly with each other without the need for a computer. For example, a keyboard can tell a rhythm machine to start, or one keyboard can play several others. However, once a computer is placed in the circuit there is almost limitless control.

Atari was an early pioneer of the music-computer interaction. The ST models have always come standard with a built-in MIDI interface and ports. The ST's have been widely used for music applications from their inception and continue to be a powerful tool for music development and performance. —E.B.C.

Types of MIDI software

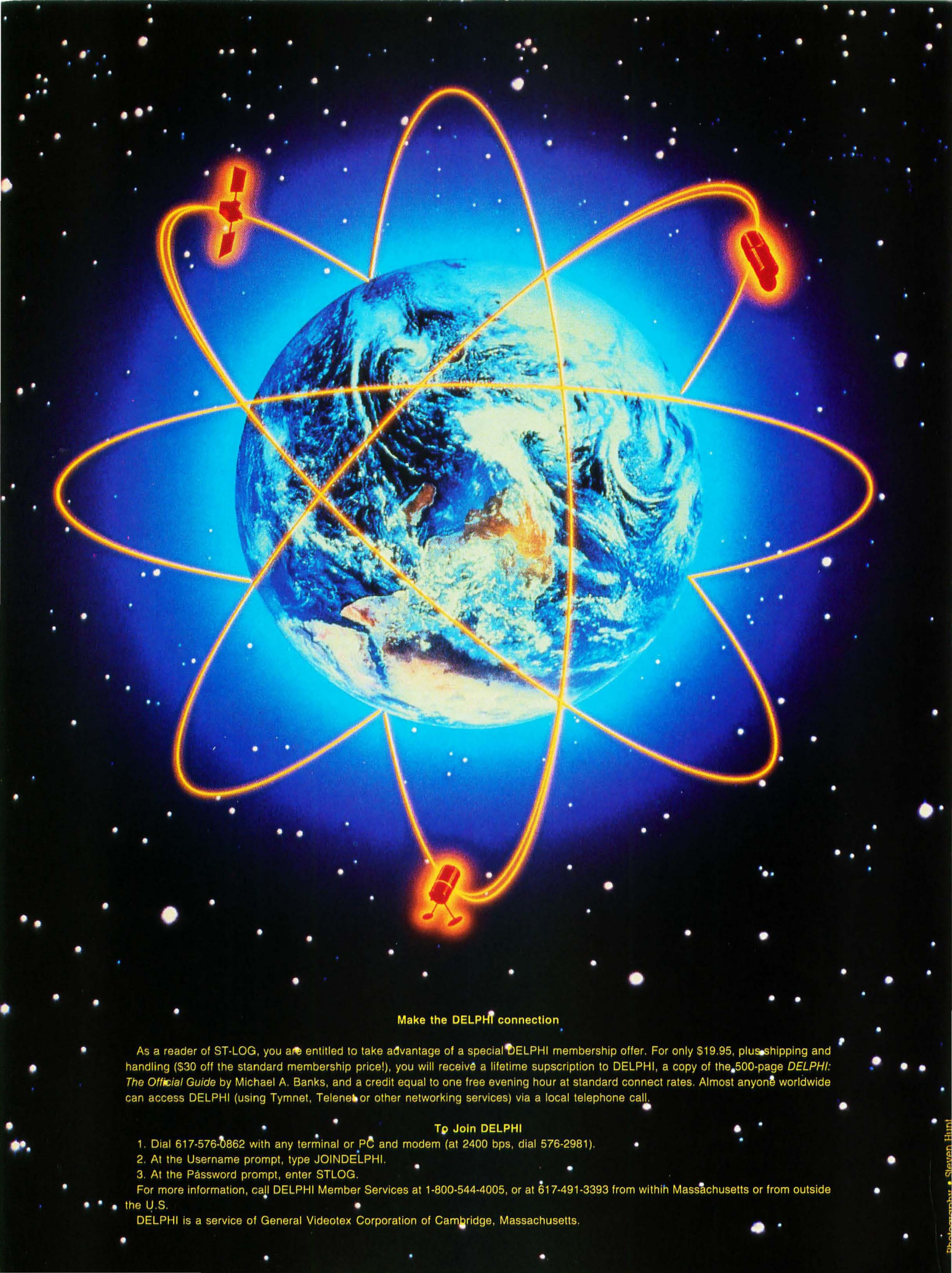
MIDI software may be placed into four basic categories: sequencing, librarian/editor, scoring and hybrid.

Sequencing programs essentially memorize a note sequence played on one or more MIDI instruments, modify it in various ways, store it and play it back through a MIDI instrument. Sequencing parallels the traditional uses for multi-track tape recorders.

Librarian/editor programs deal with the sounds within specific instruments. A keyboard synthesizer can be programmed to make the sound of a concert grand piano, trumpet, oboe or footsteps. I recently pleased myself with the sound of a helicopter in a rainstorm! Many synthesizers have several "voices" and can emit many sounds at the same time. The librarian/editor programs utilize the computer to assemble various sound patterns and then store them for later recall and loading into instruments.

Scoring programs produce written music from music played into the computer. Most programs also display it on screen in standard music notation, allow it to be edited, and then print out sheet music. Amazing stuff.

The last category of MIDI programs is comprised of hybrid and novel uses of the MIDI standard. This category is probably the most exciting in terms of an expanding future for the Atari computers and music. —E.B.C.



Make the DELPHI connection

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

To Join DELPHI

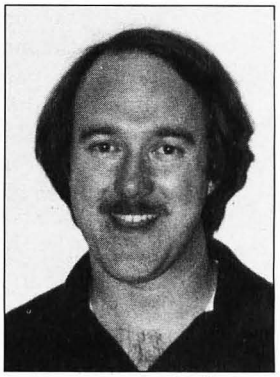
1. Dial 617-576-0862 with any terminal or PC and modem (at 2400 bps, dial 576-2981).
2. At the Username prompt, type JOINDELPHI.
3. At the Password prompt, enter STLOG.

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

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

Photography • Steven Hunt

Database DELPHI



BY ANDY EDDY

Sometimes the abilities that computers give us can be scary when you analyze them. For example, right now I'm reclining in a comfy chair in my living room, with the TV running in the background. Thanks to Craig Patchett, editor of our sister magazine, *PC LapTop Computers*, I've got a laptop computer on my lap while I plug away at this column using the IBM version of *WordPerfect*. When I'm done I can either copy it to a disk and read it on my ST at work, or stick a wire in the wall, dial up the office modem and transmit the file over the phone line. The scariest part is that I've started taking this all for granted.

All this technology is intimidating, but it brings with it a helpful dose of power also. For example, the local news used to come from the town crier, who would walk down the street clanging a bell. Later it came in printed form; but until the printing press made that more efficient, it didn't alter the time it took to get into people's hands. Nowadays a Fax machine can pass information nearly instantaneously, and the December 1988 ST-LOG detailed a news service called *XPress* that carries news to your computer over a cable-TV line.

It comes as no surprise that DELPHI can be a similar portal to news information. Sure, you can find out the latest scoop about Atari computers in either the ANALOG SIG or the ST-LOG SIG, but I'm talking about real, world-shaking news. Here and abroad. Sweat-covered sports action. Calms and storms.

All the news that fits

Let's take a look at the NEWS-WEATHER-SPORTS menu, which is located off the main menu. To get there, just type GO NEW from most any prompt in the system. Figure 1 shows what it looks like.

As you can see from the selections available, you can find out just about anything in current events that you need to. To make it easier for you, we'll run down the list to show you how to get a particular nibble of news.

The first selection, Newsbrief, gives you a comfortable overview of the important stories. The database is regularly updated, so you get fresh news, not leftover headlines. These stories are gathered from Associated Press (AP) wires. Figure 2 shows you how this looks.

If you get the chance to leave your computer desk for any length of time, you'll want to know what it's like outside. Accu-Weather Forecasts lets you preview weather in various parts of the world, a particularly valuable tool for travelers. You have a number of choices concerning the intensity of the weather report. You can pick *CITY forecasts* to provide capsule descriptions of the weather, though this only focuses on the larger cities around the country. Selecting *CAPSULE Summaries (Nationwide)*, which lets you choose one of 115 cities, passes on a brief three-day synopsis detailing projected temperature high and low, and a coded description of the general forecast for those days. Figure 3 shows how.

For a general overview of the nation's weather, you should choose the *NATIONAL Capsule Summary* selection. This lists the U.S. weather as a short combined report.

Last on the Accu-Weather menu is the *INTERNATIONAL Accu-Weather* selection. With listings similar to the *CAPSULE Summaries*, you can get three-day forecasts for locations outside the States.

Almost like an electronic newspaper,

the AP News Service menu provides a variety of categories for you to peruse. In fact, it's called AP Videotex after you get to the AP menu. Videotex was once seen as the future of information provision and was tested in various locations around the country at some TV stations; it didn't catch on very well.

As we noted in the Newsbrief section—which is duplicated in the AP menu—this link-up with Associated Press brings late-breaking news to your screen. These stories are updated almost immediately, as if you had a news ticker hooked to your computer. Figure 4 has what the AP menu looks like.

As you can see, the menu contains much of what you'd find in your local newspaper. Granted the local coverage is missing—you won't find stories about your local government or who in your neighborhood had a baby—but it's a quick and easy way to keep up on worldwide events.

With the exception of the Newsbrief and Sports selections, each choice off the AP menu brings up a list of articles that you can scan. Each article has a simple headline that gives you the basic story content. Figure 5 is a sample of the article list from the Business/Finance from March 4th.

At the time that I captured the article list you see here, the earliest of the stories had been online only about an hour. Certainly there is no newspaper that can get news to you that quickly.

To pick a particular article to read, all you need do is type in the corresponding number, and the story will come up on your screen. If you don't see one that meets your interests and the prompt contains a "More" in it, you can hit Return to get more articles.

The difference between the Sports area and the remainder of the AP Videotex area is not substantial. Because the sports world is large and contains many sub-categories, the Sports menu is broken up into topics as in Figure 6.

And the topics each have submenus also, so you can select specific information without any wasted time. When you get down to it, the topic of interest brings an article list identical in makeup to the previous article lists shown.

It's break time

Sorry to say that's all I have time for. Next month we'll finish up with the NEWS-WEATHER-SPORTS section, covering some of the other diverse entries contained there. Before we go, though, there's a tidbit of news *I'd* like to pass on. As I write this, DELPHI is reconstructing the way the file databases operate.

Perhaps the biggest complaint users have had revolves around uploads and how they are reimbursed. While other services automatically shut off the billing during an upload, DELPHI required the user to request an upload "appointment" to be granted by the SYSOP of the particular group where the file will reside. This process kept duplication of files and poor uploads to a minimum, but was cumbersome to all involved.

Next month we'll have more information on the changes that are taking place. Though they aren't complete, we can guarantee that the user will come out ahead, with an easier process to utilize for file uploads. Remember that the databases are the most accessed areas in a SIG, and offering files for others to download is a big aid to the computer community as a whole. If you've written or acquired a file that can be put in the public domain (not a commercial product), think about taking the time to upload it to the DELPHI/ST-LOG SIG.

If you aren't a member of DELPHI, you should give it a try. Take a gander at the accompanying sidebar for information on how to sign up.

Till next month, C U online. . . ■

Figure 1

```
NEWS-WEATHER-SPORTS Menu:

Newsbrief                    Press Releases: Business Wire
Accu-Weather Forecasts       Sports
AP News Service              Quiz - Your News IQ
Astro Predictions            Today in History
CompuBug                     Views on News
Financial News               HELP
Kyodo News from Japan        EXIT
Movie News & Reviews

NEWS> (AP, Weather, Sports, Movies)
```

Figure 2

```
NEWS> (AP, Weather, Sports, Movies) news
 4-MAR 20:57  Top Stories At 9 p.m. EST

March 4, 1989

   EASTERN

   MIAMI (AP) -- Striking Eastern Airlines workers paralyzed the financially
strapped carrier Saturday, overwhelming management efforts to run a skeleton
schedule and vowing to incite mass-transit chaos when the workweek starts. The
threat of rush-hour nightmares was especially acute in the congested New York
metropolitan area, where strikers planned picketing at commuter railroads and
received assurances that no rail workers would cross their lines.

More?
```

Figure 3

```
Format:
CITY                 SUN       MON       TUE

Search for> hartford

HARTFORD          46/34c    34/20sn   30/13pc
```

Figure 4

```
Copyright (C) 1989 by the Associated Press
All rights reserved.


AP VIDEOTEX Menu:

Newsbrief                    Business/Finance
International                Wall Street
National                     Stock Lists
Features                     Entertainment
Political/Elections          Sports
Government                   EXIT
Weather
```

Figure 5

```
 1   4-MAR 19:54  Time Inc., Warner To Merge
 2   4-MAR 19:09  Rail Lines Seek To Bar Strike
 3   4-MAR 18:36  How Eastern Strike Hits U.S.-1
 4   4-MAR 18:36  How Eastern Strike Hits U.S.-2
 5   4-MAR 17:48  Union Leader Wages Tough Fight
 6   4-MAR 17:43  Strikers Paralyze Eastern
 7   4-MAR 17:15  Eastern Strike Tests Lorenzo
 8   4-MAR 16:38  Eastern Strike Was Building
 9   4-MAR 15:13  Fliers Stuck At Struck Airline
10   4-MAR 14:53  Railroads Covered By Labor Act
11   4-MAR 14:29  U.S. Told To Review Debt Plan
12   4-MAR 11:28  Eastern Strike Threatens Rails
13   4-MAR 11:25  List Of Major Airline Strikes
14   4-MAR 11:24  Eastern Airlines' Bumpy Ride
15   4-MAR 11:22  Nation's Transport Net Braced
16   4-MAR 11:21  Airline Cuts Weekend Schedule
17   4-MAR 11:20  Eastern Airlines Developments
18   4-MAR 08:43  USSR To Cut Oil Exports 5 Pct.
19   4-MAR 06:14  Bordello Shares Offered At $20
20   4-MAR 05:51  4 Japan Businessmen Indicted

AP NEWS> (Article Number, More, Exit):
```

Figure 6

```
AP Sports Menu:

General Scoreboard           Soccer
Baseball                     Ski Guide
Basketball                   Tennis & Golf
College Sports               Other
Football                     Exit
Hockey

SPORTS>Which Sport?
```

The reviews are in . . .

"'A Best Buy' I'm impressed"
David H. Ahl, Atari Explorer, Nov-Dec 1987

"If you've got an Atari, you probably need this program."
Jerry Pournell, Byte Magazine, October 1987

"pc-ditto is a winner."
Charlie Young, ST World, July 1987

"This is the product we have been looking for."
Donna Wesolowski, ST Informer, August 1987

"This truly incredible software emulator really works."
Mike Gibbons, Current Notes, September 1987

NOW! RUN THESE IBM PROGRAMS ON YOUR ATARI ST.

Lotus 1-2-3	Flight Simulator	Framework	Symphony
Enable	Ability	DESQview	Q&A
Sidekick	Superkey	Norton Utilities	dBase II, III, III+
Crosstalk IV	Carbon Copy	Chart-Master	Print Shop
EasyCAD	DAC Easy Accounting	BPI Accounting	Turbo Pascal
GW Basic	Managing Your Money	Silvia Porter's	pfs:Professional File

And Hundreds More!

---

pc-ditto is a software-only utility which taps the power of our Atari St to imitate an IBM PC XT. No extra hardware is required (an optional 5.25-inch drive may be required for 5.25-inch disks). All your IBM disks will work "out-of-the-box".

pc-ditto features include:

- All ST models supported (520, 1040, & Mega)
- up to 703K usable memory (1040 & Mega)
- not copy-protected -- installable on hard disk
- imitates IBM monochrome and IBM color graphics adapters
- access to hard disk, if hard disk used
- optionally boots DOS from hard disk
- parallel and serial ports fully supported
- supports 3.5-inch 720K format and 360K single-sided formats
- supports optional 5.25-inch 40-track drives

System requirements:

- IBM PC-DOS or Compaq MS-DOS version 3.2 or above recommended
- optional 5.25-inch drive is required to use 5.25-inch disks
- 3.5-inch 720K DOS disks require a double-sided drive (Atari SF314 or equivalent)

See pc-ditto today at an Atari dealer near you, or write for free information!

pc-ditto

by
Avant-Garde Systems
381 Pablo Point Drive
Jacksonville, FL 32225
(904) 221-2904

$89.95

Avant-Garde Systems, 381 Pablo Point Dr.
Jacksonville, Florida 32225 (904) 221-2904
Yes! Please send information on pc-ditto.
Name_____
Address_____
City_____ State_____ Zip_____

CIRCLE #105 ON READER SERVICE CARD.

MIDI Capture Program

by Robert Osness

Capture is a C-language program that will allow you to record, in your ST's memory, polyphonic output from your MIDI instrument. The stored data can then be displayed in tabular form and/or played back through your MIDI instrument.

The primary purpose of this program is to show how to get started in MIDI programming on the ST, providing a foundation on which to build. The concepts used in *Capture* can be readily expanded to add key transposition, tempo modification, note editing, disk storage for data files, and many other features that will come to mind as you explore the possibilities!

Those of you who are not interested in programming may skip ahead to the section titled "Run time!" where the program's operation is described. The rest of you, read on!

MIDI I/O

Data is transmitted on the MIDI bus in much the same way as the RS-232 serial data used for modem and printer interfaces. It is transmitted in bytes, at a rate of 31.25K baud. Handling I/O data for the MIDI ports is easy, thanks to the BIOS functions *Bconin()* and *Bconout()*. For input, we use *Bconin(device)*, where Device 2 is the keyboard, and Device 3 is the MIDI port.

Availability of data at the input can be checked using *x=Bconstat(device)* which returns 0 if no data is available.

For output, we can use *Bconout(device, data)*, where data is the byte to be transmitted.

Time-tagging

Capture uses the concept of time-tagging to save and recreate the time base of the MIDI input data. The incoming data is grouped into command message blocks, and stored with a time tag indicating when it was received. On playback, the command blocks are read and transmitted at intervals determined by the difference in time tag values.

Command blocks

What is a command message block? It's a sequence of one to three bytes of MIDI data that has a common function or purpose. A command block begins with a "status" byte, in which the most significant bit is always a logic one. It may be followed by zero, one or two data bytes, in which the most significant bit is always a logic zero. The most significant four bits (nibble) of the status byte identifies the function of the command message block. The least significant nibble is the MIDI channel number to which the message block belongs.

Table 1 identifies, by command block length and function, the status byte headers. Type 8, 9 and C status words are used for nearly all MIDI bus traffic. *Capture* is designed to handle all types of message blocks, with one exception: System Exclusive block transmissions. These are usually reserved for transmitting instrument-specific data files such as patch data.

NOTE ON and NOTE OFF Commands

By far the most common command blocks are the NOTE ON and NOTE OFF commands. A NOTE ON command might contain the following hexadecimal byte sequence:

```
0x92 NOTE ON, Channel 2
0x3C Pitch = Middle C
0x40 Velocity = 0x40
```

The hexadecimal value 92 identifies a NOTE ON command for Channel 2. Next, hexadecimal data byte 3C is a pitch (frequency) value corresponding to Middle C on the piano. Notes are assigned integer pitch values, increasing with frequency. There are 12 intervals in an octave, so adding or subtracting 12 (decimal) to a sequence of pitch values would raise or lower the pitch by one octave.

The last byte value of 0x40 indicates a turn-on velocity of 64, decimal. Actually, many keyboards don't use the velocity value at all, but simply use a fixed turn-on and turn-off velocity.

Now let's look at the NOTE OFF command. A typical NOTE OFF command block might appear as:

```
0x81 NOTE OFF, Channel 1
0x3C Pitch = Middle C
0x40 Velocity = 0x40
```

Here the bytes represent a NOTE OFF status word on Channel 1, a note pitch of 0x3C (Middle C again), and a turn-off velocity of 0x40.

Implicit NOTE OFF commands

A special case of the NOTE ON command allows it to be interpreted as a NOTE OFF command. This is true whenever the velocity value is zero: a zero-velocity NOTE ON command always equals NOTE OFF for the indicated pitch value.

Program change commands

"Program Change" is the MIDI term used for changing instrument voices, such as from piano to vibraphone. A typical PROGRAM CHANGE command block might be:

```
0xC1
0x02
```

This can be interpreted as change program on Channel 1 to Voice No. 2 (as defined in the MIDI instrument user's manual). For my Casio MT-540 keyboard, this causes a voice change to vibraphone for subsequent notes.

Running status

A modified command format is sometimes used in the MIDI command message structure, allowing the command blocks to be simplified. The general rule is that the status byte need not be transmitted if it is unchanged from the preceding command block. Combining this with the implicit NOTE OFF command allows

a significant reduction in the bus traffic, as nearly everything can be transmitted as a NOTE ON command. For example, consider the following sequence:

0x90 NOTE ON, Channel 0
0x3C Pitch = Middle C
0x40 Velocity = 0x40
0x3C IMPLICIT NOTE OFF, Pitch = Middle C, Channel 0
0x00 Velocity = 0 = note off
0x3D NOTE ON, Pitch = C#, Channel 0
0x40 Velocity = 0x40
0x3D IMPLICIT NOTE OFF, Pitch = C#, Channel 0
0x00 Velocity = 0 = note off

The software

The code for *Capture* is shown in Listing 1. The program is designed simply to identify the incoming command message blocks and store them sequentially in an array, along with a value for the current time.

Array elements are organized according to the structure *mdat*, which has four data slots: three message bytes and a longword for time. The three message byte slots are identified with the abbreviations *cm*, *fr* and *ve*, for command, frequency and velocity. The longword is identified as *ti* for time. It should be noted that the data contained in *fr* and *ve* will actually be frequency (pitch) and velocity only in the case of NOTE ON/OFF commands.

In theory, time tags should have an accuracy of about plus or minus one millisecond, which is approximately the time required to transmit a three-byte command block on the MIDI bus. Although this might be considered excessively conservative, *Capture* closely approaches this goal by using a millisecond timer and an efficient processing loop.

The timer

Timing is accomplished by the ST's 68901 MFP Timer, in conjunction with the assembly-language interrupt subroutine shown in Listing 2. The timer setup is done by the function *init_tmr()*. (A detailed description of the timer is not possible here, but will be the subject of a future article.) Because the timer is interrupt driven, the assembly-language interrupt subroutine must be separately assembled and linked with the C program code. Listings 3 and 4 provide batch command files that will accomplish this with the Batch and Alcyon Compiler/Linker programs supplied with the Atari Developers Kit.

Collecting MIDI data

The main processing loop for *Capture* begins by clearing the input data array and MIDI buffer, then starts the millisecond timer. Next, a *for* statement defines the input data loop where MIDI data is collected and processed.

A second *for* loop waits for input data, which can be either a MIDI byte or a *q* from the keyboard to signal the end of MIDI input. Once a data byte has been received, it is stored in one of three temporary storage cells: *b0*, *b1* or *b2*. If it can be identified as a status byte it goes into *b0*. Otherwise, data bytes are successively stored in *b1* and *b2*, using *bnxt* to keep track. The program logic is designed to correctly handle input data in either the standard or running status format.

The last step in the input data loop is to call the function *end_c()*. This function uses a *switch* statement to evaluate the current status byte, and then uses *bnxt* to determine whether the complete command message block has been received. If so, the complete command block is stored as an array element, using the function *save()*. The millisecond timer value is also stored in the array at this time.

The array entries are in the standard MIDI command format, even if the data was received in the running status format. This is desirable to simplify any further processing of the array data that may be undertaken.

Data entry into the array continues until either the array is filled or a *q* is entered from the ST's keyboard. If the array size of 2,000 entries (about 1,000 notes) is insufficient, the value of NMAX can be increased to any size desired.

Displaying the MIDI data

With the data safely stashed away in memory, it might be nice to see what it looks like. The user is prompted with the message "Display input data? (Y/N)." Typing a *y* causes the console to display the first ten array entries, with the MIDI bytes displayed in hexadecimal and the time in decimal milliseconds. Striking any key other than *q* will cause the next ten entries to be displayed. The display loop can be aborted by typing a *q* at any time.

Playback

Next the prompt message "Play it back? (Y/N)" is displayed. Entering a *y* starts the playback process, complete with any pauses in data entry. The function *pbk()* simply steps through the array and outputs the MIDI message command blocks, separating them by a time interval equal to the difference in time tags. Though the messages are not transmitted in running status mode, the standard format commands are equally acceptable to most MIDI instruments, and there should be no discernible difference.

Upon completion of the output sequence, the program exits to the desktop. Should it be desirable to save the array permanently, a disk storage option can be readily added at the end of the main program loop.

Building the program

Those who have the program disk for this issue may proceed with a smile to the next section, and run the program. All others, let's get the grungy work over!

Type in the program as shown in Listing 1. Omit the comments at your own

TABLE 1

TABLE I. MIDI STATUS BYTES

STATUS BYTE VALUE	NUMBER OF BYTES / BLOCK	FUNCTION
8n	3	NOTE OFF
9n	3	NOTE ON
An	3	POLY-KEY PRESSURE / AFTERTOUCH
Bn	3	CONTROL CHANGE / CHANNEL MODE
Cn	2	PROGRAM CHANGE
Dn	2	CHANNEL PRESSURE / AFTERTOUCH
En	3	PITCH BEND CHANGE
Fx	1 *	SYSTEM

* EXCEPT SYSTEM EXCLUSIVE BLOCK MESSAGES

n = CHANNEL NUMBER

x = QUALIFIER BITS

risk: They sure make it easier to understand later!

The following explanation assumes the use of the Alcyon Compiler/Linker, with compiler and linker files on a common disk in Drive A. Batch files, if used, should be moved to the compiler/linker disk. The program files are assumed to be on a separate disk in Drive B. If you prefer other arrangements, make adjustments accordingly.

Type in the interrupt subroutine INTR.S, from Listing 2. Assemble it using the batch file BI2.BAT from Listing 3, using the command BI2 B:INTR.

Now compile and link the two code segments using the batch file BINT.BAT shown in Listing 4, and the command BINT B:CAPTURE.

Run time!

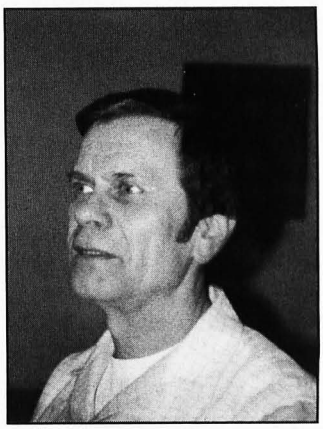
Now for the fun stuff! Hook up the MIDI cables, and double-click on CAPTURE.PRG. When the program prompts you, start playing your MIDI instrument; the data it will send through the MIDI cables will be stored in your ST's memory. (Even if you don't start playing immediately, the timer is going. Any "silence" will be recorded just as if it were musical data.)

When you're through playing, press the *q* key, and you will be asked whether you wish to view the MIDI data. If you answer "y," the data will be displayed ten bytes at a time. Pressing any key except *q* will display the next ten bytes.

Once all the data has been displayed, or you press *q* to abort the data display function, you will be asked if you wish to play back the data. Pressing *y* at this point will cause the stored MIDI data to be sent back to your instrument, after which the program will return to the desktop.

Conclusion

Keep in mind that your MIDI performance will be reproduced in precise detail, including all your blunders! Good luck, and have fun developing new and improved processing routines for your MIDI files! ■



Bob Osness, who has been programming his ST for 2 ½ years, works as an electrical engineer for Boeing Aerospace in Kent, Washington. He and his wife, Georgia, spoil grandchildren as a hobby.

MIDI CAPTURE Listing 1: C

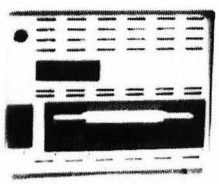
```
#define NMAX   2000            /* MIDI storage array size */

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

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

extern long timcnt;        /* msec timer value */
extern int  intr();        /* msec interrupt service routine vector */

struct    mdat {
     char cm;        /* command */
     char fr;        /* pitch/frequency */
     char ve;        /* velocity */
     long ti;        /* time */
} dummy;

struct    mdat      store[NMAX];   /* raw data array*/
struct    mdat      *p;

char b0, b1, b2;
char bnxt;


main()
{
     char k;
     int  x;
     int   i, exit;


     clr_struct(store);        /* clear input data array */

     clr_midi();               /* clear MIDI input buffer */

     init_tmr();               /* initialize & start msec timer */

     b0 = 0x90;                /* set default start conditions */
     bnxt = 0;

     printf("\nBegin MIDI input\n");
     p = store;                          /* initialize pointer */

     for( i=0; i<NMAX; i++ ) {           /* input data loop */

          for( x=0, exit=0; ; ) {        /* input wait loop */
```

Astra gives up !!!

No we are not going out of business...
We are going out for business !!!

The demise of many Atari dealers, and the reluctance of many more to carry much inventory forces us to announce:

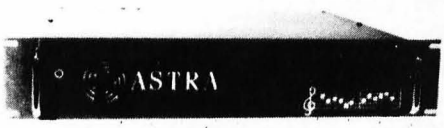
Astra Systems goes Factory Direct !



SYSTEM HD+

Our original offering for the ST computers is now available in 20, 30, and 40 megabyte capacities. With precision double sided 3 1/2" microfloppy and its tough construction it ranks as the one to beat.

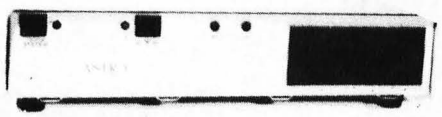
SYSTEM HD+ 20 $849.95
SYSTEM HD+ 30 $919.95
SYSTEM HD+ 40 $1069.95



RM 60/120 & RM 80/160

Our rack mount units for the MIDI musician fit portable and permanent racks.

RM 60	$1295.00
RM 60/120	$2095.00
RM 80	$1995.00
RM 80/160	$3495.00
EXPANSION 60	$ 875.00
EXPANSION 80	$1595.00



THE EXPANDER

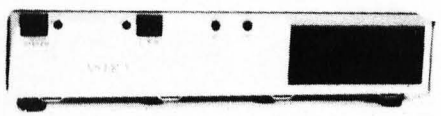
A truly expandable hard disk system. Not a chainable one as many competitors are offering. Contains space, cooling, and sufficient power supply to handle up to 120 megabytes of storage inside the under-monitor case. Built-in surge protection for its 4 AC outlets. Controls the CPU and peripherals from the front of the unit. Optional precision double sided 3 1/2" microfloppy. Twenty, thirty, and forty meg units.

EXPANDER 20 $759.95
EXPANDER 30 $819.95
EXPANDER 40 $999.00

VISA AND MASTERCARD ACCEPTED

MASTERCARD VISA

CALL (714) 549-2141
FAX (714) 546-1612
BBS (714) 546-5956



PRO MODEL

Four AC outlets control CPU and peripherals from front of unit. Available in 40, 60, and 80 megabyte units. Attractive under-monitor case. Built tough to take it !

PRO MODEL 40 $899,95
PRO MODEL 60 $999.95
PRO MODEL 80 $1499.95

SW2 MONITOR SWITCH

Switches both the power and the signal ! True one button switching. Faster and safer. Audio out and composite out (FM models only). Call for information on other exciting MODULAR units.

SW2 $50.00

Southern California Astra Systems Inc.
2500-L So. Fairview, Santa Ana, CA. 92704-9869

CIRCLE #106 ON READER SERVICE CARD.

```
            if(Bconstat(2)) {          /* test for 'q' exit */
                x = Bconin(2);
                if(x=='q' || x=='Q') {
                    exit = 1;
                    break;
                }
            }
            if(Bconstat(3)) {          /* get MIDI input byte */
                k = Bconin(3);
                break;
            }
        }

        if(exit)
            break;                  /* exit from input data loop */


        if(k & 0x80) {              /* store current input byte */
            b0 = k;
            bnxt = 1;
        }
        else if(bnxt == 0 || bnxt == 1) {
            b1 = k;
            bnxt = 2;
        }
        else if(bnxt == 2) {
            b2 = k;
            bnxt = 0;
        }

        end_c();                /* if end of cmd block,
                                   save cmds in array */
    }  /* end data input loop */

        printf("\nDisplay input data? (Y/N) ");
        x = Bconin(2);
        printf("\n");
        if(( x == 'y') || ( x == 'Y'))
            prt_array(store);               /***  display arrays ***/

        printf("\nPlay it back? (Y/N) ");
        x = Bconin(2);
        printf("\n");
        if(( x == 'y') || ( x == 'Y'))
            pbk(store);

        Jdisint(13);                /* disable timer a interrupt:
                                       required before exit!!     */
}   /* end main */

end_c()                 /* test for end of cmd block; store data */
{
        int  end_flag;

        end_flag = 0;


        switch(b0 & 0xF0) {         /* (switch uses signed int) */

            case 0xF0:                          /* 1-byte commands: */
                        save(b0, 0, 0, timcnt);
                        end_flag = 1;
                        break;
            case 0xD0:                          /* 2-byte commands: */
            case 0xC0:
                        if(bnxt == 2) {
                            save(b0, b1, 0, timcnt);
                            end_flag = 1;
                        }
                        if(bnxt == 0) {
                            save(b0, b2, 0, timcnt);
                            end_flag = 1;
                        }
                        break;

            case 0x90:              /* 3-byte command: special case */

                        if(bnxt == 0 && b2 == 0) {
                            save(b0 & 0x8F, b1, b2, timcnt);
                            end_flag = 1;
                            break;
                        }
            case 0x80:                          /* 3-byte commands: */
            case 0xE0:
```

```
            case 0xB0:
            case 0xA0:
                        if(bnxt == 0) {
                             save(b0, b1, b2, timcnt);
                             end_flag = 1;
                        }
                        break;

      }  /* end switch */

      return(end_flag);        /* returns 1 if end of cmd block */
}


save(c,f,v,t)       /* store array element & increment pointer */
char c,f,v;
long t;
{
      p->cm = c;
      p->fr = f;
      p->ve = v;
      p->ti = t;
      p++;
}

init_tmr()             /* Initialize Timer A */
{
      timcnt = 0;

      Jdisint(13);                  /* disable timer a interrupt */

      Xbtimer(0x00, 0x04, 49, intr);   /*   set up timer A:
                                       0x04 = prescale divide by 50
                                         49 = count down value

                                       intr = interrupt vector
                                              (ISR address)     */

      Jenabint(13);                 /* enable timer A interrupt */

}


clr_struct(b)              /* clear structure */
char  *b;
{
      int  a, i;

      a = NMAX * (sizeof(dummy));
      for(i=0; i<a; i++)
            *b++ = 0;
}


clr_midi()     /* clear MIDI input buffer */
{
      while(Bconstat(3))
            Bconin(3);
}


prt_array(pb)          /* display the array contents */
struct mdat  *pb;
{
      char x;
      int  i, j;

      for(i=0; pb->cm && i<NMAX; i++) {
            printf("cmd   freq  vel   time\n");
            printf("---   ----  ---   ----\n");
            for(j=0; j<10; i++, j++) {
                  printf("%02x    ",    (int) 0xFF & pb->cm );
                  printf("%02x    ",    pb->fr );
                  printf("%02x    ",    pb->ve );
                  printf("%Ld \n",      pb->ti );
                  pb++;
            }
            x = Bconin(2);
            if(x=='q' || x=='Q')
                  break;
      }
}
```

```
pbk(q)                              /* play back the array contents */
struct  mdat  *q;
{
     int  i;
     long t1, t2, t_old;

     for(i=0;  q->cm && i<NMAX;  i++ ) {

          t1 = timcnt;
          if(i) {
               do {                                   /* output timing wait loop */
                    t2 = timcnt;
               } while ((t2 - t1) < (q->ti - t_old));
          }
          switch(q->cm & 0xF0) {
               case 0xF0:                                   /* 1-byte cmds */
                                   Bconout(3, q->cm);
                                   break;
               case 0xD0:                                   /* 2-byte cmds */
               case 0xC0:
                                   Bconout(3, q->cm);
                                   Bconout(3, q->fr);
                                   break;

               case 0xE0:                                   /* 3-byte cmds */
               case 0xB0:
               case 0xA0:
               case 0x90:
               case 0x80:
                                   Bconout(3, q->cm);
                                   Bconout(3, q->fr);
                                   Bconout(3, q->ve);
                                   break;

          }  /* end switch */

           t_old = q->ti;           /* save time tag */
           q++;

      }  /* end for */
}
```

MIDI CAPTURE
Listing 2:
Assembly

```
.globl _timcnt
.globl _intr
.text
_intr:
addq.l #$1,_timcnt
andi.b #$DF,$fffa0f
rte
.data
.even
_timcnt: .dc.L 0
```

MIDI CAPTURE
Listing 3:
Batch file

```
as68 -F b: -l -u %1.s
wait
```

MIDI CAPTURE
Listing 4:
Batch file

```
cp68 %1.c %1.i
c068 %1.i %1.1 %1.2 %1.3 -f
rm %1.i
c168 %1.1 %1.2 %1.s
rm %1.1
rm %1.2
as68 -F b: -l -u %1.s
link68 [u] %1.68k=gemstart,%1,b:intr,gemlib,osbind,libf
relmod %1
rm %1.68K
wait
```

PD Parade

by George L. Smyth

The better portion of my college years were spent studying music theory and composition. For this reason I am gratified and impressed with the music software available for the Atari ST computers. Sequencers, sound digitizers and other types of software unknown to me a couple of decades ago have opened a whole new world of possibilities to me.

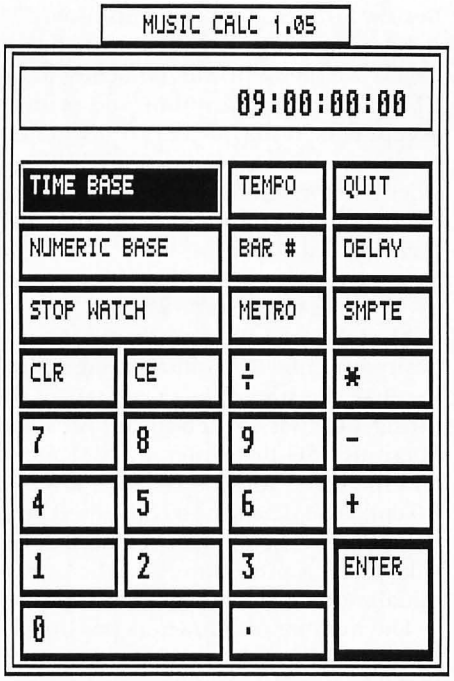
Learning to control the possibilities is another matter. Fortunately, help has come from Steve MacMillan in the form of *Music Calc*.

Music Calc V1.05, a program which runs in medium resolution, is composed of three elements: a numerical calculator, a time-based calculator and a calculator for determining position within a musical piece.

The numerical calculator, the simplest of the program's elements, provides the four basic functions: addition, subtraction, multiplication and division. The values are entered, the symbols are chosen, and the answer is returned. No bells or whistles here, this is simply a tool that has been provided as support for the other functions.

The time-based calculator can be a great help when trying to determine the length of an album side or which songs, based on their length, will fit on a 45-minute cassette. Time units are entered on the keyboard and can be added or subtracted. The time values can also be multiplied and divided by real numbers, the results of the manipulations displayed in time units. If you want help adding something like four minutes, 37 seconds, nine minutes, 46 seconds and six minutes 58 seconds, this function will prove handy.

The meat of *Music Calc* can be found in the extensive third portion of the program. The selection of the "Tempo" button allows the user to determine the speed of a song. A number of time signatures are available, in addition to the ability to enter one not displayed. To calculate the time setting, the TAP box is chosen. Pressing the spacebar begins the the real time which corresponds to the musical timing. In the example mentioned, I noted that the half note took 1005.03 milliseconds, the eighth note 251.26 milliseconds and an eighth note triplet 335.01 milliseconds. Time for four bars was determined to be 8.04 seconds. The "Bar#" screen helps determine the position within a song. The two modes associated with this choice are called "Calc Bar" and "Calc Time." Choosing the former, a selection of bars or beats is made. The bar or beat number is then entered and, as signified by the formerly computed timing, and the amount of time into the piece is then calculated. If the Calc Time option is chosen the number of bars or beats into the song is determined.



Two other handy features are the stopwatch and metronome. The stopwatch timing, and tapping the spacebar 16 more times to the beat of the music gives the program the information necessary to calculate the musical speed. This done, the calculated timing will be displayed and can be used in other sections of the program. While testing this feature, I determined that a section of Brahms Symphony #4, which had the quarter note written to equal 120 in 4/4 time, was 119.3 in the recording I was playing (very close). The "Delay" window can then calculate claims accuracy to $^1/_{100}$ of a second. Checking it with my stopwatch, I found it to be at least this accurate. The metronome feature clicks at the rate selected on the "Tempo" screen. Trying different settings allows the composer to hear what the written time settings are like, i.e., the difference between the quarter note equaling 112 and or equaling 120. Of course this can also be used to keep steady time when practicing.

The SMPTE screen allows the user to alter the default frame rate of 30 to 24 or 25 frames per second, the former being the film standard and the latter being the European EBU standard. Thirty frames per second is useful in nearly all audio productions.

When leaving the program, the user has the opportunity to save the tempo, time signature, start time and SMPTE frame rate to be used as default values the next time the program is booted. This is helpful to the user who is working with synchronization of a film and musical score.

Steve MacMillan has added a useful program to the library of ST public-domain software for musicians with Atari STs. Even if you have no immediate use for *Music Calc*, make sure that you save a copy from this month's disk. One never knows when the music bug will strike.▪

George L. Smyth has a degree in psychology from West Virginia University and is currently employed as a programmer. He is the author of a series of tutorials on programming in FORTH.

GFA BASIC SOURCE CODE PRINTER OR COLUMN-AID

by Travis E. Guy

GFA BASIC Source Code Printer (*SCP* for short) is a simple utility I wrote after first seeing the lengths of some of the listings made by GFA BASIC. Since GFA BASIC programs are restricted to one command per line, their listings tend to be narrow and long, hugging the left-hand side of the printout. A tad wasteful, it would seem.

Why not put all that clean white space on the right-hand side of the paper to use? I wondered. Saving paper has both an economical and an ecological angle to it. "No doubt about it," my programming alter ego said, "this calls for a program. *SCP!*"

SCP is quite straightforward, and fairly modular in design but not overly so. All it does is read in lines from GFA BASIC source code files (the ones listed as ASCII files, with .LST extensions) from disk, store them in a string array and print them out in a one-, two- or three-column format. And at no extra charge, it'll even do it in up to any one of three type sizes. No whiz-bang displays here—*SCP*'s not graphics dependant—so it should work in all resolutions. Just load the GFA BASIC interpreter, type in the listing, save it to disk, then select "Run" from the GFA menu bar.

What SCP does and what you should do

After a few strings are defined and cheerful greeting extended, a file-selector box asks you for the name of the file you wish to print. *SCP* checks for three things here:

1) that you did enter a filename,

2) that the file requested is on the drive specified,

3) that the file has a .LST extension.

No other checks are made on the validity of the file. SCP *cannot tell if it is being asked to print a non-text file.* Any attempts to make it do so may cause grievous trouble to befall you and your loved ones. The best that could happen is that you'll end up with pages filled with gibberish. So please don't try it, okay?

Next, the program calls a procedure to check whether a printer (attached to the ST's parallel port) is online and ready to accept data. If not, an alert box will notify you to check connections and ask whether to recheck or abort the listing. Some housekeeping chores are then performed, and its off to. . .

Question time

Alert boxes will prompt you for the desired number of columns and, following that, the size of the characters of the listing. *SCP* was written to run on a Star Micronics SG-10 printer in IBM mode, and the three size choices here are pica (10 cpi), elite (12 cpi) and condensed print (17 cpi). The size of each column is calculated via a procedure, and the correct initialization codes are sent to the printer.

The number of characters per line for the different type sizes used at this point in the program reflect a ten-inch printer carriage. For those of you with carriages of different sizes, change the values assigned to *Length%* in all three occurrences.

If you are using single-sheet paper instead of fanfold, you may opt to pause the listing between pages. Saying yes sends a command to the printer to ignore out-of-paper signals. This frees you from pesky bells, beeps, chirps and whatever else printers do when they unexpectedly run out of paper.

Some final fiddling bits are then taken care of before the program starts earning its keep. The string array that holds each page of the listing, *Page$( )*, is dimensioned, counting variables are set, and the program tells the world it is going to work on the first page. For those fascinated with GEMs, the mouse pointer is changed to the Busy Bee.

The heart of the matter

Now you've arrived at *SCP*'s main section; a three-function loop that should be no trouble to follow. Simply put, it reads in lines from the file, chops them up if needed to fit the columns and calls a procedure to print the array. This cycle continues for each page of the listing, until the end of file is reached.

To avoid confusion over where each program line ends, *SCP* appends an end-of-line delimiter (a character that isn't normally entered from the keyboard) to show exactly where the carriage return falls. In the same vein, the columns are bracketed with marks to indicate where each column line begins and ends.

When the main loop fails its end-of-file check, the printing procedure is called one last time, and then control jumps to a clean-up routine. A reset code is sent to the printer, the file is closed, and you are asked if you want to print anything else. If not, *SCP* comes to a gracious halt.

Watch it flex

SCP doesn't require any external printer driver, but does rely on certain assumptions. It is set up for 66 lines per page, out of which it uses 60 lines for the listing, allowing it to skip perforations. However these figures can be easily changed; *SCP* is very flexible when it comes to printers.

The printer codes to change type size, to reset the printer when finished, to switch off out-of-paper detection and to choose the characters used for carriage return marks and column separators are all located at the beginning of the program, and can be altered to suit individual needs and tastes. And, yes, even the number of columns and lines and the width of each column can be changed without much difficulty. I have used descriptive variable names throughout *SCP*, as well as descriptive labels, procedure names and the odd comment or two, in an attempt to try and convey a sense of what is going on.

Writing *SCP* was a breeze with GFA BASIC. Hopefully, you will find the program useful and as easy to use as it was to create. ■

GFA SOURCE CODE
Listing 1:
GFA BASIC 2.0

```
' IMPORTANT NOTE: THE "~" CHARACTER AT THE END OF SOME OF THE LINES
'   IN THIS LISTING INDICATE THAT THE LINE WRAPS AROUND TO THE
'   NEXT LINE OF THE LISTING. THE TWO LINES SHOULD BE TYPED AS ONE.
'
'
'      GFA BASIC Source Code Printer
'      Prints GFA BASIC ASCII files
'          in multiple columns
'            and text sizes.
'
' ** WARNING Non-printing characters **
'   ** will skew column alignment. **
'
' Let's define some strings and say hello.
' Note that Star SG-10 (IBM #2 mode) codes are used.
'
Column$=Chr$(240)              !Column dividing mark.
Cr$=Chr$(174)                  !Carriage return mark.
Reset$=Chr$(27)+Chr$(64)       !Initalizes printer.
Pica$=Chr$(27)+Chr$(80)        !10 cpi mode.
Elite$=Chr$(27)+Chr$(77)       !12 cpi mode.
Condensed$=Chr$(27)+Chr$(15)   !17 cpi mode.
Nopaperout$=Chr$(27)+Chr$(56) !Ignore paper-out signal.
'
Alert 1,"GFA BASIC Source Code Printer|      by Travis E. Guy ~
|         for ST-Log",1," Okay ",Dummy%
'
Pick_a_file:                        !Program starts here.
Print "Select file for printing" !A prompt.
Fileselect "\*.LST","",Filename$ !How easy can it get?
Cls                                 !Kills prompt.
'
' CANCEL or an empty OK pressed.
'
If Right$(Filename$,1)="\" Or Len(Filename$)=0 !Searches for a trailing
  Alert 3,"No file selected",1," Quit ",Dummy% !folder, or a null entry.
  End
Endif
'
' If a filename is entered, but not on the disk.
'
If Not Exist(Filename$) !Oops.
  Alert 1," File not found ",1," Retry | Quit ",Dummy%
  If Dummy%=1
    Goto Pick_a_file !One more try.
  Else
    End              !Forget it.
  Endif
Endif
'
' Not a valid GFA BASIC ASCII file extension.
'
If Right$(Filename$,4)<>".LST"
  Alert 3,"ERROR!|File does not have a .LST|extension.  May not be|~
a GFA BASIC ASCII file",1,"Go check",Dummy%
  End
Endif
'
Gosub Check_printer
Gosub Strip_folders(Filename$)
Open "i",#1,Filename$
'
' How many columns do you want.
'
Alert 2,"Number of columns|    per page",2," 1 | 2 | 3 ",Numcol%
'
' Ask for desired script size, tell the printer about it,
' and provide the size of the soon-to-be printed lines.
'
Alert 2,"    What script size for       |    file: "+File$+"    ",~
3," 10 cpi | 12 cpi | 17 cpi ",Dummy%
If Dummy%=1
  Lprint Pica$;       !10 characters per inch
  Length%=80          !on an eight inch carriage.
Endif
If Dummy%=2
  Lprint Elite$;      !12 cpi.
  Length%=96
Endif
If Dummy%=3
  Lprint Condensed$; !17 cpi.
  Length%=136
Endif
Gosub Size_column(Length%) !Calculate size for each column.
'
```

```
' And would you like a break from time to time.
'
Alert 2,"Pause between pages",2," Yes | No ",Pause%
If Pause%=1
  Lprint Nopaperout$; !Tell the printer.
Endif
'
' Initialize variables and start working.
'
Dim Page$(3,60) !3 column x 60 line array holds each page.
Whichline%=1    !Line-in-column counter.
Column%=1       !Column-on-page counter.
Page%=1         !Page-in-listing counter.
Print "Working on page ";Page%
Defmouse 2      !Looks better than, "Please Wait!"
'
' The main loop.
'
While Not Eof(#1)
  Line Input #1,Grabbed$ !Grab a line.
  While Len(Grabbed$)>Sizeline%-1       !If it is longer than the column
    Thisline$=Left$(Grabbed$,Sizeline%) !hack off a column's worth.
    Grabbed$=Mid$(Grabbed$,Sizeline%+1,Len(Grabbed$)-Sizeline%) !Remove it.
    Gosub Page_fill       !Insert it into the array.
  Wend                    !At end-of-line
  Thisline$=Grabbed$+Cr$ !tack on a mark
  Gosub Page_fill         !and send it to the array.
Wend                      !At end-of-file
Gosub Page_print          !print the last page
Goto E_o_f                !and close up shop.
'
Procedure Page_fill
  Page$(Column%,Whichline%)=Space$(Sizeline%) !Pad out current element.
  Lset Page$(Column%,Whichline%)=Thisline$    !Left-justify current line.
  Inc Whichline%          !Next line.
  If Whichline%=61        !If at bottom of page
    Whichline%=1          !reset line count
    Inc Column%           !and go to next column.
    If Column%>Numcol%    !If column count too high
      Gosub Page_print    !print the page.
      If Not Eof(#1)      !This check IS necessary.
        Inc Page%         !Next page.
        Print At(17,1);Page% !Print count.
        If Pause%=1       !If pause-between-pages set.
          Alert 1,"Paused... ",1,"Continue",Dummy%
        Endif
        Defmouse 2        !Bring back the Bee.
      Endif
    Endif
  Endif
Return
'
Procedure Page_print
  If Len(Page$(1,1))>0  !Print only if there is something to print.
    Lprint
    Lprint
    Lprint "Listing: ";File$;      !Print name of file.
    Lprint Space$(Length%/2-2-Lpos(0));"-";Page%;"-" !Center page number.
    Lprint
    For Whichline%=1 To 60         !60 lines per page.
      If Len(Page$(1,Whichline%)) !If first column not empty
        Lprint Column$;" ";Page$(1,Whichline%);" ";Column$; !print it.
      Endif
      If Len(Page$(2,Whichline%)) !Ditto for second column.
        Lprint " ";Page$(2,Whichline%);" ";Column$;
      Endif
      If Len(Page$(3,Whichline%)) !Likewise.
        Lprint " ";Page$(3,Whichline%);" ";Column$
      Else
        Lprint       !If no second or third column this
      Endif          !lprint returns printhead.
    Next Whichline% !Do one more line.
    Lprint
    Lprint          !Blank lines for bottom of page.
    Erase Page$()
    Dim Page$(3,60) !Clears and reDIMs page-keeping array.
    Column%=1
    Whichline%=1    !Reset line & column counts.
  Endif
Return
'
E_o_f:
Cls
Lprint Reset$;      !Reset printer.
Close #1            !Close file.
```

```
Alert 2,"Anything else to print?",2," Yes | No ",Dummy%
If Dummy%=1          !If "yes"
  Erase Page$()      !erase array for redimensioning
  Goto Pick_a_file   !and start over.
Else
  End                !If "no", a glorious end.
Endif
'
Procedure Check_printer
  Holding_pattern:
  If Not Out?(0)              !If something is wrong
    Alert 1,"Check printer & connections",1," Retry | Quit ",Dummy%
    If Dummy%=1               !let user try again.
      Goto Holding_pattern    !Recheck.
    Else                      !Forget it.
      End                     !Forgotten.
    Endif
  Endif
Return
'
Procedure Strip_folders(Filepath$)
  Repeat
    Dummy%=Instr(0,Filepath$,"\") !Found a backslash.
    Filepath$=Right$(Filepath$,Len(Filepath$)-Dummy%) ~
!Zap everything to its left.
  Until Dummy%=0  !Repeat till all gone.
  File$=Filepath$ !Only the filename remains.
Return
'
Procedure Size_column(Lengthline%)
  Sub Lengthline%,4+((Numcol%-1)*3) !Ignore column brackets.
  Div Lengthline%,Numcol% !Divide by number of columns.
  Sizeline%=Lengthline%   !Ta-da.
Return
```

■ **END**

ST USER

BY ARTHUR LEYENBERGER

I think I am like most ST users. When I get a new piece of software, I prefer to jump right into the program and attempt to figure it out. Rarely do I first read the manual except for the installation procedure. This is one way to test the design and operation of a program. However, major application programs do require you to read the documentation. For these, I typically read the manual cover to cover, then refer to it as I need to during the use of the program. Still, little things can often be overlooked.

Last month in this space I contrasted two operating systems used on PCs and compatibles: GEM and *Windows*. I stated that I preferred GEM to *Windows* because GEM is a more finished and complete graphical interface. Unfortunately, I made an error, and in the spirit of software truth, justice and the *ST User* way, I'd like to make a correction.

I correctly mentioned that to change directories (what GEM calls folders) in *Windows*, you click on the directory name and the computer goes to that directory and displays the list of files contained in it. However, I erroneously stated that to return to the parent directory, *Windows* requires you to type in its name, whereas GEM lets you click on the close button to move you back to preceding directory.

Windows does let you return to the preceding directory by pointing and clicking with the mouse. To do so, you point to the name of the directory you want to return to in the displayed path. In fact, *Windows* betters GEM in this regard.

GEM will let you return to the immediate parent directory by clicking on the "Window Close" button. But you can only return one level. With *Windows* you can point to any directory name in the path, allowing you to jump to any higher level directory, including the root directory itself. Being able to move one or more directory levels up is a nice touch.

Although I am a heavy *Windows* user, it wasn't until I was searching the manual for some other information that I happened across this tidbit. Now that I know about this feature, I often use it and appreciate its usefulness. I still prefer GEM in terms of speed and simplicity of use, but I wanted to set the record straight.

Casio strikes again

I like Casio. Through the use of their inexpensive but capable musical instruments, I have been able to learn about and use the MIDI ports on my ST. I first became aware of Casio's MIDI keyboards when I saw and later purchased a mini-keyboard synthesizer called the CZ-101. Later, I upgraded to their CZ-5000 full-size keyboard with sequencer. I have had this keyboard for several years and still enjoy using it alone and with my ST.

Casio's latest musical instrument is the Digital Horn (DH-100). Smaller than an alto saxophone and made out of gray plastic, the DH-100 is as fun to play as Casio's MIDI keyboards. The DH-100 comes with built-in sounds of a saxophone, trumpet, oboe, clarinet, flute and synth-reed and uses recorder-type fingering so learning is easy.

You can use either the keys like on/off switches or blow into the mouthpiece for dynamic control over volume and tone. The DH-100 is battery-operated, contains its own built-in speaker and has a two-octave chromatic scale. It also has a MIDI-out connection that can be used to control other MIDI instruments or a MIDI program.

The Casio DH-100 Digital horn is both easy to learn and fun to play. The street price of the instrument is about $100 although it retails for $150.

Game time

It's been a while since I took a break and spent some serious game-playing time on the ST. In that time, a dozen or so new titles have been piling up around here, and it's time I took a look at them. Of the new titles, a few appear to be especially interesting.

I've always been partial to golf games. Not that I play *real* golf—I tried it once and found out I'd be better off staying with the video golf games. That's why when *Zany Golf* from Electronic Arts arrived recently, I couldn't wait to play it.

I was disappointed when I first opened the package and booted up the game. *Zany Golf* is not a traditional video golf game in the sense that you try to get your best score on each of the nine or 18 holes. Instead, *Zany Golf* is more like an arcade game. You must perform well on each hole in order to move to the next hole.

At first I couldn't even get beyond the first hole. My poor performance and resulting frustration in not advancing in the game almost led me to completely dismiss *Zany Golf* and not bother mentioning it. However, I kept at it, hole by hole, and once I realized that it was more arcade game than simulation, I started to enjoy it. My success at advancing through the game also helped.

The best way to describe *Zany Golf* is to call it a fantasy miniature-golf game. Like traditional miniature golf, there are only nine holes and each hole presents a particular obstacle to overcome. What makes it a fantasy course is many of the obstacles. Bouncing hamburgers, pinball machines and flying carpets are rarely found in the real world.

Game play is straightforward. At each hole you are presented with a preview with which you can scroll around the hole by moving the mouse. Then, you place the cursor on the ball, hold down the button and drag the mouse in the opposite direction you want the ball to go—kind of like using a pool cue and drawing it backward, especially since the farther you pull the mouse back the harder you putt. Releasing the mouse button putts the ball.

As mentioned before, you must do well on the hole to go on to the next hole. Each hole has a par value and a certain number of strokes are allowed on each hole. After you get the ball in the hole, any remaining strokes are carried over to the next hole. When you use up all of your remaining strokes the game is over.

There are a couple of things that could be better in this game. One is the time required to load each hole from disk. Although it takes about 25 seconds, it seems like forever. Another is the fact that you cannot put the contents of the two disks on your hard disk. This would speed up the game and avoid the need to insert

the second disk halfway through the game.

Zany Golf was created by Will Harvey and his Sandcastle design team. The music was written by Doug Fulton, the animation created by Ian Gooding and Jim Nitchals did the ST translation. In spite of the small improvements I have suggested, it is fine effort, providing plenty of challenge to golfers and nongolfers alike.

Zany Golf sells for $40 and is available wherever fine ST games are sold. You can also order directly from Electronic Arts at 1820 Gateway Drive, San Mateo, California 94404; (or call 800-245-4525 Monday through Friday during normal business hours).

The other game that caught my eye was *Tower Toppler* from Epyx (600 Galveston Drive, Redwood City, California 94063, 415-368-3200). It seems that you have been sent to the planet Nebulus where eight giant towers rise from the depths of a toxic ocean. The goal is to reach the top of each of eight dark and deadly towers, each of which is more difficult to get to than the last. Once at the top of each tower you must destroy it by setting off special destruction mechanisms.

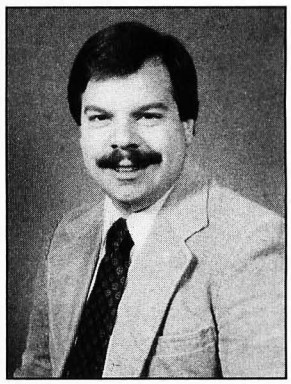
Each tower has its own set of surprises and obstacles. They are guarded by deadly rolling boulders, indestructible mutant molecules, flying phantoms, flashing blockades and other things too gruesome or nasty to mention. If you touch one of the rolling boulders you'll be bumped off the ledge and sent sailing down to sea level. If you are lucky, another ledge might catch your fall. Unless it is one of those disintegrating ledges that disappear the moment you land on them.

Your main weapon is a snowball gun that can be used to freeze or destroy enemies. In your quest for the top, you can ride emergency elevator lifts or hop over some of the obstacles. However, there's a twist. As you climb these colossal columns, they rotate with you so your vantage point changes. This constant change in view makes the action even more exciting.

There is a time limit for you to reach the top of a tower. Failing to do so ends the game. Also, you can gain extra points by catching fish between the towers. One or two players can play using joysticks, and there are two missions to choose.

Tower Toppler is one of the US Gold games from Epyx. It sells for $40 and although it has been out for a while, it is worth your consideration. The graphics and 3-D effect in this game are excellent. The music is also fun but after a while you may want to turn it off.

Tower Toppler is an interesting take-off on the "hopping game" concept. Much like one of the first games in this genre, *Q-Bert*, the game concept is simple: get to the top and avoid the monsters. But it is easier said than done, and that is where the challenge lies. *Tower Toppler* provides stunning graphics, challenging game play and the desire to play "just one more round." What more could you want from a game for your ST? ■



Arthur Leyenberger is a computer journalist and freelance writer living in beautiful New Jersey. He can be reached on CompuServe at 71266,46 or on DELPHI as ARTL.

the Animati Boardi

BY MAURICE MOLYNEAUX

While proofreading last month's installment I noticed that it wasn't the straightforward discussion of storyboarding I'd planned. I thought, *You haven't discussed presentation, visual indicators for showing movement, discussed how to link dialogue or anything! You keep coming back to ideas and refining concepts! Your focus is wandering.*

After this bout of self-abuse I started looking to correct these failings, but ended up not changing anything. Why? As I edited I realized that segregating the topic and techniques of storyboarding from the rest of the process would be an arbitrary division and not reflect the actual nature of the workings. What I'm saying is that storyboarding is not a separate part of the task of making an animation, like laying out your tools before building a shed is, but rather an integral part of the creative process, the act of developing your idea.

In animation you come up with a concept, explore it in notes and sketches, try to break it down with a storyboard and end up fleshing the idea out as you do so. The storyboard may give you a new/different viewpoint. Ideas you liked may not work as you thought and smaller things become more important. Then it's back to more notes and sketches—and another go at the storyboard. To treat storyboarding as a task and entity unto itself would be a mistake.

This realization made me aware of why I've always been so dissatisfied with animation books that discuss the process, because they invariably do make this separation. Most artistic endeavors are like living things. They start off as a seedling of an idea, growing and branching

on stand:
ng Stories

in many directions. As time passes, some branches grow fuller, others wither and die. Still others you must manually (and often painfully) prune away.

Keeping this analogy in mind, let's get back to some creative gardening.

As I was saying

Before continuing, remember that I'm giving you an example of how to create an animation. You should be applying the techniques and processes I discuss here to your own ideas, not just copying what I'm doing; otherwise we'll all end up with essentially the same thing!

Last time I was discussing how to go about breaking an idea down and roughing out a preliminary storyboard. I did just that, coming up with an idea ("an ant wanders away from the anthill, but does not get far before some disaster strikes and ends its day [and life] rather abruptly"), thinking of different approaches (realistic, scary, dramatic, comedic), sketching ideas for these, narrowing down the possibilities, then roughing out key moments in each approach in a "preliminary" storyboard. I settled on the comedic approach as the stronger one, because the ant was a real character whose actions resulted in a reaction that affected the plot. In the dramatic idea the ant was a victim, never comprehending of its plight.

In addition to this stronger "plot" element, there's another reason for choosing the comic idea. I want to get the maximum impact out of this animation. By impact I mean emotional reaction from the audience. In the dramatic approach it's most likely people would be impressed by the animation, but let's face it, it's hard to sympathize with being burned by light focused through a magnifying glass, particularly when it's depicted as happening to an ant and in a manner as stark and realistic as I'd planned.

In the comic approach, I would be trying for laughter. The way the idea is structured it would build funny ant-dealing-with-the-heat gags one on another until the climax where the ant strikes back at his attacker, but gets smashed as a consequence of his own daring. The heat gags would be amusing enough, but the kicker would come with the surprise ending of the magnifying glass falling on him.

I'm not saying that humor is always preferable. You should go with what appears to be your strongest approach. Some people think of humor as an easy way out. Not true. Good humor is one of the more difficult things to pull off. Chances are if you miss the mark in a serious approach, it won't be a complete loss. But blow a gag—make something that's supposed to be funny *unfunny*—and your audience will have no sympathy.

In this case I'll take the risk of trying to do something funny, first because I think the ideas I came up with could be sufficiently humorous, and secondly because I've done cartoon humor before and made it work. Once you've made people laugh you want to keep on making them laugh. It's a sort of addiction.

Onward and forward

So I have my concept and a preliminary storyboard (see last issue). I now have to develop this into something I can animate. Here's where I am forced to do some of that arbitrary separation of topics discussed earlier, for at this stage in the game I will usually be shuffling revised storyboards with character-design work and also will start trying to determine which animation program(s) would be best suited to the task I plan. I don't want to drag storyboarding out any further than this installment, so I'll press ahead with it and get to the other topics later.

The preliminary storyboard I drew for the last issue delineates the key sequences I'm planning, but it lacks considerable detail with regard to the actions taking place within each sequence and omits linking

material such as the transitions between sequences. I need to flesh out the concept, and that means—you guessed it—another storyboard.

Let's look at the actions presented in the preliminary layout. The ant is being heated up by an unseen assailant using a magnifying glass. The sequences already drawn are:

- Ant lays down to get tan.
- Ant sleeps in shade of sombrero that catches fire.
- Ant lays under beach umbrella that goes up in smoke.
- Ant crawls across sand like a man in a desert.
- Ant is startled to see a hand holding a magnifying glass.
- Ant leaps at hand and bites it.
- Ant walks away triumphant, brushing his hands together.
- *Wham!* Magnifying glass falls into picture, squashing ant.

What is missing are details within and links between these. This is what I will attend to now.

We need an establishing shot that sets up the situation. How about the ant cutting leaves from a plant? Okay. That's where I start. Next, I need to introduce the threat. As he works, a shadow crawls up behind him and passes over him. We change angles to see an ant's-eye view of a silhouetted magnifying glass passing in front of the sun, and then the glaring magnified light through it. The threat is now established. Now to link this to the first drawn sequence: the ant getting a tan.

Here's a solution. I cut back to viewing the ant at work. A circle of bright light focuses around him. He wipes his brow, then shades his eyes as he squints up at the light. Then—*aha!*—he gets an idea. There is a blur of action, and we see him settling back on a towel, wearing sunglasses to get a tan.

I need a link to the next sequence, featuring the sombrero. At this point, I'll establish a standard device for showing the increasing heat level: Show the magnifying glass and the bright spot in it. I'll use this "linking device" between each sequence. For good measure, each time I show it the view will be a little closer to the glass, and the bright spot in it will be brighter and more intense. This has the dual effect of making the threat seem larger and more powerful each time, in addition to establishing why the ant has to keep switching strategies in order to beat the heat.

Here's how I work this linking device in and flesh out details in the sequences. The ant is getting a tan. Cut to the glass and the light, then back to the ant, who is getting a sunburn (so he's a black ant turning red). The ant hops up, grabs the towel, snaps it and *presto!* The towel is now a sombrero. (How? Irrelevant question. This *is* a cartoon, after all.) He dons it, drops to the ground and takes a siesta. Cut back to the glass. Back to the ant, whose sombrero is starting to burn. He sniffs a bit, there is a pause, then *toing!* His eyes snap open. A slight pause as the flames continue, then a blur of action as he whips off the sombrero and stomps the flames out.

The next sequence utilizes the beach umbrella that goes up in smoke. Do you see a problem with this? I do. It's too similar to the gag that it follows. Funny once, boring twice. What to do?

The first thing that comes to mind is to come up with another gag. Easy, right? Uh, yeah, well, maybe not. Nothing appealing comes to mind right now. I could leave it, but as I know it's wrong, I won't. I can't think of a replacement, so I'm left with the option of deleting it. Leaving it would do more harm than shortening the animation, so I'll cut it. If later on I get an idea for a replacement I can always shoehorn it in.

The next sequence is the ant crawling like a man in a desert. How shall I bridge the gaps between the sombrero-stomping and this? Cutting to the glass again might be effective, but maybe I should have the ant first finish stomping out the flames then trudge off screen. I could then cut to the glass and back to him crawling.

Then again, no. Having him finish stomping, relaxing, then trudging off lengthens the animation but does little else. It establishes nothing and seems to add nothing worthwhile. The problem is that I don't want to just cut suddenly back to the glass.

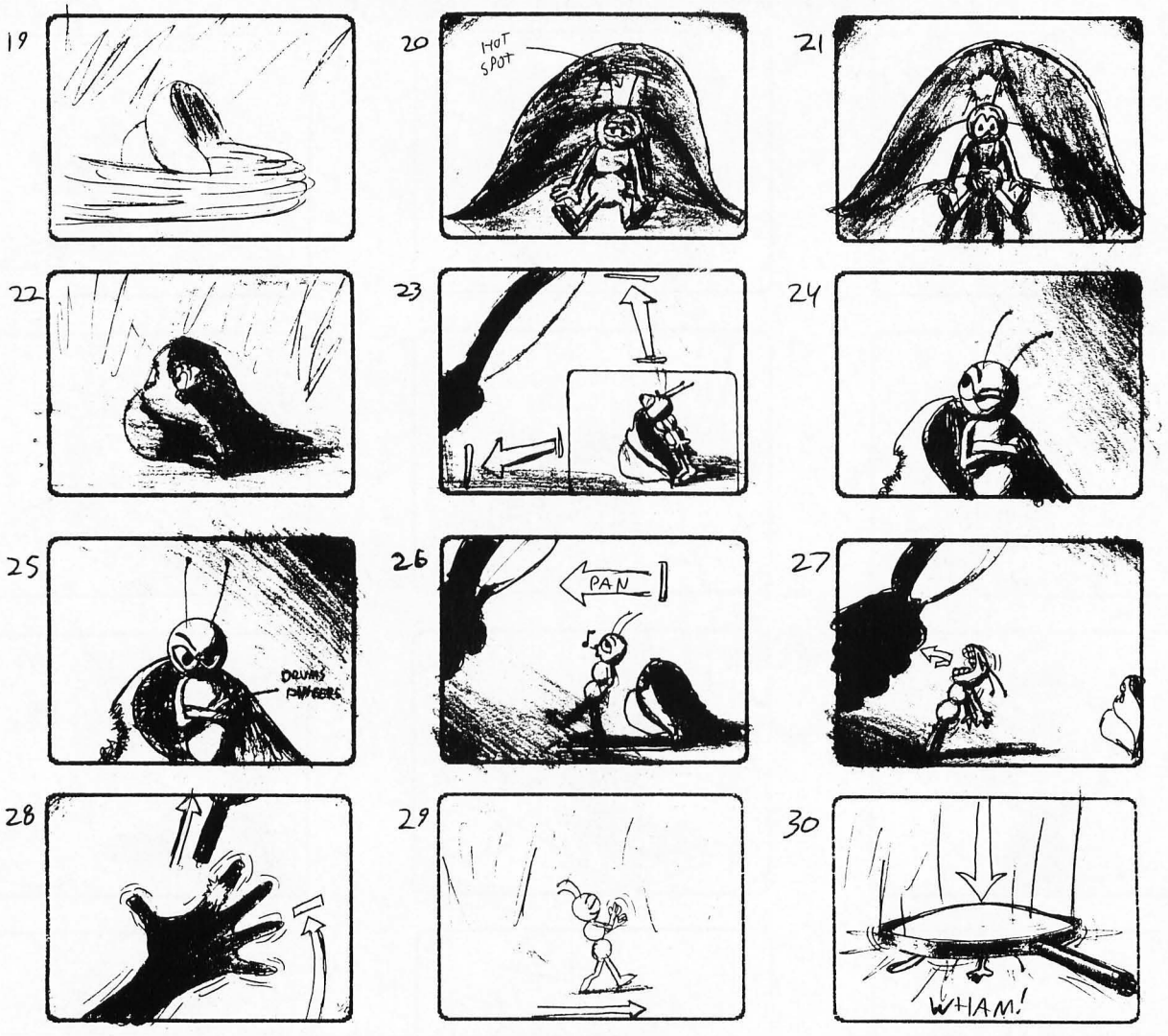
Ah, ha! I've got it! The solution here is not to cut to the glass, but to *dissolve* to it. As the ant stomps, the scene dissolves from that image to the image of the glass and the sunlight. To establish the heat as more severe than ever and that time has passed, in this case I'll hold the shot of the glass for an extra second than previously, then dissolve back to a scene of the sand, but no ant. We see a trail running across the screen, then the view shifts sideways along the trail until the ant comes into view, dragging himself across the sand, panting.

Okay, we're ready to finish up. The ant has to see the glass and hand, then attack. In the original idea he would crawl out of the light then, when the light refocused on him, would suspect foul play and look actively for the source. I like this idea, but how do I present it?

The problem here is finding the best way to establish what is going on in the minimum amount of time and in the most direct and understandable way possible—and all without being cliched and trite! Timing is critical, because thus far each sequence is linked to the next by the bridge of showing the magnifying glass, and a long transitional shot at the end would destroy the pacing (the pace should build *up* toward the end, not slow down; that's why they call the finale a climax). This use of the magnifying glass as a bridge is dramatic and eliminates the need to animate uninteresting and unnecessary transitional sequences (like the ant getting hotter, thinking and then staging his next plan). Likewise, the ending should employ some similar technique and not shift gears to an approach I've thus far rejected.

Here's a thought: As the ant crawls, I dissolve back to the light, then back to him crawling. He comes upon a large leaf, and ducks behind it to get some shade. Once in the shade, he sighs and looks relieved—but the heat focuses tightly on the leaf and burns a hole through it, letting the light hit the ant again. Now he suspects something's not right, steps out from behind the leaf. *Surprise!* He sees the hand and the magnifying glass! His expression goes from shock to anger, then slowly slides into a cunning look. He starts walking toward the hand, whistling innocently, then springs into the air, sporting huge teeth and *chomp!*

The final problem is how to make the transition between this attack and the finale. Some actions are so complex that trying to animate them is pointless. Pointless not because it can't be done, but because it's difficult, and even if you got it right, it probably wouldn't be the best solution. So, I'm not going to show the hand whisking out of the picture and the ant dropping to the ground and walking off, only to get flattened. Right as the ant bites I'll switch to a shot of the sky and sun. Suddenly, the hand snaps up into view, fingers spread and a glowing red bite on one finger. Just for effect I'll plan a blood-curdling scream here. Oh, what about the magnifying glass? Well, I'll add it here, flying out of the hand and shooting out of



the top of the picture.

Won't showing the glass flying off clue the viewers as to the end? Only if I make it obvious. The trick here is to keep the viewer's eye on the hand, so they'll only peripherally note the glass flying off. This is done by having a pulsing red bite mark on the dark silhouette of the hand, which will be shaking at screen center. The combination of the pulsing bite mark and shaking action should keep the viewers' eyes on the hand (I hope).

I finally cut back to the ant walking away, smugly brushing his hands together. Then, suddenly, *wham!* The magnifying glass flattens him!

Fade to black. Roll the credits. The end.

So, there's the plan, which I have drawn out in the revised storyboard accompanying this article. I got most of these ideas without having to sketch, but in order to make sure I won't forget any, I jot down a quick note about each idea. It's easy to forget some little bit of business when you're working on projects like this, so *always* make notes!. For your benefit, I'll update the list from before and add the new material and make other changes (compare it to the first list). Here's the continuity now presented in the new storyboard:

- Ant collects leaves, shadow passes over him.
- Shot of magnifying glass moving in front of sun.
- Ant starts to get hot, wipes brow, squints at light.
- Ant has idea and lays down to get tan.
- Close up (CU) of glass with light blazing through it.
- Ant gets sunburn and produces a sombrero to sleep under.
- CU of glass.
- Sombrero starts burning, ant realizes this and stomps it out.
- Extended extreme CU of glass.
- Ant crawls across sand like a man in a desert.
- Another extreme CU of glass.
- Ant crawls into the shade under a leaf, but the heat burns a hole through it! Ant gets suspicious.
- Ant steps out from behind leaf and is startled to see a hand holding a magnifying glass focusing the light on him.
- Ant whistles, walking nonchalantly toward hand, then he leaps at the hand and bites it!
- Loud scream, hand jerks into full open position with glowing red bite spot; glass flies up out of view.
- Ant walks away triumphant, brushing his hands together.
- *Wham!* Magnifying glass falls into picture, squashing ant.

Indicators

Let's look at the storyboard accompanying this article. Does it seem pretty clear to you? I hope so. You can probably already fathom the action and probably understand what all the arrows and such stand for. Still, we'd better talk a bit about them.

These arrows are what I call "movement indicators," and they help in establishing the directions of actions in a minimum number of drawings. Thus, to

storyboard the ant walking toward the hand, I merely have to draw one pose of him in a walking mode and use an arrow to indicate his path. To indicate an action that involves starting from one position and then moving, and/or moving then stopping, I use small start/stop lines at the head and tail of the arrows. Look at the frame where the hand displays the bite. A curved arrow comes up from the bottom of the picture and shows the path the hand takes into the picture. There is a stop line at the tip of the arrow, indicating the hand stops that movement at the indicated position. Note that the arrow for the magnifying glass just keeps going, meaning it doesn't stop.

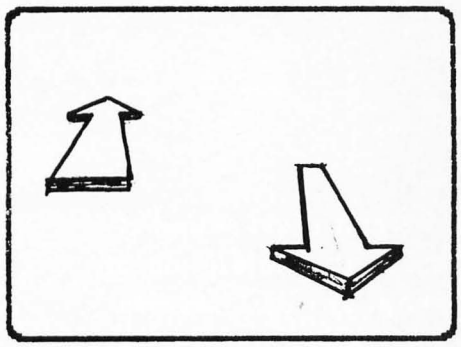


FIGURE 1

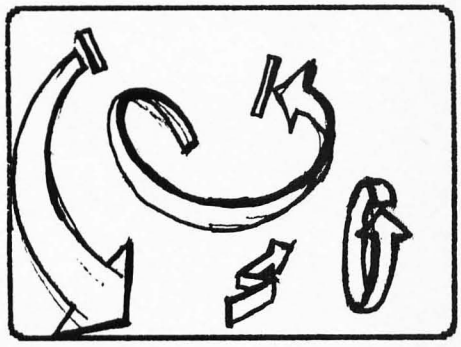


FIGURE 2

To indicate actions moving in or out from the screen, I use similar arrows, but ones drawn in perspective. If an object is to move away from the viewer, I'll draw the first arrow in Figure 1. If it's coming toward the viewer, I'll draw one like in the second arrow. You can use start/stop lines with these too.

For more complex actions, like objects turning, twisting, spinning and so forth, you can use one complex arrow or a number of arrows. Figure 2 shows examples.

For vibrating actions, like shivering, shaking and so on, you can use movement lines like those in the frame where the bitten hand shoots into view and shakes. Fast actions can be indicated with speed/blur lines, like the frame of chaotic action just before the ant appears on the blanket.

Camera angles

Although we really don't have a camera per se when making a "movie" on a computer (although some programs have digital equivalents), when planning your animation it's still best to act as if you did have a real camera. Watch any movie or television show, and you'll notice that the viewpoint changes constantly. When a character is the center of attention, you'll often get a tight close-up of that person. When several characters are interacting, you'll get a medium shot of the group, interspersed with close-ups. Wide angle "establishing shots" are used to give the viewer a look at the setting and surroundings, before moving in to closer angles on the characters. There are also what are known as "insert shots," where the camera focuses on a small but key action, like a finger closing on a trigger, a picture falling behind a desk, a spider walking down Phoebe Cates' cleavage. Uh, you get the idea.

There are still shots where the "camera" doesn't move; pans where the camera spins to one side of the scene or the other (an extremely fast pan is known as a "whip pan"). There are even dolly shots where the camera moves sideways, in addition to trucking into and out of (forward into and backing out of) a scene.

There are also point-of-view (POV) shots, such as seeing the scene from the viewpoint of one of the characters in it. In most films the viewpoint is omniscient, meaning it is not of any character in the scene—sort of like the POV of someone who isn't there!

You should try to determine which type of viewpoint is best for a given shot, which angle and range of "closeness" is best to clearly display the action planned. At the same time, you don't want to do a lot of boring side views. Try to avoid ordinary camera angles. If you want to see a lot of camera work, watch some music videos. They truck, pan, dolly, zoom and bank constantly, and cut from extreme close-ups to long shots to medium shots with reckless abandon.

You must also be careful about choosing camera angles that are dramatic, but inappropriate. For example, a high overhead view of the ant when he gets smashed might be a neat angle, but it ruins the shot because we'd see the glass falling toward him too soon, which would warn the viewer what was going to happen and ruin the gag. By the same reasoning, I never show the kid who is holding the magnifying glass. This is an ant's-eye view of the situation, so I keep to what is in the ant's world. Besides, showing the kid isn't relevant. Showing him would add nothing and probably detract from the effect because what was a faceless threat would suddenly be a person, a character. Showing the kid would raise questions I don't want to answer, and I don't want to answer them only because those questions are irrelevant to the story I'M creating.

Revisions

Did you notice that I haven't discussed actually drawing this revised storyboard? Why? Because there's not much different in regard to drawing this one than the previous one. You just sketch and explore different views until you get what you want and/or think works (they are not always one and the same). You go through your idea as I did, linking the sequences together with new action, and so forth, using interesting and clear camera positions, and draw it again and again until you get something usable. It's not as easy as it sounds sometimes.

There is one other thing that would be useful when storyboarding, particularly if you're going to use some characters and not inanimate objects, and that's "posing." But, posing is related to character design, and that's part of the next installment. So don't put away your sketch pad yet; the revised storyboard you draw may very well not be the final one!

Th-th-th—class dismissed

Okay, we're finished with Storyboarding 101. Next time it's down to the final preproduction phase, including character design, posing and something actually computer-related (*gasp!*): deciding which software to use and if your idea will work with the tools you have. ▪

Blissfully ignorant of the realities of time and space and plain old common sense, ***Maurice Molyneaux*** *hopes someone will someday discover "retroactive reincarnation" so that when he dies he can come back in a* previous *life as animation director Chuck Jones. His greatest fear would be to come back as Wile E. Coyote, and in the process have to learn some humility.*

BALLBUSTER

BY ERIC SCOTT

LOW RESOLUTION ONLY

Ballbuster **is a fast, arcade-style game with colorful graphics and great sound effects. It will run in low resolution on all Atari ST computers with a minimum of 512K of memory, although 520ST owners should boot up without any desk accessories before playing.** ***Ballbuster*** **can be run from any drive, including a hard drive. Those of you who are sweating at the thought of plugging a joystick into the ST can relax! The mouse is used for all player input.**

Note: Due to the large size of this program, it is available only on this month's disk version or in the databases of the ST-LOG user's group on DELPHI.

Illustration by · ALAN HUNTER


Ballbuster
By Eric Scott
Top six scores for Ballbuster
11660
9750
7860
6220
5140
4005 *
Click left mouse button to play
Click right mouse button to quit

BALLBUSTER


MANA:4840
Ballbuster
BALLS:2

Your task is to help Morry gather crystal balls to throw at a great spinning orb in which the Demon Lord is hording all the manna of the world. Alabarador is on a trip in a small city 200 miles away, so the fate of the Earth lies in Morry's and your hands. Hit different colored segments on the orb to release different amounts of magical energy, and each color awards you with certain amounts of points.

Starting the game

To start, double-click the BALLBUST.PRG on your disk. In a few seconds the title screen will appear. This screen will display the top six scores achieved during the current play session. If your last score made it into this list, an asterisk will be displayed next to it so that you can easily compare it with the other five. Since the scores aren't saved when you quit, the top six will initially be set to zeros each time you run the game.

The story

The title screen also introduces the two characters in this exciting drama. The little red-clad fellow is Morry Mage, a hard-working but inept apprentice of the great sorcerer Alabarador. One day he was practicing a simple spell to relieve his persistent dandruff and apparently botched the incantation, for the next thing he knew every hair had fallen out of his head. In the process he also managed to drop and break the last of his master's crystal balls.

As Morry left the ancient Hall of Enlightenment in search of more crystal balls, the golden Spring of Power hid from his view a most disturbing sight. In a far corner of the room a giant and terrible head began to form! Somehow Morry's spell had released the evil Demon Lord from the farthest reaches of the nether region to which Alabarador, after a great battle, had imprisoned him centuries ago! Well, only partially freed, for the Demon Lord's body was still trapped; even when Morry screws up, he does it wrong.

For ages the ogre had plotted his revenge, and now he began a terrible chant. A giant multicolored globe took shape in the center of the hall. Slowly at first, it began to spin. As the globe revolved faster and faster, it started to soak in mana from the far corners of the land like an immense magical sponge.

When Morry returned later that evening, hugging a sack containing three crystal balls that he had scrounged up in town, he was very worried. Strange things were happening all across the land. Flying carpets laid limply on the ground, magic washing tubs were filled with laundry that just floated aimlessly in dirty water; in fact, anything that used magic had stopped working.

He reached the Hall of Enlightenment, and stopped short when he saw the huge spinning globe with strange symbols etched into its glowing surface. As he gaped, a monstrous green head appeared above him and spoke in a hissing voice. Gloating, the Demon Lord told Morry of his terrible plans to gather all the mana in the world into the globe, where only he would have access to it. When the ogre had trapped a sufficient amount of energy, he would release himself completely from the nether region and rule the Earth forever.

Since Alabarador had gone on a trip to a small city some 200 miles away, Morry knew that any hope to defeat the Demon Lord's vile plot lay solely with him. The great globe spun far out of his reach, but perhaps a crystal ball thrown with enough force just might shatter one or more of those brightly colored segments and release the mana trapped within.

Morry took one of the balls from the sack and settled himself onto a disc of levitation. He studied the whirling orb above him and said a short prayer to the powers that be to guide him in this undertaking. Perhaps with luck he might even get the ball onto the Spring of Power! Taking aim at a red segment, Morry threw the crystal ball with all his might.

Playing the game

Your task is to help Morry release as much mana from the globe as possible. To play a game, click the left mouse button. You will now find yourself in the Hall of Enlightenment with Morry and the Demon Lord. At the top of the screen are two important readouts. MANA will show how much mana you release during the game, while the BALLS indicator will show how many of the crystal balls are left. When you have three or more crystal balls, this number will be displayed in green; if only two or one balls remain, the number will be yellow or red respectively.

In the center of the screen you will see the spinning globe of mana. Each colored segment holds a certain amount of the magical energy: Red segments hold 500 units; gold segments hold 100 units; violet segments hold 75 units; light blue segments hold 50 units; green segments hold 25 units; and dark blue segments hold ten units. For every 5,000 units of mana released, you will receive a bonus ball and be alerted to the fact by a bell tone.

The left side of the screen stands the golden Spring of Power. Should your luck and skill enable you to land a crystal ball on this amazing machine, a spell of alchemy will turn the ball to gold. After the spring shoots the golden ball back into play, any segment hit will release twice the amount of mana it normally contains. If you are truly fortunate and can get the

gold ball onto the spring one or more times, all segments will be worth triple or more their original value.

Below the globe Morry Mage sits on his levitation disc. The disc will float left or right along the bottom of the screen, responding to the movements of your mouse. To tell Morry to throw a crystal ball, press both mouse buttons simultaneously. He will thank you and toss the ball straight into the air. At first the crystal ball won't go very high, because let's face it, Morry is a wimp. Keep Morry centered under the ball, though, and it will bounce off his noggin. With each bounce the ball will fly a little higher until it nearly reaches the top border. Move him to the left or right slightly, and the ball will hit one of Morry's hands, sending the crystal ball flying in the desired direction.

Whenever a ball smashes a globe segment, the ball will fly off in a new direction and speed. The ball will rebound safely off top, left and right borders, but you must always have Morry catch it with his hands or head before the ball hits the bottom. Should you fail, the ball will splinter into useless fragments and poor Morry will be very depressed. When you lose a crystal ball, click the left mouse button, and Morry will politely thank you and ready another ball. When you have lost your last ball, click either mouse button to return to the title screen and see if your score made it into the top six.

The Demon Lord will not stand idly by while you try to destroy the globe and his evil plans! He flashes in and out of the nether region, floating up and down the walls while trying to position himself to catch the crystal balls. If he does catch one, it makes a nice, crunchy snack. There is nothing more frustrating than watching him gulp down your last ball. The Demon Lord will not touch a gold ball, however; he finds they cause him severe heartburn.

A few valuable hints

Never position Morry directly beneath the globe when throwing a crystal ball. It will almost always ricochet into the ground before you have a chance of catching it.

Work on the left or right edges of the globe. This gives you some time to react if the ball suddenly shoots toward the bottom or a corner.

A good tactic to try is bouncing the ball off one of the walls after letting it build up speed. It may land on top of the globe and "sink" down, causing a lot of damage and freeing tremendous amounts of mana. This must be done with caution, however, because the crystal ball might simply slide up and down the border and give the Demon Lord an opportunity to munch out.

While you should always try to attack the side of the globe away from the Demon Lord, the left side gives you the best chance to land a ball on the Spring of Power. If you get a gold ball you can forget about the ugly green guy for a while.

About the program

Ballbuster was written in Megamax C; the graphics were developed with *DEGAS Elite*; and the sounds with the *G.I.S.T.* sound editor. Because it uses *G.I.S.T.*, the link file SNDSUBS.O is required to create an executable program file.

The major task when writing an arcade-style game is how to coax fast graphics out of the ST without resorting to writing an all machine language program, something a little beyond my current talents. Because the only really slow C routines are the graphic functions, all I needed to concentrate on in this case was a replacement for the *vro_cpyfm( )* function. After a long period of searching through the rather limited number of books and articles dealing specifically with Atari ST graphics, I came up with a few valuable nuggets.

The ST screen memory is divided up into 16-bit words. A "sprite" that you put into that memory must be able to fall on any horizontal pixel within those 16-bit words. This is done by shifting the sprite between 0 and 15 pixels right so that it fits into the correct bit position like a jigsaw puzzle piece.

Usually this process occurs while the sprite is being drawn. When you try to have a large number of sprites moving around the screen at video-game speeds, it becomes hopeless! The normally fast Atari is so busy calculating and shifting that the graphics barely trudge along. The solution became obvious; just do all the shifting before the game even begins.

I send the sprite data and the amount to shift it to a function called *shift_sprite( )*, where an assembly routine performs all the calculations and places the shifted sprite into the proper array position. This is done up to 16 times for each sprite, although some sprites that don't need to move horizontally can get by with only one shift. The arrays will now hold sprites that need only to be plotted to a screen buffer. When you initially set up the arrays, you need to "Malloc" twice the amount of memory used in the original sprite for each position that will hold a shifted sprite. This is the only drawback to preshifting; it does use quite a bit more memory than standard methods of drawing sprites.

The function *plot_sprite( )* quickly draws a shifted sprite onto a screen buffer. This assembly routine takes the sprite array and by ANDing the X coordinate of the sprite with 15, determines which shifted sprite in the array to plot. Each word of the sprite is then ORed with the buffer at the correct XY coordinates. If "mode" was 0, it will instead erase a 16 by 16 segment of the globe by repeatedly ANDing the first two words of the "blank" sprite with the buffer. A few simple changes to this part of the routine would allow you to use masks for the sprites. Note that both the *shift_sprite( )* and *plot_sprite( )* functions can be used with any size sprite, as long as its width is evenly divisible by 16.

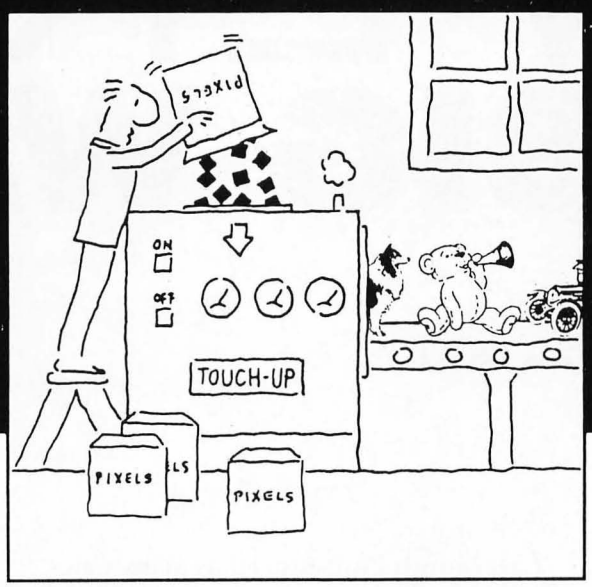
Two other small functions, *copy_buffer( )* and *flip_buffer( )*, complete all the assembly routines I needed for *Ballbuster*. The function *copy_buffer( )* rapidly copies the contents of one graphic buffer to another, while *flip_buffer( )* waits for a vertical blank, sets the screen to the HIDDEN buffer, and swaps the addresses for the HIDDEN and SHOWN buffers.

These four functions can easily be used by anyone in their own Megamax or Laser C programs to achieve near machine language speeds for their game graphics with little programing effort. ▪

Eric Scott's first computer was a Sinclair ZX-81. After he outgrew that machine, he purchased an Atari 400, followed by an 800XL. He now owns a 1040ST and hopes to make a living programming it.

Give your desktop graphics that professional touch.

Introducing Migraph Touch·Up™



Migraph Touch-Up is the complete design tool for high-resolution monochrome images.

Create, edit, enhance—you can do it all with Touch-Up.

And do it better, because Touch-Up is the first "virtual page" graphics program for the Atari ST.® A sizeable advantage indeed!

Touch-Up can handle bit-mapped images of any size and resolution (based on available memory in your computer). Which means you can now produce pixel-perfect images for all your publishing projects.

Migraph Touch-Up. A powerful tool for professional-quality results.

Ask your dealer for a demonstration of Touch-Up, or call Migraph's toll-free number for more details.

MIGRAPH INC™

200 S. 333rd St., Suite 220 Federal Way, WA 98003 (800) 223-3729 (206) 838-4677

CIRCLE #112 ON READER SERVICE CARD.

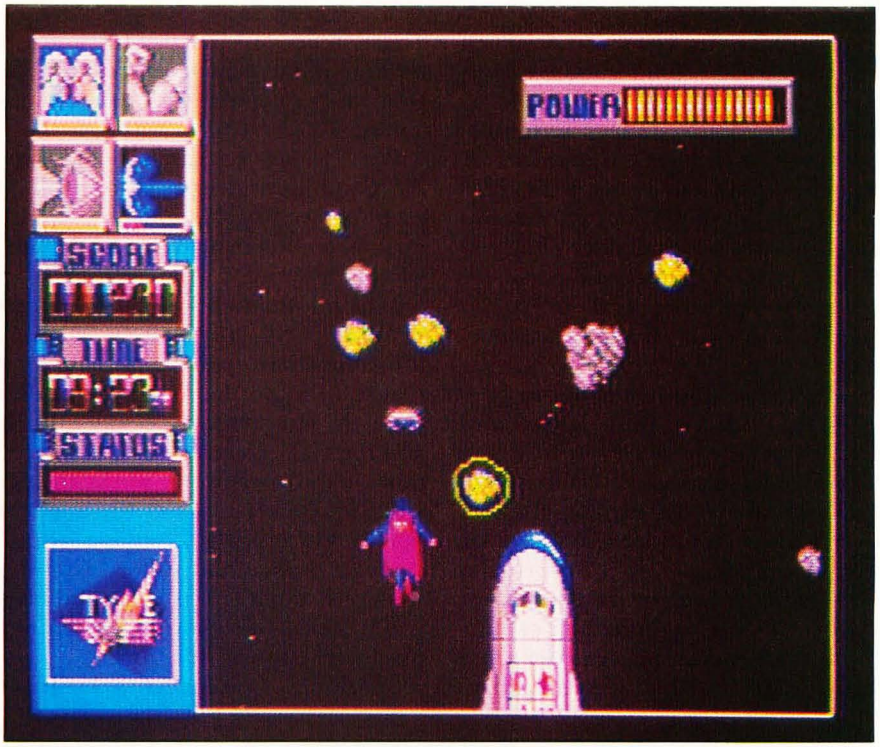
FROM OVER THE BIG WATER

By Marshal M. Rosenthal

*L*ast month I introduced as many new games from the overseas market as could be jammed into one article. But they keep on coming! Frankly, there seems to be no end to overseas software. So here's the ground rules: I'll try to mention new products as fast as they appear, which means that sometimes the information will be a bit skimpy. To be fair, I'll go into more depth on the most interesting ones later. Many of these programs will eventually make it to our shores, although there's no telling when or under what name.

Superman—Man of Steel was mentioned briefly last time and merits a closer look. As noted before, you play the guy in tights throughout a number of arcade scenarios, in pursuit of the evil Darkseid and the equally menacing Lex Luthor. Each sequence is preceded by a comic-book page that displays text on what dire event is in progress.

You begin by trying to stop Darkseid's demons from approaching the city. As they close in, you choose various powers with a click of the right mouse button. Punch a demon out of his rocket sled with a resounding thud, or use heat vision to fry his brains. Of course, these guys are firing deadly rays at you—rays you must avoid as you fly amidst them. Each power that is activated can be drained, so you need to watch the screen and monitor your power levels.



You begin by trying to stop Darkseid's demons from approaching the city. As they close in, you choose various powers with a click of the right mouse button.

SUPERMAN—MAN OF STEEL **Tynesoft Computer Software**

Now to rescue Lois from the guys in power suits. This time you also have a super kick, useful when the baddy gets close enough. Beware those deadly missiles, and whack them into oblivion. Should you succeed here, you are then given the envious (?) task of escorting a space shuttle through space. Most of the meteors can be smashed or blocked with your super body, but beware the kryptonite. Fail and you are treated to the sight of the shuttle exploding.

There are scenarios not described here in which you must compete before finally reaching the arch criminals.

I do have a few complaints about this generally fun game. Characters are a bit tiny and Supe's movements a tad slow in response to the mouse. I also found the sound effects below par—mostly thuds and krinks. We all know that the ST sound

chip is not fantastic, but it can do much better than what is found here.

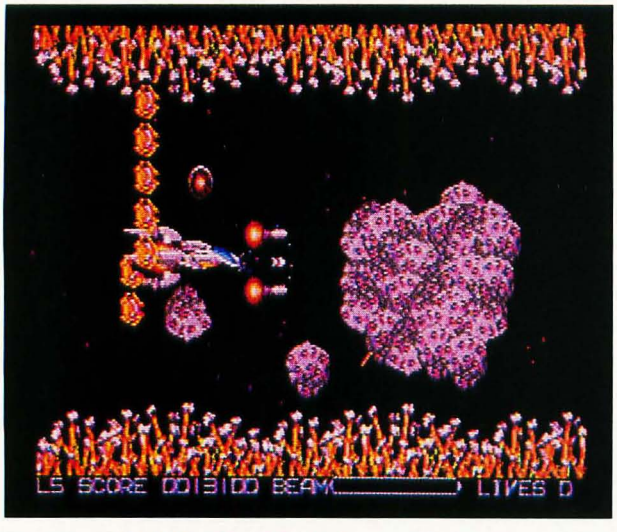
Some games are designed for rapid action, without a lot of strategy and thinking involved. Such a game is *Andes Attack*, by Llamasoft Ltd. The theme is *Defender*-like, with you piloting your fighting ship through hostile, horizontal-scrolling terrain. Your mission: to rescue desperate llamas from the mean, ugly aliens.

Squads of monsters inhabit each level. There are demonic smiley faces, floating mines, evil clutching fiends out to grab and devour those little guys, and well. . .you get the idea. Of course, you're equipped to handle such difficulties, with rapid laser fire and smart bombs that blast a full screen to bits. Use the radar screen to keep check on the llamas, who scream for help (with ever so tiny word balloons). Manage to shred their captors before they rise up, and you can catch and return the llamas to their blessed terrain. Those in need of practice will adore the training mode, which lets you practice to your heart's content with total invulnerability.

What makes this game so addictive is the graphics and ease of control. The graphics are bright, cute and big, with a cartoon feeling that adds some humor to what is, after all, a shoot 'em up. (Check out the information screens that precede play for specifications on the enemy.) As for game control, one hand stays on the mouse to move the ship up and down, while the left button accelerates the ship and the right changes its direction. The other hand meanwhile works the Shift and Control key for laser fire and smart bombs. Once you get cooking—zipping through orange and yellow explosions, while the sound effects bleep and blap—you'll feel like you're watching a psychedelic movie. It's fun!

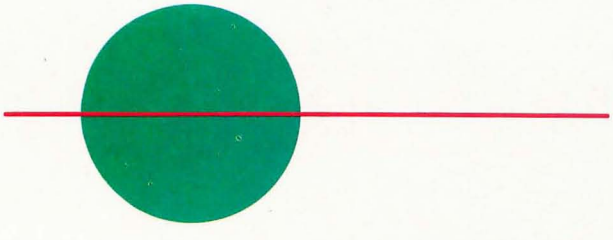
Speaking of psychedelics, Llamasoft's *Trip-A-Tron* is one involved program—but that doesn't mean it's difficult to master. You use the various screens to create moving graphics and light patterns that dance around onscreen to your favorite music. There are numerous icon-driven control screens, and these include a waveform/ build control, a star-field control, a color cooker, MIDI control and much more. Llamasoft gives you enormous options so that the final result is a personal, interactive statement between you and the music.

More options include a video sequencer that enables graphics and music tracks to be transferred to videotape, an event assign (to make certain keys on the computer bring up desired reactions), and a sillyscope for altering waveform patterns in a purely visual way. *Trip-A-Tron* requires one megabyte of memory to be most effective, and it beats the heck out of the light organs of the '60s. The visual



R-TYPE · **Activision U.K.**

R-TYPE · **Activision U.K.**



ANDES ATTACK · **Llamasoft Ltd.**

artist in you will love it.

Ubi-Soft of France takes parts of *Pac-Man* and combines them liberally with *Gauntlet* to get *Puffy's Saga*. You control a little bouncing puff ball in search of his mate, bouncing through levels to pick up useful objects like invisibility and super shots. There are also the Magic Goms, which can be used to activate a map of the present level or increase your firepower. In the meantime, there are many obstacles, not the least of which are the red blobs that won't leave you alone—even when you blast them apart. The fire breathing dragons don't help either, nor does the fact that time limits are imposed on your stay in each level. You won't appreciate the stun barriers or shape changing walls either.

Extra features have been included. Both a slow and fast mode can be activated, and the game can be saved once you reach the seventh level. Digitized sound enhances the game, but as lowering power points bring you closer to your demise, a voice warns, "Puffy, you will die!" A bit weird for my taste. Still, all contributes to an entertaining game that requires arcade skill as well as strategy.

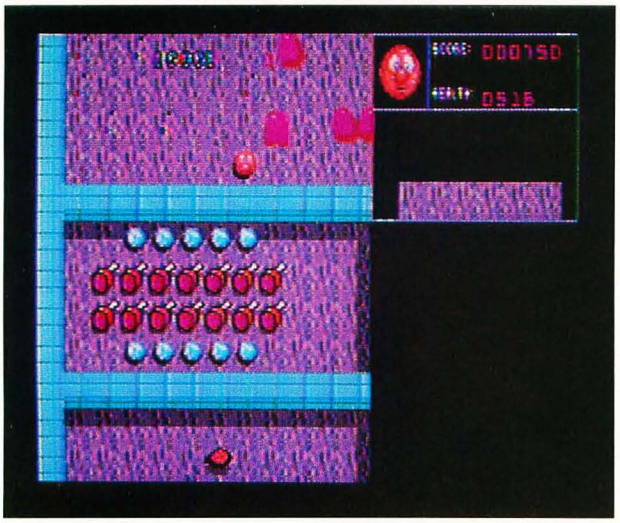
We return to space with *R-Type*. The evil Bedo empire needs to be eradicated, so grab your spaceship and head on out. All the awesome action that marked the arcade version is here: giant creatures, numerous aliens and action, action, action.

Fire till your joystick button overheats because there's no time to relax! Pick up power modules to add firepower and shield strength. Avoid alien weapons and the energy-draining landscape, but most of all keep firing. There's also a power orb robot that will fly alongside and help you. Be on the lookout for it.

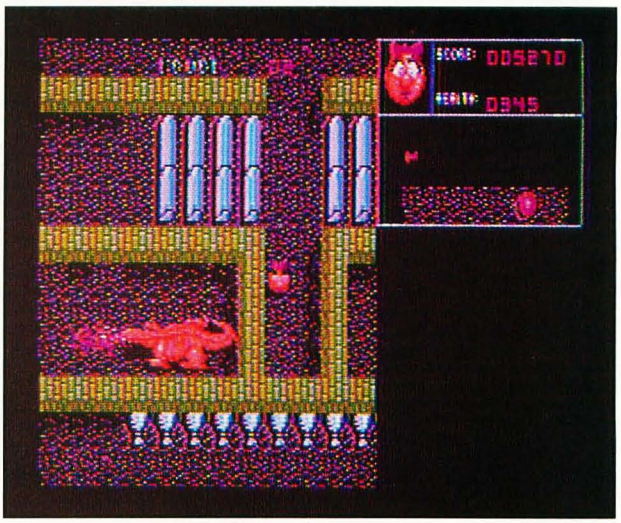
There are plenty of levels and few options but to fight it out. You do have a choice between music and sound effects, and a pause key is mercifully tossed in.

R-Type is so sensational that I feel compelled to enclose the following super cheat, but don't read further if you want to be victorious on your own merit. Here it is: When the program prompts for the second disk, first press the Help key, then type *ME* and press the Up arrow. Insert Disk 2 and continue. The F5 key will now protect you from the aliens and landscape, while F6 deactivates the effect of enemy fire.

Just crossing my desk is another superhero: Batman. *The Caped Crusader* descends upon us, courtesy of Ocean Software. Being a "night person" takes on a whole new meaning as you don the persona of this complex and somewhat violent defender of the good, in search of both the Penguin and the malevolent clown prince of evil, the Joker. More on this great game next time. □



PUFFY'S SAGA · **Ubi-Soft**



PUFFY'S SAGA · **Ubi-Soft**

ANDES ATTACK · **Llamasoft Ltd.**

Products mentioned in this article:

Andes Attack
Llamasoft Ltd.
48 Mount Pleasant
Tadley Hants
England RG26 6BN

Batman—The Caped Crusader
Ocean Software
6 Central Street
Manchester
England M2 5NS

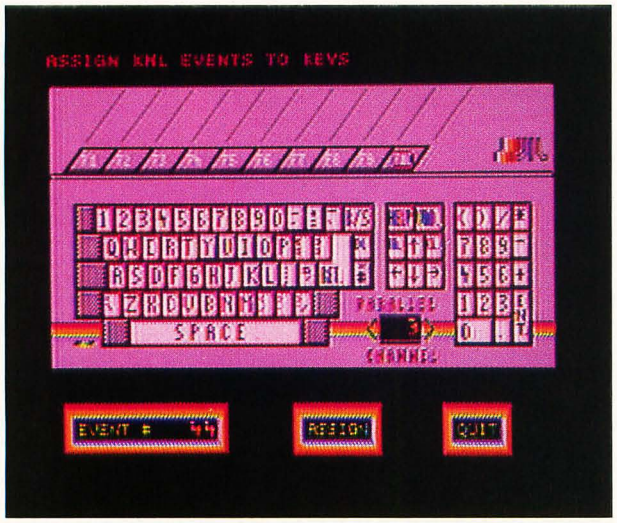
Puffy's Saga
Ubi-Soft
1, Voie Felix Eboue
94021 Creteil Cedex
France

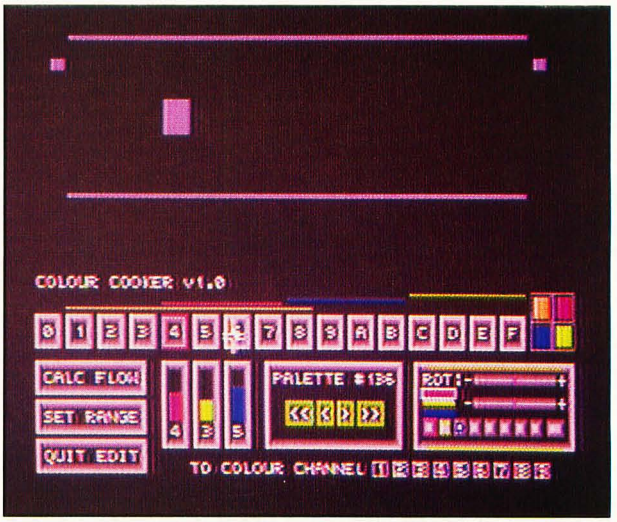
R-Type
Activision U.K.
Blake House, Manor Farm Road
Reading, Berks
England RG2 0JN

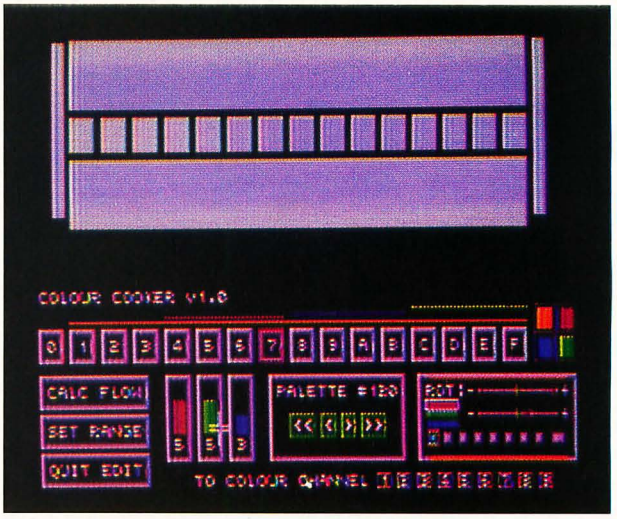
Superman—Man of Steel
Tynesoft Computer Software
Addison Industrial Estate
Blaydon, Tyne & Wear
England NE21 4TE

Trip-A-Tron
Llamasoft Ltd.
48 Mount Pleasant
Tadley Hants
England RG26 6BN

Marshal M. Rosenthal *is a photographer and writer whose work takes him throughout the world. His photographic/written projects have been published in major consumer publications in Europe and the United States.*







STEP 1:

Make Mine Music

by Maurice Molyneaux

We all know that Atari hasn't done a good job of promoting and selling ST computers in the U.S. IBM clones dominate the business world, Desktop Publishing is the domain of the Macintosh, Apple IIs infiltrate many homes and more schools than I care to think about, and more Commodore 64s collect dust in closets than the total of every Atari computer ever sold. Even in the video-game biz Atari, once the champ, is now a small-time player behind the leaders Nintendo and Sega.

Sure, Atari is a big deal in Europe, but they hardly have a toehold in the domestic market of the United States. The ST is considered by many an "also ran" computer, a market failure that never found a niche.

But wait, what's this we see? While the ST has failed to make a dent in the markets mentioned above, it seems it *has* pushed its way past all opposing brands in one particular field, and now appears to be "computer king of the hill" in this special niche market: music and, specifically, MIDI.

Atari did a number of things wrong with the ST design (and I could wax eloquent on this, but I won't), but their one stroke of brilliance was the inclusion of MIDI ports as standard interfaces on all ST computers. Until recently the ST was the *only* personal computer that had MIDI ports as part of the base unit (late model Laser [Apple II clone] machines and recent Yamaha MS-DOS computers have them also). And, while you can add MIDI interfaces to everything from Atari 8-bits to PC clones, it seems that the ST's combination of low-price, mouse-driven interface and built-in MIDI ports has made it a favorite of people buying computers for use in music.

In the recent past the Apple Macintosh was the top computer for MIDI, but it appears to have been usurped by the ST. Some estimates give the ST a share of the MIDI computer market of between 30 and 40%, leaving the Macintosh and PC clones down in the 20s! Some have even taken to using the catch phrase: "If it's not an ST, it's not MIDI!"

What all the shouting's about

If you're new to computers, electronic musical instruments, or both, you may wonder what all the fuss is about. The word "MIDI" is bandied about as if everyone and his second cousin's half-sister knows what it means. From all the hubbub it becomes fairly obvious that MIDI has to do with music, but what MIDI is and what it does are often glossed over or forgotten.

I bet you can guess where this is leading, huh?

MIDI was established by a consortium of leading musical instrument manufacturers. They set out to create a standard (both hardware- and software-wise) to define the manner in which electronic musical instruments communicate with each other and share data. In this way a Yamaha synthesizer could understand the data coming from a Roland model. This was no small thing, and was the musical world's equivalent of what it would be like if IBM, Apple, Atari, Commodore and other computer manufacturers standardized graphics file formats and the hardware and software interfaces for communicating that data. Imagine!

The term MIDI is an acronym for "Musical Instrument Digital Interface," which is quite a mouthful. As I said before, it refers to both a hardware and software standard, meaning both the interface port/connectors, and also the manner in which information is represented within and communicated through the hardware. What MIDI does is provide a standard way to digitally represent music so that it can be communicated between various electronic devices; such as synthesizers, sequencers, drum machines and so on.

A MIDI port is a small five-pin DIN plug that is the standard interface connector for MIDI-compatible devices. There are two of these on any ST, an IN port and an OUT port (located on the back of 520 and Mega models, and on the right side of 520STfm and 1040ST machines). The IN port is for data received by the ST from other MIDI-equipped devices. The OUT port is for data transmitted to such devices from the ST.

Although STs don't have one, there is a third kind of MIDI connector: the THRU port. The THRU connector just takes whatever information is coming from the IN port and passes it, unaltered, to whatever device is connected to it. While the ST does not have such a port, Atari did do a little tweaking. They connected the two signal-in lines on the IN port to two unused pins on the OUT port, thus creating an effective, but somewhat nonstandard THRU.

In truth, the ST's MIDI OUT is more like a MIDI THRU/OUT (if you'll pardon another pun). However, precisely because the THRU connection is nonstandard, the port is normally treated as an OUT only. To use the THRU you need to build a special adapter to split out both OUT and THRU ports (making such an adapter is beyond the scope of this article). If you don't want to build such an interface yourself, you can always purchase a commercial solution. One that I know of is offered by the fine folks at Practical Solutions.

MIDI doesn't record actual sound, like an audio tape (which is an analog device, not digital, like MIDI), nor does it digitize the sound (meaning breaking down the sound into numbers representing it) like the digitally recorded music on a compact disk or DAT (Digital Audio Tape). What it *does* do is store the information about the music and how it is played. This is somewhat akin to a word processor. Word processors do not store words, which are in reality just sounds, but they store a standard set of symbols (arrangements of letters) that represent those words. Do you follow me? So, just as a word processor uses characters that our mental software can translate into audible words, MIDI uses digital information which a properly "knowledgeable" device can turn into music.

MIDI has numeric values for notes, velocity (how forcefully you press the keys or hit the pads on a velocity-sensitive instrument), and so forth. For example, when you press a key, a "note on" signal and the value of the note is sent through the MIDI OUT port of the device you are playing. When you release the note, a "note off" signal is sent. Velocity information is also transmitted (if an instrument does not actively support variable velocity, it will usually send a medium or half-velocity value, and possibly simply ignore any incoming velocity values). A device on the receiving end that "records" these events and monitors the time that elapses between them can then later transmit that information from its own OUT port and replay it.

MIDI doesn't record actual sound like an audio tape nor does it digitize the sound like the digitally recorded music on a compact disk. What it *does* do is store the information about the music and how it is played.

A device and/or program that records these sequences of MIDI events is called a "sequencer," and the information it records is called, simply enough, a "sequence."

By following this established standard for communicating musical information, it is possible to link together a number of devices to use and distribute this information.

Looping and linking

While it's easy to see how you could send information from one device to another, how do you send information for one device to play and not have the data used by any others connected? This may not seem to be a problem until you look at how MIDI works. Many times a MIDI-linked system is daisy-chained in a loop. If, for example, you were using your ST acting as a sequencer, and you had a keyboard and three black-box synthesizers (meaning synths *sans* keyboards or other manual input controls), you would hook them up according to how you intended to play and playback the sequences.

Because you have only one keyboard to play on, you make that the start of the chain, running a cable from its OUT port to the ST's MIDI IN. Another cable would go from the ST's OUT (or THRU if you have a splitter) to one synthesizer's IN, from its OUT/THRU to the next, and again THRU it and to the third. If you wanted to eventually use the keyboard itself in the playback, the OUT/THRU from the last synth would be connected to the keyboard's IN.

Essentially, we have a big loop. You might think that sending information out from the ST would pass through all the connected devices, which would all use it.

This would be true if MIDI passed information on only one channel. It does not. MIDI has 16 separate channels for information exchange. Thus the simple solution to the above problem is to set the various synthesizers to play only the data on a given channel. One plays what's on Channel 1, the next Channel 2, and the last Channel 3.

Now, back to the one-person band analogy. If you have a sequencer (either a dedicated piece of hardware, or, more likely, some software and your ST) and two or more MIDI-compatible instruments, you can can become a one-person band. Let's go back to the setup mentioned previously, with a keyboard, ST, and three keyboardless synthesizers hooked up in that order. You do not "close" the loop by hooking the last synth in the chain to the keyboard, because it would then receive the same information it transmitted and try to play two notes for every one. You could do this if your keyboard has no sound-generating capability of its own, or if it has a "local off" mode, which means it won't play the keys you press, but just transmit the MIDI data for that action.

First you select some sounds on the various synthesizers. No. 1 is a guitar sound. No. 2 is a bass. No. 3 is a drum. You then set each one to respond to a specific MIDI channel. OMNI mode means the device plays everything received on all channels! If you can turn off the sound on your key-

board, you might want to do it so its own sound doesn't clutter the sound of the synthesizers.

To start, you set your keyboard and ST sequencer to the channel the synth with the drum sound uses. You work out a percussion pattern on the keyboard, hearing it through the appropriate synth (if the sequencer has a "software thru" option or you have a THRU interface attached), and record it with the sequencer software on your ST, which is set to the MIDI channel for the drum machine. You make certain that "sequence" remains assigned to the drum machine for playback.

When you're done with that, you set up to record the sequence for the synth playing bass. You then start playback of the drum sequence and play the bass part on the keyboard in time with the percussion. When you play both channels back, you have two instruments. You then repeat this procedure, setting the keyboard to the channel of the synth playing guitar, playing back the drums or bass while sequencing the guitar part.

You might want to use the keyboard for a part (assuming it too is a synth and not just a keyboard). You then hook it up as the end of the chain (keeping the local mode off), select a sound, assign it a channel and sequence the final part.

When you're finished, you play it back and *voila!* You are playing all the parts in a rock combo.

This represents a pretty simple example of how MIDI can be used. If that's all MIDI did, it would be interesting enough, but there's more.

Increasing frequency

From what I have thus far described, you might assume that you need to be a keyboard player in order to use MIDI. Not at all. Not only are there more and more MIDI-compatible instruments appearing all the time (like synth-axes, digital drums, and even wind instruments), but you can also assemble the music through software, a note at a time, if necessary.

Let's assume you're familiar enough with a piano keyboard to know Middle C from F sharp. Let's say you can play a song properly at half the actual tempo, but goof up if you try to go full tilt. Using a synthesizer you play a song at half speed, recording the music on an audio tape while simultaneously sequencing it.

When you finish the song, you play it back, first on the tape, then using the sequencer. Both are no good at that speed. Okay, since the recording is at half speed, if we play it back at *double* speed it should be at the correct speed, right?

We play back the tape at double speed. Yes, it's the right tempo, but something's wrong. The sound is too high-pitched. Your keyboard sounds like a piano belonging to Alvin and the Chipmunks. And, if you had sung along with the tune you were playing, you'd notice that you, in fact, sound like Alvin! Why is this? Because when the speed is increased, so is the frequency. The higher the frequency, the higher and squeekier the sound. That recording is useless, unless you want to audition for the voice of a chipmunk.

Conversely, if you had played the tape back at half the frequency, it would have gone *down* in frequency, and everything would sound deep and low. In such a case, if you'd sung in a falsetto, the playback would make you sound like a baritone in a deep well!

A MIDI port is a small five-pin DIN plug that is the standard interface connector for MIDI-compatible devices. There are two of these on any ST: an IN port and an OUT port.

After burning and burying the tape in an unmarked grave, you proceed—cringing all the while—to double the playback speed on the sequencer software and listen to it (it didn't record your singing luckily). Momentarily, you unclench yourself as you realize the playback sounds more or less right. The tempo is correct, the notes are correct, and the frequency is not in chipmunkland.

This method works because the sequencer stores note-on and -off events, the velocity and the time between them. When you double the playback speed in MIDI all that occurs is that the time between note events is halved. The speed is doubled with no change in frequency. In this manner, even a mediocre player can make recordings where he/she seems to play faster than a master. This, of course, does not mean you'll be anywhere near as *good* as a master—just faster.

If you don't know your way around a keyboard or other MIDI-ready instrument, there's still hope, because while you may not know how to tickle the ivories or strum the strings, you can make music on your ST and then send it to whatever MIDI-compatible device(s) you have.

This is what I call the "sheet music" approach. You sit down with a piece of software that lets you assemble a musical score, and do so. Some programs work like real sheet music, where you put notes, ties, rests and such on a staff. Some other do things like let you "paint" music by drawing. The position of a drawn line on a staff indicates the note, and the length of the line determines its duration. I used the music-staff approach to get a listen to a Tom Lehrer song for which I had the sheet music and lyrics, but which was not featured on any of his albums. The song was called "I Got It From Agnes," and it was considered too "naughty" to put on a album at that time. On the sheet music it states that the song should be "sung infectiously." I input the notes into an appropriate program, switched on the synthesizer, and got a good laugh as I put the lyrics to the music and imagined how Mr. Lehrer must have sung the song.

Editing

Once a sequence has been recorded, you can do more than just play it back. Like a word processor, with a good sequencer program you can actually *edit* your sequences. You can alter the tempo; copy, move and delete segments of the music; change notes; and so on. Thus, if you had sequenced a song in the half-speed manner described above, but found that there were some bumpy spots where you held a note too long or such, you could edit these problems out. Shorten the note, fix the tempo, adjust the timing. You could copy one instrument's part, modify it and assign it to another instrument. The possibilities are wide open.

Some MIDI software packages have features to fix many errors automatically. One piece of jargon bandied about a lot that few people understand is "quantizing," which simply means "automatic timing correction." If your rhythm is off in a sequence, and if the sequencer software you are using features quantizing, activating that function will fix the problem in most cases.

Other software tools

While sequencers are probably the most common of MIDI programs, there

are other categories with tools for specialized functions. However, most of these can (and maybe should) be used in conjunction with a sequencer or other program.

A scoring or score printing program is simply a tool for producing a musical score in notation forms based upon a MIDI sequence. Scoring programs are particularly useful for people who can play an instrument, but cannot or would rather not write the notation out, particularly when they need such sheet music for distribution to other musicians, publication, etc. Some scoring software is powerful, a lot of it just marginal. No matter how good the program, chances are that it won't produce "perfect" sheet music automatically. You may have to add ties, rests and other details after it has arranged the notes on the staffs.

There are also "patch" programs that fall into two related categories of "patch editors" and "patch librarians." Many moons ago, the sounds on synthesizers were hard-wired by connecting various units together with patch cables. To change the sound, you had to reroute the cables. These arrangements came to be known as "patches," a term which has carried on through the age of switchable ROMs and RAM-loadable sound-parameter data.

A patch editor is a program used for creating a patch (sound) for a given synthesizer. These programs are instrument specific. For example, the *DX Android* program works only with DX series synthesizers. This is because, MIDI or not, the internal arrangement of synthesizers and the way they make sounds is often as different as the way different brands and models of computers work and store data. The patch editor programs allow the user to build new sounds from scratch or edit existing patches. Since creating or modifying a patch on many synthesizers by themselves is done by fiddling with buttons, sliders and a few rudimentary displays, doing the same work in a GEM environment is much easier.

A patch librarian is a program that stores a database full of patches and that can "download" the selected ones to the synthesizer via the MIDI connection. This is useful in a performance situation, where you want to be able to select and load patches quickly and easily. Switching ROM cartridges or downloading the patch data from a cassette (as some keyboards can) is not the most elegant solution. Like the patch editor, many librarians are synthesizer-specific. A number of them are more generic and can work with a number of different synths. Of course, just because you can use the program with different synthesizers doesn't mean that the patches from one are compatible with the other!

No ultimate solution

On the surface it looks like, with MIDI, anyone can be a musician, and anyone with enough instruments can be a one-person orchestra. Sure, it looks this way, but is it really the case?

First of all, giving the best paint program in the world to someone with no drawing ability won't make him an artist. Likewise, all the technology in the world won't make a nonmusical person a musician. Just because someone can punch notes into a program and have it play them back flawlessly does not make him a George Winston. You still need musical ability to create music and to be able to perform it. No technology in existence today has eliminated that.

But you hear musicians say that MIDI has cost jobs. Has it? Actually, yes. The increasing use of drum machines and albums produced and played by a single person surely have limited the opportunities for many professional musicians. The question that arises is: Is that such a bad thing? Do we lose anything worthwhile is the process?

The answer to both is a qualified "yes." While MIDI is a marvelous thing, it is still not the same as a real musician, even if the sequence was made by a musician. MIDI times things to a computerized clock, starting and stopping events to the "ticks" of that clock. Those ticks, small as they may be, are still a little too big. If a musician hits a note between those "clicks," it's recorded at the next one. Then again, the data MIDI handles has its limitations. Nuances are not always accurately reproduced. All but the most expensive drum machines are even worse. Very few actually sound like a real drummer. There is something missing, because the sound is still too mechanical.

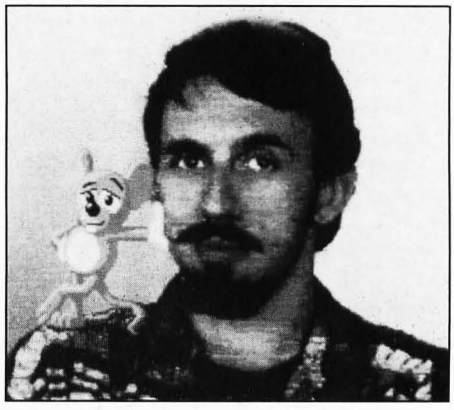
And, in a lot of cases, MIDIfied music often lacks feeling. And feeling is *everything* in music.

Is this a problem with the technology or the users? Both, really. The technology still has a ways to go before it can capture the subtle nuances beyond the current scope of MIDI. But MIDI is fairly accurate in most cases, and many times only a trained ear can hear the difference. Perhaps that is part of the problem with MIDI: It is too accurate. How many musicians can time their playing to fractions of a second, making each and every interval exactly the same length as every other? Yet a computer can and does this routinely, and many people have taken to letting the machine do that. Even when their timing is not really off, it's so easy to hit "quantize" to get rid of any and all timing errors that a lot of people do. Their timing becomes perfect, but their music starts to sound mechanical.

But, at the same time, MIDI is bringing the ability to make music and *play* with it, into the lives of people who otherwise would have no contact with music, other than listening to it. So this gives these people an interest in music that may lead to a more serious pursuit of it. It allows musicians with limited resources to produce music with a complexity that they might not otherwise be able to achieve. It allows professionals to enhance their capabilities and do even more than their already considerable talents allow.

I suppose that MIDI is to music what digital-image processing has been to the graphic arts. It is seen by many as a threat, and by others as a godsend. It is neither, really, but occupies a less dramatic middle ground. It is a *tool* that can expand the capabilities of people to make music, and as such is an important *part* of the answer to music making in our modern age, but it is not *the* answer unto itself.

Author's note: Special thanks to John Eidsvoog (cofounder of CodeHead Software and author of the *GenPatch* and *HybriSwitch* MIDI programs from Hybrid Arts) for his invaluable help in providing information for, and weeding errors out of, this article.■



Blissfully ignorant of the realities of time and space and plain old common sense, Maurice Molyneaux hopes someone will someday discover "retroactive *reincarnation" so that when he dies he can come back in a previous life as animation director Chuck Jones. His greatest fear would be to come back as Wile E. Coyote and, in the process, have to learn some humility.*

Attention Programmers!

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

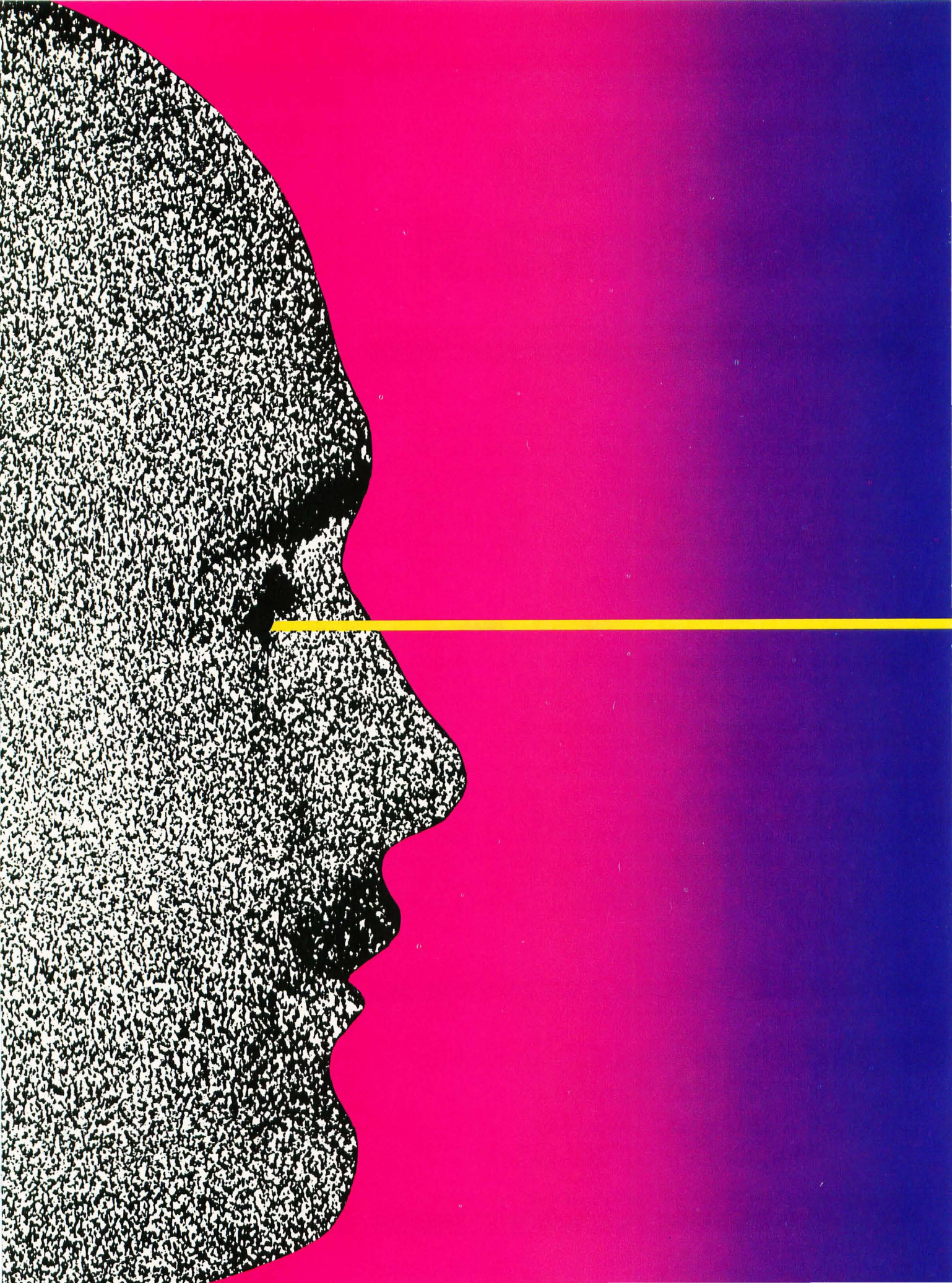
All submissions for publication, both program listings and text, should be provided in printed and magnetic form. A typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ST-LOG Magazine**, authors acknowledge that such materials,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

Send your programs and articles to:

ST-LOG Magazine

P.O. Box 1413-M.O.

Manchester, CT 06040-1413



TUTORIAL

SOFTWARE ENGINEERING

BY KARL E. WIEGERS

Is There a Doc in the House?

Some of you may be tempted to quickly turn the page and pretend you didn't see this article about writing software documentation. It will do you no good. The ostrich approach may help you get by for a little while, but documentation is a vital aspect of any serious software development. As budding software engineers, we must move beyond the slipshod software documentation characteristic of our sordid past. This article will help you with that sometimes distasteful but always essential process.

Why document?

I could sit here and claim that software documentation is good for you, that it makes you grow big and strong. You probably wouldn't believe me. The real truth is that documentation slows the rate of hair loss among computer programmers (something those of us old enough to run for President sometimes think about). Honest.

The reason for this is that the dreaded chore of software maintenance usually results in the victim tearing out his hair by the handful, whereas thorough software documentation can make the task of changing existing programs vastly easier. Not to mention the additional benefit of lower dental bills due to reduced gnashing of teeth. And the sad fact is that we may spend as much time fine-tuning and debugging programs that we thought were "done" as we do whipping up new ones.

Although you probably think of "software" as something you run on a computer, it's really much more than that. The "deliverables" (that is, end products) of a software development project include source and executable code, of course, but also critical components such as associated data files (e.g., GEM resource files) and user manuals. Test cases also represent a software deliverable. And so does documentation. Anyone paying for software you write has a right to expect specific kinds of documentation, and software development contracts frequently include such a stipulation.

The usual excuse is "I don't have time to write documentation; I have to get on to the next project." To my way of thinking, delivering (or otherwise declaring complete) a software package that isn't properly documented is the moral equivalent of omitting everything to the right of each equal sign in your program because you didn't have time to complete the statements. If you think of it as "technical

writing" rather than dullsville documentation, perhaps you'll have an easier time convincing yourself to find the time to do it properly.

Another reason cited for omitting documentation is that the comments take up valuable file space, and they can actually slow the execution of interpreted languages. This is certainly true of small computers such as 8-bit Ataris. However, you ST users can't slip by with such a flimsy excuse, since your disks hold lots of bytes, and you're probably using compiled languages that ignore comments. Nope, you just can't fool me with those old excuses.

As you read the rest of this discourse on documentation, keep in mind that our overriding goal is communication. Anything that improves the effective communication of vital information about your software to another person (or to you, after time has passed) is *good*. Anything that inhibits that communication, either by omission, error or redundancy, is *bad*.

A philosophical aside

In the scientific world, an experiment is not considered valid unless it can be duplicated by another scientist skilled in the art (science-ese for saying that the second scientist knows how to do the same things that the first one does). Of course, to accomplish such replication requires that Scientist Number 2 have access to a complete description of how the experiment was performed by Scientist Number 1. This is why documentation of scientific research is so important (and hence voluminous). Such documentation is usually done in the form of doctoral dissertations or papers published in the recognized literature. I confess to having contributed one of the former and ten of the latter to the already enormous volume of chemical research literature.

Recall that the thrust of this series of articles is that the time is here to turn software development into an engineering process, as distinguished from an art form. Along these lines, I propose that the measure of when a piece of computer software is truly complete should include a new criterion: The software should be readily modifiable by another programmer skilled in the language used. Successful modifications of existing software are greatly facilitated by thorough documentation. We should all design, write and document our programs with an eye toward maintainability.

Now on to some more concrete thoughts.

Flavors of documentation

You may be relieved to learn that software documentation comes in not 31 but just three flavors: program, system and user. The amount of effort you devote to each type of documentation depends upon the nature and scope of the software project, as well as upon its intended users.

Program documentation refers to module-level descriptions of the components that make up your software system. This is sometimes referred to as "internal" documentation, since the descriptions are usually in the form of comments imbedded in the source code files. Of course, additional documentation outside the source file may be necessary, depending on the nature and complexity of the modules involved.

System documentation describes the overall software system and how it connects to the rest of the world. How do the executable components of the system fit together? What data files are used by the program modules, and what do they look like? How is the system structured? How do your source code modules relate to the system design, and which modules satisfy which specifications from your analysis phase? The answers to these questions, and many others, should be found in your system documentation.

User documentation might be in the form of a separate printed manual, quick reference cards, online help files or a magazine article. Writing good user documentation is a skill that takes much practice; we've all encountered manuals written by people who seem to be early in this practice phase. David L. Coles had a lot of valuable advice on writing good user documentation in his article "Read Any Good Docs Lately?" in the July 1988 issue of ST-LOG.

One other useful piece of advice concerning user guides is that the first draft should be written before the program is written. This is a good way to make sure your system design will in fact satisfy your vision of how the program is supposed to work from the user's perspective. In a sense, this is another way to define the specifications of the software system.

Program documentation

I will use the terms program, module, source code and internal documentation pretty much interchangeably. In each case, I refer to the textual information that sheds more light on the purpose, structure and function of a source code module than can be gleaned simply by examining the code itself. When this text information is included right in the source code files, I call it "internal" documentation. In contrast, system documentation is generally found on paper or electronic media separate from the code, so I refer to it as "external" documentation.

Internal docs are imbedded in the source file in the form of comment statements, a feature present in all programming languages I know of. It is most convenient to collect a large chunk of pertinent information (which we'll identify momentarily) into a single big comment block, which can be placed either at the top or the bottom of the source file, whichever seems most comfortable to you. I'll call this the header information, no matter where you decide to place it. Additional details can be presented as in-line comments, located just ahead of the section of code to which they pertain.

It's important to develop a consistent style for your internal documentation. This makes it easier to read, when you know what to look for. For example, I prefer to place my in-line comments in little comment blocks, like this (the /**/syntax delimits comment lines in C and several other programming languages):

```
source line
another source line
(blank line)
/*-----------------------------------------*/
/* here is my comment block                */
/* it could be several lines long          */
/*-----------------------------------------*/
(blank line)
yet another source line
still more source lines
```

An alternative supported by many languages is to place the comments to the right of the source code statements, like this:

```
source line                 /*-----------------------------*/
another source line         /* comments go to the right of */
yet another source line     /* statements being commented  */
still more source lines     /*-----------------------------*/
```

This approach preserves the readability of the source code itself, but it has some disadvantages. The comment blocks can become fragmented if new lines are inserted; the number of characters available for comments might be limited (depending on the logical line length of your source code editor); and this style doesn't help break the code into logical blocks that your eye can distinguish at a glance. The choice is yours, but remember the value of a consistent style.

Head 'em up

Let's talk about what kind of information should be incorporated into the header comment block and in-line comments. Figure 1 shows a more or less standard header comment format that I use

for most of my program modules. To accelerate the documentation process, you could create a text file containing the general structure of this header block and simply imbed it at the top of each program source file to remind you what information to supply. This template approach also contributes to our goal of a consistent documentation style.

The header information contains the same sort of stuff that a newspaper reporter puts into his articles: what, why, how, who, when and where. Let's see what I mean by this.

What: Supply the name of the program, input the parameters it takes, error codes it returns, external files read and written, and any other auxiliary files that are needed (such as PIC or RSC files). A list of the internal procedures (subroutines and functions) called by your program, and a similar list of any external procedures called, should be supplied. By internal procedures I mean subprograms that are contained within this same source code file; external procedures reside within source files other than the one we're documenting at the moment. Don't just list these subprograms—also supply a short description of what they do. The details of how the subprograms work can be left to the header comment block that I'm confident you'll write for each separate procedure in your system.

It's not a good idea to include a list of the other programs that call this module. That can change in the future, especially if you've cleverly written a module with good potential for reuse in other software systems. Each time you do take such a step down the reusability path, give yourself a gold star.

A very important component of "what" is a list of all the variables used by the program. I construct separate lists for array and scalar (non-array) variables, indicating the type of variable (integer, character, double-precision, etc.) and the dimensions of array variables. Each variable listed should have a definition sufficient for someone unfamiliar with the program to be able to figure out what it is for. List any variables that are global to the whole software system or otherwise shared among modules in a common storage area. Finally, I sort the variables in each of these lists alphabetically.

Why: A brief statement of the purpose of this program is important. Indicate what software system it belongs to, or whether it is the main (or sole) source program for a system.

How: Define the usage syntax of the program. Illustrate how it would be called from another program (if it's a subprogram of some kind), or any input parameters it might require (if it's a standalone program).

The "how" section should also include some insight into any nontrivial algorithms used in the module. This shouldn't be a simple restatement of the code, since you can assume that anyone skilled in the language used could follow that. But explain any tricks that were used, or any particular mathematical techniques employed. References to books or articles containing more detailed descriptions are excellent substitutes for showing great detail in the internal documentation.

Who: Of course, include your name as the author of the program. If someone else actually owns or is responsible for maintaining the software, their name(s) should also be shown. I do most of my work on a huge IBM mainframe computer. It really burns me to want to speak to someone about a program on a public disk and not be able to find out who wrote it because of inadequate internal documentation.

When: State the date the program was written. Include a modification log, indicating when changes were made to the code, by whom and why. It also burns me to look into a public program that all of a sudden started giving me trouble, just to find that the date on the file is more recent than that of any of the comments in the file. This makes it much harder to figure out what changes were made in a time frame that might explain the problem.

Where: Where does this program fit in with other related programs, as part of a system? (Okay, I stretched the reporter metaphor a bit to come up with a "where"; give me a break.)

Get in line

The other kind of internal documentation consists of in-line comments, sprinkled judiciously throughout the source code. And I do mean judiciously. There's no point in duplicating in words that which is readily learned by reading straightforward, well-documented code. Instead, contribute something extra with your comments. Sometimes just providing another "view" of the computer code in the form of a free-form description in English, can be illuminating, but including pseudocode equivalent to the actual program statements doesn't add much insight.

In a source file that contains several procedures, I will *always* have a brief description at the top of each separate procedure. This paragraph or two explains the purpose of the procedure, and it also serves to visually break up the printed program listing into blocks. Your brain likes that sort of "chunking." Any new or unusual variables local to that procedure should also be defined here.

Again, a consistent comment style is important for readability. Don't let the source listing get too cluttered, or the communication-enhancing aspect of documentation may backfire on you. Rather than writing a comment line before (or appended to) each line of code in a section, insert one short comment block before that section of code, explaining the overall purpose of the section. The one exception is assembly language, where the source lines are short enough (and obscure enough) to benefit from comments on nearly every line.

There's another important point about program documentation we shouldn't overlook: It should be correct. This seems obvious, but there's a subtle trap here. Any time the program is changed, you should review the internal documentation to see if it requires updating. Algorithms may change, new variables may be added or sections of code may be deleted entirely. The only thing worse than no documentation is erroneous documentation. If you see some documentation in a program, you tend to use it as an aid to understanding and perhaps modifying the program. But if the comments are obsolete, misleading or contradictory, you don't know whether to conclude that the code is wrong or the comments are wrong. Either way, your valuable time and mental energy are wasted.

System documentation

As I mentioned earlier, system or external documentation consists of additional information pertaining to the overall software system, stored in the form of printed documents or computer files separate from the code itself. The external docs help the reader understand how the different pieces of the system fit together; they should be designed so as to give future programmers enough information to let them successfully and efficiently modify your programs. Let's look at some items I regularly include in my external documentation.

1. Begin with an overview. Why does this system exist? You should already have such a statement of purpose from the very

beginning of your structured analysis (remember structured analysis?) phase.

2. In what environment does this software run? State any hardware or software constraints, such as "requires an Atari ST or Mega ST with at least one megabyte of RAM, TOS in ROM, a blitter chip, and two double-sided floppy-disk drives." Are there other programs, such as those that are part of the operating system, that this system uses? Are there restrictions on folders where executable or auxiliary files can be placed? Are there any known conflicts with other programs or operating environments ("no desktop accessories can be resident," or whatever)?

3. The system specifications, again from your analysis phase, should be included. If you have data flow diagrams, toss them in too. Remember that if you use such items as part of your system documentation, your maintenance tasks should include updating the specifications and diagrams to correspond to actual code changes. Sometimes this can be facilitated by using the computer-aided software engineering (CASE) tools we discussed last time. Never forget that incorrect documentation is worse than no documentation at all.

You'll want to include the context diagram, data flow diagrams and any relevant printouts of your data dictionary from the system design (as distinct from specification) phase also. I usually toss out the process narratives once the modules have been written. It's too much of a chore to keep them current when the code is changed, and they really aren't good for much once the modules themselves are written and debugged. The exception would be if you are using a code generator (lower-CASE tool) to create code directly from your process narratives, but it will be several years before very many of us are operating at that level of software engineering technology.

4. I like to include what I call a "requirements trace" in my system docs. This is essentially a table in which I list every one of the numbered items from my written system specification, along with the data flow diagram number, process specification number, and program module name that satisfies each specification. An example is shown in Figure 2.

The requirements trace accomplishes a couple of useful things. First, it lets me verify that I have in fact addressed each of the requirements in the specification. Have you ever got part way through a project, only to realize that you forgot to include one of the great features you had been planning to have in there? Talk about a sinking feeling in the pit of your stomach!

In addition, this cross-reference of specification with actual system components comes in very handy during maintenance. Suppose you had originally planned to handle documents up to 100K long in the World's Greatest Word Processor that you're attempting to sell to Gigundo Software, Inc. Gigundo says, "We love your program, but before we can pay you a $1 million advance against royalties, it must be able to handle documents up to 101K." Well, you just find the specification that pertained to this 100K limit, turn to the requirements trace to find out what parts of your design and which module(s) are involved, and you know just what parts of the software need to be changed to make Gigundo happy.

5. I include a list and a short description of all of the program source files in the system, grouped by language used. This short description might be the same as the "Purpose" section of the header comment block in the source files we discussed earlier.

Figure 1

Partial Requirements Trace for Reaction Time.

Specification	Data Flow Diagram	Process Specification	Module Name
1.0	1.0	1.1	SHOW_MENU
2.0	1.0	1.2	MAKE_CHOICE
3.0	2.0	2.1	OPEN_FILE
	2.0	2.2	READ_FILE
	2.0	2.3	CLOSE_FILE
4.1	3.1.1	3.1.1.1	READ_JOYSTICK
4.2	3.1.1	3.1.1.2	MOVE_FLASK
4.3	3.1.1	3.1.1.3	FIRE_BUTTON
5.1	3.2	3.2.1	SELECT_CMPD
5.2	3.2	3.2.3	ADD_TO_EQUATION
6.1	3.3	3.3.1	SELECT_COEFF
6.2	3.3	3.3.2	ADD_COEFFS

Figure 2

Sample Header Block of Internal Documentation.

```
/*********************************************************************

     Program Name:  REACTION.SRC

     Purpose:  Main source file for Reaction Time system.  Sets up playing
               screen, lets user build equation using joystick, lets user
               balance equation, calls procedures to judge equation and
               change score.

     Written By:  Karl E. Wiegers

     Date Written:  July, 1985

     Modifications:
     --------------

       Date:                                   Programmer:
       Purpose:

     Arguments Passed:  none

     Return Codes:  0 - no errors
                    1 - joystick not plugged in

     Internal Procedures:

       SETUP_SCREEN       - set up 4 playing screen areas and borders
       MOVE_FLASK         - move flask around with joystick, within bounds
       BUILD_EQUATION     - add or replace formulas in equation line
       ADD_COEFFICIENTS   - add or replace coefficients in equation line

     External Procedures:

       EVALUATE - see if the equation built is known, valid, and balanced
       UPSCORE  - increase score if equation is correct, make sound
       DNSCORE  - decrease score if equation is wrong, make sound

     Array Variables:

       COEFFS(4)      - coefficients available for use (2, 3, 4, 6)
       COMPOUNDS(15)  - formulas of compounds available in current
                        reaction set
       RXN_COEFF(4)   - coefficients placed in current equation
       RXN_CMPD(4)    - formula numbers placed in current equation

     Scalar Variables:

       COLOR  - color to be used for next print statement
       DONE   - number of equations in current set found so far
       I      - index variable
       SCORE  - current score
       STICK  - deflection direction code for joystick
       TOGO   - number of equations in current set yet to be found

     Files Read:

       RXNDATA.X - reaction data file for selected set; main menu has 7
                   sets to choose from; 'X' = the set number chosen

     Files Written:  none

*********************************************************************/
```

6. A list of the data files read or written by the programs is essential. You should also include detailed byte-by-byte descriptions of the record formats in each file. This can be very useful when trying to track down elusive bugs. If your system accesses any true databases, the field descriptions of records in the databases should also be included here. Any additional files, such as resource or picture files, should be listed, as well. You might prefer to include these in the "Environment" section described above. Either place is fine, so long as it's perfectly clear to the reader just what files comprise the entire software system.

7. Think back to our Gigundo example. How can you convince yourself that the changes you made to your word processor to make Gigundo happy didn't introduce several inadvertent bugs? By running through your test cases again, that's how. And if a Gigundo programmer must make future changes, he'll want to look in the system documentation you supplied him to find a list of the appropriate test cases to run, test data files to use and representative output from these tests. This whole concept is related to the idea of software quality assurance, which will be the topic of a future article.

8. I like to keep a copy of the user's guide in with the other system documentation, too, for quick reference.

9. Another very useful piece of documentation for a complex system is what I call a "module hierarchy diagram." This is basically a list indicating the program modules called by each module in your system. I use an indenting scheme to indicate these dependencies. Here's an example:

```
Main Program
     INITIALIZATIONS
     SHOW_SCORES
     PLAYING_SCREEN
          SCREEN_SETUP
          SHOW_FORMULAS
          BUILD_REACTION
               TOP_SECTION
               COEFFICIENTS
               REACTION_LINE
          EVALUATE_REACTION
     CHANGE_SCORES
```

Each of the capitalized items represents one program module in a software package. The main program calls the modules indented one level (INITIALIZATIONS, SHOW_SCORES, etc.). Similarly, those modules call others indented one additional level: PLAYING_SCREEN calls SCREEN_SETUP, SHOW_FORMULAS and so on. I'm sure you get the picture.

A procedure hierarchy diagram is useful for tracking down bugs. If I see a problem with the program that I know is appearing in, say, the COEFFICIENTS section, I can use this diagram to trace a possible path by which control could have passed into COEFFICIENTS, and thereby speed up my search for the culprit code that introduced the problem. I find these diagrams particularly helpful if I have to go back and tweak an old program after enough time has passed that I've forgotten its detailed structure.

Suggestions

Whew! You're probably overwhelmed by the magnitude of what I consider adequate documentation. In reality, you are not likely to go to this much effort except for commercial-scale projects. But this doesn't mean you can completely neglect the documentation issue. Rather, select from the components I've suggested to come up with what you believe to be satisfactory for each of your own projects. And don't leave it all till the end, when you're sick of the whole thing and anxious to get on to something else. Do it as you go, and the chore won't be quite so onerous.

What else can we do to make the documentation task less of a hassle? I've suggested building templates for internal documentation, thereby saving you both some typing and some thinking for each program module. My guess is that your name doesn't change from one project to the next; you can hard-code in constant information like that so you don't have to type it each time.

If you want more ideas about formats, you can buy entire books of forms suggested for use in creating external documentation. It's kind of staggering to see the detail that some data processing or software development shops collect in the way of documentation. You can easily wind up with more pounds of explanatory paper than you have source code. But these books can give you valuable ideas about formats for describing file structures, maintaining change histories, and so on. One suggestion is *Standards and Procedures for Systems Documentation*, by Andrew W. Poschmann (Amacom, 1984). Manual preparation ideas are the topic of *Software Manual Production Simplified*, by Richard Zaneski (Petrocelli, 1982).

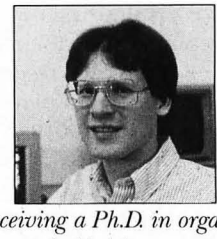
Make sure you have access to adequate word-processing software, since you'll be doing a lot of typing as you create documentation. If you wish to write particularly sophisticated user guides, you should consider the powerful desktop-publishing packages (and, of course, laser printers) that are now available. You can think of desktop publishing as one variety of CASE tool, since you're certainly using the computer to facilitate creation of some of your software deliverables.

Here's a more innovative thought: why not let the computer handle some documentation of individual source modules semi-automatically? I do a lot of programming in two languages on an IBM mainframe computer, REXX and FORTRAN. Some time ago, I wrote two programs, REXXDOC and FORTDOC, which do just this for me. Their purpose is to process a source file in the appropriate language and create a first draft of the internal header comment block illustrated in Figure 1. They create the template of prompts to remind me to enter the purpose, date, and so on. And they automatically list all of the program variables, classified by type (array and scalar) and sorted alphabetically, as well as the internal and external modules called by each source program. Not bad, eh?

I'm not sure, but I suspect that an enterprising software engineer who writes a program called CDOC to do the same sort of thing for C programs on the Atari ST just might find a market for it. I throw this thought out as a challenge. Of course, if you write CDOC in C, the first test case can be CDOC itself. This idea of automating the documentation aspect of software development is part of my basic computer philosophy: Ask not what you can do for your computer; ask what your computer can do for you.

Postscript

You think I'm being tough on you because I ask you to throw a few lousy comments into your programs? The Boeing Company estimates that it produces some two billion pages of software documentation each year. But if it helps keep the 747 in the air, I'd say it's worth it; wouldn't you? ▪



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

THE ST AND THE ROLAND JUNO 106 SYNTHESIZER (or: Yep, It's Got MIDI. So What?)

BY MICHAEL FRIESEN

This article will show you one way to use the MIDI Out port on your ST. I'm assuming that you have a Roland Juno 106 synthesizer and at least one MIDI cable. Even if you don't, the information here will help you better understand the arcane world of MIDI system exclusive codes.

One of the most unfortunate things about the J-106 is that, once you've programmed a voice, it's gone. Sure, it's still in memory, but can you remember which parameter is affecting what? That makes tweaking parameters a matter of guesswork, because it's not always clear what modulator is creating which effect. And fine-tuning parameters is nearly impossible with analog sliders.

The answer, of course, is to spend a hundred bucks on a patch editor/librarian that will let you program the synth from your ST and store hundreds of extra patches on disk. But a hundred bucks could probably be used for more useful things. And wouldn't it be fun to get your hands dirty?

The fact is that you can easily (and I mean *easily*) put together a simple program that will let you specify exactly what values to assign to which parameters.

In addition, if you have a sequencer that allows you to transmit system-exclusive information, you can embed not just program changes, but complete programs in your sequences. That means that if your J-106's memory gets wiped, you'll still be able to play your sequences with the original voices.

The brass tacks

Because the Juno 106 has such a rudimentary voice structure, it doesn't take a lot of numbers to set up a sound. So the real work is figuring out the MIDI patch-dump command sequence. The inside back cover of the J-106 owner's manual gives the whole MIDI implementation. For the purposes of this article, we're interested only in "Part 3: Exclusive Messages."

When we want to send system-exclusive data to the J-106, we have to transmit a certain series of commands before we send the data, and then we have to know what the order of the parameters is. But before we mess around with control codes, we have to make sure that the J-106 is set up to receive them. The first step is to set the function switch on the back of the machine (off to one side of the MIDI terminals) to III; that is, all the way *away* from the terminals. Now the J-106 will accept controllers, notes and system-exclusive data.

Next check to see what channel the J-106 is set to transmit and receive on. Remember this number. Write it down.

In Section 3.1 of the "Exclusive Messages" section on the MIDI Implementation page of the J-106 owner's manual, you'll see a column of letters, binary numbers, hexadecimal equivalents and a short description of what all of the above mean. Because the numbers are in hexadecimal, and the MIDI Outs like to have data sent through in decimal, we'll have to translate from hex into decimal notation. It's a good idea to learn how to do this. If you're not sure, I've included decimal equivalents for the hex codes, and there's a sidebar for converting binary to decimal numbers.

A. $F0 (240)—This is the value that tells the J-106 to expect system-exclusive data.

B. $41 (65)—Tells the J-106 that the data is coming from a Roland machine—which is not exactly the case. The data is coming from an Atari ST speaking "Rolandese."

C. $30 (48)—Tells the J-106 to expect all parameters for a single voice.

D. $00-$0F (0-15)—This number is equal to whatever channel on which the J-106 is set to respond *minus 1*. Even though the display reads 1-16, the machine will expect a number between 0 and 15. So when you press the MIDI Channel button, subtract 1 and enter that number here. For example, if the display reads 3, then enter 2.

E. $nn—The source of the data. Not terribly important for our purposes. Set it to 0 and forget about it.

F. The next 18 numbers are the actual program data. More on this below.

G. $F7 (247)—This tells the J-106 that the data transmission has concluded. In effect, it hangs up the system-exclusive telephone line.

Program Data: Section 3.4 of the MIDI Implementation gives a complete rundown of the order of parameters. In typical lab-coat fashion, they list them from 0 to 17. If it helps to think of them in terms of 1-18, go ahead. All it means is that the first number you send will set the LFO Rate, the second will set the LFO Delay Time, etc. I recommend that you make a chart such as the one below.

```
LFO Rate     =
LFO Delay    =
DCO LFO      =
DCO PWM      =
Noise Level  =

VCF Cutoff   =
Resonance    =
VCF ENV      =
VCF LFO      =
VCF KYBD     =

VCA Level    =

Attack Rate  =
Decay Rate   =
Sus Level    =
Release Rte  =

Sub Osc Lvl  =
```

Easy, huh? Just send the first bunch of information numbers, then plop in some parameter values, and you've almost got a complete voice. But now comes the tricky part.

Listed above are only 16 parameters. Chorus, wave, range and a few other settings are missing. The reason is bit-mapping of parameters. (For more on bit-mapping, see the sidebar.) It's a lovely concept, but considering that there are so few parameters in the J-106 voice, I wonder why they were so desperate to save such a small amount of memory. The fact is that the machine uses just over 2K of memory, and bit-mapping only saves about 1K.

The layout for the 17th parameter is shown in Figure 1. So if we want Chorus 1 on, we have to set the left-most two bits like this: 10. The left bit set to 1 means that Chorus 1 is selected, and the other bit turns it on. Obviously, if you set the second bit to 1 (OFF), then it won't matter which chorus you've selected with the first bit.

We turn the SAW wave on by putting a 1 to the right of the first two bits. Combined with the chorus data, this is how our byte looks so far: 101.

We want the pulse wave OFF, so we put a 0 into the fourth position: 1010.

The last three bits set the range. If we want an eight-foot range, we set the last three bits to 010. The whole seven-bit parameter now looks like this: 1010010. When we run this number though the brain-mill we get a value of 82.

The procedure for the last byte of data is much the same, only easier. The left-most bits set the high-pass filter. The next bit toggles the VCA from gate to ENV follow mode; the next sets the envelope polarity; and finally, the right-most bit sets the PWM source to either manual or LFO,

as shown in Figure 2.

In this case, a setting of 01110 would indicate that Filter 2 was being used, the VCA was set to gate, the envelope polarity is negative, and the LFO is the pulse-width modulation source. We throw this into the magic binary blender and come out with a decimal value of 14. Now that the work is over, it's a simple matter to dream up a patch, load your sequencer and throw that data in. The procedure is virtually the same for both GFA and ST (*gag*) BASIC.

Listing 1 is a simple program written in GFA BASIC that you can use to send patch data. As I mention in the sidebar, I hate programming. So if you want a flowery user-interface with bells and whistles, go ahead. This little program simply gets the job done.

Using this sort of a procedure to program your synth, you can exercise bit-resolution control over the parameters. You can fine-tune a sound until it's just right.

But don't forget: Once you've got a winner, print it out on paper or write the values into a voice chart. And whatever you do, don't forget to save the sound in the synth's memory. (And you do keep a memory map regularly updated, don't you?)

If you have a sequencer that will allow you to send system-exclusive data, you've got it made. Simply go into event edit, and insert the access codes, the parameters and, finally, the F7 EOX command. For the *MIDI Recording Studio* from Dr. T., Figure 3 is how the whole sequence would look.

When you're just jamming, the front panel is fine. And when you're under the red light, you don't want to waste time doing decimal conversions. However, there is no better way to make sure that your sounds are repeatable than to send them to the synth each time you play the song.

Leaving the patch inside the synth is fine until your battery dies. Or until someone accidentally spills beer onto it. Or until you write over it while doodling one day.

Besides, I've found that analog sliders simply do not have 127-value resolution. Working digitally, you know exactly what parameter has exactly what value. Short of buying a full-fledged librarian (simply a splurge for the struggling student), this is the best way to control your patches.

I've included some of my favorite programs in Figure 4 for use in this format. Try them out, tweak them parameters and have fun! Just remember, each time you send the data, you must first send the five identification bytes, the 18 parameters and the EOX Byte. ■

FIGURE 1

Chorus Type	Off/On	Saw wave	Pulse Wave	DCO Range
1: Chorus 1	1: Off	1: On	1: On	100: 4'
0: Chorus 2	0: On	0: Off	0: Off	010: 8'
				001: 16'

FIGURE 2

HP Filter	VCA Mode	ENV Polarity	PWM Source
1 1 : Off	1: Gate	1: -	1: Man
1 0 : 1	0: ENV	0: +	0: LFO
0 1 : 2			
0 0 : 3			

FIGURE 3

MSR-	ST	EVNT	TIME	CH	TYP	NOTE	VEL	DUR
1	1	1	0		*	240	(Sys Ex Begins)	
1	2	2	1		*	65	(Roland ID)	
1	3	3	1		*	48	(Function Type)	
1	4	4	1		*	0	(MIDI Channel 1)	
1	5	5	1		*	0	(Program #)	
1	6	6	1		*	57	- LFO Rate	
1	7	7	1		*	45	- LFO Delay	
1	8	8	1		*	0	- DCO LFO	
1	9	9	1		*	55	- DCO PWM	
1	10	10	1		*	0	- Noise Level	
1	11	11	1		*	85	- VCF Cutoff	
1	12	12	1		*	0	- Resonance	
1	13	13	1		*	0	- VCF ENV	
1	14	14	1		*	0	- VCF LFO	
1	15	15	1		*	25	- VCF KYBD	
1	16	16	1		*	52	- VCA Level	
1	17	17	1		*	59	- Attack	
1	18	18	1		*	32	- Decay	
1	19	19	1		*	86	- Sustain	
1	20	20	1		*	40	- Release	
1	21	21	1		*	0	- Sub Osc Level	
1	22	22	1		*	26	- Compound Byte # 1	
1	23	23	1		*	24	- Compound Byte # 2	
1	24	24	1		*	247	(End of Sys-Ex)	

FIGURE 4

Patch:	Whoodwhind	Metalcussion	Head Trip	BrassPad	Warble
LFO Rate	82	94	127	0	108
LFO Delay	40	0	0	0	0
DCO LFO	11	127	0	0	0
DCO PWM	0	102	102	62	0
Noise Lvl	0	116	0	0	0
VCF Cut	87	46	20	65	94
Resonance	27	0	125	39	127
VCF ENV	17	63	33	3	80
VCF LFO	0	9	0	11	23
VCF KYBD	56	84	127	127	127
VCA Level	127	0	31	59	0
Attack	7	0	0	50	0
Decay	127	95	127	100	127
Sustain	127	15	0	94	51
Release	6	45	127	44	71
Sub Level	0	0	0	0	0
Switches 1	74	2	68	82	2
Switches 2	11	17	1	25	26

Decoding Binary

Time for a revelation: I hate programs. Debugging bugs me. So instead of writing a program to convert numbers from one base to another, I've learned how to do it in my head. For those of you who still need to figure out how to get from a binary string like 01011101 to the decimal number 93, here's a quick lesson.

We've become so accustomed to the decimal system of numbers that it's difficult to really understand how numbers work and what they mean. The number 285, for example, is automatically understood to mean . . . well . . . 285. But what it really means is this:

$$2 \times 10^2 + 8 \times 10^1 + 5 \times 10^0$$

We understand that the 2 represents the value of 2 multiplied by 10 to the second power.

If this is all clear so far, you won't have any problem with binary. If it makes your head feel like arythmic tapioca, don't worry. It took me most of tenth grade to figure this out.

Now when we get a binary, the rules are the same—only the numbers have changed. Instead of going from 0 to 9 before we skip to the next place to the left, we only have 1s and 0s. And instead of 10, the base number is 2. But the exponents are always the same. The right-most position is 2 to the zero power. The next is 2 to the first power, and so on.

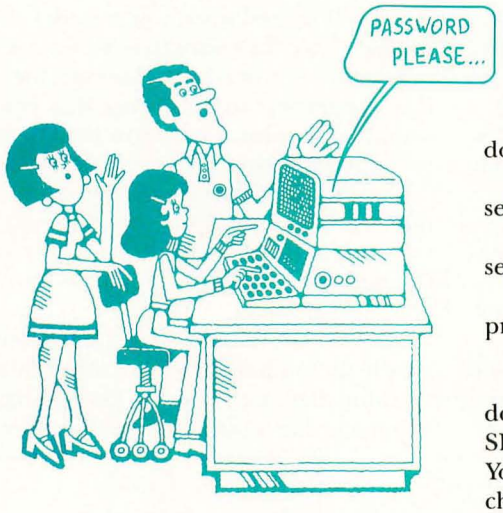
So now let's go back to our friend 01011101. This does not stand for the decimal value of 1,011,101. Until you get fluent in binary (or write a program to save you from thinking), you should work it out on paper. Because the lowest exponent value is always 0 and there are eight digits or places in this number, we write down the exponents from 7 to 0, left to right, and then place the value of each bit in the appropriate column:

```
2^7  2^6  2^5  2^4  2^3  2^2  2^1  2^0
 0    1    0    1    1    1    0    1

 0  + 64 + 0  + 16 + 8  + 4  + 0  + 1 = 93

Or: 1*26 + 1*24 + 1*23 + 1*22 + 1* 20 = 93
```

(to page 95)

Safe Keeping

by Mark E. Nelson



Safe Keeping is a file encryption program, used to keep sensitive files—either programs or data—secure from everyone except those with the correct password. The file is encrypted or encoded using the password as a key, then only the same password will decrypt the file. If you've wanted to keep your diary or journal on the computer, if you're writing a book or article that you'd like to keep private until it's thoroughly edited, if you have sensitive business documents or records stored on disk, or if you have any other privacy or security concerns, *Safe Keeping* can help.

Using the program

Double click on SAFEKEEP.PRG from the desktop, in either medium or high resolution. Click on OK to get past the copyright notice. Now select the options you prefer from the Options drop-down menu. You can choose to have the password echoed or shown on the screen as you type it. It's nice to have the echo turned off if someone is watching as you work, so that your password will remain confidential. You can also choose to have the encrypted or decrypted file that has been worked on automatically deleted. The defaults are echo on and delete off.

After you have selected your options, click on Lock/Unlock from the menu with the same name. The standard GEM file-selector box will be displayed, and you are told to select a file for encryption. The process is exactly the same for locking and unlocking files, and any type of file may be locked including programs, data, text, graphics, etc. Next you select the file name to be used for the new file, which is created by locking or unlocking a file. If you are locking a file, I suggest the extension .SAF be given the file to help you remember which files are locked and which are not.

Now you are prompted to enter a password. You may enter up to 80 characters including spaces, combination characters (such as Control-D, Alternate-K), punctuation, just about anything you can type except function keys. A password sequence that includes only alphabetic characters is the most common and the least secure. Longer password sequences are also more secure than short ones. If you have the "Don't echo password" option on, an asterisk will be shown for each character typed. Backspace to correct errors, and press Return when you're finished.

Safe Keeping will now begin the lock or unlock process on the file you have specified. The status of reading from the input file or writing to the output file is shown on the screen. If the file is larger than 100K, then multiple reads and writes will be required. If you have the delete option on, then Deleting will be shown as the file is deleted. Now you're ready to lock or unlock another file or quit.

A trial run

You may wish to go through this step-by-step trial run to get comfortable with *Safe Keeping*. A sample file called TEST.DOC is included on the disk to practice with.

1. Double-click on SAFEKEEP.PRG from medium or high resolution.
2. Click on Lock/Unlock from the drop-down menu.
3. Select TEST.DOC from the GEM file-selector box.
4. Type TEST.SAF in the new file-selector box.
5. Type "cheese" as the password and press Return.
6. Quit.
7. Show the contents of TEST.SAF by double-clicking on it and then selecting SHOW from the alert box that appears. You'll see that it is just a bunch of garbage characters since it has been locked. Compare it to the original file by showing the contents of TEST.DOC.
8. Double-click on SAFEKEEP.PRG to run it again so that we can unlock TEST.SAF.
9. Click on Lock/Unlock, select TEST.SAF from the selector box, and then type TEST.TXT in the next selector box.
10. Type "cheese" as the password.
11. Quit, then compare the unlocked file, TEST.TXT, with the original file, TEST.DOC. They should be exactly alike.

Play with the test file, trying out the different options until you feel comfortable with *Safe Keeping*. It's easy to use, but you should be sure about what you're doing before you lock an important file. Use a password that you will easily remember, but that would be hard to guess. There is no way to find out what the password was if you have forgotten it, so be careful! If you do forget your password, leave the delete option OFF and try several guesses.

How the program works

Each byte of the input file is read into a 100K buffer. Then the first byte is exclusive-ORed (XORed) with the first character in the password and written to the output file. The second byte is XOR'd with the second character and written and so on. At the end of the password, we begin with the first character again.

The XOR is used because it is a simple function with the property f(f(x)) = x. In other words, we get back what we started with if we apply the function twice.

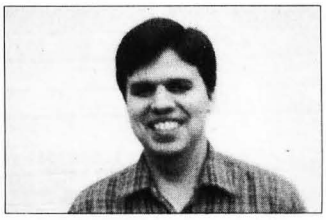
Here's how it works on the bit level. Suppose a byte in my input file contains binary 10101010, and the character I am going to XOR it with is binary 01110111. Remember that the XOR result is 1 if one of (but not both) bits is a 1, and 0 otherwise. To lock it:

```
input:          10101010
password char:  01110111
---------------------------
XOR result:     11011101
```

To unlock it:

```
locked byte:    11011101
password char:  01110111
---------------------------
XOR result:     10101010
```

I wrote the program using Personal Pascal Version 2. If you make a modification to the program that you believe makes it better, please send me a copy at the Atari Connection BBS at 801-377-1617. You can also leave any questions you have about the program there.■



Mark Nelson is a computer science student at Brigham Young University and the father of three boys, Drew, Steven and Aaron. He spends his free time coaching tee-ball, wrestling on the family room floor and watching Sesame Street.

SAFE KEEPING
Listing 1:
Pascal

```
{ "Safe Keeping" Copyright 1989 by ST-LOG }

Program Safe_Keeping;

{$I GEMSUBS.PAS}
{$I AUXSUBS.PAS}

var
  Infile_name, Outfile_name, Password: string[255];
  Infile, Outfile: packed file of byte;
  key: array[1..80] of byte;
  key_length, rez: integer;
  { menu headings }
  About_Safe, Fyle, Password_menu, Quit: integer;
  { menu items }
  About_Menu, File_Open, Pass_echo, Pass_no_echo, Quit_Quit,
       opt_dead, opt_delete, opt_no_delete: integer;
  Menu: menu_ptr;
  echo, deleet: boolean;
  path: path_name;

Function GetRez: integer;
  XBIOS(4);

Procedure Set_Up_Menus;
 Begin
  Menu:=New_Menu(10,' About Safe Keeping ');
  Fyle:=Add_Mtitle(Menu,' Lock/Unlock ');
  Password_menu:=Add_Mtitle(Menu,' Options ');
  Quit:=Add_Mtitle(Menu,' Quit ');
  Quit_Quit:= Add_Mitem(Menu, Quit, ' Quit ');
  Pass_echo:= Add_Mitem(Menu, Password_menu,     '  Echo Password        ');
  Pass_no_echo:= Add_Mitem(Menu, Password_menu, '  Do not Echo Password ');
  Opt_dead:= Add_Mitem(Menu, Password_menu,     '-----------------------');
  opt_delete:= ADd_Mitem(Menu, Password_menu,    '  Delete After Locking ');
  opt_no_delete:= Add_Mitem(Menu, Password_menu,'  Do not delete        ');
  File_Open:= Add_Mitem(Menu, Fyle, ' Lock or Unlock a File ');
  Menu_Check(Menu, Pass_Echo,True);
  Menu_Check(Menu, opt_no_delete, True);
  Menu_Disable(Menu, opt_dead);
  echo:= True;
  deleet:= False;
  Draw_Menu(Menu);
 End; { of procedure Set_Up_Menus }

Procedure Create_Key;
var
  temp: record
          case boolean of
            True:(c: char);
            False:(b: byte);
        end;
  i: integer;
begin
  for i:= 1 to (length(password) -1) do begin
    temp.c:= password[i];
    key[i]:= temp.b;
  end;
  key_length:= length(password);
end;

Function Keyin:Integer;
 var Message:Message_Buffer;
     EKey,d:Integer;
begin
 d:=Get_Event(E_Keyboard,0,0,0,0,False,0,0,0,0,False,0,0,0,0,
                    Message,EKey,d,d,d,d,d);
 Keyin:=EKey;
end; { of func Keyin }

Procedure Get_String(X,Y,MaxLength:Integer;Var Str:String);
 { returns a string entered at screen position X,Y of Maximum length MaxLength}
 var Ch:Char; I,Length:Integer; Temp:String;
 begin
  Hide_Mouse;
  Draw_String(X,Y,'_');Length:=0;Temp:='';
```

```
  Repeat
   I:=Keyin & $FF;
   If (I=8)AND(Length>0) then begin
     Length:=Length-1;
     Draw_String(X+Length*8,Y,'  ');
     Draw_String(X+Length*8,Y,'_');
     Delete(Temp,Length+1,1);
     end
     else
   If (I<>$0D) AND (I<>8) AND (I<>$1B) then begin
     If Length<MaxLength then Temp:=Concat(Temp,Chr(I))
      else begin
        Delete(Temp,MaxLength,1);
        Temp:=Concat(Temp,Chr(I));
        end;
     If Length<MaxLength then Length:=Length+1;
     Paint_Color(White);Paint_Rect(X+Length*8,Y-8,16,8);
     If echo then
       Draw_String(X+Length*8-8,Y,Chr(I))
     else
       Draw_String(X+Length*8-8,Y,'*');
     Draw_String(X+Length*8,Y,'_');
    end;
  Until (I=$0D);
  Draw_String(X+Length*8,Y,' ');
  Str:=Temp;
  Show_Mouse;
 end; { of proc Get_STring }

Function F_Delete(Var Name: Cstring): Integer;
  GemDos($41);

Function Get_Inputs: boolean;
label
 1;
var
  File_Selected, temp: boolean;
begin
  Get_Inputs:= False;
  hide_mouse;
  Draw_String(204,20*rez,'Select File to Lock or Unlock');
  show_mouse;
  File_Selected:= Get_In_File(path, infile_name);
  hide_mouse;
  paint_color(green);
  paint_rect(0,9*rez+(rez mod 2)*2,640,200*rez);
  show_mouse;
  If NOT File_Selected then goto 1;
  hide_mouse;
  Draw_String(224,20*rez,'Select New File Name');
  show_mouse;
  outfile_name:= infile_name;
  Repeat
    File_Selected:= Get_In_File(path, outfile_name);
  Until ((infile_name <> outfile_name) OR (Not File_Selected));
  hide_mouse;
  paint_rect(0,9*rez+(rez mod 2)*2,640,200*rez);
  show_mouse;
  If NOT File_Selected then goto 1;
  paint_color(white);
  paint_rect(0,50*rez,640,25*rez);
  Draw_String(1,60*rez,'Enter Password(s)');
  Get_String(1,70*rez,80,Password);
  paint_color(green);
  hide_mouse;
  paint_rect(0,9*rez+(rez mod 2)*2,640,200*rez);
  show_mouse;
  Create_Key;
  Get_Inputs:= True;

  1:          { come here if cancel selected }
end;

Function XOR(A, B: byte):byte;
var
  AA, BB, CC: record
              case boolean of
               True:(b: byte);
               False:(i: integer);
              end;
begin
  AA.b:= A;
  BB.b:= B;
  CC.i:= ((~AA.i) & BB.i) | ((~BB.i) & AA.i);
  XOR:= CC.b;
end;


Procedure Read_Write;
var
  temp: byte;
  buffer: array[1..100000] of byte;
  read_count, the_end: long_integer;
  count: integer;
```

Safe Keeping

by Mark E. Nelson

```
  been_looping: boolean;
  del_path: Cstring;
begin
  been_looping:= False;
  hide_mouse;
  begin_mouse;
  Draw_String(280,60*rez,'Reading...');
  Reset(infile, infile_name);
  Rewrite(outfile, outfile_name);
  count:= 1;
  While NOT EOF(infile) do begin
    if been_looping then Draw_String(280,60*rez,'Reading...');
    been_looping:= True;
    read_count:= 1;
    while((NOT EOF(infile)) AND (read_count <= 100000)) do begin
      read(infile, buffer[read_count]);
      read_count:= read_count+1;
    end;
    if read_count<100000 then the_end:= read_count -1
      else the_end:= 100000;
    paint_color(green);
    paint_rect(0,9*rez+(rez mod 2)*2,640,200*rez);
    Draw_String(280,60*rez,'Writing...');
    for read_count:= 1 to the_end do begin
      temp:= XOR(buffer[read_count], key[count]);
      write(outfile, temp);
      count:= count +1;
      if (count > key_length) then count:= 1;
    end;
  end;

  close(infile);
  close(outfile);
  paint_rect(0,9*rez+(rez mod 2)*2,640,200*rez);

  if deleet then begin
    Draw_String(278,60*rez,'Deleting...');
    PtoCstr(infile_name, del_path);
    count:= f_delete(del_path);
    paint_rect(0,9*rez+(rez mod 2)*2,640,200*rez);
  end;

  show_mouse;
  end_mouse;

end;

Procedure Copyright;
var Alert_String: string[255];
    d: integer;
begin
  Alert_String:=Concat('[0][         SAFE KEEPING         | |',
                       '     (c)1988 by Mark Nelson|',
    '  portions (c)1985 OSS and CCD |      Used by permission.]',
    '[   OK   ]');

  d:= Do_Alert(Alert_String,1);

end;

Procedure Set_up;
begin
  rez:= GetRez;
  if rez = 2 then paint_style(5);
  Set_up_Menus;
  Init_Mouse;
  path:= 'A:*.*';
  infile_name:= '';
  Copyright;
end;

Procedure Do_it;
begin
  If Get_Inputs then Read_Write;
end;

Procedure Get_Out;
begin
  Exit_Gem;
  Halt;
end;

Procedure Main_Event;
 Var d,k:integer;     { d is a dummy variable }
     Msg_area: Message_Buffer;
begin
 K:=Get_Event(E_Message,0,0,0,0,
              False,0,0,0,0,false,0,0,0,0,
              msg_area,d,d,d,d,d,d);

If Msg_area[0]=Mn_Selected then begin
 Menu_Normal(Menu,3);Menu_Normal(Menu,Fyle);Menu_Normal(Menu,Quit);
 Menu_Normal(Menu,Password_menu);
```

```
    if msg_area[3] = 3 then Copyright;     { About Safe Keeping... }

    If msg_area[4] = Quit_Quit then Get_Out;

    If msg_area[4] = file_open then do_it;

    If msg_area[4] = Pass_echo then begin
      Menu_Check(Menu, Pass_echo, True);
      Menu_Check(Menu, Pass_no_echo, False);
      echo:= True;
    end;

    if msg_area[4] = Pass_no_echo then begin
      Menu_Check(Menu, Pass_no_echo, True);
      Menu_Check(Menu, Pass_echo, False);
      echo:= False;
    end;

    If msg_area[4] = Opt_delete then begin
      Menu_Check(Menu, Opt_delete, True);
      Menu_Check(Menu, Opt_no_delete, False);
      deleet:= True;
    end;

    If msg_area[4] = Opt_no_delete then begin
      Menu_Check(Menu, Opt_no_delete, True);
      Menu_Check(Menu, Opt_delete, False);
      deleet:= False;
    end;

    end;
  end; { of procedure Main_Event }


begin
if init_gem >= 0 then
begin
  Set_Up;
  Repeat
    Main_Event;
  Until FALSE;
  Exit_Gem;
end;
end.
```

Safe Keeping END

INTRODUCING

VideoGames & Computer Entertainment™

12 ALL COLOR ISSUES ONLY $19.95

Save over $15 off the cover price!

- GAME REVIEWS
- ARCADE ACTION
- STRATEGY GUIDES
- TECHNICAL REPORTS
- COMPUTER SOFTWARE

VideoGames & Computer Entertainment
P.O. BOX 16927, N. HOLLYWOOD, CA 91615

☐ **Yes!** Sign me up for 12 issues for only $19.95—I'll save over $15!

☐ Payment Enclosed — Charge My ☐ VISA ☐ MC

_____ EXP_____

SIGNATURE_____

Money back on unused subscriptions if not satisfied!
Foreign—add $10 for postage.

NAME_____

ADDRESS_____

CITY_____ STATE_____ ZIP_____

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

Offer expires August 30, 1989 CJSWA

IAN'S QUEST

Ian Chadwick

What's enough? Who decides when a program is good enough for the Atari ST market?

I always wonder. What sort of mind was it that determined that *WordPerfect 4.1* was adequate for the ST user, when 4.2 had been available on the PC for a few years and 5.0 was due to be released? Who decided that the ST user was less sophisticated, needed fewer features, could do with less than the latest and greatest version?

Who decided that it was okay to release *First CADD*, the ST version of *Generic CADD*, as a stripped-down version of the PC release, making it little more than a toy? Or not to upgrade the ST *Drafix* to the Ultra version, now available on the PC? Who decided that ST CADD users are any less sophisticated or demanding than PC CADD users?

Who chose to bring out that embarrassing *Microsoft Write* as is, after the product had evolved so far beyond that level on the Mac? Who thought the Mac users deserved better than ST users?

Who decided that the GFA BASIC 3.0 manual, with all of its faults, typos, spelling mistakes, bad grammar and errata was good enough for ST buyers? Are we viewed as less literate than other computer users?

Who accepted the "index" feature in *Calamus* as good enough to release in a commercial desktop-publishing product?

The ST gets the nasty end of the stick a lot. There's a lot of "It's good enough for the ST" thinking out there, and I, for one, resent it. I'm tired of half-finished products, versions one or more generations earlier than releases on other machines, lukewarm technical support, wholly inadequate manuals, games ported from the C64, spelling mistakes and great grammatical errata in both programs and documentation—the whole bundle. I think we deserve better. And I think it reflects badly on those companies that produce these inferior or faulty products.

Is it deliberate? Sometimes, yes. It's not a malicious decision—no one puts the time and money into development of a bad product just to spite the users. But the Atari market is simply not generally viewed as serious; an image Atari Corp. itself does little to dispel. And why produce serious software for a predominantly home/hobbyist or a game-machine market?

It's also a small market, at least on this continent, and publishers play to their audience. For example, Borland, a large publisher in the MS-DOS realm, has produced *Turbo C* for the ST— sorry; only for sale in Germany. The North American market isn't big enough to appeal to them. We've only seen a trickle of the software available in the U.K., Germany and France, mostly thanks to the efforts of a few publisher/distributors such as MichTron and ISD.

And, yes, the market has a *bad* reputation as piracy-ridden. Of course, since it's true, we have only ourselves to blame for that one. As George Morrow once said, "The only industrial costs software companies have is the printing of serial numbers. What drives the prices so high is thievery." A cogent point to ponder while debating the high cost of your next purchase. I wonder if the problem is significantly less in, say, Germany, where Borland is releasing product. *Something* must have made them decide it was a better bet than the USA.

So who decides where and what to market? I dunno. I've been on that side of the fence and it's not easy, sitting in your ivory tower, pontificating, making decisions for thousands—maybe millions—of potential consumers. What I *do* know is that most firms lack any market input. They never poll the retailers, they never poll users—sometimes they conduct a limited "poll" through registration cards, but it's seldom sufficient. There's a lot of "by guess and by golly" in the business and down here at the bottom, users suffer for it.

Why doesn't someone *ask* us what we'd like to see in a product, rather than try to convince us that what gets produced is acceptable?

Morrow also said, "Without the proper software, computers make very good bookends." A step before bookends, I suppose, is game machines. I'd hate to think I have several thousand dollars tied up in a game machine. My late grandfather—whose 94 years spanned an enormous wealth of technological developments, from the first cars, through airplanes to space flight—could never understand computer games. He looked in absolute bewilderment at *Flight Simulator*, unable to see the screen full of lines and slabs of color as fields, towns, bridges and roads. I kept pointing out things to him, saying "See this? It's a road. Okay, pretend it's a road, then."

He never appreciated computer games. Couldn't see the purpose, couldn't make the jump to excitement. And, after more than a decade of computing, I'm beginning to wonder about them myself.

Anyway, let's leave him and the games

alone right now and get back onto the topic, so I can end the sermonizing.

What can we do to improve our computing lot in life and to make the publishers take us seriously? Well, for one thing, we can write to the larger software houses (those with no ST products) and ask for support. If they get enough letters from ST users, they may wake up to the fact that we're out here. We can also write to the general computer magazines and ask for coverage of the ST in their pages. Attention outside the few ST publications would improve our visibility. We can also write to existing ST publishers and tell them what we want, what we expect and what we need in software. Don't leave it up to them to determine what we ought to have.

And write to those publishers whose products aren't "good enough" and tell them so. No need to whine, but certainly there's no need for any of us to suffer these things in silence. If we do, then we only get what we deserve.

Some bits and pieces

If you're not aware already, ISD is working on *DynaCADD 2.0* for release this summer. It's a rather different approach from the original, but promises far greater flexibility, control and speed. I've seen the development version, and it's very impressive.

The main package will provide the 2-D CAD package, along with a programmer's shell (text editor and compiler/linker) to customize applications, a window, icon and menu generator, and a vector font editor. The 3-D package will be produced separately as an add-on. ISD is also bringing the product out for the PC, Macintosh and Amiga. Probably the best news is that *DynaCADD 2.0* will even run in a 520ST, since it uses a system of command overlays, rather than a single, all-encompassing program. It will remain, for now anyway, a monochrome-only program.

At the time of this writing (early March 1989) an upgrade of *Calamus*, with several bug fixes, is now available. You *must* send in your original disks, with the serial numbers. ISD is also getting ready to offer around 100 Compugraphic fonts for *Calamus* to registered users. Like *Calamus*, these will be encoded with your serial number.

Epyx sent me several games that I've been somewhat too busy to fully explore. I've looked at all of them, however, so here are my notes:

The ones I like best are *Sub Battle* and *Space Station Oblivion*. The former is a simulation of WWII submarine warfare. It's similar to *Gato, Silent Service* and *Up Periscope*, except that it includes the Atlantic theatre as well as the Pacific, and it has a side display, not merely conning tower or periscope view modes. It's an excellent game, tough, and the interface requires a lot of key tapping. But it offers many hours of playing challenge.

Oblivion is one of those odd 3-D maze-adventure-type games, full of tricks, surprises, places to explore. Also, since it's so abstract, it doesn't *feel* violent. It's not easy to understand at first, and the manual isn't clear on a lot of things, especially what you're supposed to do and how the vehicle operates. But it's fun just to explore and figure it out on your own.

A really nonviolent product from Epyx is *Final Assault*, a game of mountaineering. It's simple in concept: choose your level and your trail, grab your supplies and go. No one dies or gets killed—a wonderful change from so many shoot-'em-up games. However, the interface gets pretty tedious on long climbs, especially having to move your character's foot each and every time. Oh well, I guess it simulates the real thing. Slog, slog, slog.

If you're not already aware, ISD is working on *DynaCADD 2.0* for release this summer. It's a different approach from the original, but promises greater flexibility, speed and control.

Quite the opposite is *Techno Cop*, from US Gold, distributed by Epyx. It's the archetypal violent game: You play the role of a futuristic policeman whose job it is to kill or capture a bad guy. And in the process kill or capture anyone who attacks you. The emphasis is on *kill*. The bad guys explode in a gory mess, with a digitized scream. Juvenile, unpleasant and attractive to anyone who thinks Charlie Manson's a swell guy. To be avoided.

Tower Toppler is a new twist on the old climbing-maze theme, with some nicely humorous graphics. It's not my cup of tea, but there's no violence in it either; so it's acceptable to a broad spectrum of users.

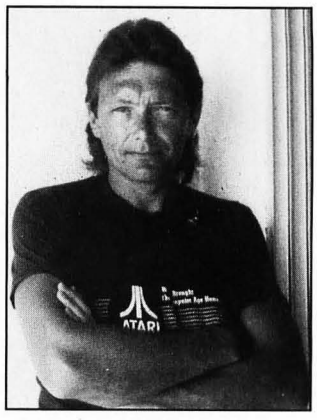
Dive Bomber is flight simulator with a "sink the *Bismark*" theme. The flight simulator isn't on a par with *FSII*; it's a lot like Avalon Hill's *Spitfire*. You have to drop a torpedo on a course with the *Bismark*. Good luck. It *looks* easy, but it's not, especially when everyone's shooting at you. It's not realistic; the computer battleship gets a lot of help the German navy didn't send out historically. Still, it's different and a challenge. Nice, polished graphics.

Epyx's biggie right now is *Art and Film Director*, which I'll leave for another column, since it deserves better.

Final notes

I recently returned from Colombia, where I met with one of the few Atari (8-bit) users in the country, a nice guy. He and about ten or 12 others in Medellin have 800 XLs and are trying to learning everything and do everything with a very limited budget, no source of supply and little help. I've been sending my old 8-bit products to him, but if you have any software ANALOG magazines, or books lying around that you don't need or have no use for these days, why not bundle it up and send it to their group? They'll appreciate all the help they can get. Write to: Armando Prieto Trillos, Apartado Aereo 1706, Medellin, Colombia.

Tell him I sent you. I thank you, in advance, on his behalf. If you have any questions about shipping to Colombia, write to me at the magazine or contact me on DELPHI, username CHADWICK. ■

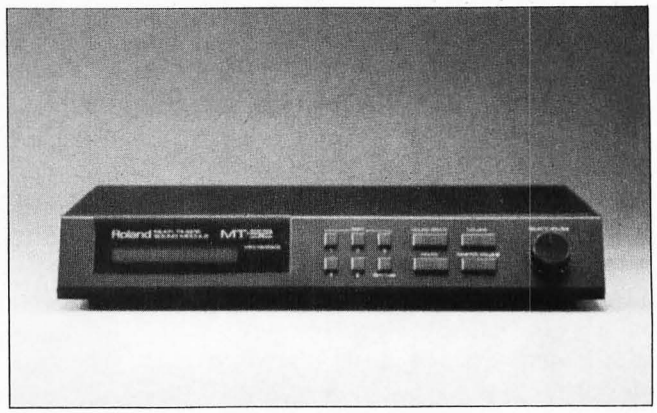


Ian Chadwick is a Toronto-based technical writer who lives in an increasingly small house with his wife, Susan, six cats, one dog, two rats and several field mice (who moved in recently, despite the cats). And that's not to mention the neighborhood's stray cats that take up residence as the mood moves them.

Roland MT-32 Multi-Timbre Sound Module.

Roland Corp
7200 Dominion Circle
Los Angeles, CA 90040
(213) 685-5141
$695

Reviewed by Michael Friesen



When Roland introduced its D-50 synthesizer, it was priced for the full professional, not for regular mortals on limited incomes. In what has become a company tradition, Roland introduced an abbreviated version of the same technology for the home market.

Dubbed the MT-32, company officials were surprised when semipros started gobbling up the units. They shouldn't have been. After all, here was a module with D-50-type sounds, multitimbral operation and a bevy of drum samples thrown in for good measure. Priced at approximately $700, the unit sold unexpectedly well to musicians who needed high performance on a low budget.

Apart from a host of great features, buyers also got what must be the oddest operating system quirk in recent synth history.

Physical attributes

The MT-32 is a small black box with an external power supply. The adapter is justified in the portability it lends to the unit. How portable is it? Extremely. With the exception of the cheaper and considerably less-powerful Yamaha FB-01, this is probably the lightest synth around. With a footprint the size of this page, and about two inches high, it's hardly a wonder that this is a favorite demo box at the trade shows.

The back panel has two ¼-inch output jacks, along with MIDI In, Out and Through. There's also a barrel-type power jack. Putting the on/off switch here was a good idea, considering how easy it would be to accidentally hit it while making front panel adjustments.

The front panel features a 16-character, backlit LCD display, six channel-select buttons and a rotary pot for selecting sounds and setting volumes.

The LCD is informative and quite readable under most lighting conditions. Unfortunately, you have to be facing it straight on for maximum visibility. There is no contrast control.

The six channel-select buttons can be used to examine what patch and volume are selected on each of the eight channels plus the rhythm channel. Six buttons for nine channels? Well, yes. You get to Channels 6 through 8 by holding down the Master button plus the selector for Channels 1, 2 or 3.

Which brings me to a minor gripe. Although the MT-32 allows the user to set a good array of features from the front panel (including an "all notes off" switch), you need either a good memory or a hand-made template to remember that hitting MASTER + PART 4 allows you to turn on the Overflow Assign function.

Along with voice and volume settings for all nine channels, you can modify reverb decay time. It's handy, but you have to set it manually every time you turn on the machine. It defaults to a the

medium-hall program with a moderate decay time.

The sounds

Until you get one of the many computer-based editors, you'll have to live with the presets. It's not too hard to do. The 128 preset sounds are warm, with good transients on the critical attack phase. Because the unit uses all-digital circuitry in the synthesis process, though, even the warm sounds can't touch the richness of some of my old Juno 106 patches.

Some of the presets suffer from sloppy programming. Syn Brass1 is one of several sounds that, when pushed to maximum velocity, will overdrive the digital to analog converters. The result is a crackling that won't go away until you drop the output volume to about 70.

Some of the sounds are pretty bad. In the tradition of L/A synthesis, there isn't a good piano anywhere in the circuitry. The synth basses, the brasses and strings are somehow thin and uninteresting. So what does that leave? Lots of great stuff.

All of the organs sound great (a local pastor tried to convert me after hearing Pipe Org 1), the synth brasses are vibrant, the saxes and basses approach the real thing, and some of the other sounds in the Synth banks are absolutely amazing. I must also note the Sho-No-Fue patch in the Special section. It's amazing. I can't describe it other than to say it's the closest thing to a synthesized bagpipe I've ever heard.

The reverb is not programmable, but just having it on a unit this cheap is great. It's a bit noisy, and you may wish for a noise gate if you're running really breathy sounds through a long decay setting. Still, it's a welcome addition.

The drum sounds appear to suffer from undersampling. You have to crank up the levels to get a good, solid thwack out of them. Of course, this also raises the noise floor to annoying levels. The range of sounds, though, is good, with both kit sounds as well as Latin instruments. There's lots of potential for the hitting-minded.

I must stress that this unit is *not* programmable from the front panel. Unless you buy or write an editor, you'll have to live with the presets—so listen carefully before you buy.

Using the MT-32

I've had my MT-32 for just over a year now, and I can heartily recommend it to anyone who already owns a MIDI keyboard—almost any MIDI keyboard. Although you can't do front panel editing, you'll be able to get at most of the sounds even from a preset-channel keyboard.

However, the real power goes to those people who have multichannel master keyboards. These people will quickly learn of the joy and frustration of having to figure out which of the great timbres you're going to put where. The performance possibilities are nearly endless.

A sequencer will let you do amazing things with this module. Because it's got 30 percussion sounds built in, as well as eight synthesizer channels and a theoretical ceiling of 32 simultaneous notes, it's possible to create incredibly complex music with just this one module.

I recently had to move most of my stuff halfway across the country. I shipped off most of the equipment two weeks before I moved, but kept my computer and MT-32 until the last day. When a friend of mine visited to take a look at all of my equipment, I found that I was able to rearrange most of my demos to run solely on the MT-32 with a minimal loss of quality. In terms of cost and sheer musical potential, the MT-32 is the most powerful piece of musical equipment I own.

The voice

Like its bigger brother (the D-50), the MT-32 uses an oscillator stacking system called Linear Arithmetic synthesis. Each voice can have between one and four partials assigned to it. Each partial is a complete synthesizer voice, with a single digital oscillator, filter, an amplifier and multipoint envelopes to control pitch, filter and amplitude.

There are 32 oscillators assigned dynamically wherever they are needed. In order to calculate the amount of polyphony, divide 32 by the number of partials used in a voice. The thickest voices use four partials, equalling eight voice polyphony. Two is usually enough for warm, organic sounds.

The oscillator wave may be either a square wave, a sawtooth or one of 128 PCM samples. Note that if you select a PCM sample, the filter becomes inoperative. You cannot filter a sampled waveform. There is good variety in the samples, with percussive sounds, attack transients and sustained loops.

The envelopes are great. The only flaw in them is that, after creating a complex attack contour with three levels and four times, all you have left is a sustain segment, and then a single release time. Being able to create complex release segments would have been nice. Still it's a big improvement over ADSR.

The filter and the envelope have programmable bias points with adjustable roll-off slopes. That means you can set up a partial to be active only in one area of the keyboard. If you set things up drastically enough, you could create virtual splits—without having a splittable master keyboard.

With four possible partials per voice, and about 100 parameters per partial, it may be just as well that you need a computer-based editor to create your own sounds. There are a number of ST-based editors for the MT-32 (one of which is reviewed elsewhere in this issue). Check your music store.

A most incredible design decision

Earlier on I mentioned a hardware/operating system quirk. Here it is. Although you can program your own sounds into the 64-program RAM area, you can't load up a bank of custom sounds and hit the road: You see, Roland didn't include a battery backup for RAM. What it did include was all the circuitry for a battery, and even a holder for the actual cell. But wait! Roland's only kidding!

What it did was write the operating system so that the RAM would be wiped on power-down, and then overwritten by the ROM patches when you power up again. A technician at Roland told me that this was to make the unit easier to operate for the home user. I'm not sure I follow the logic. Anybody with enough MIDI smarts to buy an editor, create or modify some sounds, and then send them back to the MT-32 is not going to be flustered when, on power-up, the default bank of sounds is mysteriously absent.

Furthermore, the sounds in RAM can only be accessed by a MIDI program-change command. Anybody bright enough to handle MIDI-sent program changes should also be able to deal with an absence of defaulted presets.

Other stuff

The small footprint of the MT-32 makes it ideal for desktop situations. The front panel looks nice, and all cables stay plugged in neatly behind the unit.

Because I already own a mixing board, I found the absence of a headphone jack only a minor frustration. However, it's an important factor to consider if you don't have a board and don't plan to run the unit through a home amp.

My complaints are all minor however. Most of them are taken care of in the D-110, a fully professional, industrial implementation of the same basic circuitry. Of course, it also carries a correspondingly higher price tag.

I am more than satisfied with the MT-32. It's an excellent starter or expander synth for amateur and semiprofessional users. Pound for pound and dollar for dollar, the MT-32 may be the best value on the synth market today.■

Michael Friesen *has worked as a bookbinder, a chicken shed cleaner, a cow milker, a ditch digger, a jingle composer and a synthesist for a touring theatre company. He's now a full-time journalism student at the University of Western Ontario.*

VDOS ProQueue

Marathon Computer Press
P.O. Box 68503
Virginia Beach, VA 23455-9433
(804) 460-5227 (24-hour message line)
$69.95, color or monochrome

Reviewed by D.A. Brumleve

Early in 1988, John Holder asked me to help beta-test a desktop-replacement program he was developing. He was very excited about the project. "It's wonderful!" he said. "You'll never have to use windows again!"

Wait a minute! I thought to myself. *I like windows!*

That program was VDOS, and it was eventually released as Shareware. VDOS was distributed at no cost; if you wanted the manual, you had to send in money and register your copy. Many people did send money, and the developers were encouraged to provide the much-improved version that has now been released commercially as *VDOS ProQueue*. I do like windows, but this program has nevertheless become indispensible to me.

VDOS ProQueue is a powerful operating shell program that replaces the GEM desktop, but it also provides many features that are not ordinarily available. The program requires very little memory to run (42K). It can be automatically installed upon boot-up. The authors claim there is near-complete compatibility with all existing ST software; all of the programs I have used with it have functioned normally.

ProQueue's desktop offers a series of drop-down menus that are used to perform typical GEM desktop functions such as copying files, as well as a variety of functions one might normally find in a desk accessory. In addition, the *ProQueue* desktop has an Applications Box on the right side of the screen and six pop-up archive menus along the bottom.

There are several advantages to using *VDOS ProQueue* instead of your ST's desktop, but for hard-drive owners, the premier advantage is undoubtedly the ability to instantly execute a program with a click of the mouse (or by pressing a key), no matter how deeply nested in folder upon folder that program may be. Just a bit of initial setup is required to make this possible. With point-and-click simplicity, you tell *ProQueue* where the program can be found and "define a slot" on one of *ProQueue*'s menus of immediately executable applications (that is, either in the Applications Box or on one of the archive pop-up menus). These defined slots are then saved to disk, and every time *VDOS ProQueue* is loaded, it automatically knows where your favorite programs can be found. It takes about half a minute per program to define the slots if you go about it leisurely. This procedure needs to be done only once, but if you want to change your definitions at a later time, the process is just as simple.

Ten programs can be defined in the Applications Box. The box is displayed continuously, and these programs can be executed immediately by clicking directly on them with the mouse or by pressing the corresponding key (F1 to F10). The Applications Box is reserved for those programs you use most frequently. Ninety additional program slots are provided in pop-up menus (the archive program slots). Keyboard macros are not available for archive programs. Execution speed is very fast.

The drop-down menus contain many options with which UNIX and DOS users may be familiar, but which are new to ST-only folks like myself. For example, on the disk menu are the following options: MVDIR, MKDIR, RMDIR, FORMAT, COPYDIR, CHDIR, VIRTUAL, DSKFREE, CHDRIVE and WHERE?. On the file menu are: DELETE, MOVE, COPY, FDATIME, FIND SIZE? LS -W, LS -L, CHOMD, TOUCH

and RENAME. All of the menu items have keyboard macro counterparts. In many cases, these functions are performed more efficiently with *ProQueue* than is possible with the GEM desktop.

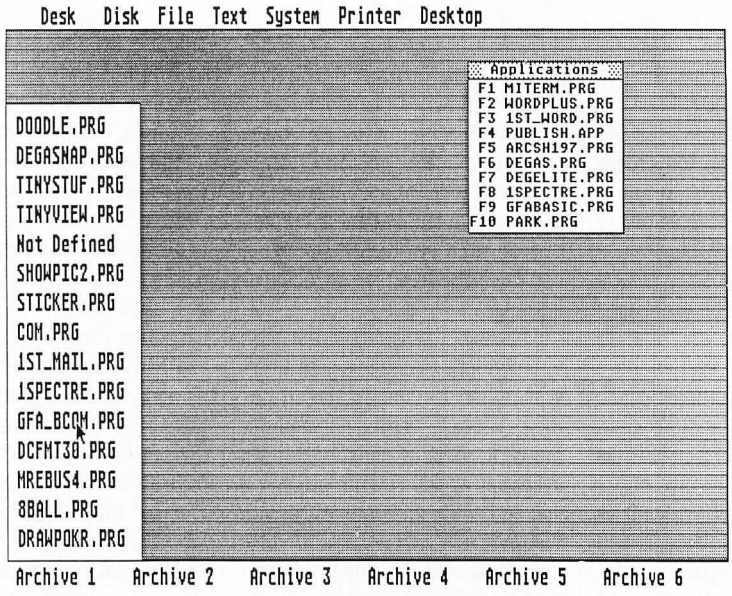
The disk and file menu offerings perform a variety of functions, some of which require more complicated sequences if performed on the GEM desktop; others are simply not possible through GEM at all. MOVE, for example, copies a file or group of files to another directory and deletes the original file(s) all in one action. Likewise, MVDIR will move a folder and its contents to another drive and directory and delete the original folder.

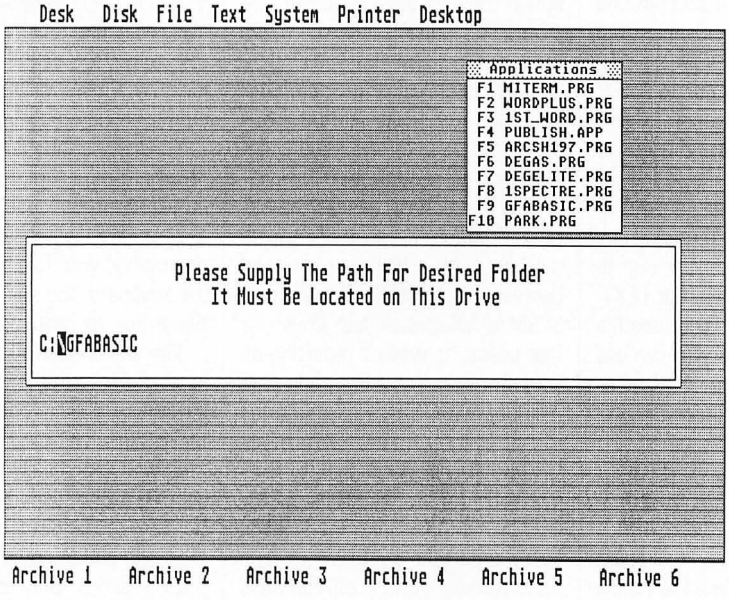
COPY lets you copy a single file, or a group of files with the same extension, or a group of files all beginning with the same letter, etc. FIND will find a file you are looking for, as long as you can remember the file's name and the drive it's on. The FORMAT option provides only the regulation single- and double-sided formats, but it displays a graphic report on the formatting process that is very helpful in diagnosing and isolating disk formatting problems.

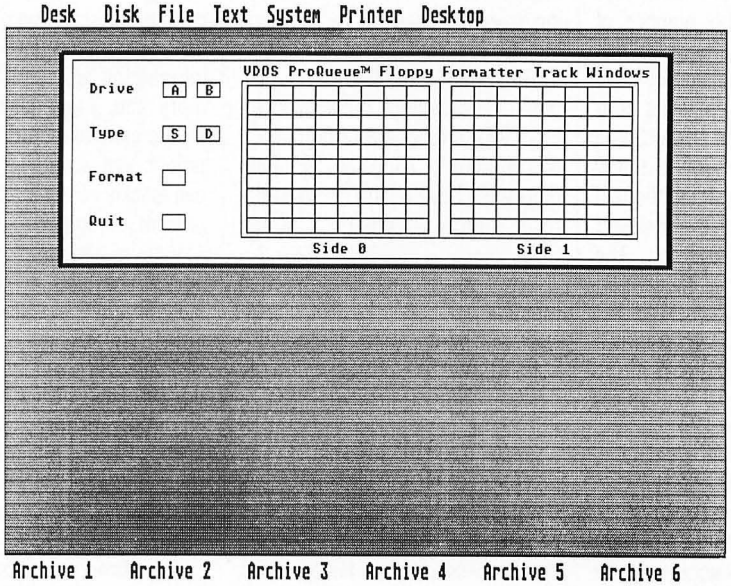
A TOUCH option allows you to update the date of a file, which proves handy for programmers. CHMOD will hide files from the GEM desktop, but *VDOS ProQueue* will recognize them. To list all the files and folders in a directory, even hidden files, you can click LS -L (for a single-column listing with very complete information) or LS -W (for a brief three-column listing). The DSKFREE option reveals the free space remaining on a drive much faster than the Show Info option on the GEM desktop.

Some of these file/directory-manipulation functions are performed using the familiar GEM file-selector box. For example, if you click on DELETE, a file-selector box displays the current path, and you simply select a file and click OK. In these cases, if the current path is not the one you need, you can alter the path as you would with a file-selector box in any program.

However, other functions, like COPYDIR, for example, require you to type the name of a file or folder in a dialog box. The box will dis-







play the current path when it appears. You cannot change the drive path with the dialog box, so before you select COPYDIR, you need to make sure you are on the proper path for the folder you want to copy. Until I became accustomed to it, the dialog box system was a problem to me. *VDOS ProQueue* makes it so easy to execute programs without regard to their directories or subdirectories that I found myself forgetting which directory or subdirectory they were in! Fortunately, the FIND option was available to refresh my memory.

The text and printer menus offer a variety of options related to text files and your particular printer setup. On the GEM desktop, you can double-click on a text file and select Show, Print or Cancel from an alert box. *VDOS ProQueue*'s DISPLAY option corresponds to Show, although it has some differences. If a line of text is over 80 columns long, *ProQueue* will divide it over two lines, for example. Like Show, DISPLAY will allow you to read farther into a file, but not back up to the beginning. However, the TEXTWIND option displays your text file within a GEM window; you can use a scroll bar to move forward or backward as you examine the file.

SEARCH will look for a text string in your chosen file. Another clever option is MERGE, which allows you to merge two text files into a single new file. LC will count the number of lines in a file. I think I would have found this more useful if it counted the number of words. Sometimes I want to see only the beginning or end of a file to jog my memory as to its contents. Clicking HEADS or TAILS will call up the first few or last few lines almost instantly. I find the text menu to be a great improvement over the standard desktop. The printer menu offers the counterpart to Print on the GEM desktop, but it also allows you to check to see if the printer is on, select pin-fed or sheet-paper options, send an escape sequence to your printer and output a text file to the screen and the printer simultaneously.

The system menu has a long list of options; many of these may currently take up a desk accessory slot on your GEM desktop. For example, *ProQueue* offers a built-in multiyear calendar, a screen saver to darken the screen when you leave your computer temporarily, a date/time display, an option to invert the colors on a monochrome monitor, a password system, an option to display free memory remaining and an auditor.

The AUDIT option is a real boon to programmers and business users who must account to the IRS or their bosses for their computer times. It keeps a constant tab on every application you run and when you begin and end your work with that program, and saves the information to disk. The audit file can then be printed. I chose the AUDIT option the first day I used the program and didn't give it much thought for some time after that. Six days later—after my computer had been turned on and off many times—the auditor was still keeping track of my program usage.

This menu also provides the EXECUTE option. If you want to run a program for which you have not defined a slot, EXECUTE will do it. SYSMAP checks all your drives for free space, as well as reporting on your printer status and free memory. $HOME returns you to the initial directory (wherever VDOSPRO.PRG is located). VERIFY lets you turn off (or on) the GEMDOS system that performs a check every time you try to write to a disk. Turning VERIFY off can save a great deal of time on file copying. It would seem more appropriate to me to have placed both VERIFY and $HOME in the disk menu. The system menu also allows you to reboot your computer; this is great for Mega owners who otherwise have a long reach to the reset button on the back of the CPU.

Yet another option in the system menu is ACCESSORY. Desk accessories are not immediately available from the *VDOS ProQueue* desktop. When you click ACCESSORY, the desktop is replaced by a single desk menu. The top option in that menu will return you to the *ProQueue* desktop; the slots below it are filled with whatever desk accessories you have booted. This is not the nuisance that it may seem. Transition between the desktop and the ACCESSORY menu screen is instantaneous.

The final menu, desktop, allows you to exit the program and return to your ST's GEM desktop.

VDOSPRO.PRG functions only in medium and high resolution. You can, however, enjoy the auto-execution benefits of *ProQueue* in low resolution through the use of an accompanying program, VDOSLOW. You cannot change resolutions from within *ProQueue*, so you have to exit to the GEM desktop, change the resolution and run VDOSLOW.PRG. Slots for ten applications can be defined. The various utilities that are offered in VDOSPRO.PRG are not available in low resolution so the only advantage to running VDOSLOW over using the GEM desktop is in the auto-execution of defined programs.

ProQueue's manual comes in an attractive three-ring binder. This is an excellent choice of bindings for a book that will need to remain open while your hands are busy at the keyboard. You can order an optional box to keep the binder in.

The explanation of the program is well-organized and, generally, clearly presented. The largest chapter lists each of the options in the menus alphabetically with accompanying explanations and examples. Some of the language is very technical; the authors often seek to explain reasons why the program functions the way it does or why certain decisions were made in the development process. This "extra" information is helpful to programmers, and, if you don't understand it, it is easy to ignore. The presentations are arranged so that you can quickly zero in on the most significant paragraphs.

Various options for the installation of the program, including installation in RAM, are explored in detail. The authors provide a bibliography, a glossary of terms (a great aid) and listings of error codes and ASCII codes. I would like to have seen an index. Even though the menu options are arranged alphabetically, this would still have been a help.

The authors promise continuing support. One improvement I'd like to see is the expansion of the Applications Box to 20 program slots, with the shift key combined with the F1 through F10 keys as macros for the ten additional programs. The archive slots do provide for 90 programs, which is plenty, but no keyboard counterparts are available for them, and the continuous visibility of the Applications Box on the *ProQueue* desktop makes for more instantaneous program execution. I would also prefer it if a file-selector box were used for those directory and file manipulations in which a dialog box is currently presented.

The authors recommend the program for 520ST owners with two drives, 1040ST owners with one drive and anyone with a hard drive. I have found the auto-execution feature to be of benefit primarily when using a hard disk, but the menu option features could well benefit anyone. There are certain features I have grown quite fond of. I wouldn't want to be without my auditor or the text-reading options or the MOVE option, for example. On the other hand, those options are only available in medium or high resolution, so I would hesitate to recommend the program for use without a hard drive if the programs used most frequently are in low resolution. For hard-drive owners, however, and for others who use their machines primarily in high or medium resolution, *VDOS ProQueue* represents a major improvement over the ST's GEM desktop.■

D.A. Brumleve is an ST programmer and writer living in Urbana, Illinois.

Sound Designer

Digidesign
1360 Willow Road, Suite 101
Menlo Park, CA 94025
(415) 327-8811
$349, color or monochrome

Reviewed by Larry Herzberg

Originally written for the Macintosh, this powerful program has recently been made available to Atari ST owners.

A digital sampler analyzes any sound by measuring the amplitude of its waveform many thousands of times a second. Each discrete measurement is called a sample, and the entire set of samples used to reproduce a sound is a sound file. Since a sound file consists of nothing more than a long string of numbers, you might think that most samplers would offer a host of features for manipulating such simple data. If you own one, however, you've probably been frustrated on more than one occasion by its editing limitations. Most affordable samplers are unable to even graphically display waveforms, let alone "zoom in," redraw, cut, paste or copy segments of them. Mixing and merging sounds is usually well beyond their capabilities.

Sound Designer, a "universal" waveform editor and digital signal processor, does all of these things and much more. Originally written for the Macintosh, this powerful program has recently been made available to Atari ST owners.

Looking at sound

For those who have only a vague idea of what digital samplers do, *Sound Designer* can be a wonderfully educational tool. The manual, which covers the basics of sound and sampling, is clear and concise, but the real lessons are learned when you begin to examine the graphic displays of the sounds themselves.

Sound Designer puts GEM through its paces to make such examinations easy. Once a sound has been transferred from your sampler to the computer, the program writes all the data to a disk file and opens a window displaying as much of the waveform as it can. If the sound file is too long to fit into the display all at once, the window can be scrolled along its horizontal axis until the portion you wish to edit is shown. Up to three windows displaying different sound files can be opened on the screen at once, and no matter how "zoomed in" you might be on the waveform in the active window, an additional "overview window" is

always available to display the entire sound. By moving the cursor around in the overview window, you can cause the active window to jump to whatever segment of the waveform you desire, including any portions that may not have been loaded yet from disk to RAM.

Once the proper waveform segment has been displayed, you can magnify any portion of it in one of two ways. By clicking on the horizontal or vertical arrows of the Scale Box, which resembles a small, empty window in the bottom-left corner of the editing screen, you can shrink or stretch either the horizontal (time) or the vertical (amplitude) axes until you've achieved the scaling you want. Then, if you've inadvertently magnified the view so much that the specific waveform segment you were concerned with is no longer showing, you can scroll around the window until you find it.

A faster way to zoom-in is by utilizing the Zoom Box. After clicking on the Zoom Box icon, the I-beam cursor used for basic editing functions is replaced with a cross hairs, and by clicking and holding down the mouse button, a rectangle can be "dragged" around any portion of the waveform. When the mouse button is released, this portion is redrawn to fill the entire window, and the time and amplitude axes are rescaled automatically.

If you wish to mark any position in a sound file for future reference, numbered icons can be dragged from the control panel into the active window. Clicking twice on such a marker displays its editable information box, allowing you to name it and catalog its purpose. To quickly locate any marker that has been placed in the active window, you simply hold down the alternate key and type the marker number. Ten markers are available for each window.

Redrawing waveforms

With each step of magnification, increasingly complex views of the sound are displayed. The differences between the waveforms produced by a flute and a saxophone, for instance, become obvious, as do anomalous noises that you might want to erase, or "draw out" of the sound.

In order to redraw any part of a waveform, you must be almost entirely zoomed-in on it, so that the scale of the time axis is reduced to a few milli- or microseconds. At that level, audible glitches spike out of the rest of the waveform like huge stalactites. To get rid of them, all you have to do is click on the Pencil icon; the cursor becomes a drawing tool that can be dragged across the base of the spike, effectively deleting it.

The loop window

Virtually all samplers allow the user to designate loop points in a sound file when a sustained sound is desired. Looping is necessary because RAM is limited on a sampler. If you use one byte to measure the amplitude of a wave at any given moment, and you sample at a rate of 40,000 samples per second, sampling a ten-second-long piano note will eat up 400,000 bytes!

The way around this excessive use of memory is to sample the note for, say, two seconds, find a segment with relatively stable amplitude and waveform, and, after the initial attack and decay, loop that segment over and over again while decreasing its volume with an independent ADSR (Attack, Decay, Sustain, Release) envelope on the sampler's amplifier circuit; this can approximate the sound's natural decay characteristics.

If you've ever tried to designate good loop points, especially within a complex sound like that of a saxophone, you know how difficult it can be. Unless the sound can be graphically represented in some detail, finding a segment with stable amplitude and waveform is not an easy matter, and that's just the beginning of the problem. If the slope of the waveform at the end of the loop does not match up exactly with the slope at the beginning, a click or pop will be heard every time the loop repeats.

Sound Designer takes the guesswork out of this process by providing a loop window that allows the user to fine-tune loop points. After special loop markers have been placed at the beginning and end of a decently stable portion of the waveform, the center line of the loop window shows precisely how the slope at the end of the loop matches up with the slope at the beginning. By clicking on the left and right arrows under each segment, either loop point can moved backward or forward one sample at a time until the optimal matchup is achieved. This is one of *Sound Designer*'s most practical features.

Editing waveforms

All of the program's basic editing functions are as easy to use as the loop window. After a portion of the waveform has been selected by holding down the mouse button and dragging the I-beam cursor over it, that segment (its color scheme now inverted) can be reversed, creating strange backward effects, or zeroed-out, eliminating unwanted noise at the beginning or end of a sound. The selected segment can also be cut and, if desired, pasted elsewhere in the waveform, or simply copied to another location without being cut. A smoothing feature, when activated before a copy or paste operation, helps make edits seamless by creating a fast cross fade between divergent amplitudes around the edit point.

In addition, a portion cut out of one sound file can be pasted onto a different sound file altogether, as long as the destination file has been opened into one of the three available windows. *Sound Designer* allows this by always saving the waveform data most recently cut or copied to a clipboard, which becomes disk-based if it runs out of available RAM. Thanks to the clipboard, editing operations can be undone, as long as no additional editing operation has taken place since the one you want to undo.

The digital mixer

Mixing and merging, two of the five functions included in *Sound Designer*'s Digital Mixer (which the user accesses by clicking on the blender icon), are operations that always involve two sound files. Mixing together instruments

Virtually all samplers allow the user to designate loop points in a sound file when a sustained sound is desired. Looping is necessary because RAM is limited on a sampler.

Attention Programmers!

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

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

Send your programs and articles to:

ST-LOG Magazine

P.O. Box 1413-M.O.

Manchester, CT 06040-1413

MOVING?

DON'T MISS A SINGLE ISSUE

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

DO YOU HAVE A QUESTION ABOUT YOUR SUBSCRIPTION?

Check the appropriate boxes below:

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

Name _____

Street Address _____

City _____ State _____ Zip _____

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

ATTACH LABEL HERE

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

that play identical parts is one of the best ways to conserve RAM on your sampler. For example, multiple horns performing staccato unison lines can easily be mixed together, and since mixing simply adds waveforms together, sample by sample, several horns mixed into a section will only take up the amount of RAM utilized by the longest single horn sound. Waveforms can be mixed together in any proportion and with a temporal offset, so getting the balance right is just a matter of experimentation.

Merging is a slightly more complicated procedure, but can be useful in obtaining special effects. For instance, you might want to create a hybrid sound with the attack of a piano but the sustain of a guitar. Or you might like the breathy attack of one sampled flute, but the vibratoed sustain of another. To merge two sound files together, markers must first be placed in each; one designating where the merge should begin, the other where it should end. Then, after clicking on the blender icon and selecting the Merge option off the Digital Mixer menu, you choose which order to merge the sounds in, and identify the relevant marker in each sound file. A final option allows you to choose between two cross-fade shapes, linear or equal power. Since there are more variables to take into account with merges, they are a little harder to get right than mixes.

Three other features are included in the Digital Mixer: Gain Change, Cross-fade Looping and Digital Equalization. Gain Change allows you to alter the relative output volume of a sound, either in terms of decibels or by specifying a percentage of increase or decrease. When the program executes this function, it displays the total number of samples clipped (increased beyond the point where digital distortion occurs) as well as the number of continuous samples clipped. The second number is significant because small quantities of continuous samples clipped are often inaudible.

The Normalize option is perhaps the most useful feature of the Gain Change module. This function increases the gain of a sound file until the peak amplitude of its loudest sample reaches 100% full scale (meaning that it is represented by the largest number available in its particular data format). If you're uncertain whether or not a sound file needs normalization, its peak value can be determined by selecting the Peak Value button and clicking on Execute in the Gain Change dialog box.

Cross-fade Looping requires that loop markers already be placed in the sound file you wish to operate on. If you were unable to find "click free" loop points using the loop window, this should do the trick, as long as your loop markers have already been placed with some care. This function mixes data from the area around the loop end with data from the area around the loop start, sort of like a mini-merge. As in the Merge function, you can choose between two cross fade types, linear and equal power.

The final feature of the Digital Mixer is the Digital Equalization (Peak/Shelf EQ) module. Equalization is used to alter the tonal characteristics of a sound, to make it "brighter" or "darker" by increasing or decreasing the amplitudes of a specified range of frequencies. Three modes of Digital Equalization are available in *Sound Designer*. The Peaking/Notching mode resembles an analog parametric equalizer, with its three parameters of center frequency, band width and boost/cut amount. The two other modes, hi/lo shelving and hi/lo pass, affect all frequencies above or below a specified corner or cutoff frequency. Hi/lo shelving allows you to enter a boost/cut amount, while cutoff frequency is the only parameter used by the hi or lo pass filters.

Frequency analysis

To aid you in altering a sound's tonal characteristics, *Sound Designer* utilizes the Fast Fourier Transform (FFT) algorithm to create a three-dimensional representation of the waveform's frequency content as it changes over time. In this display, the horizontal axis is the frequency axis, calibrated in kilohertz from left to right; the vertical axis—curiously uncalibrated—represents amplitude, and the time axis extends "into" the screen.

Analysis always begins at the location in the sound file displayed at the left edge of the active window, and can be set to continue for up to 400 milliseconds along the waveform, although a considerable amount of detail is lost when more than 200 milliseconds are graphed.

Interesting as it is to view the tonal characteristics of sound this way, using the frequency analysis to guide you in fine tuning a sound's EQ is not an obvious matter; calibrations on the amplitude axis might help, as would a brief tutorial in the manual. However, just seeing where the bulk of a sound's frequencies lie might prevent you from wasting time trying to alter a band of frequencies that aren't even part of the waveform's structure.

Listening to the sound

Of course, once you've edited a sound file, you'll want to listen to it. *Sound Designer* offers several ways of doing this. First, by clicking on the Play Sound icon, you can hear the sound through your ST monitor's speaker. You'll probably be surprised by the fidelity of this quick method; it is quite sufficient to audition rough edits or loop points.

You can also connect the ST's audio output to an external D/A box, but the most accurate way of hearing the sound is to transfer it back to your sampler; after all, that's where you'll be playing it from. Transferring a sound to and from your sampler is as easy as clicking on the proper icon on the control panel, but it's not always necessary to send the whole sound file to your sampler. A Preview option allows you to transfer only a part of the sound currently residing in the ST's memory. This cuts down on disk access time, and eliminates a few other time-consuming protocols by always sending the sound to sample Location #1 in your sampler. By pressing the shift key while clicking on the Preview icon, only the loop points will be transferred, if your sampler supports this feature. And once the sound has been transferred, you can play it from the ST using a five-octave, mouse-driven MIDI keyboard, accessed from the drop-down Tools menu.

A few nits to pick

For all its impressive features, *Sound Designer* is not entirely free of problems, and most of them have to do with the program's disk-based design. As the manual is quick to point out, such a design does enable the program to handle sound files that are larger than available RAM, but it is also slow at times. The clipboard can be configured to operate on a RAMdisk, but this doesn't speed things up much, and if your RAMdisk is large enough to store sound files on, it will probably create more of a slowdown problem than it solves by limiting the amount of memory available for waveform display and playback. If you're operating with only one floppy drive, the disk-swaps required for some merges can drive you crazy. *Sound Designer* might be the best excuse you'll ever have to buy a hard disk.

In addition to straining your patience at times, the disk-based design does not adequately deal with the all-too-human tendency not to leave well enough alone. Since basic edits are written immediately to disk and other changes are saved when the sound file is closed, it is too easy to make permanent changes you might later regret. This is especially true if the only version of the sound you have is in *Sound Designer* format, rather than in your sampler's as well.

Two factors exacerbate this problem. First, only those editing operations that use the clipboard can be undone; waveform redrawings or loop-marker relocations cannot be. Second, the way the back-up file system has been implemented is a bit weird. If Back-up Files has been selected from the Options menu, *Sound Designer* will automatically attempt to create a backup file whenever a sound file is opened. This is fine, but if there is not enough room to create the backup on the same disk as the original, the program

just throws up its hands in dismay and announces that all changes will be permanent! Why not simply request that the user insert another disk on which to write the backup file?

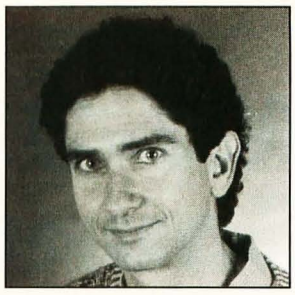
Furthermore, even if a backup file has been created, it is always deleted after the sound file is closed, whether or not you've chosen to save the changes you made. Wouldn't it make more sense to leave the backup intact, as an insurance policy against second thoughts? As it stands, you should always make a copy of any sound file you intend to open before running *Sound Designer*. If that's not possible, use the Save As option to save it under a different filename immediately after the program has opened it.

There are just a few other minor annoyances. For instance, you would think that such a disk-based program would include an option to format disks, but *Sound Designer* does not. Also, markers occasionally have problems. When one is placed at the very end of a sound file and the scale is magnified a bit, it sometimes remains out of view, even if the window is scrolled as far to the right as possible. If you then click on the bottom-right corner of the window (where you know the marker should be), it might pop up, but that corner of the window will not be redrawn properly.

A more significant problem involves the loop markers: occasionally there is a discrepancy between the time shown in a loop marker's information box and the time initially displayed on its side of the loop window. Whatever the source of this ambiguity, you can eliminate it by calibrating the horizontal axis of the active editing window in terms of consecutive samples rather than subdivisions of time; the program permits this kind of calibration in either decimal or hexadecimal form.

Supported samplers

Sound Designer supports many of the most popular samplers on the market, including the Akai S700/X7000, Akai S900, Casio FZ-1, E-mu Systems Emax, Ensoniq EPS, Ensoniq Mirage/Multisampler, Korg DSS-1, Korg DSM-1, Roland S-10/S-220/MKS-100, Roland S-50, Sequential P-2000, Yamaha TX16W, and any machine using the MIDI Sample Dump Standard. The manual is very helpful in describing the MIDI "personalities" of each of these samplers. If you own any of them and an Atari ST, check this program out.■



After receiving a BA in philosophy from (the notorious) Reed College some 13 years ago, ***Larry Herzberg*** *continued to pursue his interest in existentialism by becoming a freelance musician, writer and MIDI programmer in Los Angeles. Bucking centuries of* ST-LOG *tradition, he lives with no cats, but does share his apartment with a rather robust potted palm.*

MT-32 Sound Designer

Roland Canada Music Ltd.
13880 Mayfield Place
Richmond, B.C. V6V-2E4
(604) 270-6626
About $90

Reviewed by Michael Friesen

Roland's MT-32 is a great little synthesizer. It's got warm sounds, a full complement of PCM sampled drums, and a great multitimbral, multichannel MIDI implementation. The only problem is that it's essentially a preset machine. You can't edit any sounds from the front panel.

That means that it's impossible to modify the SynBrass1 sound so that it doesn't distort when it hits the unit's D-A convertor. That means that you can't tweak Str Sect3 to rest cleanly in the groove instead of screaming every time you push the velocity. That means that, unless you're content with the presets, you need a patch editor/librarian.

These programs perform two basic functions: They allow you to modify the sounds on your synth by using the input/output facilities of a computer, and they allow you to store and retrieve sounds to and from an archival medium.

Joel Koftinoff of British Columbia, Canada, has written a patch editor and librarian for the MT-32. The literature (I use that word guardedly) says that it was a joint project with Roland Canada. That means that Roland (Canadian division only) is handling the distribution of the program. The Canadian price is between 105 and 130 Canadian dollars.

Roland's MT-32 is a great little synthesizer. It's got warm sounds, a full complement of PCM sampled drums, and a great multitimbral, multichannel MIDI implementation.

What you get fer yer nickels

The most important program is the one that lets you edit and store your sounds. Appropriately dubbed MT_EDIT, this exists both as a stand-alone program and as a desk accessory. The latter file is a blessing for those who don't have multi-program systems. For example, it means that I can write this review while having the editor resident in RAM any time I want to check on a feature.

In more normal applications, this feature would allow you to use the editor from the desk of your GEM-based sequencer. The only caveat here is that the editor will eat up almost 80K. Those running high-end sequencers on a 520ST may find themselves quickly running out of RAM—or unable to run it at all.

Just to make matters more confusing, Dr. T's *Keyboard Controlled Sequencer* doesn't allow for GEM-based applications anyway, and Hybrid Arts' *MIDI-Track* and *SMPTE-Track* don't allocate enough accessory space to run the editor in the desktop anyway.

For those with Hybrid Arts sequencers, there is still a way to use custom patches. Koftinoff has thoughtfully included two smaller desk accessories. One of them will allow you to modify the System Area (reverb, volume, pan) as well as load banks from disk and send them to the MT-32. The other is the sender-only, and can be placed into an AUTO folder. It will then send a bank to your MT-32 whenever the ST is booted from that disk.

The sender and system controller have both been released into the public domain and may be freely traded. That means that someone with an editor can trade his sounds with someone who doesn't have the editor. Hooray! It's a good move that partly mitigates the copy protection on the other programs.

There are two banks of new sounds, two banks filled with editable versions of the ROM sounds, and a bank devoted to percussion instruments.

There's one more neat program called THRUTOS, which provides a readout of incoming MIDI events. A register shows the last program change message received on each channel. The note matrix itself can accommodate up to eight events on each of the 16 channels. While useful for debugging your MIDI system, you won't want to perform through it: It only took a four note chord on three simultaneous channels to send the thing into serious delays. It also crashed me back to the desktop when I tried a three-channel glissando. Still, it could be useful. I'm glad to say Koftinoff gave me more than I asked for.

The editor

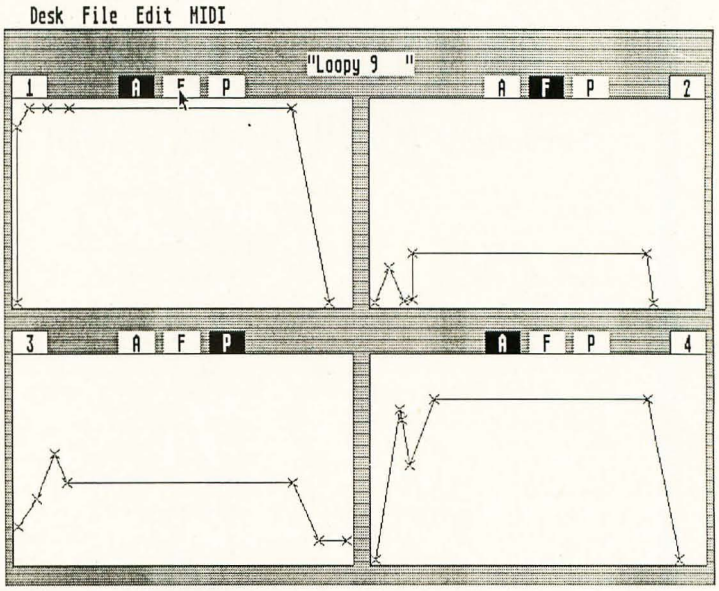
Here we come to the nub of the matter. Is it a good editor? The answer is yes. And no. On the plus side we have a user interface that allows you to move in between hierarchal levels from one point directly to another—rather than having to step through a pyramid.

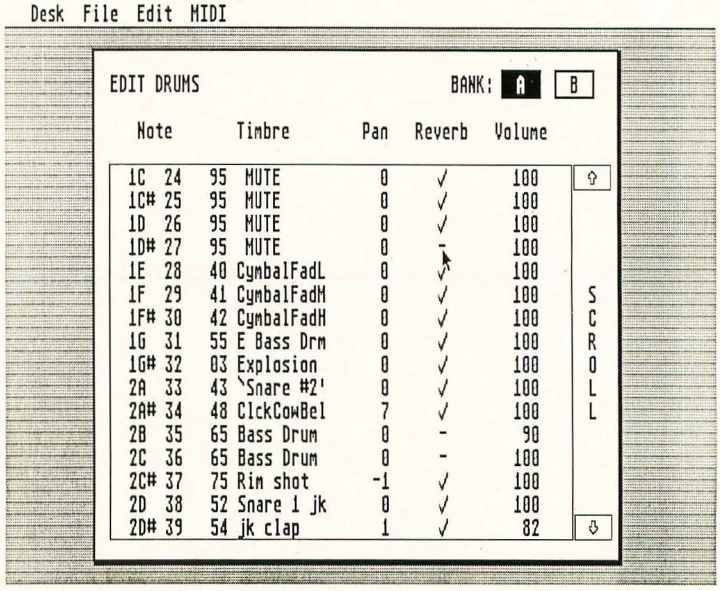
The most important program is the one that lets you edit and store your sounds. Appropriately dubbed MT_EDIT, this exists both as a stand-alone program and as a desk accessory.

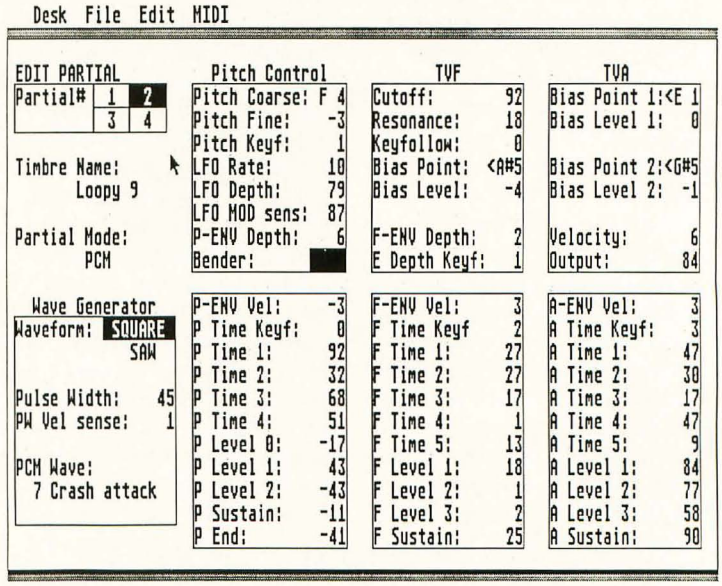
The program also remembers where you were last time you were on a certain page. You can tweak the sustain rate on the envelope page, go the system page, then return to the same parameter on the envelope page just by hitting one key.

Individual partials can be toggled on and off from any page by hitting Shift function key 1 through 4.

Having mouse-draggable points on envelopes is a lot of fun. I often







left this review to check on some feature, and then found myself spending an hour just playing with envelopes. One envelope window for each partial is displayed and each can be selected to show pitch, amplitude or filter envelopes.

Envelope parameters can be fine-tuned on the partial parameter page. The layout is logical and neat. All edits are done with the mouse. There's no way to select a parameter and then punch a value from the keypad. What's more, values do not loop—and it takes what seems to be a long time to get from 0 to 127.

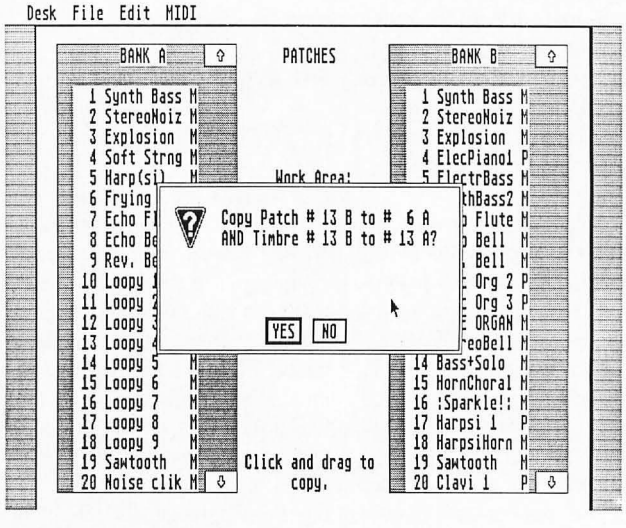
One feature I must mention is on the Edit Drums page. Whether Roland intended it or not, you can use this page to create a full keyboard on Channel 9 or 10, which can be completely split. "Completely" means that for each note you can specify sound (from the percussive files or from the normal 128 synth sounds), reverb status (on or off), pan, position and volume. I repeat, all of these can be set for *each note*! With power like this, you don't even need a multisplit keyboard!

Changes are immediately sent to the MT-32, so you can hear what effect the modification has made. You'll have to retrigger the note, however; it's not a real-time process. Now for the flip side. There are some odd features and outright omissions.

One of the oddities is the lack of a "compare" switch. It should be possible to return to the unedited sound without having to first save the timbre, bring the original into the work area, and listen to that. It should be a one-button operation.

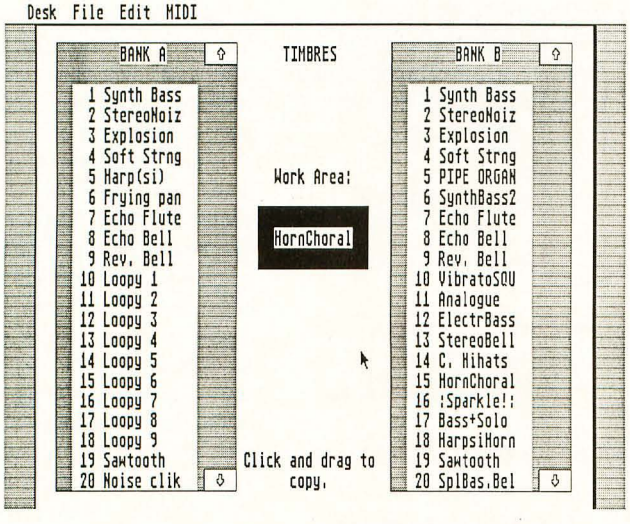
The procedure for copying timbres and patches from one bank to another is just plain weird. If I want to copy a patch from B13 to A1, the program will ask me to confirm that I want to copy the Patch Data from B13 to A1, and the Timbre Data from B13 to A13. The manual is no help here. Eventually I found that going to the Timbre page and copying the timbre from B13 to A1 would also copy the Patch Data from B13 to A1.

Time for a slight digression: Two years ago, when I was working for a theatre company, I had to come up with a whole passel of new sounds for a DX7II. What I wound up doing was copying out the parameters onto data sheets. I'd isolate operator stacks, label them and then recombine stacks from various patches. I'd design voices on paper and then do the fine-



tuning on the synth.

Now consider the MT-32. Although Linear Arithmetic synthesis isn't quite as cryptic as FM, the MT voice can have up to 400 parameters (about 100 for each of four partials). Even with the partial mute on, it's hard to remember which partial is contributing what to the whole sound. It would have been a simple thing to include a patch-print feature. It would have made programming much easier. Jeff told me that an update to include patch printing is in the works, and may be done by the time this is published.



In Spite of omissions and design problems, Koftinoff's editor has an honored place in my library.

One more complaint, and this one is directed at Roland: The omission of a battery backup for user memory is. . .well, stupid.

I bring this up because it makes the program both terribly important, and at the same time strangely futile. You need to have the ability to edit sounds—that's a cornerstone of synthesis. Yet the task of booting up the sender every time you want to play custom sounds can lose its appeal in a hurry. Some users may decide the effort isn't worth it. Others may not.

In spite of the omissions, and in spite of Roland's design problem, Koftinoff's editor has an honored place in my software library. It lets me mess around with my sounds. It lets me create the sounds that Roland should have built into the unit to begin with. It lets me customize the presets and make them my own. And as long as it does that, I'm willing to put up with the all the rest. ■

***Michael Friesen**, a full-time journalism student at the University of Western Ontario, is a MIDI fanatic who likes to annoy retailers by asking them about products that haven't yet been released.*

The ST GAME-SHELF

Out Run

by Sega
Distributed by
Mindscape Inc.
3444 Dundee Road
Northbrook, IL
60062
(312) 480-7667
$49.95, color only

Reviewed by Scott Wasser

Let's get philosophical for a minute. If a tree falls in the woods and nobody is around to hear it. . . .

Oops, sorry. Wrong philosophy class. We're talking computer software here.

Okay, let's start over. If a tree falls on your favorite coin-op arcade machine and destroys it, what are you going to do?

Yeah, I know it's a crazy question, but remember we're talking philosophy here. Philosophy is all about crazy questions. Most of the time they can't even be answered, so we're ahead of the game here because I'm going to answer this one: If your favorite arcade game happens to be *Out Run*, then buy the ST version recently released by Sega and distributed by Mindscape. It is one of the most accurate translations of a coin-op you're likely to find for any personal computer.

Ah, but—and here's where we get into the really deep philosophy, folks—that raises two new questions. How accurate a translation do we really want for our STs, and how much should we expect to pay for it?

Don't expect me to provide definitive answers to those questions. I can't do all the work for you. But, as Jed Clampett might have said on *The Beverly Hillbillies*, I will commence to do some philosophizing.

Like other driving simulations, the object in *Out Run* is to get from Point A to Point B by completing intermediate stages within the time allotted.

Seems to me the money question isn't too difficult. Think about how much you like a particular coin-op, try to figure out how many times you've already played it and guesstimate how many times you might play it if you owned it. If you multiply that last number by 25¢ and come up with a figure that's more than the going price of the home version, seems to me you might want to run right out to your nearest software retailer and pick up a copy.

If all your calculations were correct, you'll probably consider your investment a reasonable one. But that doesn't necessarily mean you'll be totally happy with the program. You see, that would contradict a philosophical theorem that was formulated not long after the first *Pong* video games found their way into homes all

across the country. The theorem states that it's impossible for a video or computer game to ever completely satisfy the person who buys it. We always expect a little more. And, unfortunately, the ST version of *Out Run* simply doesn't deliver. As close to the original as the ST version is, the tens of thousands of video-game fanatics who fed countless quarters into the *Out Run* coin-op will probably want more than just a faithful translation after they've played the home version a few dozen times.

To start with, they might want an instruction manual that's got a little more information in it. The one provided is a bit on the sketchy side. Fact is, if you eliminate the page of instructions that explain how to load the program on three different computer systems, it contains little more information than you'll find pasted to the front of a good arcade stand-alone.

I did appreciate the page containing a chart-type map of the *Out Run* course. The timing, scoring and tips section was also appreciated, although the directions explaining how to register high scores were incorrect. Another waste of space was the labeled photograph of the game screen. I can't imagine anybody being unable to figure out everything about the screen simply by playing *Out Run* one time.

These minor flaws in the documentation could be ignored if the instructions were otherwise complete. But they're not, if for no other reason than they fail to thoroughly explain the drop-down option menus in the ST version of *Out Run*. For example, one of the menus contains an option called "Continue." There's no reference to it in the instructions, and I simply couldn't figure out what it's supposed to do. What I wish it would do is give me the option of continuing a game from the point at which it normally ends. But I couldn't figure out any way to do that.

Like other driving simulations, the object in *Out Run* is to get from Point A to Point B by completing intermediate stages within the time allotted. On the coin-op, you know you're going to have to drop another coin in the slot and start back at the beginning if time runs out.

But I've always felt that purchasing the home version of an arcade hit should entitle the customer to be able to play a game out to its conclusion. And since few of us are skilled enough to beat even the simpler arcade games, that would require a real continue or resume option that *Out Run* seems to lack.

It's a shame because, like the coin-op version, the ST program is an enjoyable, addictive game. Animation is smooth, creating a feeling of real movement as your sports car cruises over dips and curves in the blacktop, and roadside scenery flashes past. Occasionally, you'll notice a little glitch in the animation (usually when passing another vehicle), but mostly the visual effects are first-rate.

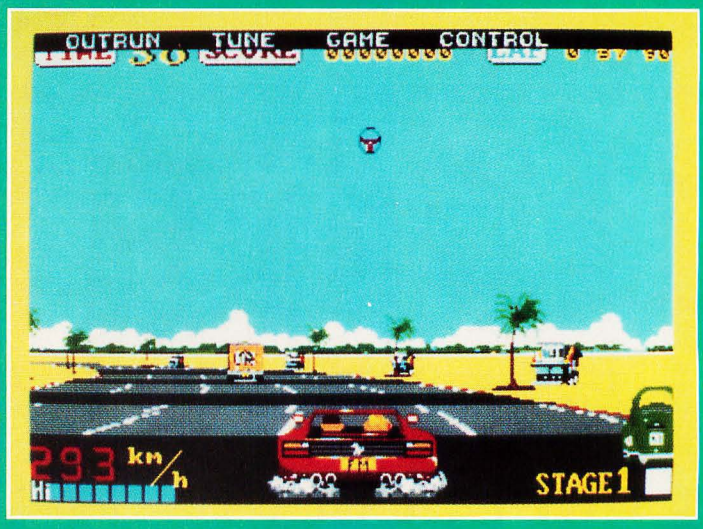
Playability is also first-rate. *Out Run* can be played using either a joystick or mouse, and response to input from either device is quick and accurate. Moving the mouse or joystick left or right steers the car, and the fire button or right mouse button shifts between high and low gear. Holding down the left mouse button or pushing the joystick forward makes the car accelerate. But even when using a good stick such as Mindscape's Powerplayer, negotiating turns and regulating your speed is a little tricky with a joystick.

Finally, *Out Run* gives players the option of listening to any one of three tunes (or no music at all) while tearing down the highway. All three songs are catchy and seem appropriate for the fast-paced on-screen action.

If you like fast-paced driving simulations or find yourself drawn to the coin-op version of *Out Run*, you'll undoubtedly enjoy Sega's ST translation. You'll have to play it at least 200 times to justify its asking price, but that's not an unreasonable expectation for a game that looks and plays as good as this one. It's just a shame that the designers didn't add features that could have given it an even longer play life.

Recommendation: Buy it. ■

If you like fast-paced driving simulations or find yourself drawn to the coin-op version of *Out Run*, you'll undoubtedly enjoy Sega's ST translation.



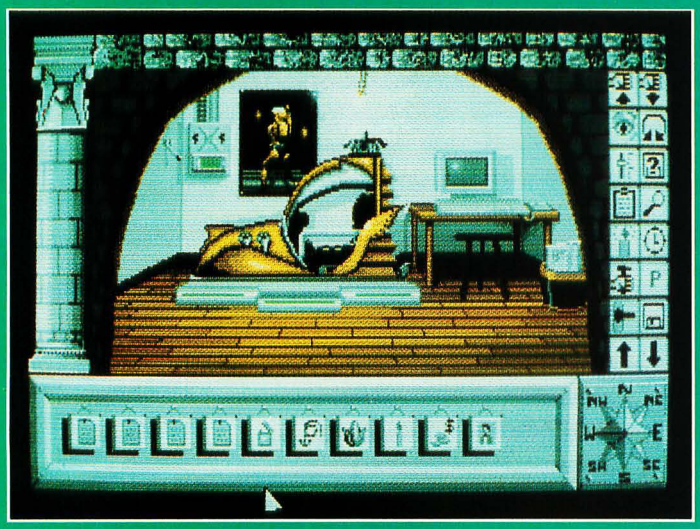
CHRONOQUEST

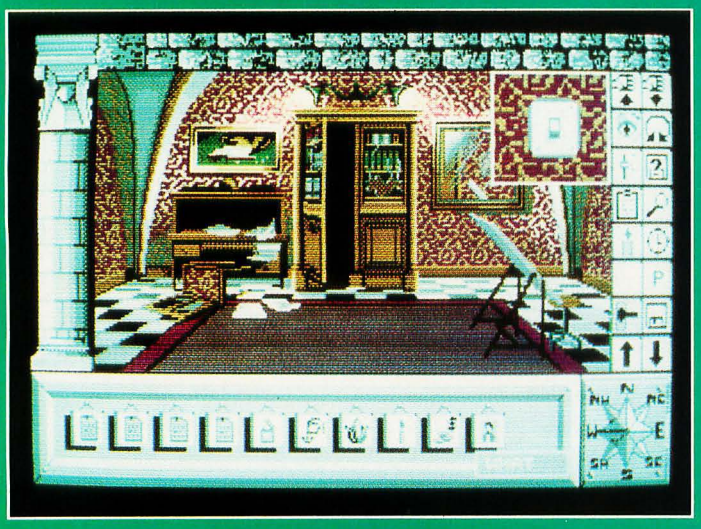
Psygnosis, Ltd.
First Floor Port
of Liverpool Building
Liverpool L3 1BY
United Kingdom
$49.95, color only

Reviewed by
Frank Eva

Psygnosis is a software house that has made a name for itself with action/strategy games and knock-out graphics. They were responsible for such titles as *Deep Space*, *Barbarian* and *Terrorpods*. The firm's latest release, *Chronoquest* is its first attempt at an adventure game. Has Psygnosis brought gamers something that is different/better than the vast quantity of adventure games already on the market?

First of all, *Chronoquest* is boxed in the familiar glitzy Psygnosis packaging. The semihard case holds the instruction manual, a poster, a semitransparent plastic overlay and four disks. That's right, *four*: a boot disk and three data disks!





The scenario: It seems that you are the estranged son of a "mad" scientist, who claims posthumously, in a letter to you, that he has invented a time travel device and has taken trips to the future and the past. The time period you initially find yourself in is 1922, and yet he speculates wildly about such instruments as computers that he found in the future world of 1987.

Your dead father used up most of his wealth perfecting time travel and found it necessary to steal artifacts from other time periods, which he sold to finance his efforts. Unfortunately, an unscrupulous house servant named Richard discovered what was going on and had to be sacked. Your father suspected that Richard was capable of evil—possibly even murder. Since you are reading a letter that was to be sent to you only upon your father's demise, you assume that Richard killed him.

Before his death, your father scattered the secrets to the time travel device throughout his own chateau. You must find the pieces to this puzzle and use the device to travel back in time. In each of four time periods, your father hid the pieces to the puzzle that unlock the secret of future time travel. These must be found in order to travel forward in time to 2125 AD, where you must attempt to find Richard, somewhere in Paris, and bring him to justice. You are warned that this trip will be disorienting. Paris is no longer the capital of France. There is a model society here, in which order and justice prevail. Due to the nature of the place, all conventional forms of direction finding are useless. You must go on courage alone.

The four time periods you will visit are:

Pre-history (33172 BC)—Useful tools have not been invented, so you must survive with brain and hands. Several references are made to "cave dwellings."

India (1605 AD)—The empire has reached its peak, but leprosy has decimated the population. You must be careful not to contract the disease yourself. The Holy Temple holds a piece of the puzzle.

Mexico (750 AD)—The Mayan civilization is in decline. The lush vegetation of the jungle contains many clues. Gain access to the temple of Chichen Itza, and you will find what you seek. This is the most difficult time zone of all.

Egypt (1100 BC)—Brave the arid deserts and find the pyramid. Make it through the tricky mazes in order to find the missing piece to the puzzle of future time travel.

The user interface is really what makes playing *Chronoquest* enjoyable. There is no need to type in text commands only to have them rejected by a finnicky parser. The game is totally mouse-driven. Two rows of icons frame the right side of each scene. Pointing and clicking on these icons performs the entire range of adventure-type commands, including Take, Drop, Use, Search, Examine, Push, Pull, Up, Down, Turn, Wait, Inventory, Load and Save. Furthermore, there is a compass icon for inputting directions of movement. All the player has to do is click on one of eight points of the compass, and if travel is permitted in that direction, the program takes him/her there.

A box framing the bottom of the screen is used for displaying status information (text), and also holds icons representing the player's current inventory.

And that's not all! There is an accompanying sound track and a spattering of animation thrown in. The animation can be turned off, or adjusted to user preference by clicking on an icon. The animation represents actions such as opening and closing doors and furniture drawers, and is accompanied by an appropriate sound.

Last but not least, the quality of the artwork is only a slight step below that found in adventures produced by a premier adventure game company, Magnetic Scrolls. Complaints? For one thing, I have not been able to discover a way to eliminate the background music without disabling the sound effects that accompany the animation, which you obviously do when you turn down the monitor's volume.

Also, Psygnosis' new method of

Continued on page 94

ROAD RAIDERS

Gray Matter
Distributed by
Mindscape, Inc.
3444 Dundee Road
Northbrook, IL
60062
$39.95 color only

Reviewed by
Peter A. Smith

You curse silently (okay, not so silently) as your ATV hits an oil slick, spins out of control and slams into the guardrail. The rebel car that's been tailing you takes full advantage, and rams you broadside. You regain control of your vehicle and blast the rebel with a shell from your forward-mounted cannon. A glance at your damage indicator tells you that repairs are needed. You know where the nearest gas station is, but first you'll need a supply of food to exchange for services and goods. You'll have to risk a trip inside one of the buildings. You know that in the buildings you can find food, gas. . . and hordes of "green, slimy mutants." Better make sure you've got plenty of ammo for your sidearm. You'll need it.

Road Raiders is one of the latest offerings from Mindscape, set in a post-disaster world. This time, the disaster was a food shortage, combined with the invention of the Slu food substitute, a substance which, the world discovers too late, turns anyone who eats it into one of the aforementioned green, slimy mutants. Your role is that of bounty hunter, and your quarry is Dr. A. Noid, sly inventor of Slu.

Road Raiders comes on two disks, one of which is copy protected. A color monitor and joystick are required. Neither hard disks nor RAMdisks are supported; a serious omission as we will see later. The graphics are fairly well done. I especially like watching the mutants clamber out of their holes, and your on-screen persona has a sort of one-too-many-car-wrecks limp as he runs. It fits in with the whole Mel Gibson/*Road Warrior* gimmick. The backgrounds are nicely detailed; many an "Eat Slu" sign can be glimpsed as you fly past in your ATV, and the insides of the buildings are littered with empty cans and boxes, which add to your challenge as well as to the ambience, since you must maneuver around them.

The controls are straightforward with almost everything accomplished by joystick, except for firing your smoke screen (spacebar), and housekeeping commands such as Save, Restore and Mode, the latter of which is a nice added feature. In Mode 1, your joystick is used in the usual driving way: forward increases speed, pulling back decelerates, and left and right turn your car accordingly. I prefer this mode as it has a nice driving feel to it, especially when you take a corner at high speed and your vehicle fishtails realistically. Mode 2 is more basic. It controls your car through the streets like Pac-Man through his maze. Point the stick and the car goes in the corresponding direction.

But, as Shakespeare's Hamlet said, "The play's the thing." And I have news both bad and good to report. Let me get this off of my chest. The disk access is horrible! It takes approximately 30 seconds to load the screens every time you enter a building, and the same to load the roadway screens when you leave the building. The gas station is a bit better: about ten seconds to enter it, and 20 seconds to leave. This wait may not seem too bad, except that you are often ducking into a building, grabbing some food, driving around the corner to a gas station, buying some ammo, then driving back and re-entering the building. The long load times break up the flow of the game. On the positive side, all the mutants "reset" when you leave a building, so this feature may dissuade you from being tempted to bend the rules a bit by bobbing in and out of the building every time things start to heat up.

Let me get up on my soapbox. First, if you don't own a hard drive, then maybe this disk access thing won't bother you too much. But as any other hard-disk owner will attest, once you have one of these beastiés, you immediately become spoiled by the decreased load times, and anything loading off a floppy irks you. Why Mindscape couldn't have supported hard-disk users is beyond me. If copy protection was the concern, then, by all means, leave Disk A protected and use a "key disk"-type scheme. But let us load our data off of our hard or RAMdisks! End of lecture.

A couple of other faults. You can save your game, and you *will* have to use this feature. The problem is, you can only save one position on your data disk. The data disk is not copy protected (although nowhere in the documentation is this mentioned), so do yourself a favor and play the game with a back-up of this disk to avoid the risks of writing to an original. Fault 2: Occasionally, the program will miss a joystick input. This inevitably leads to driving into a brick wall, or walking into an open pit.

Strange though, as much as I griped about the disk access to my much-suffering computer widow, I kept slapping those disks back in, and leaving this world for the one of *Road Raiders*. I found driving through the maze of streets *fun*. You have to watch the land mines, of course, and the ever present rebel cars. And the inside of the

Continued on page 94

SKYCHASE

Maxis Software
953 Mountain View Drive
Suite 113
Lafayette, CA 94549
(415) 376-6434
$39.95, color only

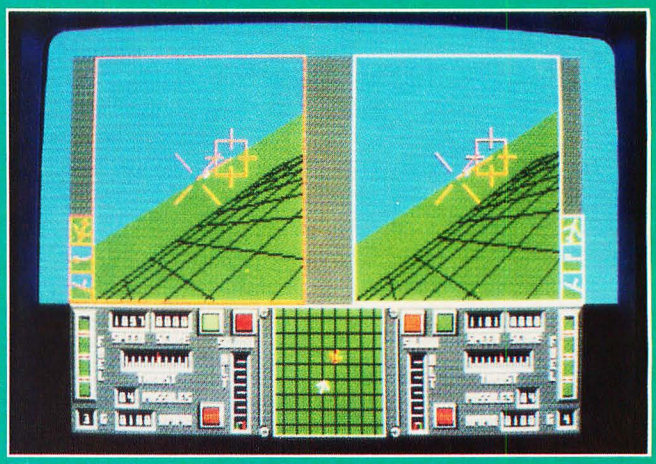
Reviewed by Scott Wasser

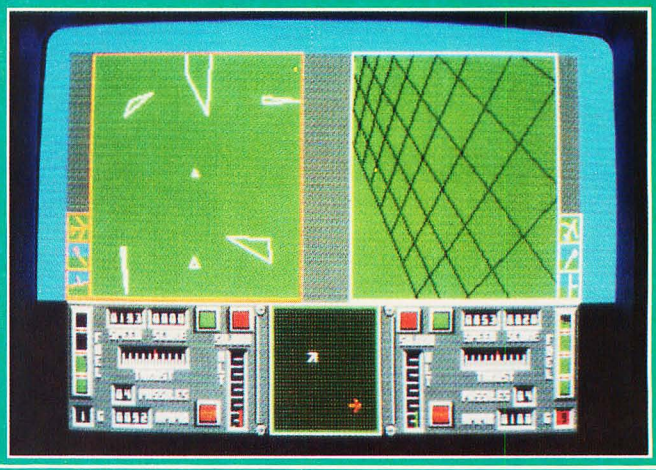
The software reviewer grumbles as he opens the package and surveys its contents. He thinks to himself, *Great! Just what I needed. Another flight simulator. Can't wait to try this one.*

He'd rather visit the dentist. Not that the reviewer has anything against fake flying. At one time in fact, he'd relished the opportunity to solo for a few hours in front of his computer monitor. Neat. Touch-and-goes by the dozen without ever having to pop a Dramamine.

But that was a while ago. The guy still enjoys flying his keyboard, but can't find much time for computerized flights of fancy. Too many new products to evaluate. Too many reviews to write.

Most are not too time-consuming. Sure, a thorough review requires hours of running the program to get a good feel for it. But that's the fun part of the job. The hard part is absorbing the documentation from some of the programs being evaluated. It can be a real grind when the program being reviewed comes with so many pages of documentation it





SPACE CUTTER

Rainbird Software
3885 Bohannon Drive
Menlo Park, CA 94025
(414) 322-0412
$29.95, color only

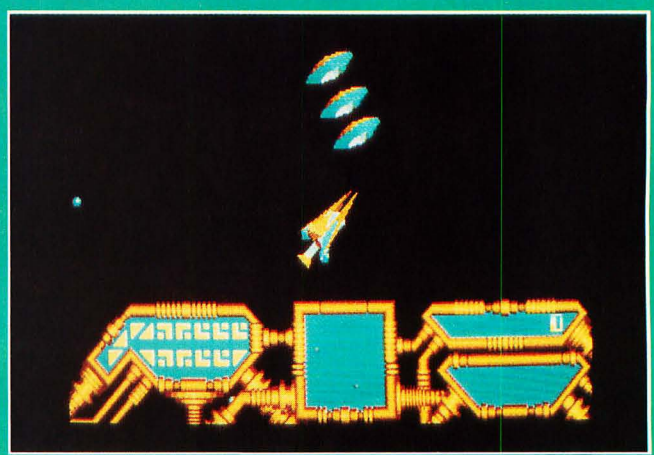
Reviewed by Frank Eva

Another in a series of arcade/adventure games, *Space Cutter* comes from the United Kingdom, where the absolute rage seems to be 3D-modeled graphics. Unfortunately, *Space Cutter* falls short in several critical game-playing areas, and is, consequently, a far cry from other Rainbird titles such as *Starglider II*.

The fictional background contained on the packaging is somewhat misleading. You are told that playing *Space Cutter* is "A daring escape to freedom." Supposedly, the premise is that you have stolen a spacecraft that no other human has ever been capable of piloting before, and that your thoughts of freedom spur you on to your quest.

Never do you really make a mad dash for freedom. In fact, you take up the role of adventurer, in search of five objects scattered throughout the area of known space referred to as the "whirligig." Along the way, you battle attackers that vary from passive to downright hostile.

The documentation consists of



would have made Gutenberg consider another line of work.

Guess which type of software usually comes with the most documentation (not counting virtually any MS-DOS program, many of which appear to have been created primarily to keep this country's paper mills in operation). Need a clue? Let's just say that if these programs had been around in Orville and Wilbur Wright's days, we'd still be riding railroads to go cross country.

Hopefully, you're now beginning to understand my aversion to flight simulators. Most of those I've tried are great, but they're accompanied by reams of documentation that make them a chore to learn. Fact is, I got through four years of college without having to read as much.

So picture my reaction when *SkyChase*, a program from Maxis Software, shows up in the mail. True, I'd accepted the assignment from ST-LOG Editor, Clayton Walnum, but surely you can understand my having second thoughts as visions of 1,000-page instruction manuals danced in my head.

But wait a minute! What's this? The package only weighs a few ounces. What did they do, forget to pack it? Let's see. Hmmmm... only one disk. And a "Flight Manual" that's only 19 pages long. Impossible!

No, not impossible—ingenious. Credit *SkyChase* designer and programmer Ralph Russell with performing an impressive feat. He has succeeded in taking the essence of modern air-to-air combat and distilling it into a exciting, enjoyable and satisfyingly realistic jet-fighter simulator.

Did I forget to mention that it's also amazingly easy to learn?

The main reason for that is *SkyChase*'s design. This isn't the kind of flight simulator in which the person at the computer works up a sweat punching keys, trying to perform every simulated action from starting the ignition to removing the wheel chocks. Russell decided instead to concentrate on simulating the action that takes place after two combat fighters begin tangling in the ozone.

The game begins with the jets already airborne. Maneuvering them is a breeze. Using a joystick (a mouse cannot be used), the pilot makes the plane climb, dive or bank (turn) by simply pushing the joystick in the corresponding direction. The only other flight element controlled by the player is thrust, which determines the aircraft's speed and is increased or decreased using the keyboard.

Maintaining control of the jet has also been simplified by keeping the number of on-screen instruments to a minimum. There is a radar-type screen that shows the jet's position and direction, and a perimeter indicator that warns when a jet is approaching the boundary of the predetermined combat area.

Other indicators show altitude, degree of thrust, speed, G-forces being exerted on the pilot, fuel and ammunition remaining and the status of the automatic missile-targeting system. Finally, there are three little graphic displays that depict the jet's attitude in terms of pitch, heading and banking.

I'll grant you that on paper that may sound like a lot of instrumentation. But compared to other flight simulator's, *SkyChase*'s cockpit display is downright sparse. And that suits this jet jockey just fine, since it enabled me to concentrate on important stuff like trying to obliterate my airborne enemy before he scattered my molecules in the wind.

But the best benefit of the scantily clad cockpit is that it allows the ST's sharp color monitor screen to show two such cockpits at the same time. Each one represents the view from a different jet. That makes for some intriguing dogfights, since you have a chance to sneak a peek at your opponent's instruments to see what he's up to. You can even torment yourself by watching as his air-to-air missile sight locks onto your soon-to-be-defunct fuselage.

The split-screen mode can be turned off if you're battling against the computer. That's just one of many play options offered by *SkyChase*. While space doesn't per-

Continued on next page

a single manual, 3×4 inches in size, totalling 16 pages. ST owners should appreciate the fact that their disks are not copy protected. However, it is hoped that this will not spawn a rash of illegal copying, which is still an infringement on Rainbird's copyright.

The modeled graphics are indeed superb. They are colored realistically, rendering a believable three-dimensional effect. The animation is also excellent. Each of the major objects rotates a full 360 degrees and looks fine in every position. This is *Space Cutter*'s best feature. On the other hand, while modeled in three dimensions, *Space Cutter* is played in only two. There is no perspective of screen depth to the game, as there is in *Virus* and *Starglider II*.

The background starfield is actually many screens linked together by multidirectional scrolling. However, since expanded playfields or overscan (popular on the 8-bit Ataris) are not supported, the player is really only viewing the starfield through a window. The unfortunate part of this is the relationship of window size compared to the size of the on-screen attackers, which are so large, they seem to be upon you before you can react.

The player's ship is always centered on the screen, while the surrounding starfield scrolls in whichever direction the ship is facing.

A control panel can be accessed by pressing the control key. This panel displays several important status indicators, including supplies of fuel and weapons. It also indicates what supply depots are present within the current sector of space. And finally, a radar map is displayed that indicates the positions of supply depots, star gates, and your own ship. Unfortunately, there is another major drawback here. The map does not display the positions of enemy ships! You have to guess that for yourself. Also, the position of "perfect solids," the objects you are looking for, are not shown on this map. So, you will doubtless wind up doing a fair amount of wandering.

The lack of joystick control is *Space Cutter*'s worst deficiency. Mouse control is extremely awkward; nothing like the system used in *Starglider II*. When asked about the possibility of adding joystick control, the technical staff at Rainbird explained that they could not put any more time into the product, without jeopardizing its price point; and they do have a point. The product's price is extremely competitive.

A point in favor of *Space Cutter* is its mathematical modeling of known space. This allows for a tremendous number of sector configurations, and consequently, many levels of play. The documentation states that over 4 billion sectors, spanning 11 levels of progressive difficulty are possible. This does not necessarily mean that you will have to conquer each sector in order to ultimately finish the game. That would take an estimated 8,171 years of continuous play! Through the use of selected star gates, you can jump whole sections of the galaxy in search of the puzzle pieces. Since the last piece is found in Sector 33550336, the player is provided with two "road maps." Mind you, it will take a minimum of 36 jumps through star gates to reach this goal. Then, you must find the puzzle piece and haul it to a nearby star gate in order to complete the game.

Unfortunately, for the arcade enthusiast, calculating the positions of objects in each new sector seemingly takes forever to accomplish. In the meantime, the player just sits and waits. When the calculations are complete, the sector map turns red, and you can then enter real space by pressing the left mouse button. Once the new sector has been calculated, though, the wait time is minimal, which means that if you lose a life, you won't have to wait for recalculations in order to get back into the heat of battle.

In conclusion, *Space Cutter* may appeal to avid space adventure fans, but it will surely disappointment arcade purists. The awkward mouse interface must take the blame for this. And with no game save routine, adventurers will have to devote many hours of continuous play, in order to finish the contest.■

Recommendation: Get a demonstration before buying.

SKYCHASE

Continued from previous page

mit listing all the options, let's say there are enough of them to extensively alter play parameters and permit some intriguing combat scenarios. Also appreciated is the design of the option menu and submenus, which are impressively easy to use.

While all of these features are wonderful, they would have been wasted on a program with mediocre animation. After all, how much fun can you have with a jet-fighter simulation in which the aircraft seem to move no faster than one pixel per minute?

Thankfully, that's not the case with *SkyChase*, which features animation as smooth and fast as any ST program I've seen. Response to joystick input was also superb. And although I'm not a big fan of the wire-frame graphics used in *SkyChase*, the effect of flight was so realistic and the simulated combat so stimulating that it was easy to forget the enemy was nothing more than a hollow line drawing.

In other words, Maxis' *SkyChase* scored a direct hit with this reviewer. If you like your flight simulators spiced with wing flaps and ailerons and all sorts of other aeronautical gadgets, it's probably not for you. But if you'd rather shoot down bogeys than read compass headings, you'll love it.■

Recommendation: Buy it.

CHRONOQUEST

Continued from page 90

copy protection may be the cat's meow when it comes to protecting the company's right to make a decent profit on *Chronoquest*—an argument that no one can legitimately question—but it's inconvenient for the user. Remember the semitransparent plastic overlay mentioned earlier? It contains a grid with numbers down the left side and letters across the top. This overlay is placed on the box top. Upon booting the game, the program prompts the player to identify the color that matches a grid randomly chosen by the computer. This is done not once, but twice. Fail, and you are forced to reboot the computer.

Needless to say, I failed more than once. The problem is identifying the most prominent color in the grid requested. Frequently, there will be more than one color, and you wind up guessing at which one they want.

All whining aside, I believe Psygnosis has a real winner in *Chronoquest*—a B + at least. The programmers, Infomedia of France, are to be congratulated for elevating the genre of adventure gaming.■

Recommendation: A must for adventurers.

ROAD RAIDERS

Continued from page 91

buildings made for a nice contrast. Make no mistake, though. This is no game for the too-much-violence-in-video crowd. The mutants come at you endlessly, and the only way through is to lay about with whatever gun you have handy.

The arena is something totally different, and not at all what I expected to find. Rather than a reckless demolition derby, it can be delicate work maneuvering around deadly crevices while trying to smash enemy vehicles. Add to this such goodies as time bombs and trap doors, and you'll be short a few patches of hair before you make it through. After defeating the master of the arena, you will gain admittance to a second, more difficult city. Altogether you will visit three cities and three arenas in your hunt for Dr. A. Noid.

Road Raiders is basically an arcade game with a hint of adventure thrown in. You'd do well to map the city streets, and each building will take some time to explore. Also, while you don't improve personal attributes, as happens in fantasy RPG's, you can improve your weapons and vehicle. Add-ons include radar, a smoke dispenser and battle armor to name a few.

There is also an economic side to the game. While you may sometimes find extra equipment, usually you will have to buy the equipment with food, which is found lying about in the mutant-infested buildings. You'll have many tough decisions to make, such as when your ATV needs both repairs and fuel, but your food supply is such that you can't afford both. Also, the advanced weapons cost big. . .um. . .calories, so you'll have a tough time scraping up enough leftovers (sorry) to buy them.

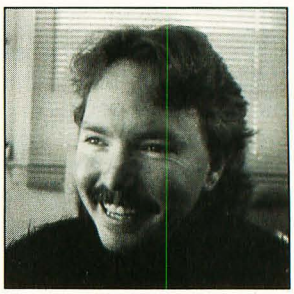
All in all, I would wholeheartedly recommend this game, if it were not for the problem with disk access. I have written to Mindscape about this failing, but to date they have not replied. If you are a big *Road Warrior* fan, or if you like a game that mixes arcade action with some simple adventuring/role-playing (and you are the patient sort), buy it.■

Recommendation: For arcade/adventure fans.

If you'd rather shoot down bogeys than read compass headings, SkyChase is the flight simulator for you.

Scott Wasser *has been a daily newspaper reporter and editor for the past 12 years and has been interfacing with computers for the last four. He has written columns and feature stories about computer hardware, software and home electronics, and is a regular reviewer for ST-LOG.*

Frank Eva *is an auditor by profession, but has been involved in the computer industry ever since his purchase of an Atari 400 many years ago. He has dabbled in programming and has had several text adventures published.*



Peter Smith *lives in Sag Harbor, New York, with his wife and in-house editor, Julie. When not in front of his computer, he is a bicycle enthusiast, a struggling musician and a poet. He can be reached on GEnie at the address Capt.Cook.*

Juno 106 Synthesizer *(from page 65)*

If you're still experiencing the tapioca effect, any introductory text on computer programming should be able to give you more information.

Incidentally, you can check your work by adding the remaining numbers ($2^7 + 2^5 + 2^1 = 162$). The "values" of the 0s added to the values of the 1s should work out to 255 for an 8-bit number, 127 for a 7-bit number and 31 for a 5-bit number.

The Strange World of Bit-Mapping

If you ever tried to program the sound chip of the C-64, you'll already be familiar with the process of bit-mapping. The whole idea is used to save memory and speed up data transmission. Consider the J-106. There are 11 noncontinuous parameters in the voice. When I say "noncontinuous," I mean that a given parameter has only a few possible values.

The LFO Rate is a continuous parameter: It can have integer value between 0 and 127. The high-pass filter, by comparison, has only four possible settings, and the pulse waveform has only two: on and off.

You could assign each of these switchable parameters to a full byte of data. A value of 0 would indicate the off position, and a value of 127 would indicate on. But why use a full byte of data when you can store the same information in a fraction of a byte? A value of 1 could indicate that the pulse wave is on, while 0 would indicate the off state.

The high-pass filter has four possible values. That range can be expressed using two binary digits where 11 is off, and 00 indicates the maximum setting of 3. (A high value on the synth doesn't always correspond to a greater binary value. Don't ask me why, I didn't build the thing.)

So now we've put two values into three binary bits. That leaves us with up to five more bits in which to store other switched data. Roland doesn't always use up the full eight: The first of the switch-bytes uses only seven bits, while the second uses only five.

Michael Friesen has worked as a bookbinder, a chicken-shed cleaner, a cow milker, a ditch digger, a jingle composer and a synthesist for a touring theatre company. He's now a full-time journalism student at the University of Western Ontario. ■

JUNO-106
Listing 1:
GFA BASIC

```
Dim Senddata%(20)
Out 3,240   ' send start of sys-ex to the MIDI Port - Out 3
Out 3,65    ' send Roland ID number
Out 3,48    ' Function Type - we're transmitting a full voice
Out 3,0     ' MIDI Channel #
Out 3,0     ' Sending to Location # 11
'
For Counter = 1 to 18
     Read Senddata%(Counter)
     Out 3, Senddata%(Counter)   ' send Program parameters
Next Counter
Out 3,247 ' Send end of sys-ex message
'
Print "Done Sending Data"
'
Data 57,45,0,55,0,85,0,0,0,25,52,59,32,86,40,0,26,24
```

JUNE 1989 · DISK LISTINGS

The ST-LOG #31 diskette contains 19 magazine files. They are listed below.

FILENAME.EXT	FILE TYPE	COMMENTS
\BALLBUST\		
BALLBUST.PRG	RUN FILE	BALLBUSTER
BALLBUST.C	C	SOURCE CODE
\GFAPRINT\		
GFAPRINT.PRG	RUN FILE	GFA SOURCE PRINTER
GFAPRINT.LST	GFA BASIC 2	SOURCE CODE
\MIDICAPT\		
CAPTURE .PRG	RUN FILE	MIDI CAPTURE
BI2 .BAT	TEXT	BATCH FILE
BINT .BAT	TEXT	BATCH FILE
CAPTURE .C	C	SOURCE CODE
INTR .S	ASSEMBLY	SOURCE CODE
\PDPARADE\		
MUSICALC.PRG	RUN FILE	MUSIC CALC
MUSICALC.TXT	TEXT	DOCUMENTATION
\SAFEKEEP\		
SAFEKEEP.PRG	RUN FILE	SAFE KEEPING
SAFEKEEP.PAS	PASCAL	SOURCE CODE

DISK INSTRUCTIONS:
Only those files with .PRG, .TOS or .TTP extensions may be run from the GEM Desktop. Other programs may require additional software as shown below.

WARNING: Be sure to read the appropriate magazine article before attempting to run the programs on this disk. Failure to do so may yield confusing results.

.EXT	DESCRIPTION
.BAS	Requires ST BASIC
.C	Requires C compiler
.PAS	Requires Pascal compiler
.S	Requires 68000 assembler
.LST	Requires GFA BASIC or GFABASRO.PRG

INDEX TO ADVERTISERS

This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.

I Was a Teenage SYSOP!

by Greg Knauss

During a recent summer, strange and horrible experiments took place at my house. Awful things. Terrible things. Things to fray nerves, to make skin crawl, and to keep one awake a night.

I became a. . . SYSOP.

A system operator (or SYSOP) is a person who runs, at great personal expense and little personal gain, a bulletin board system (BBS) from his house. A BBS is a fiendish concept designed to take all your computer equipment away from you: You set up your computer with a program to accept outside calls over your modem.

The system I ran (or rather, staggered) was S.W.A.M.P.S. (South Western Atari Message and Program Service) out of Gardena, California, [213-324-0218] taken over from a friend of mine while he and his wife had a baby. I remember setting the beast up, excited and even looking forward to the weeks ahead. I was going to be a SYSOP! I could validate users and do the housekeeping and. . . uh. . . and. . . er. . . .

Aw, who cares? I have a BBS. That first day, still giddy with the thrill of it all, I sat in front of the screen and waited for someone to call. I waited and waited and waited. Then I waited some more. I did a bit of waiting until I finally decided to wait. After that I waited, then did some much needed waiting.

Boy, that was fun.

Eventually, people did start to call. Usually, in the middle of the night. I know they called in the middle of the night because when they did, my modem gave a sound approximately equivalent to a teakettle whistle fed though an amplifier. Then my drives ran for about half an hour. *Grind, grind, whirr.*

Maybe this wasn't going to be as much fun as I had thought.

The joy of losing my computer to the BBS quickly became the indifference of losing my sleep to a noisy modem, which quickly became the horror of watching my drives catch fire from the heat of being on 24 hours a day. Mike, my friend, phoned up and said, "You might want to keep a cooling fan on them. Those older drives weren't designed to run that long."

Whimper.

After an afternoon of pure panic, I got the drives back up, and the BBS was running again. I pointed a vintage World War II fan at the cooling vents on my drives and hoped for the best. "The best" turned out to be a sleeping environment that rivaled being strapped to the wing of a Cessna. At least the BBS was running. Sort of. A good SYSOP is 30% moderator, 30% comedian, 30% encyclopedia and 30% programmer. If you noticed that that comes out to 120%, you'll know why there are so few really good SYSOPs.

Hide it though I tried, somehow it leaked out that I was not a good SYSOP: Maybe it was the fact that all the downloads disappeared one day or that message bases would suddenly spontaneously combust or that all the electronic mail was delivered to the wrong people. Complaints started coming in. People asked Mike to take back the BBS so that it could be in competent hands again. I paid 80 bucks to have a line installed for this?

I was determined to improve, though. Mike kept saying that this was an acid-test for his home-brew BBS program; if I could handle it, anybody could.

Finally, after spending about three hours a day for a week on the silly thing, the situation started to improve. The message bases were running smoothly; the number of callers was picking up; people seemed satisfied.

This, of course, is just about the time the data disk chose to get a bad sector.

When a BBS data disk gets a bad sector, it's the approximate equivalent of a brick wall suddenly appearing in front of a speeding car. Things rapidly went higgly-piggly.

As I was attempting to straighten out the bad sector mess, I accidentally erased the message allocation map, the thing that tells the computer where each message is on the disk. This is the equivalent of dropping seven tons of cement blocks on top of the car that hit the wall.

As I recall, I screamed a lot.

The BBS was a complete mess. I've seen explosions with more organization.

And so I took the chicken's way out: I reinitialized the system. I took everything and started over. This had only been done three times before in the history of the S.W.A.M.P.S., in its three years of existence. I did it after three weeks.

And after messing up, I did it again.

And then I did it again.

Three times. That's got to be some sort of record. It became sort of a running joke: Russian BBSing, you stood a one-in-six chance of having the S.W.A.M.P.S. be up. It was an elevator BBS: It's up! It's down! It's up! If the BBS blows up while you're on-line, you win a prize!

Finally, six weeks after I took the monster over, Mike wanted it back. The price of forwarding all the calls from his number to mine was higher than he had expected.

I didn't object.

When I first took over S.W.A.M.P.S., I had visions of my own BBS dancing in my head once I gave Mike his board back. I would set it up and run it and everything. It would be my BBS. Though I did enjoy running the S.W.A.M.P.S. (believe it or not), I realize now how much work goes into it. I had always respected SYSOPs before, but I had never realized how much they go through every day keeping those beasts up and running, just so users can call—and for free, at that!

When you call BBSs, be kind to the SYSOPs, respect their rules and give them support. The life of a system operator is not easy.

And I, personally, would rather have my nose cut off than run my own BBS.■

Greg "Maddog" Knauss, 21, lives in La Jolla, California and attends the University of California, San Diego. He says he is feeling much *better.*

ST-LOG invites all authors to submit essays for possible use in the Footnotes *department. Submissions should be between 1,000 and 1,500 words and may be on any aspect of Atari computing. Any style or type of essay is acceptable—opinion, humor, personal experience—but creativity is a plus. Send your submission to:* Footnotes, *c/o* ST-LOG, *P.O. Box 1413-M.O., Manchester, CT 06040-1413.*




12 Issues $28

$19 OFF THE COVER PRICE

12 Issues with Disk $79

NEW LOWER PRICE

BREAK AWAY FROM THE PACK

The world of ATARI-ST continues to grow by leaps and bounds, and ST-LOG is there every step of the way! We stand apart from the competition by offering more color, comprehensive reviews and in-depth features. SUBSCRIBE NOW!

☐ **12 Issues $28** ST LOG
MCRWW

☐ **12 Issues with Disk $79**
DCRWW

☐ PAYMENT ENCLOSED ☐ BILL ME
CHARGE MY: ☐ VISA ☐ MASTERCARD

CARD # EXP SIGNATURE

NAME

ADDRESS

CITY STATE ZIP

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires August 30, 1989.

REVOLUTIONARY NEW PRODUCT

SWITCH BACK

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

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

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

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

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

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

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

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

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

ONLY $69.95

ST Protection Techniques

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

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

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

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

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

Digisound ST

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

DIGISOUND PROFESSIONAL

All the excellent features of DIGISOUND plus these great extras

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

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

$149.95

Beat Box

Is it a Drum Machine? A sequencer? A new concept in digital sound? The answer is - YES!! It's all this - and so much more!! It's a polyphonic song construction set that turns your ST into a drum machine and digital sequencer. Now anyone can be a master composer. No musical knowlege required!

Just point and click to create facinating drum, voice, or musical patterns in four voices. Combine and arrange patterns to form complete musical compositions. Play the sounds, patterns and songs through your montior speaker or digitizer hardware.

You don't need a digitizer to enjoy Beat Box. It comes with over 35 ready to use digitized sounds. Or you can use your own sounds recorded with a Digisound ST, Professional, or other digitizer.

Beat Box $29.95

COLOR COMPUTEREYES™

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

SPECIAL OFFER – Buy both Computereyes and Powerprint and SAVE 20.00 from the total.

PowerPrint ST

BLOW YOURSELF UP

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

POWERPRINT

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

Only $39.95

DIGISPEC

DIGISPEC is an exciting new breakthrough in computer video digitizing. **DIGISPEC** works with your Color ComputerEyes to create **spectacular 512 color** video images. Now you can capture and display video pictures in unsurpassed detail and clarity, with 512 different colors on your ST screen.

DIGISPEC is easy to use. It works on any ST computer. Simply capture an image with your Color ComputerEyes and color video camera (or VCR, video disk, etc.). Then run **DIGISPEC,** and watch as your picture appears in a full rainbow of 512 true colors.

DIGISPEC includes a special shading feature to give you even more detail and color. The shading feature uses a technique called dithering, which creates and displays images in 3,375 or even 24,389 simulated colors.

DIGISPEC creates detailed, low resolution video images on any Atari ST, color video camera (or VCR), and Color ComputerEyes. Every Color ComputerEyes needs **DIGISPEC!**

Only $39.95

Polydisk

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

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

Polydisk (512K Solid state drive)
Clock option card is also available for Polydisk

24 HOUR HOTLINE — VISA & MasterCard Welcome

216-374-7469

Call or write for free catalog.

Customer Service line (216) 467-5665 M-F 9 AM-3 PM EST

ALPHA SYSTEMS

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

CIRCLE #119 ON READER SERVICE CARD.

Backup For The Future

FA•ST

Before you go any further, back up. And take a good look at what you're leaving behind.

Think about all of that data you've stored away. Information that's taken you months, even years, to enter. Not to mention all of the files you've collected. The programs you've written. The music you've composed.

Now, try to imagine the future without it. All that work, *lost*. All that time, *spent*. All that money, *gone*. And all because you failed to backup for the future.

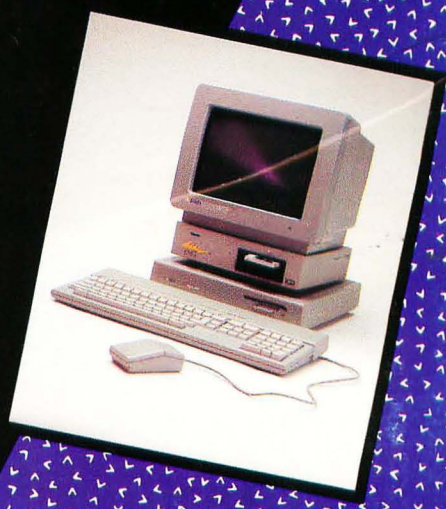
Preparing for your future is what the FA•ST Tape Backup does best. Offering the promises of tomorrow without the risks of today. It's the only tape backup system for the Atari ST that makes data loss a thing of the past— for the good of your future.

So now there's no turning back. Not with the FA•ST Tape Backup, as it secures up to 155 megabytes of valuable data at a rapid rate of six megabytes per minute. It knows exactly how to keep all that data on the right track too, either by file or partition restoration. With back-to-back enhancements that make it effortless.

Like its user friendly GEM-style interface, an image backup mode for Magic Sac or Spectre 128 partitions, a built-in battery backed clock and daisy chaining through its DMA and SCSI ports. And for the times you need to share the FA•ST Tape Backup with other computers, its one-of-a-kind push button SCSI ID switches put the selection at your fingertips.

If that's not enough, the FA•ST Tape Backup can pack a complete hard drive right inside its compact case at your request. Choose from either a 50, 100 or even 200 megabyte system— when you really need a Backup with a lot of drive.

The FA•ST Tape Backup. Think about it... a future without data loss. And then prepare for it. Because you really do have too much to lose.



Call or write for a free catalog today.

ICD

ICD, Inc. • 1220 Rock Street • Rockford, Illinois 61101
(815) 968-2228 • BBS: (815) 968-2229

CIRCLE #120 ON READER SERVICE CARD.