ISSUE 72

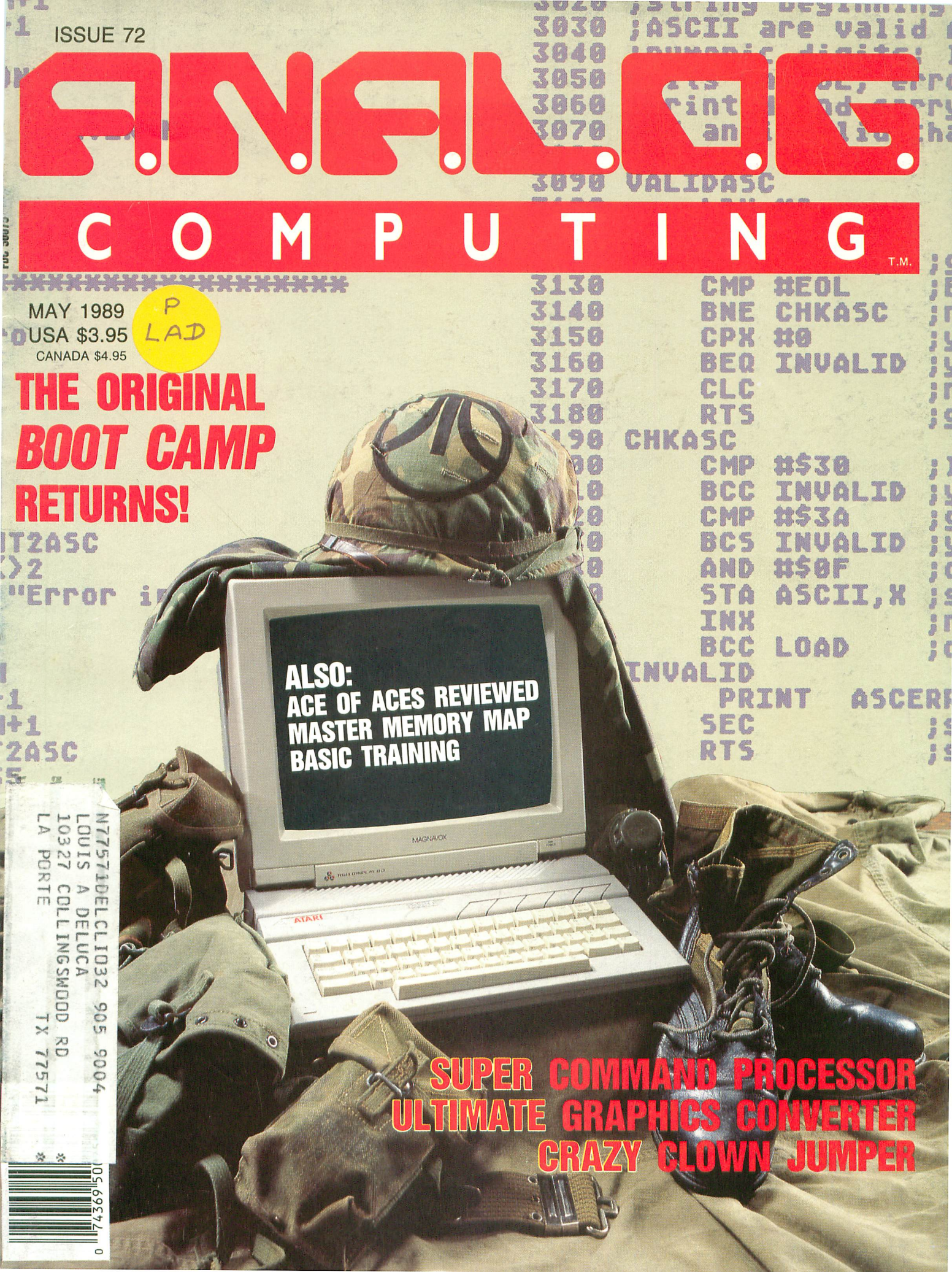
ANALOG COMPUTING

T.M.

MAY 1989
USA $3.95
CANADA $4.95

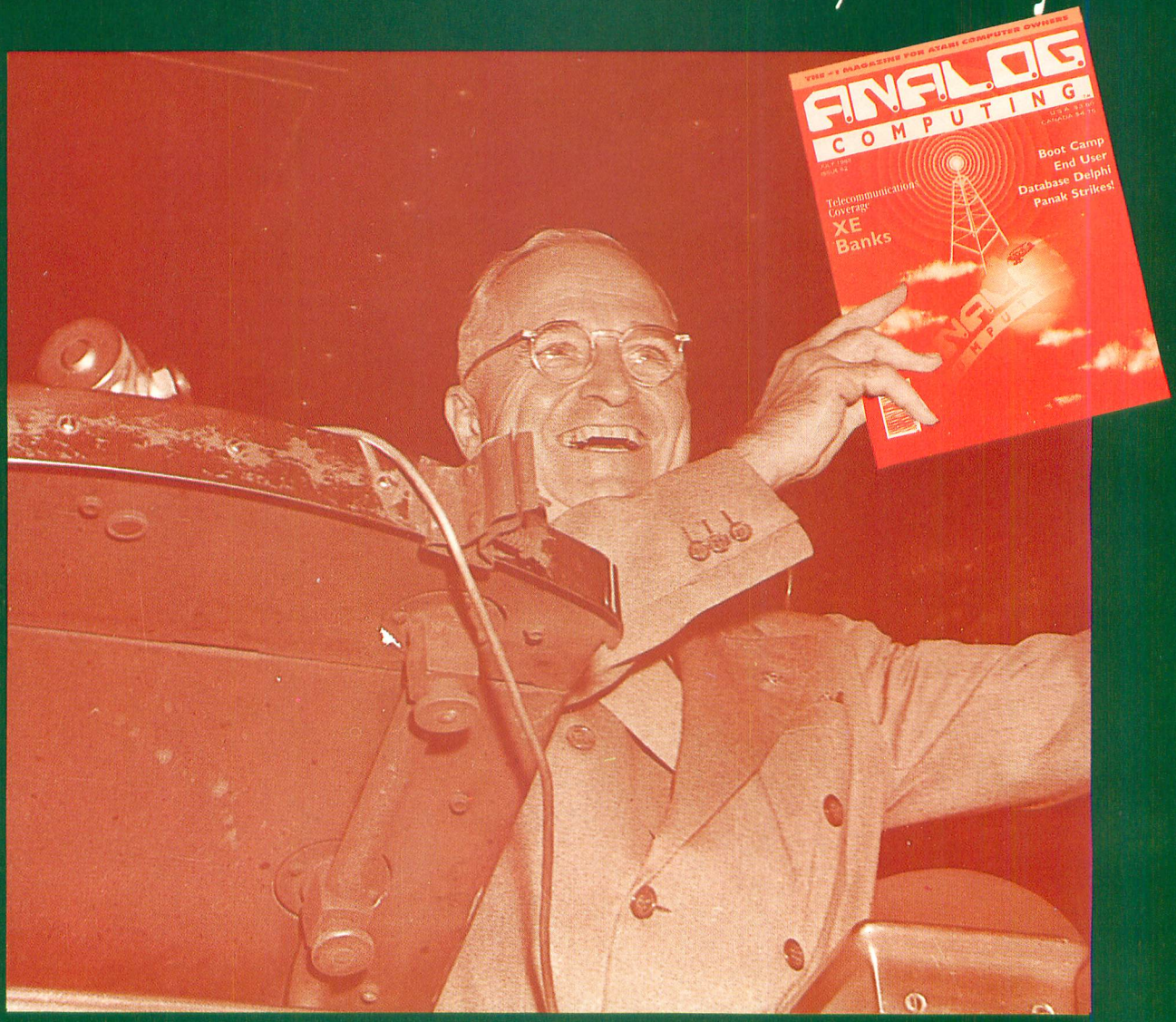
P
LAD

THE ORIGINAL *BOOT CAMP* RETURNS!



SUPER COMMAND PROCESSOR
ULTIMATE GRAPHICS CONVERTER
CRAZY CLOWN JUMPER

Give 'Em A.N.A.L.O.G., Harry!



Two Historic Facts:

1 Dewey did not defeat Truman for the Presidency in 1945. Truman went on to be known for his truthful, forthright style and as one of the nation's most popular Chief Executive Officers.

2 You can save time, and save a lot of money by subscribing to A.N.A.L.O.G. Computing Magazine. Save $19 off the cover price with the convenience of having A.N.A.L.O.G. delivered directly to your door before it even hits the newsstands. To order use the handy postage-paid order card located in the back of this magazine!

1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE!

Editorial

by Clayton Walnum

Over the last couple of months, the ANALOG reader surveys have been flooding in, along with many letters responding to the February '89 editorial. For those of you who may have missed that issue, the subject of the editorial was the future direction of ANALOG. I had stated that there was little new ground to cover and that I wondered whether readers—especially the newer readers—would like to see some of the older topics covered again. I also suggested that it might be a good idea to start reprinting Tom Hudson's old *Boot Camp* columns, because they are the best assembly-language tutorials available anywhere, and they are now all out of print.

When I made these suggestions, I really had no idea that I would be opening such a floodgate of enthusiasm. Judging by your letters (a few of which are printed in this month's *Reader Comment*), virtually *all* of you would like to see the *Boot Camps* reprinted. Also, the vast majority of you not only wouldn't mind seeing the older topics covered again, but almost insist upon it. I would estimate that at least half of the letters we received were from people who had been reading ANALOG for only the last couple of years, and so had missed a lot of important material. These letters informed me that new owners of Atari 8-bit computers are having a tough time finding the information they need to get the most from their machines.

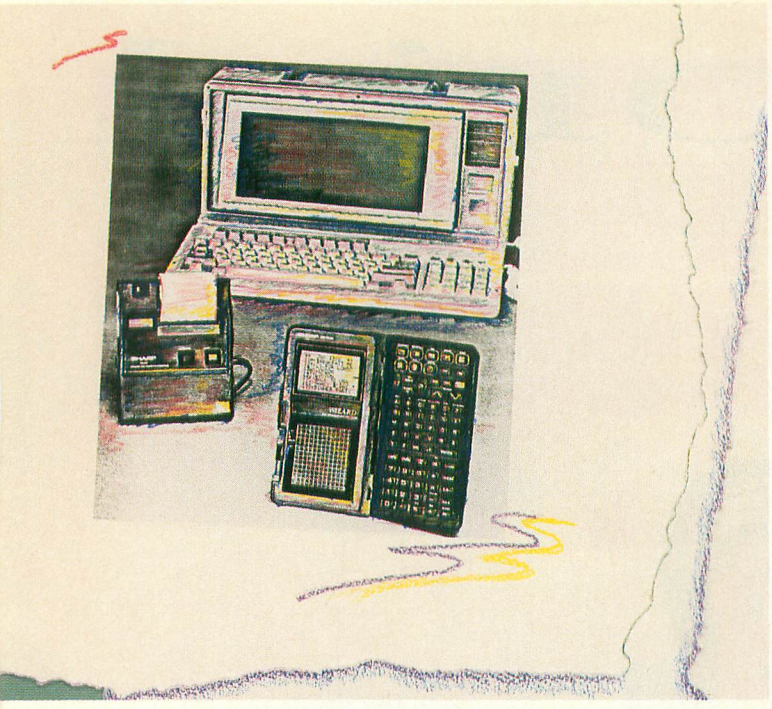
Many people even asked that specific articles, other than the *Boot Camp* series, be reprinted. I immediately envisioned a new department called *ANALOG Classics* where now out-of-print programs—the best from the past—could be reprinted for our many readers who may have missed them. If you would like to see this new department in ANALOG, write to me at the Manchester, Connecticut, address and let me know.

At any rate, by popular demand, this month we begin to reprint the classic *Boot Camp* series, starting with Column 1 published way back in Issue 13 (the cover date was September 1983)! This month also brings the debut of a new column with an old title: *BASIC Training* will tell you everything you want to know about programming your Atari in BASIC, starting with the simple essentials and advancing to the more complex topics like player/missile graphics, animation and the use of VBIs and DLIs.

Those of you who have been with us from the beginning and have no need for the novice material still have much to look forward to. We will continue to pack each issue with many exciting new programs and articles, each designed to make your Atari computing experience as rewarding as possible.

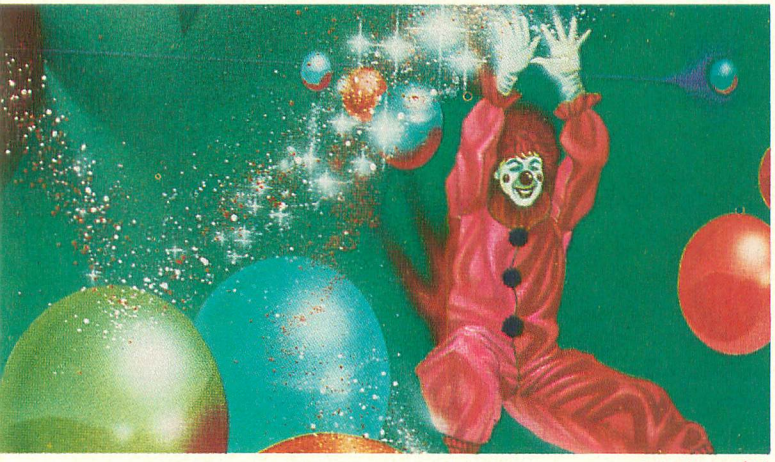
We'd like to thank all who responded with suggestions and ideas. Each letter was carefully read and considered. It was extremely gratifying to see that the Atari 8-bit community is still very much alive—and more importantly, is still willing to make an effort to see that their machines remain viable alternatives for the home-computing enthusiast.

Cont



on page 8

on page 68



on page 74

FEATURES

8

What's New in Consumer Electronics

The Winter '89 CES didn't offer much for Atari 8-bit owners, but there were many items of interest for those who want to stay on the cutting edge of technology

by Arthur Leyenberger

36

Super Command Processor

Here's a memory-resident DOS that allows you to add your own commands—and all that flexibility in less than 2,000 bytes!

by Bryan Schappel

48

The Ultimate Graphics File Converter

Share your graphics between Newsroom, Print Shop *and* MicroPainter *with this handy conversion program.*

by Lee S. Brilliant, M.D.

68

Master Memory Map, Part X

The concluding installment of ANALOG's official Atari 8-bit memory map.

by Robin Sherer

74

Crazy Clown Jumper

Can you make it in the circus? Here's your chance to try—without nets. A 100% machine-language arcade game.

by Brad Timmins

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. U.S. subscription: $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscription: Add $10 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 **ANALOG Computing** Magazine, P.O. Box 16927, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

e n t s

REVIEWS

67 **Ace of Aces**
(Atari Corp.)
reviewed by Matthew J.W. Ratcliff

COLUMNS

16 **Game Design Workshop**
by Craig Patchett

27 **Boot Camp**
by Tom Hudson

32 **Database DELPHI**
by Michael A. Banks

58 **BASIC Training**
by Clayton Walnum

64 **The End User**
by Arthur Leyenberger

DEPARTMENTS

3 **Editorial**
by Clayton Walnum

6 **Reader Comment**

7 **8-Bit News**

62 **M/L Editor**
by Clayton Walnum

80 **BASIC Editor II**
by Clayton Walnum

ANALOG COMPUTING STAFF

Publisher
LEE H. PAPPAS

Executive Editor
CLAYTON WALNUM

Art Director
KRISTEL PECKHAM

Associate Editor
ANDY EDDY

Managing Editor
DEAN BRIERLY

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
CHARLES F. JOHNSON

Contributing Editors
MICHAEL BANKS; FRANK COHEN;
CRAIG PATCHETT;
MATTHEW J. W. RATCLIFF;
ROBIN SHERER

Cover Photography
LADI VON JANSKY

Illustrations
JOHN BERADO
ALAN HUNTER

Copy Chief
KATRINA VEIT

Copy Editors
ANNE DENBOK
RANDOLPH HEARD
PAT ROMERO
KIM TURNER
SARAH WEINBERG

Editorial Assistant
NORMA EDWARDS

Chief Typographer
KLARISSA CURTIS

Typographers
DAVID BUCHANAN
B. MIRO JR.
JUDY VILLANUEVA

Contributors
LEE S. BRILLIANT, M.D.
TOM HUDSON
BRYAN SCHAPPEL
BRAD TIMMONS

Vice President, Production
DONNA HAHNER

Advertising Production Director
JANICE ROSENBLUM

National Advertising Director
JAY EISENBERG
(213) 467-2266
(For regional numbers, see right)

Corporate Ad Director
PAULA THORNTON

Subscriptions Director
IRENE GRADSTEIN

Vice President, Sales
JAMES GUSTAFSON

MAY 1989
ISSUE 72

Where to Write

All submissions should be sent to: **ANALOG Computing,** P.O. Box 1413-M.O., Manchester, CT 06040-1413. All other editorial material (letters, press release, etc.) should be sent to: *Editor,* **ANALOG Computing,** 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ANALOG Computing,** P.O. Box 16927, North Hollywood, CA 91615, or call (818) 760-8983.

Correspondence concerning a regular column should be sent to our editorial address, with the name of the column included in the address.

We cannot reply to all letters in these pages, so if you would like an answer, please enclose a self-addressed, stamped envelope.

An incorrectly addressed letter can be delayed as long as two weeks before reaching the proper destination.

Advertising Sales

Address all advertising materials to:
Paula Thornton — Advertising Production
ANALOG Computing
9171 Wilshire Blvd., Suite 300
Beverly Hills, CA 90210.

Permissions

Subscriptions

ANALOG Computing, P.O. Box 16927, North Hollywood, CA 91615; (818) 760-8983. Payable in U.S. funds only. U.S.: $28-one year, $54-two years, $76-three years. Foreign: Add $10 per year. 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.

For further information, write to **ANALOG Computing,** P.O. Box 1413-MO, Manchester, CT 06040-1413.

JE Publishers Representative
6855 Santa Monica Blvd., Suite 200
Los Angeles, CA 90038

Los Angeles	**— (213) 467-2266**
San Francisco	**— (415) 864-3252**
Chicago	**— (312) 445-2489**
Denver	**— (303) 595-4331**
New York City	**— (212) 724-7767**

READER COMMENT

The Boot Camp Question

I am writing in response to Clayton Walnum's editorial in your February 1989 issue. I have owned my computer, a 130XE, since December of 1985. The first issue of ANALOG I ever bought was Issue 43 dated June '86. I subscribed shortly after and still use one of the programs in that issue weekly, namely "Home Shopper" by Matthew Ratcliff.

Up to that time, I relied on commercial software and was trying to learn BASIC. I picked up hints and got great help from studying your type-in programs. At that time, *Boot Camp* was just something I skipped over.

As I progressed, I started to master some of the things I could do with my machine. I discovered *Mapping the Atari* through numerous references to it and dug deeper into my computer. Still, there was little help for the real beginner. Finally, I wrote some programs using DLIs and only recently got a Player/Missile program running.

As I progressed, it was obvious I needed to learn machine-language too. I bought the Atari Assembler/Editor and *Machine Language for Beginners*, but again I was on my own. The book did not describe how to access the IOCB blocks on the Atari or explain the use of all the assembly instructions available. I dug out my old ANALOG issues and, by studying the machine-language listings, was able to get screen, disk and printer access running. But I am still nowhere near understanding all I can do or how I can do it with assembly language. I finally purchased a MAC/65 cartridge just so I did not have to convert your programs to make them run with my Assembler/Editor.

So here I am. Forty-one years old and feeling like I'm in high school again trying to learn algebra, only I don't have a teacher to run to when I get stuck. Would I like to see *Boot Camp* reprinted? You bet I would! After all, I'm missing 30 issues worth of information! Would I like a review of advanced BASIC programming techniques too? Yes, Sir!

I vote for reprints of *Boot Camp*, and while you're at it, reprints or new articles on BASIC for your new readers.

Finally, I'd like to thank ANALOG for hanging in there through some tough times and providing the best and most useful programs for my Atari 8-bit.—David M. Schoch
Scotia, NY

I just received the February '89 issue in the mail. It's a real piece of art. The use of color in the titles and graphics of each of the articles really makes your publication stand out from the other magazines. After reading the editorial regarding the future direction of ANALOG Computing, I realized that I had better write and express my opinions on what I'd like to see in print.

Although I don't own an Assembler/Editor, I would like to see you reprint all of the *Boot Camp* tutorials. That would inspire me to buy an assembler and learn assembly language. I have been faithfully following your *Game Design Workshop* and have learned a lot about BASIC programming from those articles. More BASIC tutorials would be appreciated. —Kevin B. Dickinson
East Meadow, NY

I would like to see the early *Boot Camp* articles reprinted. I started my subscription to your magazine about a year ago and really enjoy it. But I have just become interested in assembly language and would really like to see the articles explaining assembly language that I missed. —Shane Graber
Stryker, OH

I would be very interested in seeing reprints of your magazine's fantastic *Boot Camp* columns. I have only been a subscriber since February 1987, and have missed many of them.

As an idea for new topics to be covered, have you considered writing technical tutorials on programming the various configurable disk drives (XF551, Happy 1050, U.S. Doubler 1050, Indus GT). I would be very interested to learn how to do this. I would also be interested in how the Ultra-speed I/O routines work for these devices. I would like to write a high-speed, self-booting sector copier for the XF551, and I need to know the formatting and sector-skew commands.

I always look forward to receiving the next issue of ANALOG, and I hope my idea has helped you. —Robert Beauchea
Edison, NJ

Wow! When we suggested the possibility of reprinting the Boot Camp *tutorials we never imagined that the response would be so overwhelmingly favorable. The number of people who would like to see the series reprinted outnumber those who don't by about 20 to one. Clearly, there's only one thing we can do: reprint* Boot Camp*!*

As for Mr. Beauchea's questions about handling the various disk drives and Ultra-speed I/O techniques, if there's someone out there who'd like to put the article together, we would certainly like to see it.

Thanks to all of you who have responded to the February editorial and to those who have filled out the survey that appeared in that issue. It's nice to know that 8-bit computer owners are still very interested in using and learning about their machines. Your help will go a long way towards making ANALOG Computing *the type of magazine you want.*

8 BIT NEWS NEWS NEWS

Napoleon Wins Award

Datasoft's strategy war game, *Napoleon In Russia—Borodino 1812*, has received the "Fire & Movement and Charles S. Roberts Award" for the best pre-20th century war game. The award was presented to the game's programmer, Steve Krenek of Krentek Software and is given "to honor significant achievements in the field of war-gaming."

Napoleon In Russia recreates the famous battle outside of Moscow and includes such features as control of artillery, cavalry and infantry; detailed scrolling maps; and the handling of such battle variables as morale, fatigue and speed of movement. The program is published by IntelliCreations and is priced at $24.95.

IntelliCreations, Inc
19808 Nordhoff Place
Chatsworth, CA 91311
(818) 885-6000

CIRCLE #196 ON READER SERVICE CARD.

Color Diskettes

If you've been wondering how to separate your utility programs from your game programs from your business programs, Memorex may have the answer for you. This large supplier of floppy-disk products has announced the availability of color disks. Designed to help you spot the disks you want in a large file, the disks come in a 10-pack including two each of blue, green, yellow, orange and red disks.

Also now available from Memorex is a 50-pack (not color) of DS/DD 5.25-inch disks for those who need to buy their disks in large quantities.

Memorex
2400 Condensa Street
Santa Clara, CA 95051-0996
(408) 957-1000

CIRCLE #197 ON READER SERVICE CARD.

Power Supply Plans

A battery backed-up power supply can prevent you from losing important data should a power outage occur during a computing session. Most of these units are fairly expensive, but now 8-bit Atari users have a low-cost alternative.

Technitron has put together complete plans for building a battery backed-up power supply for your 8-bit Atari computer, using readily available Radio Shack parts. The package includes complete plans, schematics and parts list and is priced at $4. (Note: This project is recommended only for those with electronics experience.)

Technitron
P.O. Box 1033
Wilkes-Barre, PA 18702

CIRCLE #198 ON READER SERVICE CARD.

Multiple Disk Formatter

Most computer owners are well acquainted with the chore of formatting a new box of disks. Atari DOS, with its one-at-a-time approach, just isn't suitably designed for this chore. But don't despair! Help is on the way.

Just released from Helpways is a handy utility program designed to make the formatting of multiple disks a less painful chore. *AutoPrep* will format a quantity of disks in either single or dual density, as well as write your favorite DOS to each of them. The program is priced at $17.95.

Helpways
P.O. Box H
Rochester, NY 14623
(716) 334-3928

CIRCLE #199 ON READER SERVICE CARD.

Atari Show!

The Michigan Atari Computer Enthusiasts (MACE) has announced The Michigan Atari Computer Expo to be held on May 6 and 7, 1989 at the Detroit Metro Airport Hilton, located in Romulus, Michigan. MACE, one of the nation's oldest and largest Atari Users' Groups, was the first club to sponsor an Atari-only show with the AtariFest of 1984.

"We plan on filling over 40 booths with developers, retailers and dealers—both large and small," said Pattie Rayl, MACE Coordinator. "This show is planned with the Atari user (and users' groups) in mind. Whether you have an 8-bit or an ST, you'll find lots to interest you."

The last show in Detroit area, the MAGIC Show, was praised by exhibitors and attendees alike, and MACE plans to continue the tradition.

For information on MACE, the Michigan Atari Computer Expo, booth prices, admission rates and discount airfare, call Pattie Rayl at (313) 973-8825 or write for an exhibitor or Users' Group package at 3487 Braeburn Circle, Ann Arbor, MI 48108.

CIRCLE #200 ON READER SERVICE CARD.

What's New in Consumer Electronics

by Arthur Leyenberger

Ever since I can remember, I've loved "neat things." To a young boy 30 years ago, a neat thing was a two-foot-tall spaceship by Ideal that had blinking lights, sound effects and a motorized block and tackle/arm for loading two-inch payloads. A neat thing was also an erector set that combined creative play with practical lessons about mechanical engineering.

Still another neat thing was slot-car racing. Although it was hot for only a couple of years, it provided hours and hours of enjoyment while honing one's reaction times (racing), patience (model building) and mechanical ability (working with motors, gears, etc.). In my case, it also gave me an outlet for my entrepreneurial skills since I would "soup-up" other kids' cars for a small price.

I guess some things never change. I still love neat things, but the category has been expanded to include computers, gadgets, electronics, audio, video and on and on. If I had to sum it up, I'd have to admit that I am in love with technology. It's safe to assume that the readers of ANALOG also share this feeling to one degree or another.

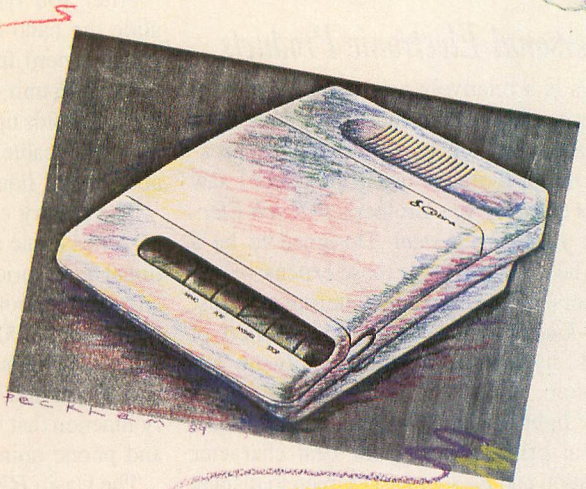
Over the last ten years, the business community has certainly embraced computers even from the early days of the Apple II and *Visicalc*. Not so the home computer market. Although computers are increasingly finding their way into homes each year, people who own computers at home are a minority. Further, people who actively use their home computers are an even smaller minority.

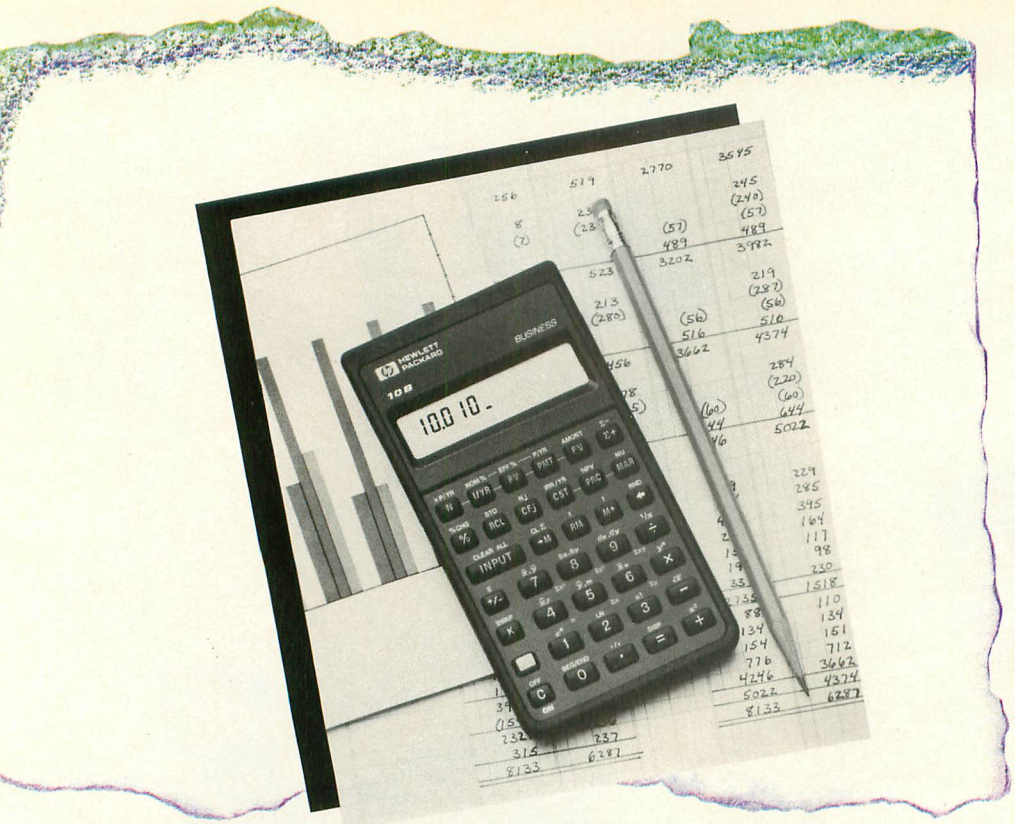
Nonetheless, the very fact that you own an Atari home computer, are an active user and are reading this magazine right now, means that you understand, appreciate and enjoy technology and what it has brought forth in the way of consumer electronics. That is why you can appreciate the semi-annual Consumer Electronics Show.

Consumer electronics is one of the largest industries in the United States, and the Winter Consumer Electronics Show (CES) in Las Vegas is *the* place to see the new products and technologies. Over 770,000 square feet (that's about 17 football fields) of exhibit space, spread across a half dozen locations, is available for the some 90,000 attendees to view. The 1,400 or so exhibitors always hope that


CASIO FX-7


SONY
Maxipak
VFD-50





their products will get the major attention during the four-day show.

Every year CES attendees look for that one product or technology that will have consumers lining up to buy. After all, CES is a trade show where retailers, buyers and distributors want to know what products will make them the most money. In past years such products as video recorders, home computers, video games, pocket TVs, Compact Discs and many others have been introduced and been tremendously successful with consumers. This year was no different as several new product categories were seen.

Personal Electronic Products

This is a relatively new category of consumer products that includes everything from language translators to pocket spelling checkers. One of the most useful products I saw in this category was Brother's *P-Touch Electronic Lettering System*. Designed to be a hand-held alternative to those expensive and cumbersome professional lettering systems, the *P-Touch* produces lettering in a variety of colors, styles and sizes for virtually any application.

The lightweight self-contained unit uses a thermal printing device for best character resolution and uses an LCD screen to display the label. A large rotating dial is used to enter each letter or character for the label, and once completed the 45-character-maximum label can be edited before it is printed. The existing label is not lost when the unit is turned off, and interchangeable ribbon/tape cartridges are used for different colored labels.

The Brother *P-Touch* is an easy-to-use labeling device that produces quality labels in five sizes and four styles. The labels are produced on self-adhesive strips that are easy to apply. It lists for $150, but I bought mine at a discount house for $99. Tape cartridges sell for about $10 apiece.

Another electronic lettering system, called the *HW-1 Digital Writer*, was shown by Casio, Inc. This product is somewhat different than the *P-Touch* in that it will print on any paper rather than just on strips of self-adhesive labels.

After you type copy into the *HW-1*, you slide the hand-held printer wand (you hold it like a pen) from left to right on paper to print. The unit beeps when printing is complete. It holds up to 2,066 characters and runs on rechargeable batteries—a full charge takes about eight hours.

The battery charger and black ink ribbon cassettes come as standard accessories. Optional accessories include blue, red, gold and silver ink ribbons, and a RAM card that will hold up to 5,300 characters. The *HW-1* offers a choice of eight different type styles and special effects and has a built-in phone directory function that lets you store up to 272 names and phone numbers.

The Casio *HW-1* will print on any flat surface and also serves as a full-featured 10-digit calculator. It retails for $300.

Sharp was showing the *Wizard*, a pocket electronic organizer. This $300 device contains all of the essential business tools such as appointment calendar, phone directory, notepad, calculator and world clock in one handy gadget. The sleek 4-inch by 6-inch unit weighs only eight ounces and has a built-in PC interface to bi-directional load schedules, phone listings and documents from a personal computer.

The *Wizard* has been available for about six months and has been a hot seller. Using a vertical format and an 8-line by 16-character LCD display, the *Wizard* is completely menu driven so that it is easy to use. It contains a 32K memory, separate alpha and numeric keys for data entry, variable screen-character size and a 200-year calendar.

One of the unique features of the *Wizard* is its ability to use plug-in IC software cards that, when slid into position, also double as a touch-sensitive panel for selecting the card's functions. Currently, three cards are available: a dictionary/thesaurus, a time-management system and an eight-language translator. Each of the cards is priced under $130. The *Wizard* can also be connected to peripherals such as printers, a cassette recorder or even another Wizard.

Another personal electronic product is the *Hexaglot* language translator from Polyglot. This smartly styled product contains translations for six different languages: English, Spanish, German, Italian, French and Portuguese. Simple to use, all you do is type in the word you want translated, in any language, and press the key corresponding to the language you want the word translated to.

The *Hexaglot* also functions as several other devices as well. It contains a four-function calculator, a currency-exchange function and a multi-lingual dictionary for spell checking. In addition to translating individual words from one language into any of the five other languages, the *Hexaglot* translates several words or a phrase at once, as long as the total number of characters is less than 128.

The *Hexaglot* weighs a mere 2.5 ounces, measures approximately 5.5 by 3 by 0.75 inches and has a two-line by 16-character LCD screen. The unit has a 60-key membrane keyboard, operates on batteries and has an automatic power-off feature. The *Hexaglot* retails for $186.

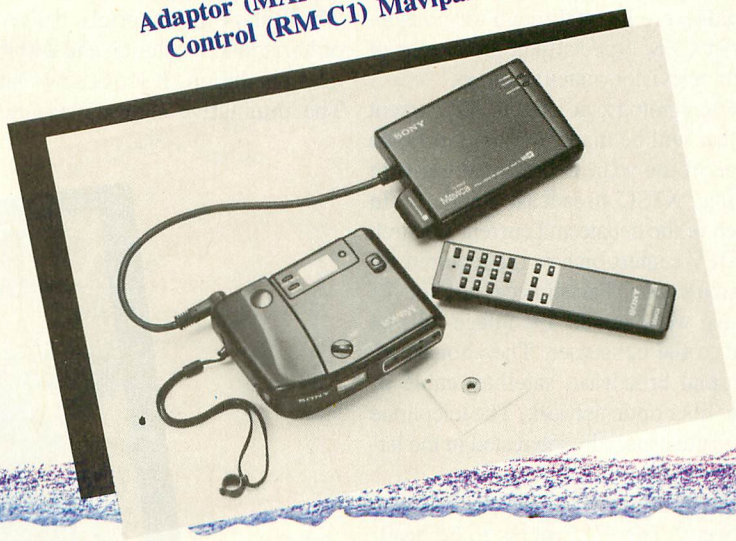
Franklin Computer, makers of the *Language Master* series of hand-held spelling checkers, introduced the first speaking dictionary. The *Language Master 4000* features an electronic dictionary, thesaurus and phonetic spelling corrector that pronounces more than 83,000 words. Not only does it provide concise dictionary definitions but it also pronounces them correctly.

Useful for people who have difficulty pronouncing words, the *LM-4000* will also be useful for people learning English as a second language. Simply type in a word the way it sounds and within seconds the LM-4000 will display its definition, parts of speech and hyphenation points on a LCD screen. Press the SAY key and the unit's electronic voice pronounces the word.

In addition to the above, the *LM-4000* has a built-in vocabulary-building tutorial that features a list of 3,500 words frequently found on such tests as the SAT, GRE and GMAT. Students can view randomly selected words and at the touch of a key display their meanings. The new speaking dictionary includes a built-in speaker, volume control and headphone jack. Retail price is set at $400, and the *LM-4000* will be available this summer.

Cobra Electronics was showing a unique answering machine. The *Cobra Timekeeper* is the first answering machine that offers a digital time stamp on each incoming message so that you know when and at what time the message was received. The $120 *Cobra Timekeeper* is a beeperless machine with multifunction remote control. The system is based on a single standard-sized cassette.

As each message is received, the date and time are recorded. When you play your messages back, a synthesized voice announces the time stamp after each message. Sounds like a useful product for someone who travels a lot and needs to know when their messages came in.



Mavica Still Camera, Playback Adaptor (MAP-T1) Optional Remote Control (RM-C1) Mavipak Disk.

Audio

Here's a twist—or rather a marriage of new and old technologies. Finial Technology, Inc. has introduced an optical turntable that can play regular LP vinyl disks using a laser. Really, no kidding. Called the *LT-1 Laser Turntable*, this product really does play LPs without anything touching the grooves and causing further wear on the record.

Not only is wear on the record eliminated but the audio quality is improved as well. With no tone arm and cartridge, there is no drag to cause any speed irregularities that can be heard as pitch changes. Further, without a tone arm, no motor noise is transmitted from the platter, resulting in no rumble. The whole thing makes more sense than it would first seem.

The *LT-1* consists of a lightweight turntable system, tracking mechanism and computer-controlled laser optical assembly. Unlike conventional turntables, the platter does not need to be heavy in order to overcome the resistance of the stylus dragging across the record surface. Speed accuracy is maintained by a closed-loop computer circuit that is updated 60 times per second. There is a variable speed control that the user can set between 30 and 50 rpm.

The tracking mechanism is also microcomputer controlled and ensures that even moderately warped disks can be properly played. Furthermore, the system memorizes boundaries and times each cut. It also maintains a constant gap between the optics and the record surface with an accuracy equal to 1/30th the thickness of a human hair.

The record grooves are "read" by the laser assembly, and since there is no physical contact, the grooves are scanned more than 40 times the amount of a conventional stylus. As a result, more musical information is extracted from the grooves with an improvement in audio quality. In addition, the *LT-1* has a noise blanking circuit to reduce ticks and pops and an in-drawer cleaning system that keeps the optical assembly clean.

The front loading *Laser Turntable* is fully programmable, and a display shows elapsed and remaining time for all cuts on the side. The *LT-1* is expected to be available by summer 1989.

Other audio products included a rash of portable cassette tape players, some new portable CD players and all shapes, sizes and colors of boom boxes, many of which appeared to be created at the art deco school of design. Sony was showing several *DiscJockey*-brand carousel CD-changer models for the home as well as the car. Most of the home units feature Sony's five-disc revolving carousel tray which lets you mix both 5-inch and 3-inch CDs. A pair of models feature a unique 10-disc magazine that lets you swap the 10-disc magazine between the home and car player.

Quite a few companies were showing "combi" players, units capable of playing CD, CD-3, CD video and laser Disks. Since CD video discs cannot be played on a standard CD player, a special player is required (the "combi" player) to play them. As you may know, CD video or CDV is an optical playback medium that looks similar to a compact disc except that the 5-inch CD video is gold colored to distinguish it from a CD. It offers 20 minutes of digital music and five minutes of regular, full-motion video on one side of a disc.

I guess the predictions that CDV would be a 1989 phenomenon is coming true. Whether it will be fully embraced by the electronics consumer will have to wait to be seen.

Video

From the halls of congress to retailers showrooms, the future of television is being hotly debated by industry experts, government officials, consumers and the press. The words heard everywhere are HDTV, which stands for high-definition television. Those same words were heard through the convention hall at CES, especially at the booths of the major television manufacturers.

HDTV technology, at least in the current version that will be made available in Japan by the end of the year, is incompatible with the existing NTSC broadcast standard. In fact, much of the debate and current research about HDTV centers on how the required second augmentation channel necessary for the razor-sharp wide screen TV image will be delivered to the consumer. The choices include normal broadcast, satellite, cable or some new fiber optic network. The telephone companies are especially interested in the latter possibility for obvious reasons.

Quality is one area where compromise may have to occur. The FCC seems to be heading in the direction of wanting any HDTV format in the U.S. to be compatible with the existing NTSC broadcast standard. However, doing this may jeopardize the quality of the image. Viewing true HDTV, like the MUSE system about to be introduced in Japan, is an amazing experience. The picture quality is crystal clear and almost appears three-dimensional.

In addition to wanting NTSC compatibility, the FCC and Congress want the high-definition system to be American developed and sets manufactured in the U.S. A number of companies are proposing systems that are compatible, but the maximum resolution of these images are not as good as they could be. Regardless of the final system chosen, industry experts predict that HDTV will arrive in the U.S. in the early 1990s.

Equally important as the home entertainment aspect of HDTV is the potential use of it for a variety of other uses. From theatrical distribution of feature films to high-speed, high-quality color printing to video teleconferencing, HDTV promises to be a technology that will serve many purposes. Professional users will not settle for less than the highest quality picture possible. For example, HDTV could be used for telesurgery, but I know that I would not want an operation where the doctor had to rely on anything less than the sharpest quality image.

Anyone who has seen an HDTV display can't help but be excited about eventually having this technology in their home. Once the political and economic issues are sorted out, I too hope we end up with the best quality system possible.

Another interesting, although not as practical, new product was shown by Hitachi. It is a 5-inch color LCD TV for cars. The tiny TV displays 115,000 pixels, delivers 480 lines of horizontal resolution and 240 lines of vertical resolution. It looks surprisingly good. The diminutive screen measures only 6.5 inches wide by 5.7 inches high by 1.5 inches deep.

The TV's tuner uses a dual antenna system with an arrangement of four separate antenna elements that assures quality reception, even in a moving vehicle. An advanced system of antenna switching lets the TV receive broadcasts from the antenna with the strongest signal up to 60 times per second.

Hitachi's "auto video" was designed primarily with the future of on-board navigation systems in mind, but it also functions as a normal TV. The small set includes A/V inputs for connecting external video sources such as VCRs or video games. The LCD color TV also has an infrared remote control and an on-screen display.

Although the color LCD TV was designed for the Japanese market and Hitachi has no current plans for U.S. introduction, it sounds like a great product to mount in the back seat to keep the little tykes occupied on those long trips.

Toshiba announced and showed a new 4-inch color LCD television called the *LCD-048*. The high-resolution screen incorporates an active filter matrix with a thin-film transistor color filter that provides a bright, clear image. Resolution is 220 vertical by 480 horizontal lines.

Other features of the *LCD-048* include audio/video input and output terminals, switchable internal backlight for improved viewing under low-light conditions, earphone jack, telescoping antenna and a provision for an external antenna, as well as electronic tuning and on-screen channel and volume settings. Pricing has not yet been established.

Still Video

Not "still video" (as in more information on video products) but still video (as in using video technology for making still images or snapshots). Still-video cameras have been in development for several years and prototypes have been shown at CES in prior years. However, this was the first time that companies like Canon, Sony and Olympus were showing actual production models.

All still-video cameras use a 2-inch floppy disk that stores 50 frames or images. Each image is stored as digital data and can be retrieved instantly. Some companies like Canon use the camera itself to play back the images through a television whereas Sony and Olympus require separate players to view the image.

Based on the Hi-Band video standard, all of these still-video cameras have the same quality pictures and basic features. All are capable of capturing images with up to 300 lines of resolution. The magnetic disk used to store the electronic images is erasable and reusable and is compatible between all makes of still-video cameras. The major differences between different cameras are the particular features of each. Some have a built-in telephoto adapter while others have a macro adapter for close-up shots.

Most of the cameras have a built-in flash, can take individual frames or multiple frames per second and are about the same size. Automatic exposure and white balance are also standard features. Other features include self timers, rechargeable batteries and point-and-shoot ease of use.

The Sony *Mavica* (an acronym for Magnetic Video Camera) MVC-C1 weighs just over a pound and measures 5¾ by 2¼ by 4¼ inches and is designed to fit in the palm of the hand. The *MVC-C1* includes a 15mm f/2.8 fixed-focus lens that provides sharp images of subjects at a distance of 1.5m to infinity. The built-in flash is automatically activated in low-light conditions, and the camera is capable of shutter speeds from 1/60th to 1/50th of a second.

In addition to single picture shooting, the Sony still-video camera offers continuous high-speed image recording at either four or nine frames per second. A "blank search" function automatically advances the disk to a blank frame so accidentally erasing an image is avoided (to erase a picture, the separate playback adapter must be used).

The playback adapter, the *MAP-T1* is powered by either batteries or AC and can use an optional wireless remote control for convenience. The *MAP-T1* also fully charges the camera in one hour. The camera will retail for $650, and the playback adapter will sell for $250. Both will be available by the time you read this.

The Canon still-video camera is called the *Zap Shot*. In addition to the standard features like built-in flash, auto-exposure, etc., it features a built-in macro mode for extreme close-ups at 12 inches. Recording and playback functions are all contained in the 15-ounce body, and it uses a 11mm f/2.8 fixed focus lens.

The *Zap Shot* can take pictures either individually or at three images per second and contains a built-in self-timer. When taking pictures, you can let the camera automatically find the next highest blank frame, or you can manually set it to any frame you want. The *Zap Shot* will sell for under $1,000 and be available by the time you read this.

Olympus was the third company showing a still-video camera, but it was still a prototype. Like Canon, Olympus uses a separate playback unit for viewing the video images. Also, the company announced that they will have a stand-alone still-video recorder/player.

The Olympus *V-100* is almost twice the weight of the other still-video cameras shown at CES, but the added weight means added features. It has the fastest continuous shooting mode (15 frames per second), and the camera contains a built-in 9-27mm 3x power zoom lens (equivalent to a 35mm camera 49-147mm lens). The *V-100* also offers automatic date recording (year/month/day) on the video image and automatic backlight exposure compensation.

The *V-200* AC-only playback unit also

offers unique features. Multi-screen playback allows 4, 9, 16 or 25 pictures to be simultaneously displayed on the screen at once. Digital effects like negative reverse and digital solarization, adjustable intervals between individual pictures (much like a slide projector) and high-speed continuous viewing of 15 images per second are possible. Other features include wireless remote-control operation, single-track and all-track erase capability and on-screen display of date stamp and track number.

The *V-300* still-video recorder/player offers features aimed primarily at the professional user. The unit is compatible with many video sources such as external video cameras, VCRs, laser-disk players, etc., can record either continuously or individual images and has S-VHS (Y/C) type output connections. In addition, interval recording is possible from one second to 99 minutes. Interval playback is also possible with a range of 1-99 seconds.

The *V-300* also offers high-speed continuous playback at 2, 5 or 10 frames per second. Further, a programmable playback function is provided and random access to any frame is possible. And, as if this were not enough, a wireless remote control is provided as standard equipment.

Olympus clearly has the professional or high-end user in mind with their line-up of still-video products. At press time, pricing and availability had not been established. However, Olympus representatives say that the products should be out sometime around the middle of the year.

Home Automation

Home automation is a relatively new product category. There are existing products that fall into this category such as smart security systems, automatic sprinkler systems and programmable AC outlets and switches. One problem with these products is that each company has designed their own techniques for device control. As a result, products from one company will not work with products from another company.

A small exhibit at CES was displaying a new home-automation standard designed to make the automated home a reality. This technology is intended to allow home entertainment products, major appliances, security systems and heating and air conditioning equipment to work together by means of a standard bus.

Called CEBus, it is an evolving Electronic Industries Association home-automation standard. Using a CEBus system, you can choose which electrically powered products in your home you want to operate automatically—products as different as your TV, electric corn popper, garage-door opener or computer.

Another area where CEBus products and home automation in general will be useful is for people with disabilities. Disabled individuals will be able to better control their environment with safety, security and convenience. For example, a visitor at the door can first be identified and then let in without the person having to get up. Since the CEBus standard is flexible, new products can be designed especially for the elderly, bedridden or other people with special needs.

Toshiba: LCS-048 Color TV.

More Calculators

Hewlett-Packard makes the finest calculators money can buy. They have been making calculator models for business, science, engineering and statistics since the early 1970s. Although their products don't compete with the kind that you buy at the supermarket, HP has broadened their line to now include low-end models that start at about $50 retail.

Two new models were introduced at CES this year: the *HP-10B* business calculator and the *HP-20S* scientific model. Celebrating their 50th year in business, HP says they are able to offer low-priced calculators because their research and development and manufacturing departments have been working together at an earlier product-design stage which reduces overall costs.

Aimed at business professionals and students, the *HP-10B* is HP's first business calculator with a list price of under $50. It offers the same quality and reliability of HP's other more sophisticated models and contains the essential business and math functions for solving problems such as loan calculations and cash-flow analysis.

The *HP-10B* uses an algebraic entry system and features statistical functions, basic math functions and a 15-register memory. Also, it contains an automatic constant function for repetitive calculations, forecasting and linear-regression functions and a one-line by 12-character LCD display.

The *HP-20S* scientific calculator also uses the algebraic entry system and contains 15 math, ten trigonometry and nine statistical functions. In addition, the *HP-20S* features base conversions and arithmetic in decimal, binary, octal and hexadecimal modes. Further, polar-rectangular, hour/minute/second-decimal hour, degree-radian and English-metric conversions can be performed.

The *HP-20S* also features keystroke programming whereby the machine "records" your keystrokes as you solve a problem and then lets you "play them back" as a program. The programming capability also includes conditional testing and branching. The calculator has ten storage registers and a 99-step program memory. The *HP-20S* uses a one-line by 12-character LCD display and sells for $50.

Cellular Telephones

It appears that cellular telephones are really

taking off these days. The technology is not that old but the convenience it offers and the continual improvements made by the manufacturers have spurred the growth. The industry reports that there are now about a million subscribers and another million are expected to hook up in 1989. With those kinds of numbers, it's not surprising that cellular telephones are one of the hottest areas of consumer electronics today.

One of the fastest growing segments of the cellular scene is the hand-held portable cellular telephone. These high-tech communications devices look like Walkie-Talkies and allow telephone communication just about anywhere. In fact, while at CES in Las Vegas, I was having lunch at a local restaurant when a policeman came in for his lunch break. He was carrying a Walkie-Talkie *and* a hand-held portable cellular phone. I noticed during his meal that he was in constant contact with his dispatcher via the cellular phone.

With prices continuing to drop and improvements being made with the hardware, consumers are lining up to buy these phones and start using them on the go. Whether it's a consumer product or a business tool, the market has grown so large that these phones are available from stores like Radio Shack, the Crazy Eddie's type and telephone company-owned retail outlets.

When portables first came out they were priced at about $2,000. Now, most sell for closer to $1,000, and typical users spend around $100 per month on the service. Recently, two companies, Motorola and STS introduced hand-held models for about $700. Although these units are heavy by today's standards (roughly 28 ounces each compared to the under 20-ounce weight of most portables) interest has been very strong, and the companies are trying to met the demand.

NEC America, Inc. continues with its *P9100* series of portable cellular telephones featuring dual telephone-number capability, alphanumeric display and 832-channel capability. Up to 40 names and telephone numbers can be stored, displayed and automatically dialed from memory. The *P9100* also has last-number redial and a call-duration timer.

The NEC *P9100* offers 45 minutes of talk time and up to eight hours of standby, and when not in use, 20 hours of continuous standby. The unit weighs 23 ounces and sells for under $1,000.

The Mitsubishi Electric *DiamondTel 90X* weighs just 18 ounces and is a full-featured model. It offers a full 832-channel operation, dual telephone-number capability, an alphanumeric liquid-crystal display and a 100-number memory. In addition, the *90X* provides one and a half hours of talk time and 13 hours of standby time. A scratch pad memory allows you to enter a number into the memory during a call and then speed dial that number by simply pressing one button after you complete the initial call. The *90X* sells for under $1,800.

Oki's new *Model 700* portable is also lightweight but full-featured. It measures 7.48 by 2.1 by 1.37 inches and features dual telephone-number capability, automatic credit-card dialing, a 100-name phone directory and electronic menu. It sports a 100-minute talk time or 18 hours of standby operation. Pricing has not yet been announced.

Portable cellular phones are expected to surpass 10% of the cellular market in 1989. In addition, by 1990 fully half of the cellular telephone market is expected to be comprised of transportable and portable telephones. The models mentioned above are just a small sampling of what is currently available. The future looks very bright for portable cellular telephones.

Other Products

There were also a number of useful and clever products shown at CES. The *Eyeopener* is a $10 eye shield that mounts on a video camera. When flipped into position, it lets you keep both eyes open but makes it much easier to focus through the finder. You don't need to squint and can concentrate better on what you are doing. The *Eyeopener* attaches easily and quickly to any video camera and is adjustable for either left-handed or right-handed use.

Another interesting product seen was the *Private Eye* from Reflection Technology. This postage-stamp-sized video screen is capable of displaying 80 columns by 22 lines with the clarity of a 12-inch screen as seen from two feet away. Although it sounds wacky, the *Private Eye* works amazingly well. The resolution of the red display (it could also be green) is 720 by 280 pixels, although the company says that higher resolution can be achieved "depending on yield."

Although not yet a commercial product, the applications for this technology seem unlimited. Imagine the *Private Eye* as an alternative to a laptop computer display. You would wear the headband containing the display tethered to the machine. Or imagine its use by a surgeon. As the surgeon works, the display could show the patient's X-rays or vital signs.

Casio usually has some interesting things at every CES, and I wasn't disappointed this time. In the past, Casio was the first to introduce a mini-keyboard synthesizer at a reasonable price: the *CZ-101*. I bought one of these when they first came out. A couple of years later, Casio introduced the first mini-keyboard digital sampling keyboard: the *SK-1*. I bought one of those too.

Casio's latest musical gadget is the *DH-100 Digital Horn*. It is a tad smaller than an alto saxophone and is made out of grey plastic. It sells for $150 retail but can be found for about $100 at discount stores. The *DH-100* comes with built-in sounds of a saxophone, trumpet, oboe, clarinet, flute and synth-reed and uses recorder-type fingering, making learning is easy. *continued on page 56*

Workers test a TV picture tube for color quality.

GAME DESIGN Workshop

by Craig Patchett

Fighting Back

We're now at a point in the game where we could do a number of things, and different people will recommend doing different things. For example, we've ignored the next-level and end-of-game sections of the program so far, and some game designers will argue that these sections should be completed next. We also haven't done anything about the score yet, and we haven't given the invaders the ability to fire at the player. So what's our choice? Well, the point I'm trying to make is that it could be any of these. I personally have chosen to give the invaders some fighting spirit, so that's what we'll do. You should realize, however, that at this point in the game the order of doing things is not as important as it was before.

Firing Back at the Players

Before we start the programming again, let's stop and take a look at the idea behind what we're going to do. To state it simply, we're going to make the invaders fire missiles at the player. No big deal, right? Not in this game, but giving the computer the ability and intelligence to fight back against the player can quickly turn into an extremely complicated task in other games. As a matter of fact, as I mentioned in the column on logic, a good part of a game's logic is usually involved with making the computer seem intelligent.

Unfortunately, this kind of thing takes a lot of programming and, therefore, a lot of time, which is why most BASIC games are either not very intelligent or very slow. BASIC invaders falls into the "not very intelligent" category and is still somewhat on the slow side. So, if you feel the urge to devote a lot of time to designing a game that shows incredible intelligence, I would suggest either doing something where speed isn't important (such as a strategy game of some sort) or learning machine language.

Okay, back to our game. What we'd like to do is have the invaders in the bottom row fire missiles at the player's base. We'll make it easy on the player by only allowing the invaders to fire one missile at a time and having them fire randomly. (You may like to try changing the program later so that the invader nearest to the player's base is the one that fires.) So let's jump in and make some program changes:

```
400 X=USR(ADR(MISCLR$),PA+7
68,255,243):POKE 1721,0:RET
URN
1100 IF PEEK(1721)<>0 THEN
GOSUB 400
```

```
1120 IF STRIG(0)=1 OR PEEK(
1700)<>0 OR PEEK(1720)<>0 O
R EF>1 THEN 1170
1170 IF PEEK(1701)<>0 OR PE
EK(1721)<>0 OR LINE+BOTROW*
2=19 THEN 1250
1180 X=USR(ADR(MI5CLR$),PA+
768,255,243):X=PEEK(20)
1190 IF PEEK(20)=X THEN 1190
1200 POKE 53278,0:FC=INT(RND
(0)*8)*2:FR=BOTROW*48
1210 IF INV$(FR+FC+27,FR+FC+
27)="♥" OR INV$(FR+FC+27,FR+
FC+27)="
" THEN FC=FC-2+16*(
FC=0):GOTO 1210
1220 POKE 1693,FC*8+55+5CROL
L+COAR5E*8:POKE 53253,PEEK(1
693):FV=BOTROW*16+LINE*8+46
1230 POKE 1697,FV:FV=FV+PA+7
68:POKE FV,PEEK(FV)+4:POKE F
V+1,PEEK(FV+1)+4
1240 POKE 1713,4:POKE 1717,1
:POKE 1701,1
5340 POKE 1700,0:POKE 1701,0
:POKE 1720,0:POKE 1721,0
5350 POKE 1704,0:POKE 1705,0
:POKE 1708,129:POKE 1709,1
```

And now yet another incredibly in-depth explanation:

400 This is our collision routine for the invader missile. For now it simply erases the missile and clears the collision flag.

1100 Here we check to see if the invader missile has collided with anything important and go off to the collision routine if it has.

1120 A simple change to this line so that it now goes to line 1170 if we aren't going to fire a player missile.

1170 Now we decide whether or not we're ready to fire an invader missile. If there's already one in the air, if an old collision hasn't been taken care of yet or if the invaders are on the last line before the base (at which point the player would have no time to react to the missile), then we skip ahead.

1180 If we are ready to fire an invader missile (i.e., if the program got past the previous line), then we can clear out Missile 1 and clear the system clock.

1190 We wait here until a jiffy has passed (to make sure that the missile is completely off the screen).

1200 Now we clear the collision registers and randomly choose an invader on the bottom row.

1210 The next step is to make sure that the chosen invader still exists, so we check its place in INV$. If we find a blank or an explosion, then we go one invader to the left (wrapping around to the right side if we're on the leftmost invader already) and try again. We don't leave line 1210 until a live invader has been found. Incidentally, the two funny-looking characters in this line are produced with CTRL-COMMA and inverse CTRL-M.

1220 We now have the position of our attacking invader, so we position the missile horizontally and figure out the vertical position.

1230 Here we position the missile vertically and turn it on within the PMG area. You should notice that instead of turning on the missile by POKEing two 4s into the PMG missile area, we are adding 4 to the values that are already there. Why? Remember that the player missile is in the same group of bytes as the invader missile, and there is a chance that the player missile may be in the same two bytes that we're going to be putting the invader missile into. To make sure that we don't erase the player missile, we add 4 instead of POKEing 4. What about lines 1140 and 1150, though, where we turn on the player missile by simply POKEing? Isn't it possible that we'll accidentally erase the invader missile? It is, but by the time the invader missile gets that low on the screen, we don't need to worry about it anymore. If it bothers you, however, there's no reason why you can't update lines 1140 and 1150 so that the invader missile is left intact.

1240 We tell PMOVE to watch for the invader missile colliding with Playfield 2 (the barriers) or Player 0 (the base), and then we tell PMOVE to start the missile moving.

5340-5350 These lines are part of the PMOVE initialization section, and we've changed them so that they initialize Missile 1 so that all the flags are clear and it's set to move upwards when we turn it on.

Well, that wasn't too difficult, and as a result we now have the invaders firing happily away at us. Of course if you shoot away all but one invader on the bottom row, you'll notice that things tend to slow down quite a bit. Unfortunately, we can't do anything about this unless we were to introduce some more machine language. It's something you might like to try if you know machine language, but I won't do it here. This column has been designed to teach you how to program a game from BASIC, and machine language is only used when it's in a form that can easily be adapted to your own games. Such is not the case now.

Invader-Missile Collisions

The next step is to take care of the invader-missile collision routine at line 400. Let's take care of a small problem first that you may or may not have noticed. We're using Missile 1 for the invader missile, and Missile 1 has the same color as Player 1. Well, we're using Player 1 for the alien saucer, which means that the invader missile has the same color as the alien saucer. As you can see if you run the program with the above changes, this seems to be perfectly acceptable. But try shooting down the alien saucer and see what happens. When the saucer fades out, so does the invader missile, and it doesn't return to normal until another saucer appears (which is when we restore Player 1's color). The solution to this is to restore Player 1's color after the saucer has been destroyed. To do this, just make the following change to line 220:

```
220 POKE 53249,0:POKE 53250
,5:POKE 705,40
```

Now we're ready to take care of collisions, so go ahead and make the following changes:


```
400 X=USR(ADR(MISCLR$),PA+7
68,255,243):POKE 1721,0:IF
PEEK(1717)<>255 THEN 470
410 POKE 1664,255
420 X=USR(ADR(MISCLR$),PA+7
68,255,252):POKE 1700,0
430 FOR X=117 TO 250 STEP 4
:POKE 704,112+INT((250-X)/9
):NEXT X
450 POKE 53248,128:POKE 168
4,128:POKE 704,15
460 POKE 1664,0:RETURN
470 X=PEEK(1693)-47+2*(RND(
0)>0.5)-1:Y=PEEK(1697)-159:
GOTO 260
```

400 Instead of RETURNing at the end of this line, we now check to see whether a collision with the base has occurred. If it hasn't, then the missile must have collided with one of the barriers, so we skip ahead to line 470.

410 The missile has collided with the base, so we temporarily disconnect the base from the joystick (so that the player can't move the base while we're destroying it).

420 We also want to get rid of the player missile if it's in the air, so we clear it and turn it off.

430 Now we fade out the base in the same way that we faded out the saucer.

450-460 We want to give the player a new base, so we position it in the middle of the screen, restore its color, reconnect it to the joystick, and then return to the main part of the program.

470 This line takes care of the missile colliding with a barrier. Since the code to do the actual explosion is already in place starting at line 260, we may as well make use of it, so all we do here is figure out where the explosion is to take place and then skip over to line 260.

Keeping Track of Player Bases

We now have all our explosions in place, but when the player's base gets destroyed, he automatically gets a new one, regardless of how many he's already lost. As you're probably well aware, video games do not generally give you an infinite number of bases, or lives, unless the game is timed. In BASIC invaders the game ends when all your bases are destroyed or when the invaders reach the level of your base, not when your time runs out. This means that we're going to have to somehow keep track of the number of bases that we give out. These changes will do the trick:

```
440 BASES=BASES-1:IF BASES=
0 THEN POKE 53248,0:GOTO 20
20
4020 BASES=3
```

440 The variable BASES now keeps track of how many bases the player has left. Each time a base is destroyed, we subtract 1 from BASES and check to see if BASES is equal to 0, which would mean that the player has no bases remaining. If BASES is equal to 0, we move Player 0 off the screen and go off to the end-of-game routine.

4020 The player is given three bases to begin with, although you can change this line if you'd like to start with more (or less).

How Many Bases Are Left?

So far so good, but we need to have some way of showing the player how many bases are left. We could just print the values of BASES next to the score somewhere, but that can be a little confusing. You may recall that we included a base character in our redefined character set, and it was for this exact reason that we did so. If you make the following changes to the game, the extra bases will appear on the screen right next to the score:

```
450 POKE 87,1:POKE 88,SCRL:
POKE 89,SCRH:POSITION 19-BA
SES,0:? #6;" ";:POKE 53248,
128:POKE 1684,128:POKE 704,
15
5190 POKE 87,1:POKE 88,SCRL
:POKE 89,SCRH:POSITION 0,0:
? #6;" SCORE:     0    ■■"
```

450 Each time we give the player a new base, we will now erase one of the extra bases from the screen. This line sets up the computer so that we can print on the score line, and then replace one of the extra base characters with a space.

5190 In case you were wondering where the extra base characters came from, this line has been modified to print two extra bases alongside the score. Just in case you've forgotten, the base character is typed in as CTRL-0.

Now we're all set.

Keeping Score

Now that we're keeping accurate track of extra bases and the like, we may as well keep track of the score as well. The score is an integral part of a video game, since it is what most people use to gauge how well they have played. Some people play to better their own score, some to get the best score on the machine, and there are even those who spend hours and hours playing one game, in an attempt to hold the dubious honor of being the best in the world at that game. In any case, because of all this emphasis on score, it is important to design the game so that the score is representative of how well the player has done. This is not as easy as it sounds, however, and before we add scoring to BASIC invaders, let's take a look at just what's involved.

What Changes the Score?

The first step in designing the scoring for a game is to decide which events on the screen are going to produce a change in the score. For example, some games credit the player for shooting anything that moves, some give points for each time unit that the player remains alive, and others reward the player for avoiding objects on the screen.

Now it's true that how things are done depends a lot on the type of game that's involved, and there's no reason why we couldn't apply all the above mentioned scoring techniques to our simple BASIC invaders. (We won't though.) How? Well, we could give points for shooting the invaders and the saucer, for destroying all the invaders in as short a time as possible, and for avoiding the invader missiles by as much room as possible.

Trying to avoid the missiles by as much room as possible is pretty stupid, and wouldn't be something that we'd want to give points for. Destroying the invaders as quickly as possible isn't a bad idea, although we won't be using it here (you may want to try adding it to the program yourself). We will be giving points for shooting the invaders and saucer, so let's look at the different ways we can approach this seemingly simple task.

How Many Points for What?

Once we've decided what we're going to give points for, the next question is "how many points?", and there are a few guidelines we can follow here. First of all, today's video game players are used to high scores in the 1,000s. No matter how good a game is, a player isn't likely to be too enthusiastic if he does really well and gets a high score of 357. A high score of 357,000, on the other hand, is something that he or she will feel good about, even if the only difference is three extra 0s. As a matter of fact, you'll find that most games tend to pad out their scores with extra 0s for this exact reason. As a result, you'll see high scores like 54,200, but never like 54,237. It's a silly world. Anyway, we're going to pad out our score with one 0, as you'll see later.

For now, we still have to figure out how many points to put behind the 0s. Usually, or ideally, the points given for shooting something are representative of how difficult the object was to shoot. We have three different types of invaders on the screen, as well as an alien saucer. The invaders are the easiest to hit, so we'll give more points for hitting the saucer. The invaders vary in size from large to small, with the smaller ones being slightly more difficult to hit, so we'll give more points for hitting the smaller invaders.

Picking some numbers now, we'll give 10 points for the large invaders, 20 for the medium, 30 for the small, and 300 points for an alien saucer. Where do these numbers come from? Wherever you want. We could have given 50, 60, 70, and 500 if we wanted, or any combination that looked good. You may not agree with the numbers I'm using, and are more than welcome to change them. The only thing that matters in scoring is that harder tasks are awarded with more points. The actual numbers themselves will only affect how high the scores will be.

Well, with all that raving out of the way, we're now ready to make the necessary changes to our program. And here they are:

```
220 POKE 53249,0:POKE 53250
,5:SCORE=SCORE+30:GOTO 350
340 SCORE=SCORE+3-INT(R/96)
350 SCORE$=STR$(SCORE):POKE
 88,SCRL:POKE 89,SCRH:POKE
87,1:POSITION 12-LEN(SCORE$
),0:? #6;SCORE$;
390 RETURN
3030 DIM MLANG$(90),INV$(57
8),DAT$(16),ROW(5),DL$(30),
SCORE$(6)
4020 BASES=3:SCORE=0
```

220 When the saucer is shot down, we now add 30 to SCORE (which keeps track of the score, of course) and skip ahead to a routine that prints the updated score on the screen. In case you're wondering why we're adding 30 instead of 300, don't forget about the extra 0 that we're using to pad out the score. That 0 is permanently on the screen and is not included in SCORE. It's actually just as easy to keep the whole thing in SCORE. When you're designing a game in machine language you look at the score as a series of digits instead of a whole number. Since I do a lot of machine-language programming, I'm used to doing things this way, and so that's my excuse for doing things in a funny way here. Besides, it's a sneaky way of emphasizing the padded score.

340 You'll recall that R is equal to the row that the shot invader is in, times 48. The rows are numbered from 0 to 5, so if we divide R by 96 and take the integer part of the result, we'll get 0, 1 or 2, depending on which row the invader was in. Well, this also tells us what type of invader was shot, and we can use it to adjust SCORE. Which is exactly what this line does.

350-390 This is a simple routine that puts the updated score on the screen. Our first step is to convert SCORE into a string. We do this so that we can tell how many digits are in the score, which in turn allows

us to right-justify it on the screen (try changing this line so that it prints SCORE, not SCORE$, and you'll see why we're doing this). Next we set up the computer so that it's ready to print in the score section of screen memory, position the cursor appropriately, print the string, and then return back to the main part of the program.

3030 We need to reserve some space for SCORE$.

4020 And we also have to initialize SCORE to 0.

That's about all there is to scoring, with one exception. As a further incentive to getting a good score, most games give bonus lives (or time) to the player when he or she reaches a certain score. Some games give only one extra life, while others give many. Some limit the number of extra lives you can keep in reserve, some let you continue to accumulate lives as long as you can keep earning them. It's entirely up to the game designer. For our game, we're going to give one bonus base when the player reaches 10,000 points. The following lines take care of it for us:

```
360 IF SCORE<1000 OR BB=1 T
HEN 390
370 BB=1:BASES=BASES+1:POSI
TION 20-BASES,0:? #6;"■";
4020 BASES=3:SCORE=0:BB=0
```

360 We don't want to award a base if the score is less than 10,000 (remember the extra 0) or if the base has already been awarded. BB is a flag that gets set to 1 when the bonus base is awarded. I originally forgot to include such a flag, and as a result the game awarded me an extra base every time I shot something, once I'd gotten 10,000 points. Be careful of little details like this when you add a new feature to your game.

370 If we are going to award a base, then we set the bonus base flag, add 1 to our base count, and print the new base next to the other extra bases.

4020 Here we just make sure that BB is initialized to 0.

Of course right now the game won't let you get 10,000 points, since it ends when all the invaders are shot down. We'll take care of this in the next section, but you can change line 360 so that the extra base is awarded at a lower score if you want to see for yourself that the above lines do indeed work.

A New Level

We now have all the basic game play elements in place, but we're still limited to one screen of invaders, or one level of the game. Our next logical step, therefore, is to extend the game to more than one level, and that's exactly what we're going to do in this section.

In all of the early video games, a new level meant a more difficult version of the level before it. But that soon changed as games like *Donkey Kong, Tempest* and *Miner 2049er* hit the market. In these games a new level introduced a different challenge, which meant that the game took longer. Each level was progressively more difficult, and this is the basic guideline that all good games should follow. If a player is not consistently challenged by a game, then he or she will quickly tire of it, and this is something the game designer would prefer to avoid.

Increasing the Challenge

How do we go about increasing the challenge? As I mentioned already in the section on game logic, the most common ways are to speed things up, add more opponents, and give the opponents more strength. In other words, you want to give the computer more of an advantage over the player.

At the same time, you want to make sure that the player still has a chance. It's very important to maintain the impression that *hey, I made a dumb mistake but I'll do better next time,* rather than *there's no way I could have gotten out of that.* After all, why should anyone play a game they know they have no chance at?

It's very hard to predict how difficult the higher levels will get in some games, so you have to be careful when you're programming the difficulty changes. Another thing to watch out for is the fact that you will tend to be very good at your own game. As a result, a beginner may tend to find the game very difficult. It's a good idea to have a friend try the game out and give you their opinion. Make sure that they're not afraid to be honest, however!

It's extremely difficult to talk specifically about things like this, since each game is, and should be, different. You might want to spend some time at your local arcade, playing a variety of games and noticing what they do to make each level more difficult.

As far as our own game is concerned, we don't have many choices. Because of the speed problems with BASIC, we can't really do anything to speed things up. We do have a little room to add more opponents, but not much. We need something that we can use level after level.

As far as making the invaders stronger is concerned, we could add an extra missile, but again, we can't do this at every level. So what is left? What if we begin each level with the invaders a little closer to the bottom of the screen? That way the player has less time in which to destroy them all (since the game is over when they reach the bottom). We'll place a limit on how far down they can start, of course, since we have to make sure the player has a chance. This is a simple but effective solution that is also easy to do, something which we need to consider when using BASIC. Let's go ahead and do it:

```
2000 X=USR(ADR(VBLOFF$))
2010 X=USR(ADR(MOVMEM$),ADR
("♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
"),MEM1+LINE*24+TMP*48+24,2
3):GOTO 5000
4020 BASES=3:SCORE=0:BB=0:L
EVEL=0
5000 POKE 559,0:POKE 54276,
0:X=USR(ADR(MISCLR$),PA+768
,255,240):LEVEL=LEVEL+1
5005 X=USR(ADR(VBLOFF$))
5010 DLIST=PEEK(560)+PEEK(5
61)*256:IF SCRH<>0 THEN POK
E DLIST+4,SCRL:POKE DLIST+5
,SCRH
5190 IF LEVEL=1 THEN POKE 8
7,1:POKE 88,SCRL:POKE 89,SC
RH:POSITION 0,0:? #6;" SCOR
E:     0     ■■"
5280 LINE=LEVEL-2:IF LINE>8
 THEN LINE=8
```

2000 When we finish a level, the first thing we want to do is turn off the VBLANK routines, mainly so that the player can't move the base around while we're getting things ready. This line takes care of that for us.

2010 We also have to take care of the fact that the last invader explosion is still on the screen. The solution to this is to simply move a string of blanks (remember that the internal code for a blank is the ATASCII code for

a heart) into the line of the screen where the last explosion occurred. After we do this we're ready for the new level, so we skip ahead to the code that initializes a level. I'll go into a little more detail on why things are arranged this way after the rest of the explanation.

4020 We now have to keep track of which level we're on, which we'll do with the variable LEVEL. Here we just initialize LEVEL to 0.

5000 This is the beginning of the code that initializes a level. First we make sure that the screen is turned off. Then we initialize the fine scroll register to 0, clear the player and invader missiles, and increase the level number.

5010 If IF SCRH <>0 is true, this means that we've already completed at least one level, in which case we want to restore the first LMS in the display list (to get rid of the coarse scrolling we've done).

5190 We only want to set up the score and extra bases if this is the first level. If it isn't, then the score and bases are already on the screen and we don't want to change them.

5280 As our last step, we want LINE to reflect the level that we're on. (Remember that LINE specifies how far down the screen the invaders are). At the same time, we also want to make sure that the invaders don't start too far down, so we let LINE start no higher than 8. If LINE is equal to 8, then the bottom row of invaders will start two lines above the base. This gives the player just enough time to destroy the bottom row before it reaches the base.

You've no doubt noticed that lines 5000 through 5280 fall right in the middle of the initialization part of BASIC invaders that we'd already written. As a result, we only have a few lines to add rather than a whole bunch. Let's take a few minutes to look at how the initialization section of the program has been organized, since I've been somewhat sneaky about it.

Initialization Routines

In a game such as BASIC invaders, the initialization section can be divided into three sections. We'll call these sections "FIRST RUN," "NEW GAME," and "NEW LEVEL." Their purposes match their names.

FIRST RUN takes care of things that only need to be done when the program is first run. The FIRST RUN section of BASIC invaders takes up lines 3000 through 3310 and handles things like setting up the machine-language routines, the PMG area, and the character set.

NEW GAMES takes care of things that need to be done each time a new game is begun. It takes up lines 4000 through 4020 in BASIC invaders and handles the title page, initializing the screen, and initializing a few variables such as LEVEL and SCORE.

NEW LEVEL is the section of the code that we're dealing with now, and takes up lines 5000 through 5490. Apart from the tasks we just added, it also sets up the display list, draws the barriers, sets up INV$, initializes some more variables, and takes care of a few other sundry items.

As you design your game, you should try and group your initialization code into these three categories. I have to admit that I didn't when I first wrote BASIC invaders, and ended up having to go through and sort things out later. Try not to find yourself in the same predicament (easy to say).

Another Change

And now, back to the program. I conveniently left something very important out of the above additions, but you won't notice what until you reach Level 7. Can you guess what it is? Well, at Level 7 the invaders reach the barriers, and at the moment the program is only set up to handle this if it occurs during a level, not at the beginning of one. So, we have to make another change to the NEW LEVEL section:

```
5130 IF LEVEL<7 THEN BARLIM
=0:GOTO 5180
5140 TMP=LEVEL-7:IF LEVEL>9
 THEN TMP=2
5150 BARLIM=TMP+1:FOR X=1 T
O BARLIM:POKE DLIST+20+X,22
:NEXT X
5160 X=USR(ADR(MOVMEM$),DLI
ST+31+10*TMP,ADR(DL$),29-10
*TMP)
5170 X=USR(ADR(MOVMEM$),ADR
(DL$),DLIST+22+TMP,29-10*TM
P)
5220 IF BARLIM=3 THEN 5270
```

5130 If we aren't at Level 7 yet, then we haven't struck the barriers, so we set BARLIM to 0 and skip over the next part.

5140 TMP is actually equal to BARLIM-1, as you'll see in the next line. The only reason that we bother with it is because we'll need to use BARLIM-1 a lot in the next few lines, and using TMP instead is a lot simpler.

5150 Here we set BARLIM, and then we POKE the correct number of CHR 6 HSCs into the display list. You may want to read the "Onwards and Downwards" section again if you're not sure why we're doing this.

5160-5170 Now we want to "scrunch" the display list to get rid of the extra CHR 14 lines, so we use the same technique that we did in lines 1440-1450. You can also see now why we needed TMP.

5220 If BARLIM is equal to 3, which means that the invaders are starting far enough down so that the barriers are completely gone, then there's no point in drawing them.

Now everything works as it should. One final note before we move on. In the arcade version of *Space Invaders*, the difficulty of each level is further increased by the invaders speeding up as their numbers decrease. By the time you get down to the last invader, it is zooming back and forth across the screen and is much more difficult to hit before it reaches the bottom. Why don't we add this to our game? In case the answer isn't obvious, the problem is speed. If, however, you can get your hands on a BASIC compiler, which speeds BASIC programs up considerably, you could add a time delay between invader movements, and then have this delay decrease with the number of invaders remaining. Since you should seriously consider investing in a compiler if you're doing a lot of BASIC game programming, I thought I would just mention this now, since it pertains somewhat to this section.

Sound

Here are the changes to BASIC invaders that will get some simple sound effects up and running:

```
200 FOR X=4 TO 0 STEP -1:SO
UND 2,16,4,X:POKE 705,112+X
*3:POKE 706,112+X*3:FOR L=1
 TO 10:NEXT L:NEXT X
300 FOR X=55 TO 50 STEP -1:
SOUND 1,X,8,55-X:NEXT X:SOU
ND 1,0,0,0:RETURN
335 FOR X=250 TO 50 STEP -2
5:SOUND 1,X,10,8:NEXT X:SOU
ND 1,0,0,0
380 FOR X=1 TO 5:FOR Y=8 TO
 0 STEP -1:SOUND 1,10,10,Y:
NEXT Y:NEXT X:FOR PAUSE=1 T
O 10:NEXT PAUSE
430 FOR X=117 TO 250 STEP 4
:SOUND 1,X,8,INT((250-X)/9)
:POKE 704,112+INT((250-X)/9
):NEXT X
1010 FOR X=100+PI TO 116+PI
 STEP 2:SOUND 0,X,2,116+PI-
X:NEXT X:PI=PI+20:PI=PI*(PI
<70)
1160 FOR X=16 TO 0 STEP -2:
SOUND 1,20,8,X:NEXT X
1300 POKE 675,15:POKE 1685,
235:POKE 1686,240:POKE 705,
40:POKE 706,40:SAUCER=1:SOU
ND 2,10,4,4
1370 IF SAUCER=1 AND PEEK(1
686)<40 THEN SAUCER=0:SOUND
 2,0,0,0:POKE 705,15:FOR PA
USE=1 TO 10:NEXT PAUSE
2005 SOUND 2,0,0,0
2025 FOR X=0 TO 3:SOUND X,0
,0,0:NEXT X
```

And here's the explanation:

200 When the alien saucer has been hit, we now fade out its sound as well as its color.

220 This just turns off the sound of the missile, since it has collided with the saucer and therefore doesn't exist anymore.

300 This is at the end of the barrier explosion section, and makes a small explosion sound every time a barrier is hit. Notice that we're changing both pitch and volume.

335 An invader has been hit, so we make an explosion sound by varying the pitch on one of the "noise" distortion modes.

380 Here we make a "ting-ting-ting-ting-ting" sound when the bonus base is awarded. Notice that we're using a high-pitched pure note and just changing the volume.

430 The player base has been hit, so make an explosion sound as we fade it out (this line is similar to line 200 above, with the exception that here we're varying the pitch instead of the volume).

1010 This line is interesting, because it adds a lot to the effect of the game, but at the same time it also slows it down. You may or may not want to remove it. What it does is make a "tromping" or footstep sound as the invaders move across the screen. Because of its repetition it tends to draw the player deeper into the game, in a way that's almost hypnotic. Try it and see what I mean. In any case, the sound is created by varying the volume and pitch of one of the "noise" distortion modes. We also change the pitch between tromps, so that there are four differently pitched tromps arranged in a "one-two-three-four, one-two-three-four" pattern.

1160 Here we've just fired a player's missile, so we make a simple firing sound by varying the volume on a "noise" distortion mode.

1300 When we turn the saucer on, we also want to turn its sound on as well. The distortion mode that we use in this line is a good one to use for engine-type sounds, since at the right pitch it has a throbbing quality. Thus we can just set it once to get the effect we want, without having to worry about varying pitch or volume.

1370 We want to turn the saucer sound off when it goes off the edge of the screen, and that's exactly what we do here.

2005 There is a chance that the player will finish a level with the saucer in mid-flight, so we want to make sure that the saucer sound is turned off at the end of each level.

2025 Finally, at the end of a game we want to turn all sounds off.

The End

Don't be misled by the title of this section; it isn't the end of our programming. Instead we're going to be discussing the end of the game, that inevitable moment when the computer finally overcomes the player. Since there isn't really that much that happens at the end (other than things ending), this is going to be a pleasantly short section, so let's get right into it.

For the most part, a video game has two responsibilities at its termination. First, it should tell the player that the game is, indeed, over. As seemingly silly as this may sound, there are quite a few games out there, mostly in the home market, in which the player suddenly finds himself back in the title page with absolutely no idea of how they got there. At least give the player a chance to realize that they've lost.

The second responsibility is, of course, to go to the title page so that the player can take another shot at it (so to speak). And that's all there is to it. Of course some game designers feel the need to come up with some elaborate way of presenting the "you blew it" message, such as blowing up the screen or having the aliens stand there and laugh. This tends to annoy the player to the point where they play the game again just to get revenge. I've even seen some games with end screens that are so darn cute that I play the game again just so I can lose. You may wish to adopt one of these methods if you have the extra time and memory. Just keep in mind that an exploding screen won't work too well in a "Saturday morning cartoons" type of game, and a cute ending is totally out of place in a "destroy the galaxy" type.

Letting the Player Know What's Going On

Enough talk; let's get down to action. We're not going to add anything fancy to BASIC Invaders, just a simple message to let the player know what's going on. We'll also take care of a few other details that I forgot to mention above, such as cleaning up the screen a little. Here are our new additions:

```
2020 X=USR(ADR(MEMCLR$),MEM
1,528)
2030 POKE 54276,0:POKE 559,
0:POKE DLIST+5,INT(MEM1/256
):POKE DLIST+4,MEM1-PEEK(DL
IST+5)*256
2040 X=USR(ADR(MOVMEM$),ADR
("♥♥♥♥♥♥♥game♥over♥♥♥♥♥♥♥"
),MEM1+192,23)
2050 FOR X=53248 TO 53255:P
OKE X,0:NEXT X:POKE 559,62
2060 IF SCORE*10>HISCORE TH
EN HISCORE=SCORE*10
2070 POKE 20,1
2080 IF PEEK(20)<>0 THEN 20
80
2090 GOTO 4000
```

2020 Our first step is to clear the screen, since there are still invaders on it.

Our MEMCLR routine does this quickly for us.

2030 We're going to be printing a message on the screen and we want to make sure it's centered, so we set the fine scroll register back to 0. We also want to reset the coarse scrolling, so we turn off the screen and give the first LMS in the display list its original value back.

2040 Now we put up our message (avoiding the temptation to be nasty about it).

2050 Next we move all the players and missiles off-screen and turn the screen back on.

2060 This is as good a time as any to check for a new high score, so we do.

2070-2080 We need to give the player time to realize what's going on (i.e. time to read the message), so we set the system clock to 1 and then wait for it to reach 0 again (which takes 4.25 seconds).

2090 We're all done, so it's off to the NEW GAME section of our initialization code.

See, I told you this was going to be an easy section.

The Finishing Touches

Here we are with an actual working game. Of course I'll be the first one to admit that it's not quite as fast as we'd like, but that's the problem with BASIC. A good BASIC compiler will take care of this problem for us, but before we worry about speeding things up, we still have a few embellishments to add to the program.

The first thing we'd like to do is give the program a beginning. As it stands now, the game starts as soon as the program is RUN, and a new game begins as soon as the old one ends. It is preferable to have some sort of screen that comes up before the game begins, thereby giving the player an opportunity to start when he or she is ready.

This screen is called the title screen and you'll find it, in one form or another, in every video game on the market. Some of these screens are as simple as a few words announcing the name of the game (as ours will be), while others are almost an entire program within themselves.

Why should so much work be put into such a seemingly meaningless part of the program? In the old days, when video games were restricted to the arcades, it was the title screen that attracted players to the game. When video games came home, especially on the home computers, game designers essentially imitated the whole style of the arcade games, and thus the fancy title screens made their way into the home. Of course the homes games were bought on the basis of media advertising and not on the basis of the title screen, but programmers didn't seem to care. Not that a fancy screen isn't a nice touch, but it tends to take up a lot of precious memory.

The Title Screen

Since we're trying to keep BASIC invaders as short as possible, our title screen is going to be short but sweet. It will announce the name of the game, the authors and copyright notice. It will also show the high score to date, and will sit patiently on the screen until the player presses the Start button. We'll do a little bit of adjusting to the display list to get a nice-looking screen layout, but otherwise the rest of it is quite straightforward. See for yourself:

```
4000 GOSUB 6000:GRAPHICS 24
:POKE 559,0:POKE 756,CB+2
6000 GRAPHICS 0:POKE 559,0
6010 D2=PEEK(560)+PEEK(561)
*256
6020 SETCOLOR 4,7,0:SETCOLO
R 2,7,0:SETCOLOR 1,9,15:SET
COLOR 3,4,8:SETCOLOR 0,12,1
0
6030 POKE D2+7,7:POKE D2+16
,6:POKE D2+25,6
6040 POKE 752,1:POKE 82,0:P
OKE 83,39:POSITION 3,2:? "b
asic invaders"
6050 POSITION 28,2:? "By Cr
aig Patchett & You"
6060 POSITION 22,10:? "HI S
CORE:":POSITION 37-LEN(STR$
(HISCORE)),10:? HISCORE
6070 POSITION 0,19:? "PRESS
 start TO BEGIN"
6080 POSITION 22,20:? "(C)
1984 Educational Software,
Inc."
6090 POKE 559,34
6100 IF PEEK(53279)<>6 THEN
 6100
6110 RETURN
```

This shouldn't be necessary, but here's the explanation:

4000 Before we start the game, we now go off to our title-screen routine.

6000 We get the computer to set up a graphics mode 0 display list, and then turn off the screen.

6010 Then we set the screen colors to something that will be a little more appealing than the regular colors.

6020-6030 Next we figure out where the display list is and add a graphics mode 2 line and two graphics mode 1 lines to it.

6040-6080 We don't want to see the cursor on the screen so we turn it off. We also set the margins so that we can print across the whole screen, and then we print the text.

6090 Our title screen is all set up now, so we go ahead and turn it on.

6100 All that's left now is to wait until the Start button is pressed.

6110 And then return back to line 4000.

Disabling the Break Key

As you can see, the result is simple but serves a purpose. If you're feeling adventurous, you may like to spruce it up a little. A simple but effective change would be to add some of the invader characters and have them walk in place or even across the screen. Use your imagination.

We have one more final change to the game that will make it complete. It's a small change, and one that isn't really necessary other than as a precaution. We're going to disable the Break key so that the player can't accidentally stop the program. All it takes is a simple two-line routine that has to be executed each time the GRAPHICS command is used (since this command sets the Break key back to normal). And now (drum roll), here are the last additions to BASIC invaders.

```
90 GOTO 3000
3000 GOSUB 7000
4000 GOSUB 6000:GRAPHICS 24
:GOSUB 7000:POKE 559,0:POKE
 756,CB+2
6000 GRAPHICS 0:GOSUB 7000:
POKE 559,0
7000 IF PEEK(16)-128>=0 THE
N POKE 16,PEEK(16)-128:POKE
 53774,PEEK(16)
7010 RETURN
```

And, of course, the last explanation:

90 A simple change to this line due to the addition of the next line.

3000 Here we've added the first call to our routine, right after the GRAPHICS command.

6000 Again, a call to the routine after the GRAPHICS command.

7000-7010 This is the routine. You don't need to understand what it's doing, just make sure that you use it exactly as it appears here.

Congratulations! Our game is now complete!

The Final Listing

Now that you have hopefully typed everything in and have a working version of the game it may help to see all of the code together. The following listing is here for that purpose.

To create your complete copy of BASIC Invaders, first type in Listing 1 (checking your work with BASIC Editor II, found elsewhere in this issue) and save it to disk. (Don't bother trying to type lines 29000 through 32510; Listing 2 will do that for you.) Now type in Listing 2, save it and run it. When the program has finished creating the lines containing the ML strings, type LIST "D:LINES.LST",29000,32510 to save the newly created lines to disk. Now load the program you typed from Listing 1 into your computer's memory and type ENTER "D:LINES.LST" to merge the lines created by Listing 2 with the main program. Finally, save the completed program to disk.

LISTING 1: BASIC

```
IU 90 GOTO 3000
UU 180 X=USR(ADR(MISCLR$),PA+768,255,252)
   :POKE 1720,0:IF PEEK(1716)<>255 THEN 2
   30
QI 190 POKE 675,15
AK 200 FOR X=4 TO 0 STEP -1:SOUND 2,16,4,
   X:POKE 705,112+X*3:POKE 706,112+X*3:FO
   R L=1 TO 10:NEXT L:NEXT X
GS 210 SOUND 2,0,0,0:POKE 1685,0:POKE 168
   6,5:POKE 705,15
MH 220 POKE 53249,0:POKE 53250,0:SCORE=SC
   ORE+30:GOTO 350
JN 230 TMP=PEEK(53248):IF TMP<4 OR TMP=8
   THEN 310
WF 240 SOUND 1,0,0,0
FK 250 X=PEEK(1692)-47+2*(RND(0)>0.5)-1:Y
   =PEEK(1696)-158-8*BARLIM
BL 260 POKE 89,INT((MEM7+320*BARLIM)/256)
   :POKE 88,MEM7+320*BARLIM-256*PEEK(89):
   POKE 87,7:COLOR 0
OL 270 YT=Y-3:YT=YT*(YT>=0)
AC 280 PLOT X-2,YT:DRAWTO X+2,YT+6:PLOT X
   ,YT:DRAWTO X,YT+6:PLOT X+2,YT:DRAWTO X
   -2,YT+6
EO 290 PLOT X-2,Y*(Y>=0):DRAWTO X+2,Y*(Y>
   =0)
DI 300 FOR X=55 TO 50 STEP -1:SOUND 1,X,8
   ,55-X:NEXT X:SOUND 1,0,0,0:RETURN
AS 310 IF TMP=0 THEN 250
PX 320 R=48*INT((PEEK(1696)-38-8*(LINE))/
   16):C=2*INT((PEEK(1692)-SCROLL-COARSE*
   8-55)/16):R=R*(R>=0)
TN 330 INV$(R+C+28,R+C+29)="▄▀":INV$(R+C+
   315,R+C+316)="▀▄":EF=2
UN 340 SCORE=SCORE+3-INT(R/96)
MH 350 SCORE$=STR$(SCORE):POKE 88,SCRL:PO
   KE 89,SCRH:POKE 87,1:POSITION 12-LEN(S
   CORE$),0:? #6;SCORE$;
GH 360 IF SCORE<1000 OR BB=1 THEN 390
UG 370 BB=1:BASES=BASES+1:POSITION 20-BAS
   ES,0:? #6;"▗▖";
LL 380 FOR X=1 TO 5:FOR Y=8 TO 0 STEP -1:
   SOUND 1,10,10,Y:NEXT Y:NEXT X:FOR PAUS
   E=1 TO 10:NEXT PAUSE
FT 390 FOR X=250 TO 50 STEP -25:SOUND 1,X
   ,10,8:NEXT X:SOUND 1,0,0,0:RETURN
QH 400 X=USR(ADR(MISCLR$),PA+768,255,243)
   :POKE 1721,0:IF PEEK(1717)<>255 THEN 4
   70
CU 410 POKE 1664,255
CI 420 X=USR(ADR(MISCLR$),PA+768,255,252)
   :POKE 1700,0
AD 430 FOR X=117 TO 250 STEP 4:SOUND 1,X,
   8,INT((250-X)/9):POKE 704,112+INT((250
   -X)/9):NEXT X
XF 440 BASES=BASES-1:IF BASES=0 THEN POKE
    53248,0:GOTO 2020
BK 450 POKE 87,1:POKE 88,SCRL:POKE 89,SCR
   H:POSITION 19-BASES,0:? #6;" ";:POKE 5
   3248,128:POKE 1684,128:POKE 704,15
SV 460 POKE 1664,0:RETURN
CO 470 X=PEEK(1693)-47+2*(RND(0)>0.5)-1:Y
   =PEEK(1697)-159-8*BARLIM:GOTO 260
UJ 1000 X=USR(ADR(MOVMEM$),ADR(INV$)+5B,M
   EM1+LINE*24,287-48*(5-BOTROW))
AE 1020 IF EF=0 THEN 1080
AD 1030 INV$(R+C+28,R+C+29)="♥♥":INV$(R+C
   +315,R+C+316)="♥♥"
PB 1040 TMP=R/48:ROW(TMP)=ROW(TMP)-1:IF R
   OW(TMP)<>0 OR TMP<>BOTROW THEN 1080
GK 1050 FOR LP=BOTROW TO 0 STEP -1:IF ROW
   (LP)=0 THEN NEXT LP:GOTO 2000
DS 1060 POP :BOTROW=LP
PO 1070 X=USR(ADR(MOVMEM$),ADR("♥♥♥♥♥♥♥♥♥♥
   ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"),MEM1+LINE*24+TMP*48+
   24,23)
FM 1080 5B=5B+287*(5B=0)-287*(5B=287)
PD 1090 IF PEEK(1720)<>0 THEN GOSUB 180
HZ 1100 IF PEEK(1721)<>0 THEN GOSUB 400
DF 1110 POKE 53278,0
KY 1120 IF STRIG(0)=1 OR PEEK(1700)<>0 OR
    PEEK(1720)<>0 OR EF<>0 THEN 1170
HT 1130 SOUND 1,0,0,0
SI 1140 X=USR(ADR(MISCLR$),PA+768,255,252
   ):POKE 1692,PEEK(1684)+2:POKE 53252,PE
   EK(1692):POKE 1696,201:POKE PA+969,1
U5 1150 POKE PA+970,1:POKE 53278,0:POKE 1
   712,15:POKE 1716,6:POKE 1720,0:POKE 17
   00,1
FJ 1160 FOR X=16 TO 0 STEP -2:SOUND 1,20,
   8,X:NEXT X
QG 1170 IF PEEK(1701)<>0 OR PEEK(1721)<>0
    OR LINE+BOTROW*2=19 THEN 1250
BX 1180 X=USR(ADR(MISCLR$),PA+768,255,243
   ):X=PEEK(20)+1:IF X=256 THEN X=0
PL 1190 IF PEEK(20)<X THEN 1190
IN 1200 POKE 53278,0:FC=INT(RND(0)*8)*2:F
   R=BOTROW*48
KB 1210 IF INV$(FR+FC+28,FR+FC+28)="♥" OR
    INV$(FR+FC+28,FR+FC+28)="▀▀" THEN FC=F
   C-2+16*(FC=0):GOTO 1210
NO 1220 POKE 1693,FC*8+62+SCROLL+COARSE*8
   :POKE 53253,PEEK(1693):FV=BOTROW*16+LI
   NE*8+46
HW 1230 POKE 1713,4:POKE 1717,1
GK 1240 POKE 1697,FV:FV=FV+PA+768:POKE FV
   ,PEEK(FV)+4:POKE FV+1,PEEK(FV+1)+4:POK
   E 1701,1
HZ 1250 EF=EF-(EF<>0)
LQ 1260 IF PEEK(20)<30 AND PEEK(19)=0 THE
   N 1330
NI 1270 IF PEEK(53251)=0 THEN 1330
RR 1280 CHANGE=-CHANGE:POKE 1791,129-PEEK
   (1791):POKE 19,0:POKE 20,0:POKE 53255,
```

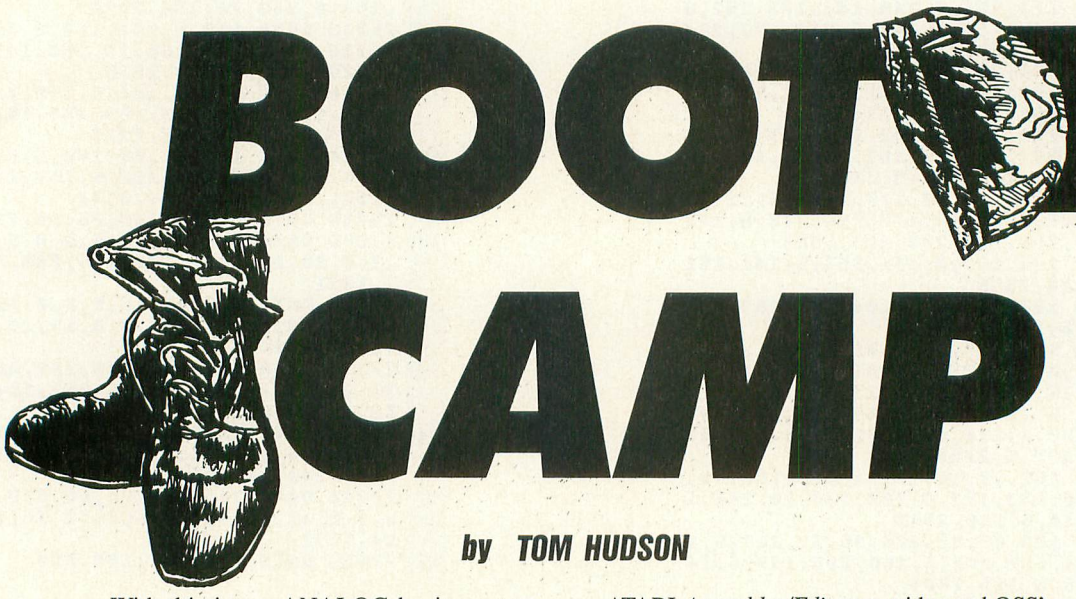
```
   127+77*CHANGE:LINE=LINE+1
GW 1290 IF CHANGE=1 OR LINE+BOTROW*2=20 T
   HEN 1400
HF 1300 POKE 675,15:POKE 1685,235:POKE 16
   86,240:POKE 705,40:POKE 706,40:SAUCER=
   1:SOUND 2,10,4,4
UW 1310 POKE 675,11
OW 1320 GOTO 1400
TJ 1330 SCROLL=SCROLL+CHANGE
CK 1340 IF SCROLL>15 THEN SCROLL=SCROLL-1
   6:COARSE=COARSE+2:POKE 1790,2:GOTO 136
   0
JC 1350 IF SCROLL<0 THEN SCROLL=SCROLL+16
   :COARSE=COARSE-2:POKE 1790,2
MZ 1360 POKE 1789,SCROLL:POKE 1788,1
ST 1370 IF SAUCER=1 AND PEEK(1686)<40 THE
   N SAUCER=0:SOUND 2,0,0,0:POKE 705,15:F
   OR PAUSE=1 TO 10:NEXT PAUSE
DP 1380 IF PEEK(1790)<>0 THEN 1380
NV 1390 GOTO 1000
HO 1400 POKE 53278,0:IF LINE+BOTROW*2<>20
    THEN 1430
UU 1410 X=USR(ADR(MOVMEM$),ADR(INV$)+5B,M
   EM1+LINE*24,287-48*(5-BOTROW))
ON 1420 GOTO 2020
LE 1430 IF LINE+BOTROW*2<>15+BARLIM OR BA
   RLIM=3 THEN 1000
CI 1440 POKE 559,0:POKE DLIST+21+BARLIM,2
   2:X=USR(ADR(MOVMEM$),DLIST+31+BARLIM,A
   DR(DL$),29-10*BARLIM)
UM 1450 X=USR(ADR(MOVMEM$),ADR(DL$),DLIST
   +22+BARLIM,29-10*BARLIM):POKE 20,0
OX 1460 IF PEEK(20)<2 THEN 1460
VT 1470 POKE 559,62:BARLIM=BARLIM+1:GOTO
   1000
DT 2000 X=USR(ADR(VBLOFF$)):X=USR(ADR(MOV
   MEM$),MEM1-44,ADR(DL$),19)
DT 2010 X=USR(ADR(MOVMEM$),ADR("♥♥♥♥♥♥♥♥♥
   ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"),MEM1+LINE*24+TMP*48
   +24,23):GOTO 5000
SD 2020 FOR R=0 TO 21:X=USR(ADR(MOVMEM$),
   ADR("♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥"),MEM1+2
   4*R,23):NEXT R
PY 2025 FOR X=0 TO 3:SOUND X,0,0,0:NEXT X
HZ 2030 POKE 54276,0:POKE 559,0:POKE DLIS
   T+5,INT(MEM1/256):POKE DLIST+4,MEM1-PE
   EK(DLIST+5)*256
LR 2040 X=USR(ADR(MOVMEM$),ADR("♥♥♥♥♥♥♥♥ga
   me♥over♥♥♥♥♥♥♥♥"),MEM1+192,23)
RC 2050 POKE 53277,0:FOR X=53261 TO 53265
   :POKE X,0:NEXT X:POKE 559,62
XP 2060 IF SCORE*10>HISCORE THEN HISCORE=
   SCORE*10
IT 2070 POKE 20,2
ON 2080 IF PEEK(20)>1 THEN 2080
OX 2090 GOTO 4000
ZF 3000 GOSUB 7000
QQ 3010 POKE 559,0
SP 3020 FOR X=53248 TO 53255:POKE X,0:NEX
   T X
SZ 3030 DIM MLANG$(90),DL$(30),INV$(578),
   DAT$(16),SCORE$(6),ROW(5)
PG 3040 DIM VBLOFF$(20):GOSUB 29000:VBLOF
   F$=MLANG$
GE 3050 DIM MOVMEM$(41):GOSUB 29500:MOVME
   M$=MLANG$
IV 3060 DIM MISCLR$(26):GOSUB 30000:MISCL
   R$=MLANG$
XG 3070 DIM MEMCLR$(36):GOSUB 30500:MEMCL
   R$=MLANG$
VT 3100 FOR BYTE=0 TO 10:READ DAT:POKE 15
   36+BYTE,DAT:NEXT BYTE
KM 3110 DATA 72,169,212,141,10,212,141,26
   ,208,104,64
DI 3120 FOR BYTE=1 TO 40:READ DAT:POKE 17
   37+BYTE,DAT:NEXT BYTE
LF 3130 DATA 252,243,207,63,0,128,0,128,1
   28,2,2,3,3,1,0,0,0,0,0,4,5,6,7,3,76,12
   8
LA 3140 DATA 64,76,80,64,76,177,64,76,5,6
   5,76,88,65,0
NE 3150 PB=PEEK(740)-8:CB=PB-4:POKE 106,C
   B-4:CA=CB*256:PA=PB*256
CE 3160 GOSUB 31000
QK 3170 X=USR(ADR(MOVMEM$),ADR(MLANG$),CA
   -256,78)
JE 3180 MEM=PA
SQ 3190 FOR SEC=0 TO 7:GOSUB 32000+10*SEC
   :X=USR(ADR(MOVMEM$),ADR(MLANG$),MEM,LE
   N(MLANG$)-1)
YD 3200 MEM=MEM+LEN(MLANG$):NEXT SEC
RQ 3210 X=USR(ADR(MOVMEM$),57344,CA,1023)
VL 3220 MEM=CA+512:FOR SEC=0 TO 1:GOSUB 3
   2500+10*SEC:X=USR(ADR(MOVMEM$),ADR(MLA
   NG$),MEM,LEN(MLANG$))
YM 3230 MEM=MEM+LEN(MLANG$):NEXT SEC
OR 3240 X=USR(ADR(MOVMEM$),CA+128,CA+640,
   336)
RB 3250 X=USR(ADR(MEMCLR$),PA+768,1280):P
   OKE 54279,PB:POKE 623,4
CD 3270 FOR BYTE=201 TO 208:READ DAT:POKE
    PA+1024+BYTE,DAT:NEXT BYTE
YN 3280 DATA 16,16,56,56,124,124,198,198
AB 3290 FOR BYTE=30 TO 38:READ DAT:POKE P
   A+1280+BYTE,DAT:READ DAT:POKE PA+1536+
   BYTE,DAT:NEXT BYTE
SS 3300 DATA 15,16,31,24,63,28,106,22,106
   ,22,255,31,255,31,56,28,16,8
CF 3310 FOR BYTE=39 TO 205:POKE PA+768+BY
   TE,192:NEXT BYTE
ZA 4000 GOSUB 6000:GRAPHICS 24:GOSUB 7000
   :POKE 559,0:POKE 756,CB+2
MJ 4010 SETCOLOR 0,4,14:SETCOLOR 1,4,6:SE
   TCOLOR 2,15,14:SETCOLOR 3,4,10:SETCOLO
   R 4,7,0
FB 4020 LEVEL=0:BASES=3:BB=0:SCORE=0
ZK 5000 POKE 559,0:X=USR(ADR(MISCLR$),PA+
   768,255,240):LEVEL=LEVEL+1
QH 5005 X=USR(ADR(VBLOFF$))
CU 5010 DLIST=PEEK(560)+PEEK(561)*256:IF
   SCRH<>0 THEN POKE DLIST+4,SCRL:POKE DL
   IST+5,SCRH
WH 5020 POKE DLIST+3,86
MJ 5030 L=PEEK(DLIST+4)+44:POKE DLIST+5,P
   EEK(DLIST+5)+(L>255):POKE DLIST+4,L-25
   6*(L>255)
TB 5040 FOR X=6 TO 20:POKE DLIST+X,22:NEX
   T X:FOR X=24 TO 50:POKE DLIST+X,14:NEX
   T X
EF 5050 MEM7=PEEK(88)+PEEK(89)*256+600
UJ 5060 POKE DLIST+21,78:POKE DLIST+23,IN
   T(MEM7/256):POKE DLIST+22,MEM7-INT(MEM
   7/256)*256
CR 5070 POKE DLIST+31,78:POKE DLIST+33,IN
   T((MEM7+320)/256):POKE DLIST+32,MEM7+3
   20-PEEK(DLIST+33)*256
IR 5080 POKE DLIST+41,78:POKE DLIST+43,IN
   T((MEM7+640)/256):POKE DLIST+42,MEM7+6
   40-PEEK(DLIST+43)*256
QL 5090 POKE DLIST+51,22:POKE DLIST+52,22
RN 5100 POKE DLIST+53,150:POKE 512,0:POKE
    513,6:POKE 54286,192
KY 5110 POKE DLIST+54,112:POKE DLIST+55,7
   0:POKE DLIST+56,PEEK(88):POKE DLIST+57
   ,PEEK(89)
FV 5120 POKE DLIST+58,65:POKE DLIST+59,PE
   EK(560):POKE DLIST+60,PEEK(561)
PS 5130 IF LEVEL<7 THEN BARLIM=0:GOTO 518
   0
EW 5140 TMP=LEVEL-7:IF LEVEL>9 THEN TMP=2
UI 5150 BARLIM=TMP+1:FOR X=1 TO BARLIM:PO
   KE DLIST+20+X,22:NEXT X
JR 5160 X=USR(ADR(MOVMEM$),DLIST+31+10*TM
   P,ADR(DL$),29-10*TMP)
LY 5170 X=USR(ADR(MOVMEM$),ADR(DL$),DLIST
   +22+TMP,29-10*TMP)
MH 5180 MEM1=PEEK(DLIST+4)+PEEK(DLIST+5)*
   256:SCRL=PEEK(DLIST+56):SCRH=PEEK(DLIS
   T+57)
YC 5190 IF LEVEL=1 THEN POKE 87,1:POKE 88
   ,SCRL:POKE 89,SCRH:POSITION 0,0:? #6;"
    SCORE:     0    ▪▪"
YZ 5200 POKE 559,62
ML 5210 POKE 87,7:POKE 89,INT(MEM7/256):P
   OKE 88,MEM7-PEEK(89)*256:COLOR 3
VR 5220 IF BARLIM=3 THEN 5270
OJ 5230 RESTORE 5260
KH 5240 FOR X=1 TO 10:READ N,XS,Y,XE:FOR
   T=0 TO N-1:FOR Z=0 TO 3:PLOT Z*40+9+XS
   ,Y+T:DRAWTO Z*40+9+XE,Y+T:NEXT Z
```

```
KU 5245 NEXT T
LV 5250 NEXT X
ON 5260 DATA 2,4,0,17,2,3,2,18,2,2,4,19,1
   0,1,6,20,2,1,16,7,2,14,16,20,2,1,18,6,
   2,15,18,20,4,1,20,5,4,16,20,20
UT 5270 SCROLL=0:CHANGE=1:SB=0:EF=0:BOTRO
   W=5:COARSE=0:SAUCER=0
AN 5280 LINE=LEVEL-2:IF LINE>8 THEN LINE=
   8
DR 5290 POKE 54276,0
LB 5300 POKE 1664,0:POKE 1665,43:POKE 166
   6,43:POKE 1667,255
US 5310 POKE 1668,50:POKE 1669,0:POKE 167
   0,5:POKE 1672,200:POKE 1673,235:POKE 1
   674,240
VD 5320 POKE 1676,201:POKE 1677,30:POKE 1
   678,30:POKE 1680,201:POKE 1681,30:POKE
    1682,30
BR 5330 POKE 1684,128:POKE 1685,0:POKE 16
   86,5:POKE 1688,201:POKE 1689,30:POKE 1
   690,30
AZ 5340 POKE 1700,0:POKE 1701,0:POKE 1720
   ,0:POKE 1721,0
VQ 5350 POKE 1704,0:POKE 1705,0:POKE 1708
   ,129:POKE 1709,1
YU 5360 POKE 1789,0:POKE 1790,0:POKE 1791
   ,128
YV 5365 POKE 675,15
FE 5370 INV$="♥":INV$(578)="♥":INV$(2)=IN
   V$
NC 5380 RESTORE 5410
QD 5390 FOR LP=0 TO 4 STEP 2:READ DAT$:IN
   V$(LP*48+28,LP*48+43)=DAT$:INV$(LP*48+
   363,LP*48+378)=DAT$
UI 5400 READ DAT$:INV$(LP*48+76,LP*48+91)
   =DAT$:INV$(LP*48+315,LP*48+330)=DAT$:N
   EXT LP
CE 5410 DATA ┣┃┣┃┣┃┣┃┣┃┣┃┣┃┣┃,┛┛┛┛┛┛┛┛┛┛┛┛
   ┛┛┛┛,EFEFEFEFEFEFEFEF,GHGHGHGHGHGHGHG
   H,IJIJIJIJIJIJIJIJ,KLKLKLKLKLKLKLKL
SC 5420 FOR R=0 TO 5:ROW(R)=8:NEXT R
PI 5430 POKE 19,0:POKE 20,0:POKE 53278,0
LP 5440 POKE 704,15:POKE 705,15:POKE 706,
   70:POKE 707,112
HF 5450 POKE 53248,128:POKE 53249,0:POKE
   53250,5:POKE 53255,205
HH 5460 GOSUB 31500:X=USR(ADR(MLANG$),CA-
   256)
PV 5470 X=USR(PA,PB,PB)
FU 5480 POKE 53277,3
OB 5490 GOTO 1000
QN 6000 GRAPHICS 0:GOSUB 7000:POKE 559,0
BV 6010 D2=PEEK(560)+PEEK(561)*256
ES 6020 SETCOLOR 4,7,0:SETCOLOR 2,7,0:SET
   COLOR 1,9,15:SETCOLOR 3,4,8:SETCOLOR 0
   ,12,10
RQ 6030 POKE D2+7,7:POKE D2+16,6:POKE D2+
   25,6
YU 6040 POKE 752,1:POKE 82,0:POKE 83,39:P
   OSITION 3,2:? "basic invaders"
HE 6050 POSITION 28,2:? "By Craig Patchet
   t & You"
UP 6060 POSITION 22,10:? "HI SCORE:":POSI
   TION 37-LEN(STR$(HISCORE)),10:? HISCOR
   E
TZ 6070 POSITION 0,19:? "PRESS start TO B
   EGIN"
SR 6080 POSITION 22,20:? "(C) 1983 Educat
   ional Software, Inc."
ZK 6090 POKE 559,34
TE 6100 IF PEEK(53279)<>6 THEN 6100
AJ 6110 RETURN
GR 7000 IF PEEK(16)-128>=0 THEN POKE 16,P
   EEK(16)-128:POKE 53774,PEEK(16)
AI 7010 RETURN
MS 29000 MLANG$="h"d _)/ \d"d b)\ \d◆":RE
   TURN
MW 29500 MLANG$="hh,Oh,Nh,Qh,Ph* )\P |h(1N
   ┏P/@)PwfOfQJ◆)Pi◆":RETURN
SU 30000 MLANG$="hh,Oh,Nhh(hh,P1N%P┏N/@)P
   U◆":RETURN
QN 30500 MLANG$="hh,Lh,Kh*)♥ )◆♥P┤h()♥┏K/
   @)PyfLJ◆)Pj◆":RETURN
GG 31000 MLANG$="- |/pF-K/ H T-(|/p3-0 ┏L-1 |
   ,M H -)/0-1L┴XM(|/┏LH1Li♥┏L♣+1L8XM(|/┏LH1
   Li♥┏L)♥ (|/ |/L_d":RETURN
PQ 31500 MLANG$="hh*h()/ \d◆":RETURN
HW 32000 MLANG$="hhh (/hh d/ g/ i/ W/nW/
   p/np/))G/" H=/ )+p )G/"┼ H/ =N/
   =/J +V)\.p/ 1 \d◆ n/=/┴mH/":RETURN
VO 32010 MLANG$="(MG/♣┘,G/=/8mH/ I/ 1LH,
   L/P,LI/8r)♥┏L◆ n/=/8mH/(8 ♥=/┴mH/ I
   /H1L/┏LHLG/p\LI/♣opm)♥┏L◆\H"":RETURN
WO 32020 MLANG$="┤ n/h*=/8mH/(8 ♥=/┴mH
   / I/1L=J/┏LHH=J/I)1L/ ┏L┏LHHLG/8\LI/♣
   ]p[=J/1L┏L/=J/1L┏L◆\H"┤ n/h*":RETURN
FL 32030 MLANG$="=/┴mH/(MG/♣┘,G/=/8mH/
   I/H1L=J/┏L/=J/I)1LHH┏L┏L/P,LI/8◆=J/1
   L┏LH=J/1L┏L◆==/,L┴X-(/B/,M)":RETURN
YF 32040 MLANG$="♥,M◆"♥(C/0 j9x ) P┬=O/]/ /
   p+)♥,M(O/=O/↓♥PP↑9x )┤P_)♥,M=O/I(/p ^O
   /=O/↓♥P(C/9x ) P┏=/]♣/p←(/":RETURN
VV 32050 MLANG$=" e/\+_9x ) P_=/]_/p/^/
   b/h◆┤P_"♥=$/p5=(/p2+┬^↑/^↑/=↑/↓┤PI/8
   )♥↓$/p5(↑/(↑/=↑/↓┤PIP♣ )♥↓$/":RETURN
RC 32060 MLANG$="p jPD=,/p9+┬^ /^ / h/= /I
   +8')♥↓$/pJ( /( / k/= /┴Xi+MG/8┤)pP\)♥↓
   $/p*=8/=♥Pp ) ↓8/↓8/)♥↓$/=4/":RETURN
QU 32070 MLANG$="=Pp ) ↓4/↓8/)♥↓$/h◆┤P,L
   bd♥":RETURN
IK 32500 MLANG$="♥♥♥♥♥♥♥♥┣┘\‾, h\♥@◆8p@ P
   ┣┘\‾,,┐H♥@◆8p ♣ H,↓→┬● |♣ p8xh(@ |●┬↓→,
   ┤@(h8xp ♣┘, →| →/ ,0@px┴x◆♣┘, ":RETURN
QE 32510 MLANG$="→| →‾┴"@px┴x8┴0♥,┐♥" ♥┐ ,♥
   @♥◆♥@ ♣♣88||FF":RETURN
```

LISTING 2: BASIC

```
VW 100 GRAPHICS 0:? "Make sure you have s
   aved a copy of":? "this program before
    RUNning it":FOR X=1 TO 1050:NEXT X
SQ 110 ? :?
RO 120 DIM LN(8):FOR X=1 TO 8:READ DAT:LN
   (X)=DAT:NEXT X
PE 130 DATA 20,41,26,36,112,11,657,128
OJ 140 FOR X=1 TO 8:TOT=0:N=0:GOSUB 1000
NH 150 FOR N=1 TO LN(X):READ DAT:TOT=TOT+
   DAT
HP 160 IF N/25<>INT(N/25) THEN 190
QP 170 T=TOT:TOT=0:READ DAT:IF DAT<>T THE
   N ? "...ERROR":STOP
QY 180 GOSUB 1000
JW 190 NEXT N:READ DAT:IF DAT<>TOT THEN ?
    "...ERROR":STOP
LM 200 NEXT X
AJ 210 RESTORE 20000
OV 220 FOR X=1 TO 8:L=28500+500*X:GOSUB 1
   010
BP 230 FOR N=1 TO LN(X):READ DAT:? CHR$(2
   7);CHR$(DAT);
TJ 240 IF N/25=INT(N/25) THEN READ DAT
NF 250 IF N/90=INT(N/90) THEN GOSUB 1020:
   L=L+10:GOSUB 1010
RQ 260 NEXT N:READ DAT:GOSUB 1020
MA 270 NEXT X
OH 280 END
LW 1000 ? :? "CHECKING LINE ";19000+1000*
   X+10*INT(N/25);:RETURN
DJ 1010 GRAPHICS 0:POSITION 2,4:? L;" MLA
   NG$=";CHR$(34);:RETURN
CU 1020 ? CHR$(34);":RETURN":? "CONT":POS
   ITION 0,0:POKE 842,13:STOP
UF 1030 POKE 842,12:RETURN
UG 20000 DATA 104,162,228,160,95,169,6,32
   ,92,228,162,228,160,98,169,7,32,92,228
   ,96,2548
QY 21000 DATA 104,104,133,207,104,133,206
   ,104,133,209,104,133,208,104,170,160,2
   55,138,208,2,104,168,177,206,145,3719
EW 21010 DATA 208,136,192,255,208,247,230
   ,207,230,209,202,224,255,208,233,96,33
   40
TH 22000 DATA 104,104,133,207,104,133,206
   ,104,104,168,104,104,133,208,177,206,3
   7,208,145,206,136,192,255,208,245,3931
```

```
YN 22010 DATA 96,96
JJ 23000 DATA 104,104,133,204,104,133,203
   ,104,170,169,0,160,255,224,0,208,4,104
   ,168,169,0,145,203,136,192,3396
PM 23010 DATA 255,208,249,230,204,202,224
   ,255,208,234,96,2365
FT 24000 DATA 173,251,6,240,104,173,252,6
   ,141,4,212,173,253,6,141,5,212,173,254
   ,6,240,79,173,48,2,3327
JY 24010 DATA 133,204,173,49,2,133,205,16
   0,3,177,204,201,65,240,61,201,1,240,52
   ,41,112,201,64,144,48,3114
IX 24020 DATA 201,80,144,42,200,173,255,6
   ,48,18,177,204,24,216,109,254,6,145,20
   4,200,177,204,105,0,145,3337
SL 24030 DATA 204,144,20,177,204,56,216,2
   37,254,6,145,204,200,177,204,233,0,145
   ,204,144,2,200,200,200,208,3984
NY 24040 DATA 189,169,0,141,254,6,141,251
   ,6,76,95,228,1556
IE 25000 DATA 104,104,170,104,168,169,6,3
   2,92,228,96,1273
HM 26000 DATA 104,104,104,141,188,6,104,1
   04,141,228,6,141,231,6,141,234,6,141,2
   37,6,238,237,6,141,240,3235
WO 26010 DATA 6,238,240,6,169,127,141,199
   ,6,162,9,160,4,173,47,2,41,16,240,9,16
   9,255,141,199,6,2765
FA 26020 DATA 162,19,160,8,140,200,6,160,
   9,189,206,6,153,189,6,202,136,16,246,1
   69,7,174,240,6,160,2969
OH 26030 DATA 108,32,92,228,96,32,238,6,1
   89,152,6,24,109,200,6,168,205,199,6,14
   4,3,172,199,6,189,2809
BK 26040 DATA 152,6,56,237,200,6,141,201,
   6,136,177,204,200,145,204,136,240,5,20
   4,201,6,176,242,169,0,3450
BE 26050 DATA 145,204,96,32,238,6,189,152
   ,6,56,237,200,6,168,176,2,160,0,189,15
   2,6,24,109,200,6,2759
MY 26060 DATA 141,201,6,200,177,204,136,1
   45,204,200,204,199,6,240,7,204,201,6,1
   44,239,240,237,169,0,145,3855
TM 26070 DATA 204,96,138,72,162,4,32,238,
   6,104,170,189,160,6,56,237,200,6,168,1
   76,2,160,0,189,160,2935
HO 26080 DATA 6,24,109,200,6,141,201,6,13
   6,177,204,61,202,6,145,204,200,200,189
   ,202,6,73,255,49,204,3206
OF 26090 DATA 136,136,17,204,145,204,200,
   200,204,199,6,176,7,204,201,6,144,221,
   240,219,189,202,6,49,204,3719
UU 26100 DATA 145,204,136,189,202,6,49,20
   4,145,204,96,138,72,162,4,32,238,6,104
   ,170,189,160,6,24,109,2994
IH 26110 DATA 200,6,168,205,199,6,144,3,1
   72,199,6,189,160,6,56,237,200,6,141,20
   1,6,200,177,204,61,3152
BK 26120 DATA 202,6,145,204,136,136,189,2
   02,6,73,255,49,204,200,200,17,204,145,
   204,136,136,240,5,204,201,3699
CO 26130 DATA 6,176,224,189,202,6,49,204,
   145,204,200,189,202,6,49,204,145,204,9
   6,189,189,6,133,204,24,3445
GY 26140 DATA 216,173,188,6,125,194,6,133
   ,205,169,0,133,77,96,162,0,188,128,6,4
   8,106,185,120,2,41,2707
MS 26150 DATA 8,208,23,189,148,6,221,136,
   6,240,43,169,0,133,77,254,148,6,189,14
   8,6,157,0,208,208,2931
WH 26160 DATA 28,185,120,2,41,4,208,21,16
   9,0,133,77,189,148,6,221,132,6,240,9,2
   22,148,6,189,148,2652
WY 26170 DATA 6,157,0,208,188,128,6,185,1
   20,2,41,2,208,17,189,152,6,221,144,6,2
   40,30,254,152,6,2668
UX 26180 DATA 32,229,6,138,16,21,185,120,
   2,41,1,208,14,189,152,6,221,140,6,240,
   6,222,152,6,32,2385
EX 26190 DATA 226,6,232,224,4,208,140,162
   ,0,189,164,6,240,83,189,168,6,240,50,1
   6,23,222,156,6,222,3182
NF 26200 DATA 156,6,189,156,6,157,4,208,2
   01,47,176,32,169,0,157,164,6,240,53,25
   4,156,6,254,156,6,2959
EL 26210 DATA 189,156,6,157,4,208,201,208
   ,144,9,169,0,157,164,6,240,106,208,196
   ,189,172,6,240,57,16,3208
XO 26220 DATA 23,222,160,6,222,160,6,32,2
   32,6,189,160,6,201,16,176,39,169,0,157
   ,164,6,240,74,254,2920
AI 26230 DATA 160,6,254,160,6,32,235,6,18
   9,160,6,24,216,105,16,205,199,6,176,4,
   41,240,208,7,169,2830
AX 26240 DATA 0,157,164,6,240,42,189,176,
   6,61,0,208,240,13,169,255,157,176,6,15
   7,184,6,169,0,157,2938
NV 26250 DATA 164,6,189,180,6,61,8,208,24
   0,13,169,255,157,180,6,157,184,6,169,0
   ,157,164,6,232,224,3141
KA 26260 DATA 4,208,145,76,98,228,0,759
NF 27000 DATA 0,0,0,0,0,0,0,0,1,3,7,13,15
   ,2,5,10,128,192,224,176,240,64,160,80,
   1,1321
MD 27010 DATA 3,7,13,15,5,8,4,128,192,224
   ,176,240,160,16,32,8,4,15,29,31,23,20,
   2,16,32,1403
PT 27020 DATA 240,184,248,232,40,64,2,20,
   23,29,31,15,4,8,64,40,232,184,248,240,
   32,16,3,15,31,2245
CW 27030 DATA 25,31,6,9,48,192,240,248,15
   2,248,96,144,12,3,15,31,25,31,13,24,12
   ,192,240,248,152,2437
XA 27040 DATA 248,176,24,48,0,9,5,0,12,0,
   5,9,0,32,64,0,96,0,64,32,16,16,56,56,1
   24,1092
UU 27050 DATA 124,198,198,520
```



by TOM HUDSON

With this issue, ANALOG begins a new column. *Boot Camp* will examine assembly language on the ATARI computer systems, while presenting useful subroutines to illustrate the techniques discussed in the column.

The Ground Rules

Before starting to learn assembly language, let's lay down the ground rules.

First, you should have a good reference guide to assembly-language operation codes. I suggest 6502 *Assembly-Language Programming* by Lance Leventhal (OSBORNE/McGraw-Hill). Of course, there are many such books, and the final choice of what book to use is up to you. Just be sure it covers the 6502 operation codes clearly and completely.

Second, since many program concepts will be shown in BASIC, you should have a working knowledge of BASIC. Assembly language requires a solid background in programming logic, and working in BASIC helps develop this skill. In addition, assembly-programming concepts can be grasped more easily if they are first shown in a language the reader is familiar with, such as BASIC. This should not be a problem for most readers, since BASIC is usually the first language learned by personal-computer owners. Therefore, from this point on, I will assume that all readers of this column are fluent in BASIC.

Third, you will need an assembler/editor package. All assembly-language listings in this column will be compatible with the ATARI *Assembler/Editor* cartridge and OSS's *EASMD* and *MAC/65* assemblers. You can use other assemblers, but some code conversion may be necessary.

Fourth, you should be able to read flowcharts. Flowcharts are a good way to visualize a program's operation before actually writing any code.

Numbering Systems

Everybody is familiar with the decimal numbering system. We all use this numbering system in everyday mathematical calculations. The word "decimal" is derived from the Latin *decem*, or ten. Therefore, this numbering system is known as "Base 10", since there are ten digits, 0-9. Let's take a closer look at the decimal numbering system.

Figure 1 shows how a six-digit Base 10 number can be broken down into individual digits. Each digit can range from 0-9 in value.

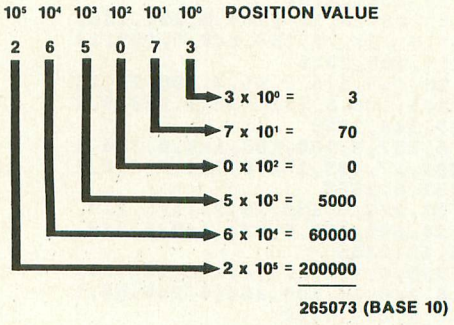


Figure 1.

Above each digit is that digit's position value. The position value is the amount each digit is multiplied by to get the actual value of the digit. You will notice that the position values are shown in powers of 10, since we are working in Base 10. The 1's position is shown as 10 to the 0 power. Any time a number is raised to the 0 power, the result is 1. Therefore, to get the value of the 3 in the last position of the number, we would calculate:

DIGIT x POSITION VALUE = VALUE

In this case the calculation would be:

3 x 1 = 3

We would conclude that the last position in the number would have a value of 3.

The next position, containing the digit 7, has a position value of 10 to the first power, or 10. The calculation of this digit's value would be:

7 x 10 = 70

When we repeat this calculation for each digit in the number and add all the values, we will obtain the value of the number, 265,073 (Base 10).

Here's another concept that we may not think about, but is very interesting. What happens to the number if we shift all the digits to the left, as shown in Figure 2?

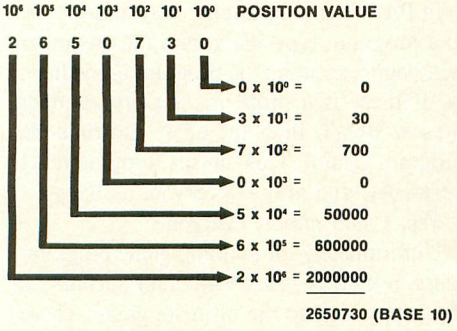


Figure 2.

By looking at the final results, you can see that the number has effectively been multiplied by 10, with a result of 2,650,730 (Base 10)!

Why did this happen? The answer is actually very simple. When each digit is shifted to the left, its position value is increased by a power of 10. The resulting number will be ten times larger than if it were not shifted. Try shifting the number to the right and see what happens.

What Do We Care About All This?

Now that we know exactly how our normal numbering system works, let's apply what we know to a different system, binary.

The word "binary" comes from the Latin *bis*, or "double". As you may know, digital computers work with two electrical states, on and off. This situation is perfectly suited for the binary numbering system, or Base 2.

The binary numbering system uses only two digits, 0 and 1, but the principle of the numbering system is the same as Base 10. Figure 3 shows a number in Base 2 and how it can be converted to Base 10.

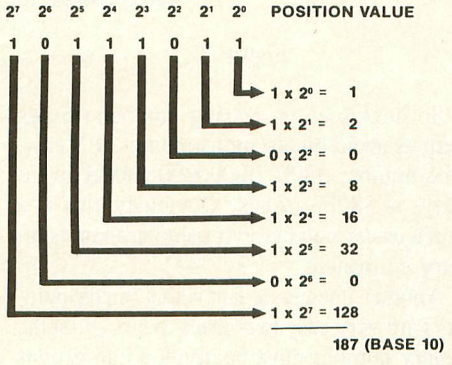


Figure 3.

Once again, the number is shown with the position values above each digit in the number. In Base 2, you will notice that the position values are powers of 2. This means that, unlike the decimal progression of 1, 10, 100, etc. the binary system has a progression of 1, 2, 4, 8 and so on. As a result, the number 10111011 (Base 2) is 187 in Base 10. Figure 4 shows the binary equivalents of the numbers 0-19. Try using the method shown in Figure 3 to convert these numbers to the Base 10 equivalents shown.

BASE 10	BASE 2	BASE 10	BASE 2
0	0	10	1010
1	1	11	1011
2	10	12	1100
3	11	13	1101
4	100	14	1110
5	101	15	1111
6	110	16	10000
7	111	17	10001
8	1000	18	10011
9	1001	19	10011

Figure 4.

Remember how a Base 10 number multiplied by 10 when we shifted it left one digit? Let's look at how a binary number is affected by such a shift. Figure 5a shows the number 7 in binary before the shift and Figure 5b shows the number after the shift.

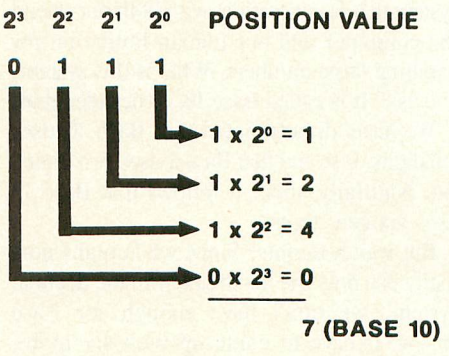


Figure 5a.

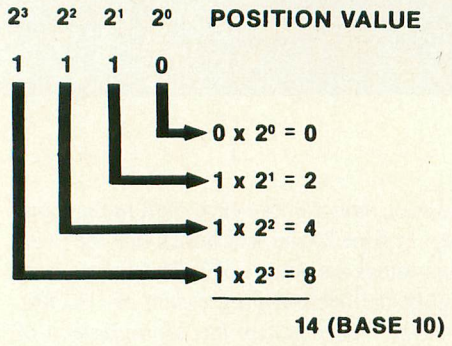


Figure 5b.

The number has been multiplied by 2! By examining this result and the above shift in Base 10, we can see that by shifting the digits in a number left to right, we multiply or divide the number by its base number. This concept will come in very handy in later installments of this column, so keep it in mind.

"Funny" Numbers

The mechanics of the binary numbering system are extremely important, but they can cause some problems.

Let's say you want to look at what is in Memory Location 33011, but don't want to give the number in decimal for some reason. The most logical choice, as far as your computer is concerned, is binary. Unfortunately for us humans, this number comes out as 1000000011110011, and is cumbersome, to say the least. We don't like to handle numbers like this—there are just too many chances to make a mistake.

Fear not! There is yet another numbering system that is compatible with both our friend the computer and our human limitation for handling large numbers. What is this system, you ask? It is called Base 16, or hexadecimal.

We have already noted that Base 10 uses ten digits, 0-9, and that Base 2 uses two digits, 0-1. Naturally, then, it follows that Base 16 uses sixteen digits.

But wait a minute! Since we humans normally use only the ten digits from the decimal system, we don't have enough for Base 16—we'll have to come up with six more. Rather than invent six new digit symbols, we'll use the letters A-F, which are already in existence. Figure 6 shows the 16 digits used in hexadecimal and their decimal equivalents.

BASE 10	BASE 16
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Figure 6.

Once again, the principle of the hexadecimal (hex) numbering system is the same as the other systems we have examined so far; the only difference is that the letters A-F must be thought of as the numbers 10-15. Figure 7 shows the conversion of the hex number $F4BE (all hex numbers should be preceded by a "$") to decimal.

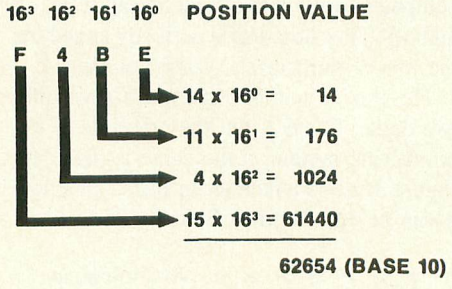


Figure 7.

So how does expressing numbers in hex help us avoid binary monstrosities? It's easy. The number 33011 (1000000011110011 in binary) is $80F3 in hex. Obviously, this is a much easier number to remember than its binary equivalent.

Another interesting fact is that binary numbers are very easy to convert to hex. First the binary number must be divided into groups of four digits, from right to left. Then each group of four digits (ranging in value from 0-15) can easily be converted to the corresponding hex digit 0-F. Figure 8 illustrates this technique.

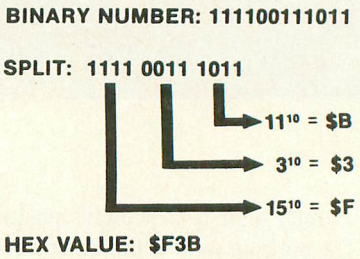


Figure 8.

Bytes and Bits

All readers who have owned their computers for more than a few days have at least heard of the terms "byte" and "bit". Usually this term pops up when the memory capacity of the computer is being discussed.

The byte is the unit most often used when referring to memory size. If your system has "16K" of memory, it has 16 x 1024 bytes, or 16384 bytes total. Each byte is made up of eight bits (short for binary digits). Each bit can be either off or on, corresponding to the 0 and 1 digits in the binary numbering system. With eight digits, this means that each byte can have 2 to the 8th power (or 256) combinations. This is why BASIC limits values in the POKE command to the range of 0-255.

In the process of learning assembly language, we will learn to manipulate the memory of your computer to do the things we want. Study these concepts carefully as they will be used in almost every assembly-language program you write.

How Assembly Works

In BASIC, a programmer can simply type in a program, type RUN, and the computer will begin executing the program immediately. If there is a problem, the programmer presses Break, finds the error and runs the program again. This makes programming very easy, and almost everyone is happy.

Yes, I said *almost* everyone.

Unfortunately for budding game programmers, a kind of brick wall soon appears on the happy road to the ultimate game. These programmers soon find that BASIC is far too slow to handle the complex graphics and

game logic necessary for an arcade-style game. At the very least, assembly-language subroutines are necessary to speed things up.

Why is BASIC so slow? Inside the computer is a device called a 6502 microprocessor. This little chip of silicon is what makes your computer work. It is capable of performing hundreds of thousands of operations per second, and does so every second your computer is powered on!

Sadly, all this computing power is lost as soon as a BASIC cartridge is inserted into your machine. You see, the microprocessor doesn't understand a single word of English, and the BASIC cartridge must act as an interpreter.

All of this interpreting takes time, and instead of doing the work you want, the poor microprocessor winds up spending most of its time translating BASIC into a language it can understand: binary. And this translation doesn't happen just once—it happens every time a BASIC command is executed! What a waste.

Assembly language, on the other hand, uses what is known as an assembler to perform this translation just once. The programmer writes a program in a special format. This is known as the source code. When ready to execute the program, the programmer processes it with an assembler, which translates the source code into object code, which is the actual binary machine-language. This code can be loaded at any time and executed as fast as the computer can go. It only has to be assembled once.

There are a few trade-offs involved when using assembly language, however.

First, the programmer must re-assemble a program each time a change is made. This can take quite a bit of time when a large program is involved. For this reason, it is a good idea to flowchart each program before writing any code. This helps reduce logic errors.

Second, the programmer must know where the program will be located in memory. Since the computer's operating system has certain needs, the programmer must be aware of what memory locations are available.

Third, errors can be hard to find. When a program is executing at hundreds of thousands of operations per second, an error cannot always be easily traced to a certain instruction. For this reason, a good debugging package is a must.

Fourth, all arithmetic must be handled explicitly by the programmer. Assembly language does not have square root, sine or cosine functions. It cannot multiply or even divide! Unless the programmer specifies otherwise, the addition and subtraction instructions can only produce numbers from −128 to 127. In the course of this column, we will examine the arithmetic functions that are possible in assembly language and how they are coded.

This may sound like a lot of limitations, but the 6502 processor allows the programmer to use the computer's built-in operating system directly, which BASIC has a hard time doing. And, of course, assembly language can be thousands of times faster than BASIC, allowing the programmer to write real-time simulations and arcade-style games.

Now that we've laid the groundwork for assembly-language programming, let's look at the 6502 itself.

Chip Off the Old Block

The 6502 processor chip has six registers that we are concerned with. These registers hold specific information and provide work areas for the programmer, and are shown in Figure 9.

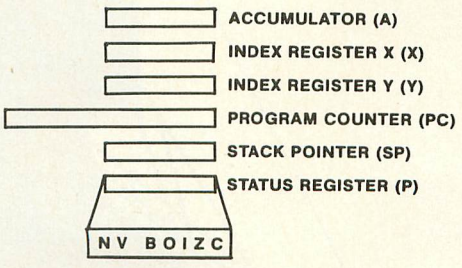


Figure 9.

The accumulator (A) is the most important register as far as the programmer is concerned. It is used for all arithmetic operations and most data manipulation. The accumulator is used more than any other register.

The index registers (X and Y) are used to hold memory indexes, counters, or offsets into tables. They can also be used as temporary storage areas.

The program counter (PC) is used by the 6502 to keep track of what instruction is being executed. This register is 16 bits long, enabling it to point to any byte in memory (up to 65535, or 64K). Since this register is maintained by the 6502, we will not be referencing it very often.

The stack pointer (SP) is used by the 6502 to keep track of a temporary storage region known as the stack. The stack holds subroutine return addresses and other temporary data. Since this registration is maintained by the 6502, we will not be referencing it very often, either.

The processor status register (P) is made up of seven individual "flags", or indicators, which inform the programmer of the 6502's current status.

The sign flat (N) is 0 when the result of an operation is positive, and 1 when the result is negative.

The overflow flag (V) is set to the exclusive-or of Bits 6 and 7 of the result of an arithmetic operation. The exclusive-or will result in a TRUE result if either bit being evaluated is TRUE, but not if both are TRUE. The overflow flag is rarely used, and is not important at this point.

The break flag (B) is set to 1 when a BRK instruction is executed. We will be using the instruction during program testing to stop program execution.

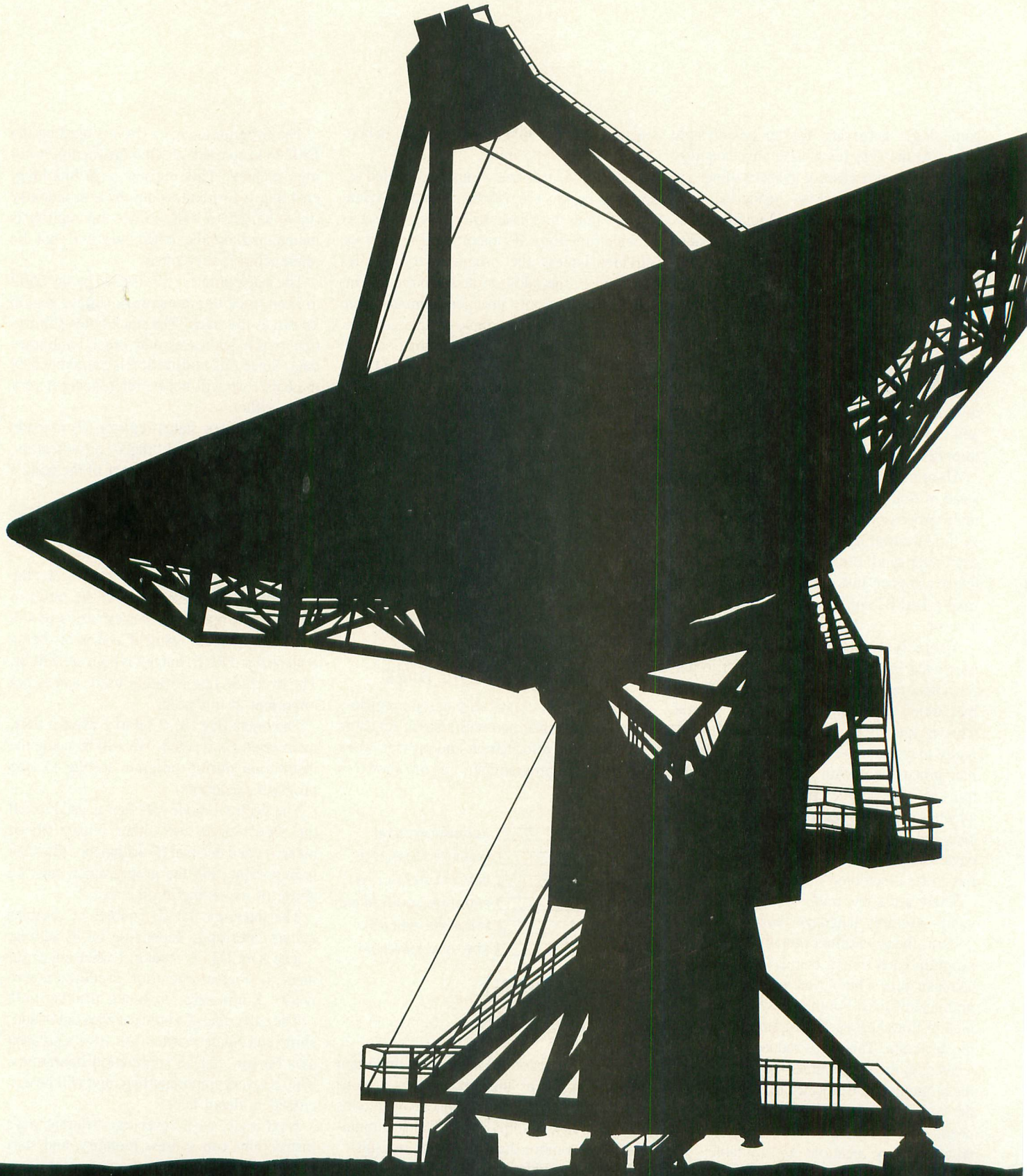
The decimal mode (D) flag is used to tell the processor to use either binary (0) or binary-coded decimal (1) arithmetic. This flag is important, and the programmer must be aware of its setting at all times.

The interrupt flat (I) enables or disables system-interrupts, depending on its setting.

The zero flag (Z) is set to 1 when any arithmetic or logical operation produces a zero result. A non-zero result sets the flag to 0.

The carry flag (C) holds carries out of add, shift, and rotate instructions. It is also used as a borrow flag in subtraction operations. This is a very important flag, and will be discussed in detail later.

Next issue, we'll cover the different ways instructions can address memory, and start studying arithmetic operations, subroutines, and several other areas. Until then, study what has been covered here until you understand it thoroughly.



DATABASE DELPHI

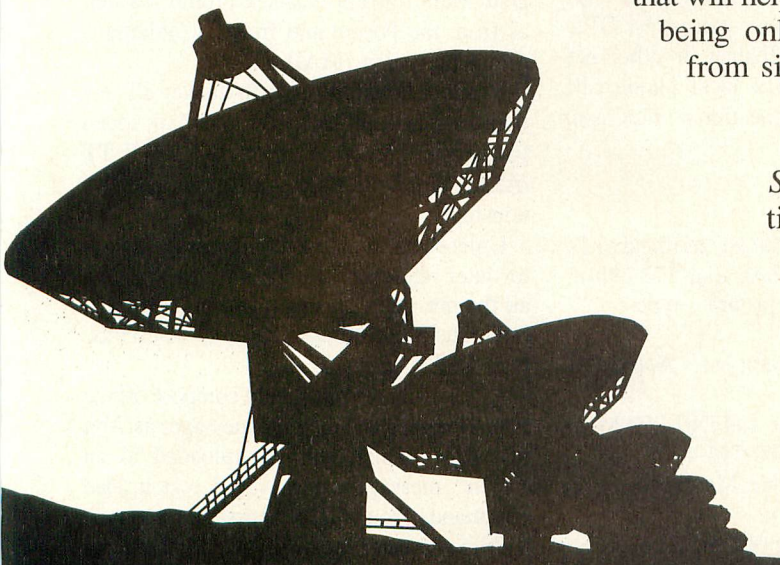
by Michael A. Banks

Saving Time and Money

I've been doing a lot of thinking about money lately, due mainly to the fact that my daughter, Susan, starts college this September. Those of you in similar situations (or attending college yourself) know the feeling. Even with lots of scholarship and grant money, there are expenses. . .

I'm also buying new computer equipment, looking to move to a new home, and doing other things that shouldn't be possible on my budget, but somehow are. But, what the heck, that's part of being American, isn't it?

Thinking about money naturally prompted some thoughts about the cost of being online. Budget-conscious as I am, I couldn't help but be inspired to turn my mind to how Atari SIG members can save money. So, this month I'll pass along some tips that will help you save money and, not incidentally, make being online a more pleasant experience—literally from sign-on to sign-off!



Signing on and off

Setting a default menu: You can eliminate the time spent moving from the DELPHI main menu and the Groups & Clubs menu by setting your default menu to the Atari SIG. The default menu is a setting that causes you to move immediately to a pre-selected menu at sign-on.

With a default menu active, you see no intervening menus or prompts, and you don't have to type any commands. You also get there faster!

To set the default menu to take you directly to the Atari SIG whenever you sign on, type USING SET from the DELPHI main menu (or, GO USING SET from the Atari SIG menu). This takes you through the Using DELPHI menu to the Settings menu, where you should type DEFAULT. You'll be prompted to enter a default menu: type GR AT and you're all set. The next time you sign on (and every time thereafter, unless you change this setting), the first thing you'll see following the DELPHI greeting will be the Atari SIG menu.

(To return to the Atari SIG after setting your default menu, type EXIT twice, then GR AT. Or, you can simply type GO GR AT at the SETTINGS> prompt.)

Signing off: Many DELPHI users think they have to sign off from the DELPHI main menu. This isn't so; you can sign off DELPHI at almost any prompt by typing BYE or, at some few prompts, /BYE.

Type BYE at all major prompts except Mail. (You must exit Mail before you can sign off.) If you're in Conference, type /BYE. You'll be signed off immediately, without having to back through previous menus.

Using the Forum

Capturing messages for offline reading: If you're a frequent Forum user, you know that reading all or most new Forum messages each time you enter the Atari SIG can be quite time-consuming. Fortunately, there's a way to cut the time you spend online viewing new messages by half or more. Simply capture all new Forum messages, read them offline, and then select the messages to which you wish to reply at your leisure. Messages can be made to scroll by far faster than you can read.

The process is easy—and you don't even have to press Return after each message. First, go to the Forum, and open your capture buffer or set a capture-to-disk file. Then, type READ NEW NS at the FORUM> prompt. The "NS" specification in the command displays all new messages nonstop.

If you're not sure you want to see all the new messages, capture a directory for offline browsing with the command DIR NEW NS. (You can stop the nonstop display of messages or a directory with a Control-O.)

Cutting things down to size: Many Atari SIG members find that they're not interested in all the message topics available. If this is the case for you, type TOPIC and "de-select" the topics that don't interest you at the menu that is displayed. (Follow the prompts.) This eliminates all messages under de-selected topics—which means you won't have to view them, even in a directory.

Replying offline: Once you've read captured Forum messages offline, you may find there are some to which you wish to reply. Rather than signing on and typing your replies in, you can compose them offline and insert them into Forum messages by uploading them. (After all, your computer can send text much faster than you can type it online! And you have the added convenience of being able to use the text editor of your choice offline.)

There are a couple of ground rules to follow. First, the replies must be in 7-bit ASCII text. And, your communications software must be able to transmit (upload) the files as ASCII, not binary, text. (And be sure to note the numbers of the messages to which you intend to reply and to give your reply files appropriate names.)

Once you've prepared your replies, sign on to DELPHI, enter the Forum, and read the first message to which you're going to reply. Then type REP, press Return, and instruct your program to upload the reply for that message. After your reply file has been sent, enter a Control-Z. Repeat the procedure with each message for which you've prepared a reply.

This technique can also be used to add new messages. The only differences are that you'll have to type ADD, and enter the message addressee, subject and topic before you can upload your message.

E-Mail

The techniques just described for reading and composing Forum messages can be applied to E-mail, though the commands are somewhat different.

Getting there: You don't have to be at the Atari SIG main menu to get to Mail. You can go to Mail from any database prompt, as well as from the Forum and from a Conference group by typing /MAIL.

Reading E-mail offline: To display all new mail-messages nonstop, enter Mail, open your capture buffer and type EXTRACT TT /ALL. This will display all mail-messages nonstop.

Unless you want to keep a message online for later reference, it's a good idea to delete all the messages once you've captured them (and thus avoid storage charges). To do this, type DELETE /ALL.

Offline replies: You should compose offline E-mail replies and original messages as 7-bit text files, which can up uploaded to an "open" message using ASCII text upload commands. You will, of course, have to type SEND and enter the message addressee and subject; or you can type REPLY after displaying an existing message to open a reply.

This technique can also be used with original messages.

Sending and receiving E-mail via workspace: If you have a really long reply or message to send (1K or larger), it's usually faster and more accurate to upload the file to your workspace using a binary file-transfer protocol such as Xmodem or Kermit, then send it from the MAIL> prompt.

Once a file is in your workspace, you can send it as a mail message by typing SEND <filename.ext> at the MAIL> prompt (where <filename.ext> is the filename and extension of the uploaded file. Note that you must include the " . " even if the file doesn't have an extension.) To send a file in reply to a message that's still in your Mail file, read the message, then type REPLY <filename.ext>.

Sometimes, due to length or other reasons, you may wish to download an E-mail message using a binary protocol. You can't do this at the MAIL> prompt, but you can copy the message to your workspace and download it from there. Read the first screen of the message, then type FILE <filename>. A new file will be created in your workspace; you can then download the file in whatever manner you wish. (Here again, you'll want to delete the original E-mail message. And remember to delete all uploaded and filed/downloaded messages from your workspace once you're done with them.)

Database Timesavers

Search!: The best way to save time in the databases is to plan ahead. If you are looking for certain kinds of files, determine in which database(s) the files are likely to be found, and determine what keywords you'll need to use in search for the files before you sign on.

File-transfer protocols: An important timesaving element is the file-transfer protocol you use. Determine, through research or experimentation, which protocol is fastest for you, then set it as your default file-transfer technique. (Type DOWN MENU at the database ACTION> prompt, and follow the prompts to do this.)

Database shortcuts: As with messages, you can capture database directories and descriptions faster than you can read them.

You can capture an entire database directory by typing DIR NS. Note, however, that this can be a lengthy process, as the Atari SIG databases contain a large number of files. So, you'll probably want to do a search first, and capture only a directory of items that meet your criteria. Once you capture a directory, you can determine at leisure which items you wish to view descriptions of.

Changing database topics: When you change database topics, don't back out to the Atari SIG main menu. Instead, type SET followed by the name of the desired database topic; you'll move immediately to that topic, whether you are at the database name prompt or the ACTION> prompt.

Miscellany

Some DELPHI members are unaware that two of the most commonly used general SIG commands—WHO and SEND—don't have to be issued at the SIG main menu. You can use WHO and SEND at any SIG prompt (even in Conference) by preceding them with a "/" like this: /WHO and /SEND.

/WHOIS <membername> also works at almost all SIG prompts, as do /TIME and, as previously noted, /BYE.

Conscientious use of the tips I've provided here can add up to quite a savings in money, and—equally important—make your time on DELPHI more efficient. I also suggest that you make liberal use of the HELP command (quite often you'll discover information on new features).

If you're relatively new to DELPHI, or haven't been online in a while, you'll find the Guided Tour of benefit. This is the same tour that you were "walked through" the very first time you signed on to DELPHI as a new user. You can take the tour any time you wish by typing USING GUIDE from the DELPHI main menu.

If you don't already have them, I recommend that you obtain the DELPHI Command Card and *DELPHI: The Official Guide*. Type USING MANUAL at the DELPHI main menu for order info.

Finally, read Database DELPHI for more tips, instruction, and the latest information on new DELPHI features.

Weekly Conferences

As always, you are invited to join in the weekly SIG ATARI conferences. Stop by any Tuesday at 10 P.M., EST, to meet other Atari users and the SIG managers. Bring your questions, programming and hacking secrets, and comments to this friendly meeting-place.

And, speaking of conferences, I'll have a few surprises for you next month when I take an in-depth look at using the Atari SIG's Conference area.

In addition to science fiction novels and non-fiction books on model rocketry and other topics, Michael A. Banks is the author of DELPHI: The Official Guide *and* The Modem Reference, *both from Brady Books. You can contact him on DELPHI by sending E-mail to membername KZIN.*

SUPER COMMAND PROCESSOR

by Bryan Schappel

Do you use DOS 2.0 or 2.5? If so, this article is for you. If you tire of waiting for DUP.SYS to load, and hate MEM.SAV, you will enjoy this memory-resident DUP.SYS file, *The BBK Command Processor.*

The idea of a command processor is not new; they have been around for quite some time. Have you ever used UNIX, OSS A+ or SpartaDOS? All these operating systems use a system of input called Command Processing. Put simply, you type in commands, followed by a string of arguments which are then interpreted by the computer. This is what the *BBK Command Processor* does and what is better is that it gives you most of the functions of DUP.SYS (and some DUP never thought of) in less than 2,000 bytes!

Typing It In

Listing 1 is the data used to create the AUTORUN.SYS file of *BBKCP.* Please refer to the *M/L Editor*, found elsewhere in this issue, for typing instructions.

Listing 2 is the MAC/65 source code which need not be typed in to use the program, but is provided so you can see how the program works and so we can refer to it in later discussions.

About the BBKCP

BBKCP has ten internal commands and can have an extensive library of external commands. The built-in commands are given, with syntax, below:

COMMAND	DESCRIPTION
ERASE fname	delete file
PROTECT fname	lock file
UNPROTECT fname	unlock file
RENAME old new	rename file
DIR [Dx:]	disk directory
COPY fn1 fn2	copy file
RUN [hex addr]	run at address
CAR	run cartridge
TYPE fname	type file
KILL	kill *BBKCP*

In the table above, *fname* stands for any legal DOS filename in the format of [Dx:]filename.ext, where *x* is any legal drive number. Anything in brackets is considered optional. For optional arguments, certain default conditions will be used. For example, just giving a filename following a command will tell *BBKCP* that this command is issued to the default device. (The default device is used as the prompt.) So if you typed PROTECT MYFILE.BAS, *BBKCP* would expand this to PROTECT D1:MYFILE.BAS—assuming D1: is the default device.

To change the default device, just enter it on the command line. For example, if the prompt is D1:, and you type D2:, you are now prompted with D2:.

Any legal Atari device may be used for the default device, such as P:, E:, S: and C:. You should remember, though, that most of the commands are used for disk operation and will give errors with another device.

BBKCP uses the Space (ATASCII 32) as a delimiter to separate commands and arguments on the command line. You may put as many spaces between arguments as you like, however there may be *no* spaces before a command.

To enter the *BBKCP* from BASIC or MAC/65 (or any other language cartridge), simply use the DOS command. Any program

currently in memory will remain safe unless you use the COPY command or use an external command. (We'll discuss external commands in a little while.)

About the Commands

You are only required to enter the first three letters of any internal command, but in all examples the entire command name will be used.

ERASE. This command accepts a filename as the argument (wildcards are allowed) and will delete the matching file(s) from your disk. You are not asked for verification, so be careful with this command.

PROTECT. Again only specify the file to be used, and *voila*, it's locked. Use UNPROTECT to reverse the locking process. Wildcards are allowed.

RENAME. This requires two arguments, namely the current name of the file and its new name. Separate the names by at least one space to ensure the rename will function correctly. Wildcards are allowed—but be careful not to rename two files with the same name.

Do not supply a device prefix for the new filename with a rename command, as it will cause unpredictable results. It will not destroy your disk, but it will cause some headaches. It is, however, perfectly legal to put a device prefix on the old filename.

DIR. This command takes an optional argument, the directory specifier. The default specifier is [*].[*], but you may use D2:[*].BAS or anything else you may think of. To get the directory from anything but the default device, just type the device name after the DIR command.

COPY. This command will copy one file to another disk drive or file. Wildcards will produce strange results—*Don't use them*. COPY will make as many passes as needed to duplicate the file. COPY is mainly for a two-drive system; if used with a one-drive system, it can only duplicate the file to the same disk under a separate name. (Note: If you are using one drive with COPY, make certain that you use unique filenames for the source and destination files. Otherwise you could loose the source file forever!)

Another great use for COPY is to copy a text file to the screen or printer. To do this your commands would look like this:

```
COPY MYFILE.TXT E: (screen)
COPY MYFILE.TXT P: (printer)
```

Using the COPY command will destroy any program in memory, so save your work before performing a COPY.

RUN. This command takes an optional hexadecimal address as its parameter and is used to execute a machine-language program in memory. If an address is supplied, RUN will execute at the supplied address. If no address is given, the last run address will be used. At the end of your machine-language routine, execute an RTS instruction to return to *BBKCP*.

RUN (with no argument) may also be used to re-run the last external command, assuming it still resides in memory.

The RUN command checks to see if the address given as an argument contains only legal hexadecimal digits and that the number contains no more than four digits. If these conditions are not met, an ERROR 180 is given, and control passes back to *BBKCP*.

CAR. This command will attempt to pass control to the left cartridge, if it is present. (In the case of an XL or XE, the "left" cartridge is either the built-in BASIC or whatever cartridge is plugged into the computer.) If no cartridge exists, the message "No Cart" will appear, and you are returned to *BBKCP*; otherwise, the cartridge will be entered.

TYPE. This command will print the contents of a text file (the name of which you supply) to the screen, assuming each line of the text has a maximum of 64 characters per line. Mainly this is used to show the contents of a BATCH file. (More on BATCH files later.)

KILL. Use of this command will (1) remove *BBKCP* from memory, (2) wipe out any program in memory and (3) pass control to either a cartridge (if present) or go to DOS.

External Commands

If you enter something that *BBKCP* does not understand, it assumes that it is an external command, and *BBKCP* attempts to binary load the file. If what you typed has no filename extension, *BBKCP* adds on a .COM extension. Executing an external command is destructive to memory, in most cases.

External commands must be binary, load-and-go machine-language files. If the file doesn't begin with a $FF $FF header, an ERROR 181 is given and control passes back to *BBKCP*.

Next month, a library of external commands will be published.

Batch Files

Batch files are a wonderful way of automating certain processes. A batch file is simply a text file full of commands that either *BBKCP* understands or commands that the left cartridge understands. You may use a batch file to perform simple operations like copying a few files or running other external commands.

Use any text editor to make a batch file or simply use the COPY command like this:

```
COPY E1: D1:MYBATCH.BAT
```

When using this command, type your text one line at a time (maximum length is 64 characters), and press Return after each line. When you are finished press Control-3 to terminate the COPY.

To execute a batch file, type a "[*]" before the filename at the prompt. You are also allowed to chain batch files if the last line in the batch file looks like "[*] batfile".

If no file extension is supplied on the filename a .BAT extension is used.

Here is a sample batch file:

```
;BATCH FILE TO GIVE DIRECTORY AND
;RUN THE CART.
;ANY LINE STARTING WITH A ';'
;IS IGNORED BY BBKCP!
;
DIR
CAR
```

Upon power-up, after it has loaded, *BBKCP* will attempt to run the batch file called AUTORUN.BAT from Drive 1. If the file does not exist you are left in *BBKCP*, or control is passed to a cartridge (if present).

The AUTORUN.BAT file is extremely powerful. You could use it to copy files to a RAMdisk or go to the cartridge and run a program or simply print a "hello" message whenever you boot up your system. You can think of this file as giving you an infinite number of AUTORUN.SYS files.

Next Month

Next time around, we will provide some library functions and tell you how to interface into *BBKCP* so you can write your own commands!

By the time you read this, Bryan and Carol will have been happily married for a while. Their new (read: same) apartment contains a new Mega ST2, which shares the computer room with the 800XL. The compu-kids get along very well—if you overlook the constant battle for the printer.

1D7C 7548
23C6 9158 =1610

LISTING 1: M/L EDITOR DATA

FF FF 7C 1D C6 23

```
1000 DATA 255,255,124,29,198,35,160,0,
,132,226,140,224,2,140,225,2,7642
1010 DATA 32,212,31,160,0,32,237,33,16
69,121,160,35,32,163,32,32,2915
1020 DATA 221,30,32,195,34,32,149,31,1
169,4,157,74,3,32,86,228,3431
1030 DATA 16,7,76,165,31,160,181,208,2
249,32,81,30,166,220,232,208,1852
1040 DATA 244,166,221,232,208,239,32,1
158,32,169,118,141,226,2,169,30,8118
1050 DATA 141,227,2,32,81,30,166,220,2
232,208,8,166,221,232,208,3,823
1060 DATA 32,81,30,165,220,133,216,165
5,221,133,217,166,226,208,10,230,4112
1070 DATA 226,141,182,35,165,216,141,1
181,35,32,81,30,165,220,56,229,8435
1080 DATA 216,133,218,165,221,229,217,
,133,219,230,218,208,2,230,219,32,4146
6
1090 DATA 43,30,48,9,32,16,30,76,191,2
29,108,226,2,32,221,30,2625
1100 DATA 173,224,2,170,13,225,2,240,9
9,142,181,35,173,225,2,141,7353
1110 DATA 182,35,108,181,35,165,216,13
33,214,165,217,133,215,162,16,169,1717
7
1120 DATA 7,157,66,3,165,214,157,68,3,
,165,215,157,69,3,165,218,8231
1130 DATA 157,72,3,165,219,157,73,3,76
6,86,228,169,220,133,214,169,1388
1140 DATA 0,133,215,133,219,169,2,133,
,218,208,210,133,212,132,213,160,4077
1150 DATA 0,177,212,240,11,32,119,30,2
230,212,208,243,230,213,208,239,6730
1160 DATA 96,168,173,7,228,72,173,6,22
28,72,152,96,162,0,169,64,6231
1170 DATA 141,190,2,169,0,133,214,133,
,219,169,64,133,218,169,5,133,9097
1180 DATA 215,76,55,30,169,0,133,226,1
133,212,133,213,185,0,5,201,8448
1190 DATA 155,208,8,165,226,240,2,24,9
96,56,96,162,15,221,105,35,5069
1200 DATA 240,5,202,16,248,48,242,6,21
12,38,213,6,212,38,213,6,6672
1210 DATA 212,38,213,6,212,38,213,138,
,5,212,133,212,230,226,200,165,4010
1220 DATA 226,201,5,176,212,144,197,16
62,16,208,2,162,32,138,72,169,7488
1230 DATA 12,157,66,3,32,86,228,104,17
70,96,169,81,160,35,32,97,4793
1240 DATA 30,32,130,30,16,6,32,165,31,
,76,240,30,173,0,5,201,3083
1250 DATA 155,240,231,96,165,6,208,7,1
169,72,160,35,76,97,30,108,4280
1260 DATA 250,191,32,35,31,133,8,32,10
0,31,108,10,0,169,255,133,3420
1270 DATA 10,169,255,133,11,173,194,32
2,133,12,173,195,32,133,13,172,6407
1280 DATA 216,35,169,0,153,26,3,169,22
28,153,27,3,169,255,141,231,9860
1290 DATA 2,169,255,141,232,2,169,0,14
41,217,35,96,169,33,208,40,6709
1300 DATA 169,35,208,36,169,36,208,32,
,32,212,31,164,223,32,237,33,7227
1310 DATA 166,222,169,44,157,186,35,18
85,0,5,157,187,35,232,200,201,169
1320 DATA 155,208,244,169,32,72,208,4,
,72,32,212,31,32,221,30,104,4675
1330 DATA 157,66,3,32,149,31,157,74,3,
,32,86,228,48,17,96,169,3413
1340 DATA 183,157,68,3,169,35,157,69,3
3,169,0,157,75,3,96,132,2929
1350 DATA 212,169,0,133,213,32,221,30,
,32,170,217,32,230,216,160,255,2729
1360 DATA 200,177,243,153,100,35,16,24
48,41,127,153,100,35,169,155,153,8177
1370 DATA 101,35,169,0,153,102,35,169,
,93,160,35,76,97,30,173,82,4344
1380 DATA 35,141,183,35,173,83,35,141,
,184,35,162,0,185,1,5,201,4620
1390 DATA 58,240,24,185,2,5,201,58,208
8,25,185,1,5,141,184,35,4179
1400 DATA 185,0,5,141,183,35,200,200,2
200,208,8,185,0,5,141,183,7590
1410 DATA 35,200,200,185,0,5,157,186,3
35,201,155,240,10,201,32,240,9976
1420 DATA 6,200,232,224,13,208,236,169
9,155,157,186,35,169,0,157,187,710
1430 DATA 35,134,222,96,185,0,5,201,15
55,240,15,32,154,30,176,13,4965
1440 DATA 165,212,141,181,35,165,213,1
141,182,35,108,181,35,160,180,76,8919
1450 DATA 165,31,169,6,208,2,169,4,141
1,112,32,32,212,31,173,186,6361
1460 DATA 35,201,155,208,11,162,3,189,
,86,35,157,186,35,202,16,247,8312
1470 DATA 32,221,30,32,195,34,32,149,3
31,169,6,157,74,3,32,86,1536
1480 DATA 228,48,204,162,16,32,137,30,
,48,27,160,0,185,0,5,201,2693
1490 DATA 155,240,3,200,208,246,169,0,
,153,1,5,169,0,160,5,32,2930
```

```
1500 DATA 97,30,76,121,32,76,221,30,16
69,0,133,8,96,133,212,132,6254
1510 DATA 213,162,255,160,4,232,189,18
86,35,201,46,240,13,201,155,208,2022
1520 DATA 244,177,212,157,186,35,232,1
136,16,247,96,32,255,255,32,35,9213
1530 DATA 31,162,0,32,227,30,32,195,34
4,169,90,141,68,3,169,35,3811
1540 DATA 141,69,3,142,75,3,169,12,141
1,74,3,32,86,228,216,172,6796
1550 DATA 4,228,174,5,228,200,208,1,23
32,140,82,34,142,83,34,160,7232
1560 DATA 0,185,26,3,201,69,240,7,200,
,200,200,192,33,144,242,200,2754
1570 DATA 140,216,35,169,234,153,26,3,
,169,35,153,27,3,162,80,32,3114
1580 DATA 227,30,160,15,185,0,228,153,
,234,35,185,144,3,153,250,35,8862
1590 DATA 136,16,241,169,71,141,238,35
5,169,34,141,239,35,173,217,35,9276
1600 DATA 208,61,238,217,35,165,12,141
1,194,32,165,13,141,195,32,169,7673
1610 DATA 193,133,12,169,32,133,13,165
5,10,141,36,31,165,11,141,40,2472
1620 DATA 31,169,118,133,10,169,33,133
3,11,173,231,2,141,67,31,173,5632
1630 DATA 232,2,141,72,31,169,10,141,2
231,2,169,36,141,232,2,96,5981
1640 DATA 32,228,32,169,255,133,8,216,
,32,240,30,173,0,5,201,42,5608
1650 DATA 208,3,76,8,34,201,59,240,238
8,160,0,32,212,31,173,186,8720
1660 DATA 35,201,155,208,15,173,183,35
5,141,82,35,173,184,35,141,83,6943
1670 DATA 35,76,125,33,169,131,133,212
2,169,35,133,213,162,0,160,0,7018
1680 DATA 185,0,5,209,212,208,26,200,1
192,3,208,244,189,161,35,141,1272
1690 DATA 210,33,189,171,35,141,211,33
3,32,237,33,32,255,255,76,125,9409
1700 DATA 33,165,212,24,105,3,133,212,
,144,2,230,213,232,224,10,208,1997
1710 DATA 205,32,124,29,76,125,33,185,
,0,5,201,32,240,7,201,155,6666
1720 DATA 240,3,200,208,242,185,0,5,20
01,32,208,3,200,208,246,132,1525
1730 DATA 223,96,160,0,169,32,141,0,5,
,32,237,33,32,212,31,169,4570
1740 DATA 126,160,35,32,163,32,169,12,
,32,112,34,169,183,141,254,35,7238
1750 DATA 169,35,141,255,35,169,4,141,
,4,36,169,3,32,112,34,16,818
1760 DATA 10,169,12,132,212,32,112,34,
,32,167,31,76,125,33,169,155,5293
1770 DATA 208,35,173,250,35,16,7,162,0
0,160,1,76,255,255,169,0,7126
1780 DATA 141,2,36,141,3,36,169,7,32,1
112,34,48,234,173,233,35,5944
1790 DATA 72,32,119,30,104,162,0,160,1
1,96,141,252,35,32,137,34,4221
1800 DATA 162,80,32,86,228,141,233,35,
,16,9,152,72,162,80,32,227,6567
1810 DATA 30,104,168,152,72,160,15,185
5,32,0,72,185,218,35,153,32,5496
1820 DATA 0,104,153,218,35,185,144,3,7
72,185,250,35,153,144,3,104,7058
1830 DATA 153,250,35,136,16,225,104,16
68,96,157,66,3,169,0,157,72,5534
1840 DATA 3,157,68,3,169,48,157,73,3,1
157,69,3,96,169,3,157,3752
1850 DATA 66,3,96,32,165,31,76,225,30,
,32,212,31,32,221,30,32,3447
1860 DATA 149,31,32,195,34,169,4,157,7
74,3,32,86,228,48,228,169,7255
1870 DATA 183,160,35,32,97,30,164,223,
,32,237,33,32,212,31,32,225,6878
1880 DATA 30,32,149,31,32,195,34,169,8
8,157,74,3,32,86,228,48,3765
1890 DATA 194,32,158,32,162,16,169,7,3
32,175,34,32,86,228,48,6,2817
1900 DATA 169,1,133,226,208,4,169,0,13
33,226,189,72,3,133,224,189,306
1910 DATA 73,3,133,225,162,32,169,11,3
32,175,34,165,224,157,72,3,6191
1920 DATA 165,225,157,73,3,32,86,228,1
165,226,208,200,32,221,30,76,9540
1930 DATA 225,30,78,111,32,67,97,114,1
116,155,0,155,68,49,58,0,1940
1940 DATA 42,46,42,155,69,58,155,69,11
14,114,111,114,45,32,32,32,1930
1950 DATA 32,155,0,48,49,50,51,52,53,5
54,55,56,57,65,66,67,9809
1960 DATA 68,69,70,155,77,79,67,46,155
5,84,65,66,46,69,82,65,2268
1970 DATA 80,82,79,85,78,80,82,69,78,6
68,73,82,67,79,80,82,2445
1980 DATA 85,78,67,65,82,84,89,80,75,7
73,76,82,86,90,94,72,3024
1990 DATA 207,42,10,76,24,31,31,31,31,
,32,34,32,31,32,31,118,7947
2000 DATA 30,68,49,58,65,85,84,79,82,8
85,78,46,66,65,84,155,3106
2010 DATA 0,10,36,139,36,162,0,142,217
7,35,134,8,232,134,9,32,4394
2020 DATA 199,32,169,83,160,36,32,97,3
30,32,225,30,32,195,34,32,2731
2030 DATA 149,31,169,4,157,74,3,32,86,
,228,8,32,225,30,40,16,1997
2040 DATA 10,165,6,208,3,108,10,0,108,
,250,191,169,81,160,35,32,5894
2050 DATA 97,30,169,186,160,35,32,97,3
30,104,104,76,28,34,125,155,4029
2060 DATA 66,66,75,32,67,80,32,45,32,4
40,67,41,32,49,57,56,8780
2070 DATA 55,32,65,78,65,76,79,71,32,6
67,111,109,112,117,116,105,4599
2080 DATA 110,103,155,98,121,58,32,66,
,114,121,97,110,32,83,99,104,4308
2090 DATA 97,112,112,101,108,155,0,226
6,2,227,2,10,36,0,0,0,9327
```

LISTING 2: ASSEMBLY

```
0100      .OPT NO LIST
0110 ;------------------------------
0120 ;
0130 ;Super Command Processor
0140 ;
0150 ;by: Bryan Schappel
0160 ;
0170 ;If the typed line does not
0180 ;contain a command, then the
0190 ;filename entered will be loaded
0200 ;as a binary file.
0210 ;
0220 ;------------------------------
0230 ;
0240 ;*** Commands ***
0250 ;
0260 ;ERA [fname] = delete file
0270 ;REN [f1,f2] = rename f1 to f2
0280 ;PRO [fname] = lock file
0290 ;UNP [fname] = unlock file
0300 ;DIR [Dn:]   = directory of D#:
0310 ;RUN [adr]   = run at address
0320 ;COPY        = copy a File
0330 ;CAR         = enter Cart
0340 ;TYPE        = read BATCH file
0350 ;KILL        = KILL BBK CP
0360 ;
0370 NUMCOM = 10      ;# OF COMMANDS
```

```
0380 IOCB5 = $0390    ;Adr of IOCB5
0390 MYBUF = $0500    ;I/O Buffer
0400 ZIOCB = $20      ;Zpage IOCB
0410 ;
0420 ;Zero Page Variables
0430 ;
0440     *=  $D6
0450 SL   .DS 1       ;Load adr lo
0460 SH   .DS 1       ;Load adr hi
0470 STL  .DS 1       ;start adr lo
0480 STH  .DS 1       ;start adr hi
0490 BLL  .DS 1       ;length lo
0500 BLH  .DS 1       ;length hi
0510 BAL  .DS 1       ;buffer byte 1
0520 BAH  .DS 1       ;buffer byte 2
0530 XSAV .DS 1       ;save loc
0540 LNPOS .DS 1      ;CP line pos
0550 LENSAV .DS 2     ;COPY length
0560 COUNT .DS 1      ;counter
0570 ;
0580 ;OS Equates
0590 ;
0600 CIOV =  $E456    ;CIO Vector
0610 ICCOM = $0342    ;CIO Command
0620 ICBAL = $0344    ;CIO buffer lo
0630 ICBAH = $0345    ;CIO buffer hi
0640 ICBLL = $0348    ;CIO length lo
0650 ICBLH = $0349    ;CIO length hi
0660 AUX1 =  $034A    ;CIO aux 1
0670 AUX2 =  $034B    ;CIO aux 2
0680 RUNAD = $02E0    ;run address
0690 EOL =   $9B      ;end of line
0700 FR0 =   $D4      ;FP number
0710 TRAMSZ = $06     ;cart in?
0720 BOOT? = $09      ;boot flag
0730 DOSVEC = $0A     ;DOS Vector
0740 DOSINI = $0C     ;init vec
0750 WARMST = $08     ;warm st flag
0760 MEMLO = $02E7    ;lo mem pntr
0770 INITAD = $02E2   ;init addr
0780 HATABS = $031A   ;Handler Table
0790 SHFLOK = $02BE   ;Caps toggle
0800 IFP =   $D9AA    ;Int to FP
0810 FASC =  $D8E6    ;FP to ASC
0820 INBUFF = $F3     ;FP pntr
0830 EDITRV = $E400   ;E: Handler Tab
0840 ;
0850 ;BUMP Macro
0860 ;
0870     .MACRO BUMP
0880     INC %1
0890     BNE @BUMP
0900     INC %1+1
0910 @BUMP
0920     .ENDM
0930 ;
0940 ;PRINT Macro
0950 ;
0960     .MACRO PRINT
0970     LDA # <%1
0980     LDY # >%1
0990     JSR EPRINT
1000     .ENDM
1010 ;
1020 ORIGIN = $1D7C
1030     *=  ORIGIN
1040 ;
1050 ;This routine attempts to
1060 ;Binary Load a file.
1070 ;If the file is not Binary an
1080 ;Error #181 is given.
1090 ;
1100 ;ENTER:
1110 ;MYBUF=Filename to load minus
1120 ;      a .COM extension.
1130 ;      Load is on IOCB #1
1140 ;
1150 LOADIT LDY #0
1160     STY COUNT
1170     STY RUNAD
1180     STY RUNAD+1
1190     JSR FINDFILE
1200     LDY #0
1210     JSR FINDARG
1220     LDA # <COM
1230     LDY # >COM
1240     JSR ADD_EXT
1250     JSR CLOSE1
1260 ;
1270     JSR SET_OPN
1280     JSR SET_DEV
1290     JSR SET_4
1300     BPL LOAD_MAIN
1310 GO_ERR JMP IOERROR
1320 NO_BIN LDY #181
1330     BNE GO_ERR
1340 ;
1350 ;Main load Loop
1360 ;
1370 LOAD_MAIN JSR READ2
1380     LDX BAL
1390     INX
1400     BNE NO_BIN
1410     LDX BAH
1420     INX
1430     BNE NO_BIN
1440     JSR SET_DEST
1450 ;
1460 GET_FIL LDA # <EPRDN
1470     STA INITAD
1480     LDA # >EPRDN
1490     STA INITAD+1
1500     JSR READ2
1510     LDX BAL
1520     INX
1530     BNE NOT_FF
1540     LDX BAH
1550     INX
1560     BNE NOT_FF
1570     JSR READ2
1580 NOT_FF LDA BAL
1590     STA STL
1600     LDA BAH
1610     STA STH
1620     LDX COUNT
1630     BNE PASS2
1640     INC COUNT
1650     STA ADDRESS+1
1660     LDA STL
1670     STA ADDRESS
1680 PASS2 JSR READ2
1690     LDA BAL
1700     SEC
1710     SBC STL
1720     STA BLL
1730     LDA BAH
1740     SBC STH
1750     STA BLH
1760      BUMP  BLL
1770     JSR GET_DATA
1780     BMI JSTART
1790     JSR JINIT
1800     JMP GET_FIL
1810 ;
1820 JINIT JMP (INITAD)
1830 JSTART JSR CLOSE1
1840     LDA RUNAD
1850     TAX
1860     ORA RUNAD+1
1870     BEQ LOAD_GO
1880     STX ADDRESS
1890     LDA RUNAD+1
```

```
1900      STA ADDRESS+1
1910 LOAD_GO JMP (ADDRESS)
1920 ;
1930 ;Get DATA from file
1940 ;
1950 GET_DATA LDA STL
1960      STA SL
1970      LDA STH
1980      STA SH
1990 ;
2000 GET_REC LDX #$10
2010      LDA #7
2020 CALL_CIO STA ICCOM,X
2030      LDA SL
2040      STA ICBAL,X
2050      LDA SH
2060      STA ICBAH,X
2070      LDA BLL
2080      STA ICBLL,X
2090      LDA BLH
2100      STA ICBLH,X
2110      JMP CIOV
2120 ;
2130 ;Read 2 bytes from file
2140 ;
2150 READ2 LDA # <BAL
2160      STA SL
2170      LDA #0
2180      STA SH
2190      STA BLH
2200      LDA #2
2210      STA BLL
2220      BNE GET_REC
2230 ;
2240 ;Set AUX1 to 4
2250 ;
2260 SET_4 LDA #4
2270      STA AUX1,X
2280      JMP CIOV
2290 ;
2300 ;E: Print Routine
2310 ;
2320 ;ENTER:
2330 ;A=LSB of string
2340 ;Y=MSB of string
2350 ;
2360 ;EXIT:
2370 ;A=zero
2380 ;
2390 EPRINT STA FR0
2400      STY FR0+1
2410 EPL LDY #0
2420      LDA (FR0),Y
2430      BEQ EPRDN
2440      JSR EPUT
2450      INC FR0
2460      BNE EPL
2470      INC FR0+1
2480      BNE EPL
2490 EPRDN RTS
2500 ;
2510 ;E: Put Byte Routine
2520 ;
2530 ;ENTER:
2540 ;A=character to print
2550 ;
2560 EPUT TAY
2570      LDA EDITRV+7
2580      PHA
2590      LDA EDITRV+6
2600      PHA
2610      TYA
2620      RTS
2630 ;
2640 ;E: Input Routine
2650 ;
2660 EINPUT LDX #$00
2670      LDA #$40
2680      STA SHFLOK
2690 ;
2700 ;Get a 64 byte line in MYBUF
2710 ;
2720 ;ENTER:
2730 ;X=IOCB Index ($10,$20,...)
2740 ;
2750 INP_MYB LDA #0
2760      STA SL
2770      STA BLH
2780      LDA #64
2790      STA BLL
2800      LDA #5
2810      STA SH
2820      JMP CALL_CIO
2830 ;
2840 ;Get a Hex # from MYBUF
2850 ;
2860 ;ENTER:
2870 ;Y=offset into MYBUF where
2880 ;  the Hex # starts
2890 ;
2900 ;EXIT:
2910 ;FR0,FR0+1 = binary number
2920 ;
2930 ;
2940 GRAB_HEX LDA #0
2950      STA COUNT
2960      STA FR0
2970      STA FR0+1
2980 G4LOOP LDA MYBUF,Y
2990      CMP #EOL
3000      BEQ HEX_OUT
3010      CMP #$20
3020      BNE TESTIT
3030 HEX_OUT LDA COUNT
3040      BEQ HEX_BAD
3050 HEX_GOOD CLC
3060      RTS
3070 HEX_BAD SEC
3080      RTS
3090 ;
3100 TESTIT LDX #$0F
3110 G4SCAN CMP HEXDIG,X
3120      BEQ GOTG4D
3130      DEX
3140      BPL G4SCAN
3150      BMI HEX_BAD
3160 ;
3170 GOTG4D ASL FR0
3180      ROL FR0+1
3190      ASL FR0
3200      ROL FR0+1
3210      ASL FR0
3220      ROL FR0+1
3230      ASL FR0
3240      ROL FR0+1
3250      TXA
3260      ORA FR0
3270      STA FR0
3280      INC COUNT
3290      INY
3300      LDA COUNT
3310      CMP #5
3320      BCS HEX_BAD
3330      BCC G4LOOP
3340 ;
3350 ;Close IOCB #1
3360 ;
3370 CLOSE1 LDX #$10
3380      BNE CLOSEIT
3390 ;
3400 ;Close IOCB #2
3410 ;
```

```
3420 CLOSE2 LDX #$20
3430 ;
3440 ;Close Any IOCB
3450 ;
3460 ;ENTER:
3470 ;X=IOCB Index ($10,$20,...)
3480 ;
3490 ;EXIT:
3500 ;X=IOCB Index
3510 ;
3520 CLOSEIT TXA
3530      PHA
3540      LDA #$0C
3550      STA ICCOM,X
3560      JSR CIOV
3570      PLA
3580      TAX
3590      RTS
3600 ;
3610 ;CP's PROMPT + INPUT Routine
3620 ;
3630 INPUT  PRINT  PROMPT
3640      JSR EINPUT
3650      BPL INPUTLV
3660      JSR IOERROR
3670      JMP INPUT
3680 INPUTLV LDA MYBUF
3690      CMP #EOL
3700      BEQ INPUT
3710      RTS
3720 ;
3730 ;################################
3740 ;
3750 ;Start of Commands
3760 ;
3770 ;################################
3780 ;
3790 ;Enter Cart
3800 ;
3810 GOCART LDA TRAMSZ
3820      BNE TRY_CART
3830      LDA # <NOCART
3840      LDY # >NOCART
3850      JMP EPRINT
3860 TRY_CART JMP ($BFFA)
3870 ;
3880 ;Kill CP
3890 ;
3900 KILL JSR UNHOOK
3910      STA WARMST
3920      JSR GOCART
3930      JMP (DOSVEC)
3940 ;
3950 ;Un-Hook OS Patches
3960 ;
3970 UNHOOK LDA #$FF
3980      STA DOSVEC
3990 CP_HI LDA #$FF
4000      STA DOSVEC+1
4010      LDA RESET+1
4020      STA DOSINI
4030      LDA RESET+2
4040      STA DOSINI+1
4050      LDY INDEX
4060      LDA # <EDITRV
4070      STA HATABS,Y
4080      LDA # >EDITRV
4090      STA HATABS+1,Y
4100 OLD_ML LDA #$FF
4110      STA MEMLO
4120 OLD_MH LDA #$FF
4130      STA MEMLO+1
4140      LDA #0
4150      STA SETFLAG
4160      RTS
4170 ;
4180 ;Erase a File
4190 ;
4200 ERASE LDA #33
4210      BNE DOXIO
4220 ;
4230 ;Lock a File
4240 ;
4250 PROTECT LDA #35
4260      BNE DOXIO
4270 ;
4280 ;Unlock a File
4290 ;
4300 UNPROTECT LDA #36
4310      BNE DOXIO
4320 ;
4330 ;Rename a File
4340 ;
4350 RENAME JSR FINDFILE
4360      LDY LNPOS
4370      JSR FINDARG
4380      LDX XSAV
4390      LDA #',
4400      STA FNAME,X
4410 REN_LP LDA MYBUF,Y
4420      STA FNAME+1,X
4430      INX
4440      INY
4450      CMP #EOL
4460      BNE REN_LP
4470      LDA #32
4480      PHA
4490      BNE XIO_ENT
4500 ;
4510 ;Perform an XIO
4520 ;
4530 ;ENTER:
4540 ;A=XIO Number
4550 ;Y=Offset to Filename in MYBUF
4560 ;
4570 DOXIO PHA
4580      JSR FINDFILE
4590 XIO_ENT JSR CLOSE1
4600      PLA
4610      STA ICCOM,X
4620      JSR SET_DEV
4630      STA AUX1,X
4640      JSR CIOV
4650      BMI IOERROR
4660      RTS
4670 ;
4680 ;Set ICBAL/H to DEVICE
4690 ;
4700 ;ENTER:
4710 ;X=IOCB Index
4720 ;
4730 SET_DEV LDA # <DEVICE
4740      STA ICBAL,X
4750      LDA # >DEVICE
4760      STA ICBAH,X
4770      LDA #0
4780      STA AUX2,X
4790      RTS
4800 ;
4810 ;Handle I/O error
4820 ;
4830 ;ENTER:
4840 ;Y=I/O Error #
4850 ;
4860 ;EXIT:
4870 ;Channel #1 is closed
4880 ;
4890 IOERROR STY FR0
4900 IO_ERR LDA #0
4910      STA FR0+1
4920      JSR CLOSE1
4930      JSR IFP
```

```
4940      JSR FASC
4950      LDY #$FF
4960 IOERR INY
4970      LDA (INBUFF),Y
4980      STA ERBUF,Y
4990      BPL IOERR
5000      AND #$7F
5010      STA ERBUF,Y
5020      LDA #EOL
5030      STA ERBUF+1,Y
5040      LDA #0
5050      STA ERBUF+2,Y
5060      LDA # <ERTXT
5070      LDY # >ERTXT
5080      JMP EPRINT
5090 ;
5100 ;This will grab a filename from
5110 ;MYBUF and put it in the
5120 ;device buffer.
5130 ;
5140 ;ENTER:
5150 ;Y=Offset to Filename in MYBUF
5160 ;
5170 FINDFILE LDA DEFDEV
5180      STA DEVICE
5190      LDA DEFDEV+1
5200      STA DEVICE+1
5210 ;
5220 FIND1 LDX #0
5230      LDA MYBUF+1,Y
5240      CMP #':
5250      BEQ X2
5260      LDA MYBUF+2,Y
5270      CMP #':
5280      BNE FIND2
5290      LDA MYBUF+1,Y
5300      STA DEVICE+1
5310      LDA MYBUF,Y
5320      STA DEVICE
5330      INY
5340      INY
5350      INY
5360      BNE FIND2
5370 ;
5380 X2   LDA MYBUF,Y
5390      STA DEVICE
5400      INY
5410      INY
5420 ;
5430 FIND2 LDA MYBUF,Y
5440      STA FNAME,X
5450      CMP #EOL
5460      BEQ FIND3
5470      CMP #$20
5480      BEQ FIND3
5490      INY
5500      INX
5510      CPX #13
5520      BNE FIND2
5530 FIND3 LDA #EOL
5540      STA FNAME,X
5550      LDA #0
5560      STA FNAME+1,X
5570      STX XSAV
5580      RTS
5590 ;
5600 ;This handles the 'RUN' command
5610 ;
5620 MEMRUN LDA MYBUF,Y
5630      CMP #EOL
5640      BEQ RUN_IT
5650 ;
5660 PULL_IT JSR GRAB_HEX
5670      BCS BAD_RUN
5680      LDA FR0
5690      STA ADDRESS
5700      LDA FR0+1
5710      STA ADDRESS+1
5720 RUN_IT JMP (ADDRESS)
5730 BAD_RUN LDY #180
5740 MEM_ERR JMP IOERROR
5750 ;
5760 ;Get disk Directory
5770 ;
5780 DIRECT LDA #6
5790      BNE DIR_GO
5800 ;
5810 ;Get a TYPE of a File
5820 ;
5830 TYPE LDA #4
5840 DIR_GO STA DIR_COM+1
5850      JSR FINDFILE
5860      LDA DEVICE+3
5870      CMP #EOL
5880      BNE OPEN_DIR
5890      LDX #3
5900 CP_DIR LDA DIRNAM,X
5910      STA DEVICE+3,X
5920      DEX
5930      BPL CP_DIR
5940 ;
5950 OPEN_DIR JSR CLOSE1
5960      JSR SET_OPN
5970      JSR SET_DEV
5980 DIR_COM LDA #6
5990      STA AUX1,X
6000      JSR CIOV
6010      BMI MEM_ERR
6020 ;
6030 DIROK LDX #$10
6040      JSR INP_MYB
6050      BMI DIRDONE
6060      LDY #0
6070 DIRCK1 LDA MYBUF,Y
6080      CMP #EOL
6090      BEQ DIRCK2
6100      INY
6110      BNE DIRCK1
6120 DIRCK2 LDA #0
6130      STA MYBUF+1,Y
6140       PRINT  MYBUF
6150      JMP DIROK
6160 DIRDONE JMP CLOSE1
6170 ;
6180 ;Set Destroy Flag
6190 ;
6200 SET_DEST LDA #0
6210      STA WARMST
6220      RTS
6230 ;
6240 ;Add File Extension
6250 ;
6260 ;ENTER:
6270 ;A=LSB of extension
6280 ;Y=MSB of extension
6290 ;
6300 ADD_EXT STA FR0
6310      STY FR0+1
6320      LDX #$FF
6330      LDY #4
6340 ;
6350 EXT_SCAN INX
6360      LDA FNAME,X
6370      CMP #'.
6380      BEQ EXT_RTS
6390      CMP #EOL
6400      BNE EXT_SCAN
6410 EXT_ADD LDA (FR0),Y
6420      STA FNAME,X
6430      INX
6440      DEY
6450      BPL EXT_ADD
```

```
6460 EXT_RTS RTS
6470 ;
6480 ;This is where RESET comes
6490 ;
6500 RESET JSR $FFFF
6510      JSR UNHOOK
6520 ;
6530 HOOKUP LDX #$00
6540      JSR CLOSEIT
6550      JSR SET_OPN
6560      LDA # <EDEV
6570      STA ICBAL
6580      LDA # >EDEV
6590      STA ICBAH
6600      STX AUX2
6610      LDA #12
6620      STA AUX1
6630      JSR CIOV
6640 ;
6650 HOOK1 CLD
6660      LDY EDITRV+4
6670      LDX EDITRV+5
6680      INY
6690      BNE NO_UPX
6700      INX
6710 NO_UPX STY EGET+1
6720      STX EGET+2
6730      LDY #0
6740 ;
6750 FIND_E LDA HATABS,Y
6760      CMP #'E
6770      BEQ GOT_E
6780      INY
6790      INY
6800      INY
6810      CPY #33
6820      BCC FIND_E
6830 ;
6840 GOT_E INY
6850      STY INDEX
6860      LDA # <NEW_E.HAN
6870      STA HATABS,Y
6880      LDA # >NEW_E.HAN
6890      STA HATABS+1,Y
6900 ;
6910      LDX #$50
6920      JSR CLOSEIT
6930      LDY #$0F
6940 COPY_E LDA EDITRV,Y
6950      STA NEW_E.HAN,Y
6960      LDA IOCB5,Y
6970      STA BAT_IOCB,Y
6980      DEY
6990      BPL COPY_E
7000 ;
7010      LDA # <[BAT_GET-1]
7020      STA NEW_E.HAN+4
7030      LDA # >[BAT_GET-1]
7040      STA NEW_E.HAN+5
7050 ;
7060 SKIP_BAT LDA SETFLAG
7070      BNE INIT_RTS
7080      INC SETFLAG
7090      LDA DOSINI
7100      STA RESET+1
7110      LDA DOSINI+1
7120      STA RESET+2
7130      LDA # <RESET
7140      STA DOSINI
7150      LDA # >RESET
7160      STA DOSINI+1
7170 ;
7180      LDA DOSVEC
7190      STA UNHOOK+1
7200      LDA DOSVEC+1
7210      STA CP_HI+1
7220      LDA # <GO_CP
7230      STA DOSVEC
7240      LDA # >GO_CP
7250      STA DOSVEC+1
7260 ;
7270      LDA MEMLO
7280      STA OLD_ML+1
7290      LDA MEMLO+1
7300      STA OLD_MH+1
7310      LDA # <PROGEND
7320      STA MEMLO
7330      LDA # >PROGEND
7340      STA MEMLO+1
7350 INIT_RTS RTS
7360 ;
7370 ;COMMAND PROCESSOR ENTRY
7380 ;
7390 GO_CP JSR HOOK1
7400      LDA #$FF
7410      STA WARMST
7420 ;
7430 MAIN CLD
7440      JSR INPUT    ;get line
7450      LDA MYBUF
7460      CMP #'*      ;Batch?
7470      BNE NO_BAT
7480      JMP BATCH
7490 ;
7500 NO_BAT CMP #';
7510      BEQ MAIN
7520      LDY #0
7530      JSR FINDFILE
7540      LDA DEVICE+3
7550      CMP #EOL
7560      BNE NO_REM
7570      LDA DEVICE
7580      STA DEFDEV
7590      LDA DEVICE+1
7600      STA DEFDEV+1
7610      JMP MAIN
7620 ;
7630 NO_REM LDA # <COMTAB
7640      STA FR0
7650      LDA # >COMTAB
7660      STA FR0+1
7670      LDX #0
7680 COMLP LDY #0
7690 COMCK LDA MYBUF,Y
7700      CMP (FR0),Y
7710      BNE TRYNEXT
7720      INY
7730      CPY #3
7740      BNE COMCK
7750 ;
7760      LDA COMADRL,X
7770      STA COMJSR+1
7780      LDA COMADRH,X
7790      STA COMJSR+2
7800      JSR FINDARG
7810 COMJSR JSR $FFFF
7820      JMP MAIN
7830 ;
7840 TRYNEXT LDA FR0
7850      CLC
7860      ADC #3
7870      STA FR0
7880      BCC TRY_NH
7890      INC FR0+1
7900 TRY_NH INX
7910      CPX #NUMCOM
7920      BNE COMLP
7930      JSR LOADIT
7940      JMP MAIN
7950 ;
7960 ;Find an Argument on Line
7970 ;
```

```
7980 ;ENTER:
7990 ;Y=Offset to start search
8000 ;
8010 ;EXIT:
8020 ;Y=1st char of argument
8030 ;
8040 FINDARG LDA MYBUF,Y
8050      CMP #$20
8060      BEQ NEXTARG
8070      CMP #EOL
8080      BEQ NEXTARG
8090      INY
8100      BNE FINDARG
8110 NEXTARG LDA MYBUF,Y
8120      CMP #$20
8130      BNE FOUNDARG
8140      INY
8150      BNE NEXTARG
8160 FOUNDARG STY LNPOS
8170      RTS
8180 ;
8190 ;Handle a BATCH File
8200 ;
8210 BATCH LDY #0
8220      LDA #$20
8230      STA MYBUF
8240      JSR FINDARG
8250      JSR FINDFILE
8260      LDA # <BAT
8270      LDY # >BAT
8280      JSR ADD_EXT
8290 BAT_GO LDA #12
8300      JSR BAT_CIO
8310 ;
8320      LDA # <DEVICE
8330      STA ICBALB
8340      LDA # >DEVICE
8350      STA ICBAHB
8360      LDA #4
8370      STA AUX1B
8380      LDA #3
8390      JSR BAT_CIO
8400      BPL BAT_MAIN
8410      LDA #12
8420      STY FR0
8430      JSR BAT_CIO
8440      JSR IO_ERR
8450 BAT_MAIN JMP MAIN
8460 BAT_9B LDA #EOL
8470      BNE BAT_XIT
8480 ;
8490 BAT_GET LDA BAT_IOCB
8500      BPL BAT_PROC
8510 BAT_NORM LDX #$00
8520      LDY #1
8530 EGET JMP $FFFF
8540 ;
8550 BAT_PROC LDA #0
8560      STA ICBLLB
8570      STA ICBLHB
8580      LDA #7
8590      JSR BAT_CIO
8600      BMI BAT_NORM
8610      LDA BAT_ZIO+15
8620      PHA
8630      JSR EPUT
8640      PLA
8650 BAT_XIT LDX #$00
8660      LDY #1
8670      RTS
8680 ;
8690 ;Perform BATCH CIO
8700 ;
8710 BAT_CIO STA ICCOMB
8720      JSR BAT_SWAP
8730      LDX #$50
8740      JSR CIOV
8750      STA BAT_ZIO+15
8760      BPL BAT_SWAP
8770      TYA
8780      PHA
8790      LDX #$50
8800      JSR CLOSEIT
8810      PLA
8820      TAY
8830 ;
8840 ;Swap IOCB Blocks
8850 ;
8860 BAT_SWAP TYA
8870      PHA
8880      LDY #$0F
8890 BAT_SLP LDA ZIOCB,Y
8900      PHA
8910      LDA BAT_ZIO,Y
8920      STA ZIOCB,Y
8930      PLA
8940      STA BAT_ZIO,Y
8950      LDA IOCB5,Y
8960      PHA
8970      LDA BAT_IOCB,Y
8980      STA IOCB5,Y
8990      PLA
9000      STA BAT_IOCB,Y
9010      DEY
9020      BPL BAT_SLP
9030      PLA
9040      TAY
9050      RTS
9060 ;
9070 ;Set COPY IOCB
9080 ;
9090 ;ENTER:
9100 ;X=IOCB Index
9110 ;A=CIO Command
9120 ;
9130 ;EXIT:
9140 ;CIO set for I/O of 12K block
9150 ;at $3000
9160 ;
9170 SET_CPY STA ICCOM,X
9180      LDA #0
9190      STA ICBLL,X
9200      STA ICBAL,X
9210      LDA #$30
9220      STA ICBLH,X
9230      STA ICBAH,X
9240      RTS
9250 ;
9260 ;Set ICCOM for Open
9270 ;
9280 ;ENTER:
9290 ;X=IOCB Index
9300 ;
9310 SET_OPN LDA #3
9320      STA ICCOM,X
9330      RTS
9340 ;
9350 ;Copy I/O Error
9360 ;
9370 ;ENTER:
9380 ;Y=CIO Error #
9390 ;
9400 ;EXIT:
9410 ;Channels #1 and #2 are closed
9420 ;
9430 CPY_IOR JSR IOERROR
9440      JMP CLOSE2
9450 ;
9460 ;Handle COPY Verb
9470 ;
9480 COPY JSR FINDFILE
9490      JSR CLOSE1
9500      JSR SET_DEV
9510      JSR SET_OPN
```

```
9520      JSR SET_4
9530      BMI CPY_IOR
9540       PRINT  DEVICE
9550      LDY LNPOS
9560      JSR FINDARG
9570      JSR FINDFILE
9580      JSR CLOSE2
9590      JSR SET_DEV
9600      JSR SET_OPN
9610      LDA #8
9620      STA AUX1,X
9630      JSR CIOV
9640      BMI CPY_IOR
9650 ;
9660      JSR SET_DEST
9670 COPY.2 LDX #$10
9680      LDA #7
9690      JSR SET_CPY
9700      JSR CIOV
9710      BMI EOF
9720      LDA #1
9730      STA COUNT
9740      BNE SAVLEN
9750 ;
9760 EOF LDA #0
9770      STA COUNT
9780 ;
9790 SAVLEN LDA ICBLL,X
9800      STA LENSAV
9810      LDA ICBLH,X
9820      STA LENSAV+1
9830      LDX #$20
9840      LDA #11
9850      JSR SET_CPY
9860      LDA LENSAV
9870      STA ICBLL,X
9880      LDA LENSAV+1
9890      STA ICBLH,X
9900      JSR CIOV
9910      LDA COUNT
9920      BNE COPY.2
9930      JSR CLOSE1
9940      JMP CLOSE2
9950 ;
9960 ;Program Text/Buffers
9970 ;
9980 NOCART .BYTE "No Cart",EOL,0
9990 PROMPT .BYTE EOL
010000 DEFDEV .BYTE "D1:",0
010010 DIRNAM .BYTE "*.*",EOL
010020 EDEV .BYTE "E:",EOL
010030 ERTXT .BYTE "Error- "
010040 ERBUF .BYTE "   ",EOL,0
010050 HEXDIG .BYTE "0123456789"
010060    .BYTE "ABCDEF"
010070 COM .BYTE EOL,"MOC."
010080 BAT .BYTE EOL,"TAB."
010090 ;
010100 ;Command Tables
010110 ;
010120 COMTAB .BYTE "ERAPROUNPREN"
010130    .BYTE "DIRCOPRUN"
010140    .BYTE "CARTYPKIL"
010150 ;
010160 COMADRL .BYTE  <ERASE
010170    .BYTE  <PROTECT
010180    .BYTE  <UNPROTECT
010190    .BYTE  <RENAME
010200    .BYTE  <DIRECT
010210    .BYTE  <COPY
010220    .BYTE  <MEMRUN
010230    .BYTE  <GOCART
010240    .BYTE  <TYPE
010250    .BYTE  <KILL
010260 ;
010270 COMADRH .BYTE  >ERASE
010280    .BYTE  >PROTECT
010290    .BYTE  >UNPROTECT
010300    .BYTE  >RENAME
010310    .BYTE  >DIRECT
010320    .BYTE  >COPY
010330    .BYTE  >MEMRUN
010340    .BYTE  >GOCART
010350    .BYTE  >TYPE
010360    .BYTE  >KILL
010370 ;
010380 ADDRESS .WORD EPRDN ;RUN Addr
010390 DEVICE .BYTE "D1:"
010400 ;
010410 ;Buffers
010420 ;
010430 FNAME .BYTE "AUTORUN.BAT"
010440    .BYTE EOL,0
010450    .DS 17
010460 INDEX .DS 1   ;HATABS Index
010470 SETFLAG .DS 1 ;set up?
010480 ;
010490 ;Wedge/Handler Space
010500 ;
010510 ;For Batch Processing
010520 ;
010530 BAT_ZIO .DS 16 ;Zpage IOCB
010540 NEW_E.HAN .DS 16 ;E: Handler
010550 BAT_IOCB .DS 16 ;IOCB #5 Copy
010560 ICCOMB = BAT_IOCB+2
010570 ICBALB = BAT_IOCB+4
010580 ICBAHB = BAT_IOCB+5
010590 ICBLLB = BAT_IOCB+8
010600 ICBLHB = BAT_IOCB+9
010610 AUX1B = BAT_IOCB+10
010620 ;
010630 PROGEND = *
010640 T_LENGTH = PROGEND-ORIGIN
010650 ;
010660 ;Entry Point to Start CP
010670 ;
010680 ENTRY LDX #0
010690    STX SETFLAG
010700    STX WARMST
010710    INX
010720    STX BOOT?
010730    JSR HOOKUP
010740     PRINT  CREDITS
010750    JSR CLOSE2
010760    JSR SET_OPN
010770    JSR SET_DEV
010780    JSR SET_4
010790    PHP
010800    JSR CLOSE2
010810    PLP
010820    BPL HAV_BAT
010830    LDA TRAMSZ
010840    BNE G_CART
010850    JMP (DOSVEC)
010860 G_CART JMP ($BFFA)
010870 HAV_BAT  PRINT  PROMPT
010880     PRINT  FNAME
010890    PLA
010900    PLA
010910    JMP BAT_GO
010920 ;
010930 CREDITS .BYTE $7D,EOL
010940    .BYTE "BBK CP - (C) 1987 "
010950    .BYTE "ANALOG Computing",EOL
010960    .BYTE "by: Bryan Schappel"
010970    .BYTE EOL,0
010980 ;
010990 ;Set Run Address
011000 ;
011010    *=  RUNAD
011020    .WORD ENTRY
011030    .OPT LIST
011040    .END
```

B&C ComputerVisions

3257 Kifer Road
Santa Clara, CA 95051
(408) 749-1003

STORE HOURS
TUE - FRI 10am - 6pm
SAT - 10am - 5pm
CLOSED SUN - MON

800/XL/XE SOFTWARE
ALL TITLES ON DISK

ENTERTAINMENT

12 ADAMS ADVENTURES ..	14.95
ALIANTS	26.95
ALT. REALITY CITY.....	26.95
ALT. REALITY DUNGEON.	26.95
BEYOND CASTLE WOLF....	14.95
BISMARK	26.95
BOP & WRESTLE (64K)...	26.95
BORDINO:1812	26.95
BOULDERDASH CONSTR.SET	17.95
BRUCE LEE	17.95
CASTLE WOLFENSTEIN....	14.95
DALLAS QUEST	7.95
D-BUG	7.95
F-15 STRIKE EAGLE	31.50
FIGHT NIGHT	17.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS.....	22.50
GUNSLINGER	26.95
HARD HAT MAC	7.95
JAWBREAKER	9.95
KARATEKA	13.50
KNICKERBOCKERS	13.50
KORONIS RIFT	13.50
LAST V-8	8.95
LEADERBOARD	13.50
MAIL ORDER MONSTERS ..	13.50
MICROLEAGUE BASEBALL..	35.95
MONTEZUMA'S REVENGE...	14.95
MOUSEQUEST.....	17.95
MOON SHUTTLE	7.95
NINJA	8.95
OIL'S WELL	9.95
O'RILEY'S MINE	9.95
PIRATES OF BARB. COAST	22.50
PREPPIE I & II	9.95
RESCUE ON FRACTALAS...	13.50
SILENT SERVICE	31.50
SPEEDKING	8.95
SPIDERMAN	7.95
SPITFIRE 40	31.50
STARFLEET I	44.95
SPY VS. SPY III	17.95
STOCKMARKET	22.50
STRIP POKER	26.95
SUMMER GAMES	17.95
TAX DODGE	9.95
THE HULK	7.95
TOMAHAWK (64K)	26.95
TOP GUNNER	17.95
TOUCHDOWN FOOTBALL ...	13.50
TRAILBLAZER	26.95
UNIVERSE	44.95
ZAXXON	13.50

PROGRAMMING

ACTION!	71.95
ACTION! TOOLKIT	26.95
BASIC XL	53.95
BASIC XL TOOLKIT	26.95
BASIC XE	71.95
DISK I/O	26.95
DRAPER PASCAL	44.95
KYAN PASCAL	62.95
LIGHTSPEED C	35.95
LOGO	29.95
MAC/65	71.95
MAC/65 TOOLKIT	26.95
MACRO ASSEMBLER	22.50
PILOT	19.95
SPARTA DOS X	71.95
TOP DOS 1.5 PLUS	35.95

PRODUCTIVITY

ANIMATION STATION	89.95
ATARIWRITER+.	39.95
ATARI BOOKKEEPER	24.95
ATARI MUSIC II	14.95
AWARDWARE (1050)	13.50
BANK STREET WRITER....	14.95
BLAZING PADDLES	31.50
CELEBRITY COOKBOOK ...	26.95
COMPUTE YOUR ROOTS ...	35.95
DATAMANAGER	17.95
FAMILY FINANCE	6.95
GUITAR WIZARD	26.95
HOME ACCOUNTANT	19.95
HOME FILING MANAGER. .	6.95
HOMEPAK	24.95
INVENTORY MASTER	80.95
LETTER WIZARD	29.95
MUSIC CONSTRUCTION SET	13.50
NEWSROOM (1050 - 64K).	13.50
NEWS STATION	26.95
NEWS STA. COMPANION. .	26.95
PAGE DESIGNER	26.95
PRINT POWER (1050)....	13.50
PRINTKIT (1050)	13.50
PRINTSHOP	34.95
P.S. COMPANION (64K) .	24.95
P.S.GRAPHICS LIBRARY 1	17.95
P.S.GRAPHICS LIBRARY 2	17.95
P.S.GRAPHICS LIBRARY 3	17.95
PROOF READER	17.95
PUBLISHING PRO	35.95
RUBBER STAMP	26.95
SYNTREND	14.95
SUPER MAILER	35.95
THE LOTTO PROGRAM	17.95
TIMEWISE	6.95
TURBOWORD/80 COLUMN REQUIRES XEP80.....	44.95
VIDEO TITLESHOP (64K).	26.95
GRAPHICS COMPANION.	17.95
VIRTUOSO	29.95
VISICALC	24.95

EDUCATION

ATARI LIGHT MODULE (REQ.ATARILAB STARTER)	9.95
BUZZWORD	35.95
GRANDMA'S HOUSE (-10)	9.95
HEY DIDDLE (AGE 3-10).	9.95
MASTER TYPE	14.95
PLANATARIUM	22.50
STATES AND CAPITALS ..	9.95
TOUCH TYPING	9.95
CBS (AGE 3-6):	
ASTROGROVER	8.95
BIG BIRD SPEC DELIVE	8.95
ERNIE'S MAGIC SHAPE.	8.95
DESIGNWARE:	
MATHMAZE (6-11)	35.95
MISSION ALGEBRA (13+)	35.95
SPELLICOPTER (6-11).	35.95
TINK TONK (AGE 4-6):	
ABC'S	8.95
COUNT AND ADD	8.95
SMART THINKER	8.95
SPELLING	8.95
SUBTRACTION	8.95
THINKING SKILLS	8.95
ALL 6 TINK TONKS..	39.95
UNICORN:	
10 LITTLE ROBOTS (PRE-SCHOOL)	26.95
FUN BUNCH (6-ADULT)	26.95
RACECAR RITHMETIC (AGE 6+)	26.95
WEEKLY READER (PRE-SCHOOL):	
STICKY BEAR SHAPES .	26.95
STICKY BEAR NUMBERS.	26.95
STICKY BEAR ABC'S ..	26.96
STICKY BEAR OPPOSITE	26.95

800/XL/XE SOFTWARE
ALL TITLES ON CARTRIDGE

ENTERTAINMENT

ALIEN AMBUSH	9.95
ACE OF ACES .(XL/XE) .	24.95
ARCHON	19.95
ASTEROIDS	15.95
ATARI TENNIS	9.95
BALL BLAZER	19.95
BARNYARD BLASTER	24.95*
BATTLEZONE	19.95
BLUE MAX	19.95
CAVERNS OF MARS	14.95
CENTIPEDE	14.95
CHICKEN	9.95
CHOPLIFTER	14.95
CLAIM JUMPER	9.95
CLOUDBURST	9.95
CRIME BUSTER.....	24.95*
CROSSBOW	24.95*
CROSSFIRE	9.95
DAVIDS MIDNIGHT MAGIC	19.95
DEFENDER	14.95
DELUXE INVADERS	7.95
DESERT FALCON	19.95
DIG DUG	19.95
DONKEY KONG	5.00
DONKEY KONG JR.	19.95
EASTERN FRONT (1941).	19.95
E.T. PHONE HOME	9.95
FIGHT NIGHT	19.95
FINAL LEGACY	19.95
FOOD FIGHT (XL/XE)...	19.95
FOOTBALL	14.95
FROGGER	14.95
GALAXIAN	19.95
GATO	24.95
GORF (400/800)	5.00
GYRUSS	14.95
HARDBALL	19.95
INTO THE EAGLES NEST	19.95
JOURNEY TO PLANETS ..	9.95
JOUST	19.95
JUNGLE HUNT	19.95
KABOOM!	14.95
LODE RUNNER	24.95
MARIO BROS.	19.95
MILLIPEDE	19.95
MISSILE COMMAND	5.00
MOON PATROL	19.95
MR. COOL	9.95
MS. PAC MAN	19.95
NECROMANCER	19.95
ONE ON ONE (XL/XE)...	19.95
PAC MAN	5.00
PENGO	19.95
POLE POSITION	19.95
POPEYE	14.95
Q-BERT	14.95
QIX	14.95
RESCUE ON FRACTALAS .	19.95
RETURN OF THE JEDI ..	14.95
ROBOTRON:2084	19.95
SKY WRITER	14.95
SLIME (400/800).....	9.95
SPACE INVADERS	14.95
STAR RAIDERS	5.00
STAR RAIDERS II	19.95
SUPER BREAKOUT	9.95
TRACK & FIELD	24.95
TURMOIL	9.95
WIZARD OF WOR.....	5.00

* REQUIRES LIGHT GUN

PRODUCTIVITY

ATARIWRITER	19.95
MICROFILER	22.50

EDUCATION

ATARILAB STARTEER SET	29.95
MATH ENCOUNTERS	9.95
FISHER PRICE (PRE SCHOOL):	
DANCE FANTASY	8.95
LINKING LOGIC	8.95
LOGIC LEVELS	8.95
MEMORY MANOR	8.95
SPINNAKER (AGE 3-10):	
ALF IN COLOR CAVES .	9.95
ALPHABET ZOO	9.95
DELTA DRAWING	9.95
FACEMAKER	9.95
KIDS ON KEYS	9.95
KINDERCOMP	9.95
(AGE 7 - ADULT):	
ADV.CREATOR (400/800).	9.95
FRACTION FEVER	9.95

THE BASIC TUTOR
Learn to program in BASIC
Requires a 410 OR 1010 Program Recorder
COMPLETE PACKAGE ONLY **$9.95**

BOOKS ONLY

DE RE ATARI	10.00
LOGO	10.00
ATARIWRITER	10.00
DOS 2.5	12.95
BASIC REFERENCE	5.00
BOOKKEEPER	10.00

SUPER SPECIALS
RECONDITIONED ATARI MERCHANDISE
30 DAY WARRANTY

800 (48K) COMPUTER $79.95	**SPACE AGE JOYSTICK $5.00**	**1030 MODEM WITH EXPRESS! $24.95**
400 (16K) COMPUTER $29.95	**ATARI TRACKBALL $9.95**	**1010 PROGRAM RECORDER $29.95**
1020 COLOR PRINTER/PLOTTER $19.95 40 COLUMNS WIDE (NEW IN BOX)	**ATARI BOOKKEEPER $14.95 - NO BOX** **ATARI NUMERIC KEYPAD $7.95**	**DISKETTES AS LOW AS 20 CENTS** 10 FOR $4.00 100 FOR $29.95 1000 FOR $200 MOST ARE UNNOTCHED WITH OLD SOFTWARE

SHIPPING INFORMATION - Prices do not include shipping and handling. Add $5.00 for small items ($8.00 Min. for Canada). Add $8.00 for disk drive. Add $2.75 for C.O.D. Calif. res. include 7% sales tax. Mastercard and Visa accepted if your telephone is listed in your local phone directory. Orders may be pre-paid with money order, cashier check, or personal check. Personal checks are held for three weeks before order is processed. C.O.D orders are shipped via UPS and must be paid with cash, cashier check or money order. International and APO orders must be pre-paid with cashier check or money order. $20.00 minimum on all orders. All sales are final - no refunds - prices are subject to change.

Phone orders accepted TUESDAY THROUGH FRIDAY from 10:00 am to 6:00 pm PST.

We carry a complete line of ATARI products and have a large public domain library.
Write or call for free catalogue. (408) 749-1003 TUE - FRI 10AM - 6 PM

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

CIRCLE #104 ON READER SERVICE CARD.

THE ULTIMATE GRAPHICS FILE CONVERTER

It seems to me that there are two types of computer enthusiasts in this world: programmers and users; and each group has its own philosophy and language. You can usually tell the difference by the way they act: programmers program and users use.

I guess I fall in the former category. I can tell because of what happened to me when *Newsroom* was released. Of course I bought the whole set of five disks and decided to learn how to use it and in the process make a family newsletter. I started out making the banner at the top of the page and immediately ran into trouble. You see, I wanted to use my family symbol on the header, and in spite of the 2,000+ graphics in the *Newsroom* set, none of them is a compass star. In fact, there are no stars at all. So I figured I could use the one from my *Print Shop* disk, but *Newsroom* does not have a routine to incorporate outside graphics or to save graphics you create onto a *Newsroom* format disk. (You can save them on DOS disks as photos or banners but not as *Newsroom* clip art disks).

The only recourse I had was to use the built-in graphics editor which is both clumsy and slow. Too bad I couldn't use *MicroPainter* and transfer my creation over. Immediately wheels started turning, and before I ever got started on the newsletter, I was pounding the keyboard. That's how I know I am a programmer, not a user! Here is what I humbly call the *Ultimate Graphics File Convertor* (UGFC)!

Typing It In

To create your copy of UGFC, first type in Listing 1 (checking your work with BASIC Editor II, found elsewhere in this issue) and save it to disk. Then type in Listing 2, save it and run it. A file named TEMP.LST will be written to your disk. Now load the program you typed from Listing 1 into your computer's memory and type ENTER "D:TEMP.LST" to merge the lines created by Listing 2 with the main program. Finally, save the completed program to disk.

For Users

With UGFC you have four formats to work from: Graphics 8, *MicroPainter*, *Print Shop* and *Newsroom*. Operation is quite simple. First you will select a Load Format then a Save Format (not the same type). After inserting the proper type of disk, the menu for that disk will appear. You then select the file you want and the desired picture will load.

HICS

by Lee S. Brilliant, M.D.

Next a set of four lines will appear which you move around the screen to define the area you wish to save. *Print Shop* saves are a fixed size while the others can be any size.

Finally, you give the Save filename and your screen is saved. The disk you save to *must* be an unneeded disk, because it will be formatted and dedicated to the single file. From this disk, you can save Graphics 8 and *MicroPainter* files to any DOS disk, while *Print Shop* saves can be loaded into the *Print Shop* editor then re-saved to your *Print Shop* file disk. *Newsroom* clip art files cannot easily be added to another disk, but you can convert then to a photo or banner and save them to a DOS disk.

Tutorial

Newsroom and *Print Shop* both display their graphics as monochrome, high resolution (i.e. Graphics 8) pictures. These can readily be converted to standard 62-sector DOS files, as well as the other way around, within size constraints. The problem arises when you try to convert between Graphics 8 screens and *MicroPainter*'s Graphics 15 (or 7½ or "E"). While both types occupy 7,680 bytes, the graphics mode 15 has only one half the horizontal resolution of Graphics 8. By assigning two bits to each pixel, you lose resolution but gain four colors.

You can load a Graphics 15 picture into a Graphics 8 screen, but you will still have only 160 columns of resolution, and you will lose your color information (although there is artifacted color which can look quite good). Conversely if you load a Graphics 8 picture into a Graphics 15 screen, you can display only 160 pixels per line, but the additional information creates strange color artifacts. (See Figure 1.)

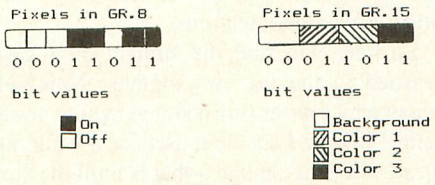


FIGURE 1

With careful attention paid to selection of color values and fill patterns you can reasonably exchange Graphics 8 and Graphics 15 screens.

Storage Techniques

Nothing really needs to be said about Graphics 8 storage. Each file is simply a 7,680-byte screen transferred to disk in a block. *MicroPainter* is the same except that after the screen bytes, the values of the four color registers are added. The real difficult storage forms are *Print Shop* and *Newsroom*.

Print Shop graphics disks (not including the original *Print Shop* disk) have a directory in Sectors 362 to 393 with each entry occupying 32 bytes instead of 16 like DOS. Sector 361 has the essential disk ID info and a VTOC or sector map. Each entry contains the starting sector of the graphic and an ID byte. Each graphic is 88*52 pixels or 572 bytes long. Each sector contains 126 bytes of data and two bytes containing the number of the next sector much the way DOS does. (See Figure 2.)

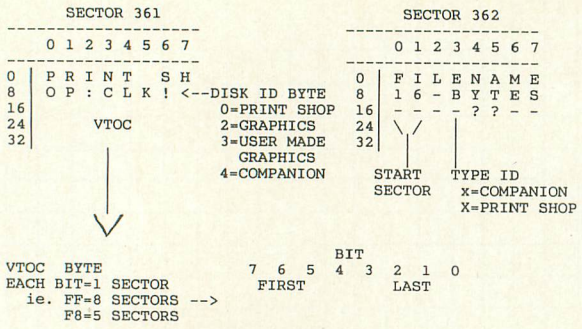


FIGURE 2

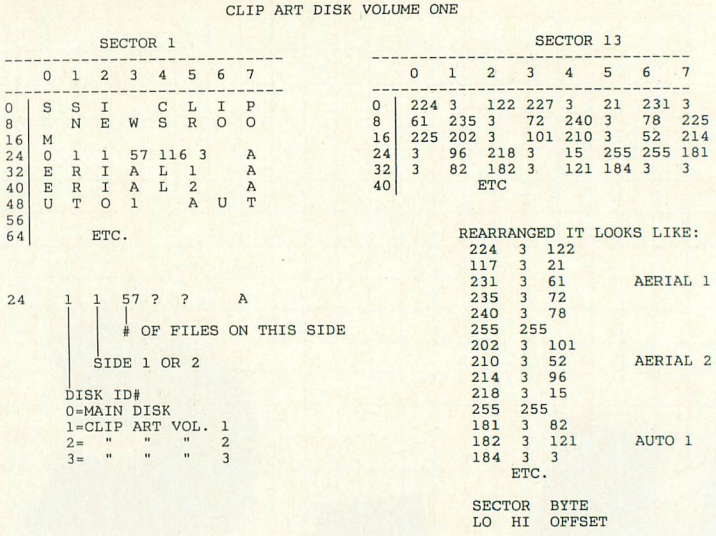


FIGURE 3

Each *Newsroom* clip art disk is in enhanced density mode and contains around 1,000 sectors. Fortunately, the disk drive and resident disk handler in the OS readily handle such problems. The process of loading a file begins with Sector 1. Here you find an ID header followed by the name of each file stored as ASCII text. Each name is separated by a character 0. The directory space extends to Sector 12 but doesn't seem to be used beyond Sector 4. Beginning at Sector 13 and extending up to about Sector 24 is a table of sector/byte locations for each graphic on the disk. Graphics are segregated into files, each one making up one screen and having one filename. Files are separated by headers consisting of two consecutive character 255s. (See Figure 3.)

First, you read the directory while counting the number of filenames you pass in order to find the one you want. Then you must search the sector table looking for and counting the file separators until you reach the one you want. For example, in Clip Art Volume 1, AUTO1 is the third name in the directory so you search until you find the second header in the sector table. The next three-byte entry will be the first graphic in the AUTO1 file. Now by our example this would be 181,3,82. The sector number is 181+3*256 or 949 and the offset into sector 949 is byte 82. The data for this file is illustrated in Figure 4.

The first four bytes give the screen boundaries of the graphic. Following are two types of data, single and repeat. Single data is a single byte whose value is placed directly on screen. To save disk space, when a byte is used more than once in a row, a special code is used to "compress" the data (most pictures contain large amounts of repetitive data). Each repetition is three bytes long: a 0 indicates a repetition followed by the number of repeats then the value to be repeated. Of course that means that any blank space, or 0, must be represented as a repetition: A single blank would then be 0, 1, 0. Since blanks are the most common repeat, it works out well.

Finally, bytes are scanned vertically, not in the usual horizontal fashion, proceeding column by column from the upper left to the lower right. Each sector uses all 128 bytes so there is no forward chaining or jumping sectors. Each sector must be contiguous with the last as are all graphics within a file. The left-hand margin of the screen area is actually 8, so any graphic saved past that margin will overwrite the Icon area and will be "inverted." Try not to save graphics past the left margin, but don't worry about *Print Shop* graphics; they are automatically marginated.

Final Notes

With UGFC you now have a way of changing one format to another among the best and most commonly used graphics programs. While UGFC should run on any type of Atari computer, I doubt it would work on an 810 disk drive because of the number of sectors on the disk; but if your drive will run *Newsroom*, it will run UGFC.

To round out UGFC, you will need a Graphics 8 paint program. The only one I am aware of is *Graphics Master* by Courtney Goodin which is not in current distribution. I happen to be lucky enough to be in the same users' group as Courtney but for the rest of you, hound ANALOG to get one in print! *[Okay, software authors: Anyone want to submit a Graphics 8 paint program?—ed.]* One exciting possibility is the ability to incorporate digitized photos or illustrations into your *Newsroom* creations.

So here you find the sum of all I have learned in the last two months. Now you know why I never finished making my newsletter! Maybe I can be a user for a while instead of a programmer—that is until the next idea hits me!

Addendum

As an afterthought I have included a short no-frills program to convert *Newsroom* Photos and Banners to Graphics 8 files. You need only a load-and-save filename. *Newsroom* stores photos and banners as DOS binary files with the first two characters of the filename "PH" for a photo and "BN" for a banner then a 6-byte filename without extenders. The graphics file consists of an 8-byte header, then the graphic margins.

The next group of bytes is variable and contains the word "Newsroom" at least once. Following are some blank spaces and assorted data which includes the disk and graphic

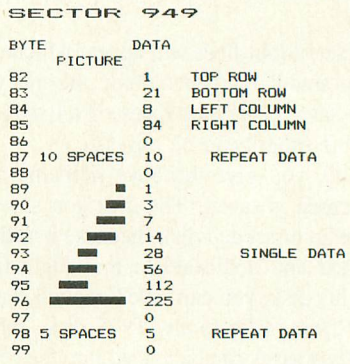


FIGURE 4

number of the original *Newsroom* clip art graphic. All this ends in a header that is a 255 followed by a 0. The banner or photo is then stored as a single file with no compression, horizonal scan, but in inverted (black-on-white background) form.

To type in this program, first type Listing 3 and save it to disk. Next type Listing 4, save it and run it. A file named TEMP.LST will be created on your disk. Merge this file with the program from Listing 3 in the manner described in "Typing it in" above.

LISTING 1: BASIC

```
EI 1 REM ********************************
CV 2 REM *      ULTIMATE GRAPHICS       *
PA 3 REM *        FILE CONVERTOR        *
OD 4 REM *    by Lee Brilliant, M.D.    *
TC 5 REM *                              *
FS 6 REM *       Copyright 1989         *
TC 7 REM *     by ANALOG COMPUTING      *
EP 8 REM ********************************
QQ 9 GOTO 900
GE 10 REM *** 10-500 BASIC SUBROUTINES FO
   R SPEED
PQ 20 IF BUF$(N,N)="ψ" THEN 60
SG 30 FN$(S)=BUF$(N,N):S=S+1
AD 35 N=N+1
JY 40 IF N=129 THEN N=1:SECT=SECT+1:GOSUB
    10000
RV 50 GOTO 20
SR 60 DIR$(COUNT*32+1,COUNT*32+LEN(FN$))=
   FN$:COUNT=COUNT+1:POKE 755,( NOT (PEEK
   (755)))*2
IW 70 IF N=128 THEN N=0:SECT=SECT+1:GOSUB
    10000
ZG 80 S=1:IF COUNT=SIZE THEN RETURN
UG 90 GOTO 35
TY 100 DIR$="":S=0:FOR SECT=13 TO 24:GOSU
   B 10040:DIR$(LEN(DIR$)+1)=BUF$(1,128):
   NEXT SECT
CJ 110 FOR N=1 TO LEN(DIR$):IF S=COUNT TH
   EN POP :GOTO 150
MZ 120 IF DIR$(N,N+1)="►)" THEN S=S+1:N=N
   +1:POKE 755,( NOT (PEEK(755)))*2:REM E
   SC-CNTL)
HV 130 NEXT N
NQ 150 SECT=INT(N/128):BYTE=N-SECT*128:SE
   CT=SECT+13:RETURN :REM S&B OF 1ST GRAP
   HIC
BM 180 X=ASC(BUF$(BYTE)):BYTE=BYTE+1:IF B
   YTE>128 THEN BYTE=1:SECT=SECT+1:GOSUB
   10000
ZP 190 RETURN
ZD 300 A=USR(ADR(MOVE$),ADR(UNCOMP$),1536
   ,LEN(UNCOMP$)):A=USR(1536,START,R,SIZE
   ,BYTE-1,ADR(BUF$)):RETURN
GZ 400 A=USR(ADR(MOVE$),ADR(COMP$),1536,L
   EN(COMP$)):A=USR(1536,START,R,SIZE,ADR
   (DIR$)):RETURN
VK 499 REM *** 500-900 CURSOR CONTROL SUB
   ROUTINES
GT 500 GET #2,K:IF K=155 THEN RETURN
GN 510 IF K=27 THEN POP :POP :GOTO 1005
WH 520 IF K=32 THEN FLAG= NOT (FLAG):GOTO
    700
GS 525 IF K>27 AND K<32 THEN ON K-27+D GO
   SUB 600,602,604,606,610,620,630,640,65
   0,660,670,680:GOTO 550
PV 526 IF D>0 THEN IF K>48 AND K<51 THEN
   D=(K-48)*4
NE 530 GOTO 500
RB 550 IF D=0 THEN A=USR(1560,ROWT,COLL,R
   OWT+52,COLL+88):GOTO 500
GH 551 A=USR(1560,ROWT,COLL,ROWB,COLR):GO
   TO 500
QG 600 IF ROWT>0 THEN ROWT=ROWT-1
ZF 601 RETURN
JG 602 IF ROWT<139 THEN ROWT=ROWT+1
ZL 603 RETURN
UC 604 IF COLL>0 THEN COLL=COLL-1
ZR 605 RETURN
DT 606 IF COLL<231 THEN COLL=COLL+1
ZX 607 RETURN
QI 610 IF ROWT>0 THEN ROWT=ROWT-1
ZH 611 RETURN
WK 620 IF ROWT+1<ROWB THEN ROWT=ROWT+1
ZJ 621 RETURN
TW 630 IF COLL>0 THEN COLL=COLL-1
ZO 632 RETURN
IC 640 IF COLL+1<COLR THEN COLL=COLL+1
ZN 641 RETURN
PW 650 IF ROWB-1>ROWT THEN ROWB=ROWB-1
ZP 651 RETURN
OK 660 IF ROWB<191 THEN ROWB=ROWB+1
ZR 661 RETURN
ZK 670 IF COLR-1>COLL THEN COLR=COLR-1
ZT 671 RETURN
WY 680 IF COLR<319 THEN COLR=COLR+1
ZV 681 RETURN
DW 700 IF FLAG=0 THEN GOSUB 13020:GOTO 50
   0
AR 710 GOSUB 13030:? "ЌUSE ⇖↑⇗←⇒ TO MO
   VE FRAME. USE ESC":? "TO GO BACK, RETU
   RN TO SAVE GRAPHIC,"
NE 720 ? "AND SPACE BAR TO TOGGLE WINDOW.
   ":IF T0=4 THEN ? "1 & 2 SELECT CURSOR
   LINES.";
NG 730 GOTO 500
YC 800 POSITION 0,OLDY:? #6;" ":POSITION
   0,NWY:? #6;"→":OLDY=NWY
HA 810 GET #2,K
PM 820 IF K=28 OR K=45 THEN NWY=NWY-2:GOT
   O 860
LX 830 IF K=29 OR K=61 THEN NWY=NWY+2:GOT
   O 860
NH 840 IF K=155 THEN 890
PA 850 GOTO 810
SV 860 IF NWY<YMIN THEN NWY=YMAX
SR 870 IF NWY>YMAX THEN NWY=YMIN
OV 880 GOTO 800
JW 890 SELECT=(NWY-YMIN+2)/2:RETURN
PU 900 REM DIMENSION VARIABLES
PW 901 OPEN #2,4,0,"K:"
QL 910 DIM CALL$(5),BUF$(129),FN$(16),DIR
   $(7680),MOVE$(49),ORA$(51),CURSOR$(243
   ),ROR$(43),ROL$(44)
ZM 920 DIM COMP$(239),UNCOMP$(194)
BM 963 CURSOR$(74,74)=CHR$(157):CURSOR$(2
   27,227)=CHR$(255)
MA 1000 OPEN #6,12,0,"S:":DL=PEEK(560)+25
   6*PEEK(561):POKE DL+2,70:POKE DL+3,PEE
   K(DL+4):POKE DL+4,PEEK(DL+5)
DP 1001 POKE DL+5,6:POKE 710,0:POKE 709,1
   4:POKE 82,0
TM 1005 POKE 752,1:? CHR$(125);"  ULTIMAT
   E GRAPHICS    file convertor"
WD 1010 ? " FORMAT TO LOAD FROM ":? :? "
    1)MICROPAINTER(62 SECTOR)"
VV 1020 ? :? "  2)GRAPHICS 8 SCREEN":? :?
    "  3)PRINT SHOP ICON"
KZ 1030 ? :? "  4)NEWSROOM":? :? " FORMAT
   TO SAVE TO "
JU 1040 ? :? "  1)MICROPAINTER(62 SECTOR)
   ":? :? "  2)GRAPHICS 8 SCREEN"
LZ 1050 ? :? "  3)PRINT SHOP ICON":? :? "
    4)NEWSROOM"
VQ 1060 POKE 703,4:GOSUB 13010
RJ 1070 OLDY=3:NWY=3:YMIN=3:YMAX=9:GOSUB
   800:FROM=SELECT
VA 1080 OLDY=13:NWY=13:YMIN=13:YMAX=19:GO
   SUB 800
NK 1090 IF SELECT<>FROM THEN 1100
RA 1095 ? "Ќ":? "YOU CAN'T USE THE SAME F
   ORMAT AS FROM":GOSUB 13000:GOSUB 13010
   :GOSUB 800:GOTO 1090
NV 1100 T0=SELECT
AX 1200 POKE 703,24:POKE 752,1:? "Ќ":? "I
   NSERT SOURCE DISK. PRESS ANY KEY."
HG 1210 GET #2,K
```

```
WY 1300 ON FROM GOSUB 13100,13100,13300,1
   3500:REM GET DIRECTORY
ZL 1350 POKE 752,0:TRAP 40000:? "K":? "EN
   TER FILE NAME:";:INPUT FN$
US 1400 IF FN$="" THEN 1200
GA 1500 REM GET FILE NAME
LF 1501 ON FROM GOSUB 2000,2000,4000,5000
QK 1515 GOSUB 13030:TRAP 40000
UQ 1520 ? "KENTER SAVE FILE NAME:";:INPUT
    FN$
AC 1525 IF FN$="" THEN 1520
IY 1530 ? "KINSERT BLANK DISK & PRESS A
   NY KEY.":GOSUB 13000
HV 1540 GET #2,K:? "PRESS RETURN TO FOR
   MAT & SAVE FILE,":? " ESC TO QUIT.":G
   ET #2,K:IF K=27 THEN RUN
UO 1550 IF K<>155 THEN 1530
YR 1560 TRAP 11300:? "KFORMATTING DISK":X
   IO 254,#1,0,0,"D:":GOSUB 13020
BV 1600 ON T0 GOSUB 6000,7000,8000,9000
KR 1700 GRAPHICS 1:POKE 710,0:? #6;" YOU
   HAVE FINISHED":? #6;"       USING":? #
   6:? #6;"          the":? #6
TW 1710 ? #6;" ultimate graphics":? #6;"
    file convertor":POKE 752,1
PJ 1720 ? "PRESS RETURN FOR ANOTHER CONVE
   RSION."
VJ 1750 GET #2,K:IF K=155 THEN RUN
TN 1760 GOTO 1750
QW 2000 REM MICROPAINTER GET
QD 2010 BUF$="D:":BUF$(3)=FN$:FN$=BUF$
AP 2020 TRAP 11100:OPEN #3,4,0,FN$:GOSUB
   13020:? #6;CHR$(125)
VE 2100 POKE 882,7:POKE 884,PEEK(88):POKE
    885,PEEK(89):POKE 888,0:POKE 889,30
GX 2110 A=USR(ADR("h▯0 Vd♦")):CLOSE #3:TR
   AP 40000:RETURN
FF 3000 REM GR.8 GET SAME AS MICROPAINTER
MV 4000 REM PS GET
EV 4010 FN$(LEN(FN$)+1)="
      ":? "K":? "LOADING ";FN$
SH 4020 FOR N=1 TO COUNT*32 STEP 32:IF DI
   R$(N,N+15)=FN$ THEN POP :GOTO 4050
PY 4030 NEXT N:POKE 195,170:POP :GOTO 111
   00
YD 4050 SECT=ASC(DIR$(N+16))+256*ASC(DIR$
   (N+17))
FG 4060 FOR N=1 TO 379 STEP 126:GOSUB 100
   40:DIR$(N)=BUF$(1,126):SECT=ASC(BUF$(1
   27))+256*ASC(BUF$(128)):NEXT N
XZ 4070 GOSUB 10040:DIR$(N)=BUF$(1,68):PO
   KE 755,2:GOSUB 13020:? #6;CHR$(125)
QQ 4080 START=PEEK(88)+256*PEEK(89)
QR 4100 FOR N=1 TO 572 STEP 11:A=USR(ADR(
   ORA$),ADR(DIR$)+N-1,START,11):START=ST
   ART+40:NEXT N
AH 4110 RETURN
XP 5000 REM NEWSROOM GET
BM 5010 FN$(LEN(FN$)+1)="
   ":? "K":POKE 752,1:? "LOADING ";FN$
UZ 5020 FOR N=1 TO COUNT*32 STEP 32:IF DI
   R$(N,N+15)=FN$ THEN POP :GOTO 5050
PZ 5030 NEXT N:POKE 195,170:POP :GOTO 111
   00
RV 5050 COUNT=(N-1)/32:GOSUB 100:GOSUB 13
   020:? #6;CHR$(125):GOSUB 10000
MU 5070 GOSUB 180:LO=X:GOSUB 180:HI=X:GOS
   UB 180:OFF=X
GY 5080 SECT=LO+256*HI:BYTE=OFF+1:IF BYTE
   >128 THEN BYTE=1:SECT=SECT+1:GOSUB 100
   00
JD 5090 FOR TIMES=1 TO 16:DIR$(1)="♥":DIR
   $(7680)="♥":DIR$(2)=DIR$
KF 5100 GOSUB 10000:GOSUB 180:ROWT=X:GOSU
```

```
   B 180:ROWB=X:GOSUB 180:COLL=X:GOSUB 18
   0:COLR=X
PU 5110 C=INT((COLR-COLL)/8)+1:R=ROWB-ROW
   T+1:SIZE=C*R:START=ADR(DIR$)+40*(ROWT-
   1)+INT(COLL/8)
VZ 5120 GOSUB 300:SECT=PEEK(778)+256*PEEK
   (779):BYTE=PEEK(1726):SHIFT=COLL-INT(C
   OLL/8)*8:IF SHIFT=0 THEN 5125
QG 5122 FOR N=1 TO SHIFT:A=USR(ADR(ROR$),
   ADR(DIR$),7680):NEXT N
YP 5125 A=USR(ADR(ORA$),ADR(DIR$),PEEK(88
   )+256*PEEK(89),7680)
LB 5130 NEXT TIMES
AJ 5300 RETURN
NT 6000 REM MICROPAINTER PUT
OA 6001 TRAP 40000:GOTO 6500
LY 6010 GOSUB 9002:DIR$(1)="♥":DIR$(7680)
   ="♥":DIR$(2)=DIR$:START=PEEK(88)+256*P
   EEK(89):OFF=ROWT*40+LBYTE
PJ 6015 LPRINT ROWT,ROWB,COLL,COLR,LBYTE,
   WIDE,C,OFF
GZ 6020 FOR N=ROWT TO ROWB:A=USR(ADR(MOVE
   $),START+OFF,ADR(DIR$)+OFF,WIDE):OFF=O
   FF+40:NEXT N
CZ 6030 BUF$="D:":BUF$(3)=FN$:TRAP 11200:
   OPEN #3,8,0,BUF$:AD=ADR(DIR$):HI=INT(A
   D/256):LO=AD-256*HI
FJ 6040 POKE 882,11:POKE 884,LO:POKE 885,
   HI:POKE 888,0:POKE 889,30:A=USR(ADR("h
   T0 V♂♦"))
AT 6050 RETURN
RA 6500 GOSUB 6010:PUT #3,0:PUT #3,14:PUT
    #3,14:PUT #3,14:CLOSE #3:RETURN
PA 7000 REM GR.8 PUT
   ="5SI CLIP NEWSROOM" THEN 13600
WU 13510 ? "ß":? "NOT A NEWSROOM DISK!":?
    "INSERT A NEWSROOM CLIP ART DISK AND"
   :? "PRESS RETURN OR ESC TO GO BACK."
NE 13515 GOSUB 13000
DN 13520 GET #2,K:IF K=155 THEN 13500
WR 13530 IF K=27 THEN POP :GOTO 1005
CA 13540 GOTO 13520
PE 13600 DIR$(1)=" ":DIR$(7680)=" ":DIR$(
   2)=DIR$:? "ß":POSITION 8,10:? "  READIN
   G DIRECTORY  ";
CM 13610 SIZE=ASC(BUF$(28)):COUNT=0:S=1:N
   =32:GOSUB 20:POKE 755,2:POKE 752,1:A=0
   :GOTO 13410
   K(195);". CORRECT AND PRESS ANY KEY.":
   GET #2,K
ZW 11210 ON T0 GOTO 6000,7000
XP 11300 GRAPHICS 0:? "FORMATTING ERROR.
   USE NEW DISK.":END
QF 13000 SOUND 0,200,10,10:A=1^1:SOUND 0,
   250,10,10:A=1^1:SOUND 0,0,0,0:FOR DELA
   Y=1 TO 100:NEXT DELAY:RETURN
JN 13010 ? "ß":? "USE ↑ & ↓ THEN PRESS
   RETURN":RETURN
GM 13020 GRAPHICS 8+16+32:POKE 710,0:POKE
    709,14:RETURN
KK 13030 GRAPHICS 8+32:POKE 710,0:POKE 70
   9,14:RETURN
YT 13100 REM DOS DIRECTORY
BQ 13101 TRAP 11000:OPEN #1,6,0,"D:*.*":P
   OKE 201,14:POKE 675,0:POKE 676,2:POKE
   677,0:POKE 678,8:POKE 679,0:? "ß"
5S 13110 INPUT #1,FN$:IF FN$(6,9)="FREE"
   THEN 13130
RK 13120 ? FN$(3,13);"▸";:GOTO 13110
DP 13130 CLOSE #1:TRAP 40000:POKE 703,4:R
   ETURN
JE 13300 REM PRINT SHOP DIRECTORY
NZ 13301 ? "ß":? :SECT=361:GOSUB 10000:IF
    BUF$(1,15)="PRINT SHOP:CLK!" THEN 133
   50
AS 13310 ? "ß":? "NOT A PRINT SHOP DISK":
   ? "INSERT PRINT SHOP GRAPHICS DISK AND
   ":? "PRESS RETURN OR ESC TO GO BACK."
MY 13315 GOSUB 13000
AV 13320 GET #2,K:IF K=155 THEN 13300
WL 13330 IF K=27 THEN POP :GOTO 1005
AS 13340 GOTO 13320
GO 13350 DIR$(1)=" ":DIR$(7680)=" ":DIR$(
   2)=DIR$:POSITION 8,10:? "  READING DIRE
   CTORY  ";:COUNT=0:TRAP 40000
XT 13360 FOR SECT=362 TO 393:GOSUB 10040
RY 13370 FOR N=0 TO 96 STEP 32:FN$=BUF$(N
   +20,N+20):IF FN$="X" OR FN$="x" THEN 1
   3380
BN 13375 GOTO 13400
LU 13380 DIR$(COUNT*32+1,COUNT*32+32)=BUF
   $(N+1,N+32):COUNT=COUNT+1
YP 13400 NEXT N:NEXT SECT:POKE 755,2:POKE
    755,2:POKE 752,1:A=0
HR 13410 ? "ß":FOR N=0 TO 11:POSITION 6,1
   +N:? DIR$(32*(N+A)+1,32*(N+A)+16);"  "
   ;DIR$(32*(N+12+A)+1,32*(N+12+A)+16)
IR 13420 NEXT N
IV 13430 POKE 703,4:? "ßPRESS ↑ & ↓ FOR
    MORE TITLES,  ESC ":? "TO GO BACK, &
    RETURN  TO CHOOSE TITLE."
KL 13440 GET #2,K
YW 13450 IF K=27 THEN POKE 703,24:POP :GO
   TO 1005
SE 13460 IF K=155 THEN RETURN
EM 13465 IF K=61 OR K=45 THEN 13480
CO 13470 IF K<28 OR K>29 THEN 13440
JD 13480 IF K=28 OR K=45 THEN IF COUNT>A+
   24 THEN A=A+24:POKE 703,24:GOTO 13410
AM 13490 IF K=29 OR K=61 THEN IF A>0 THEN
    A=A-24:POKE 703,24:GOTO 13410
EG 13495 GOTO 13440
MA 13500 REM DIRECTORY FOR NEWSROOM
QB 13501 SECT=1:GOSUB 10000:IF BUF$(1,17)
XY 10020 POKE 779,INT(SECT/256):POKE 778,
   SECT-INT(SECT/256)*256:POKE 769,1:A=US
   R(ADR(CALL$))
XH 10030 IF PEEK(771)=1 THEN RETURN
ZC 10035 GRAPHICS 0:? "DISK ERROR NUMBER
   ";PEEK(771):END
UB 10040 POKE 755,( NOT (PEEK(755)))*2:GO
   TO 10000
WC 10050 POKE 770,87:GOTO 10010
TK 11000 CLOSE #1:IF PEEK(195)=136 THEN 1
   3130
JS 11010 ? "ERROR ";PEEK(195):STOP
LA 11100 CLOSE #3:IF PEEK(195)=170 THEN ?
    "ß":? "FILE NOT FOUND! PRESS ANY KEY"
   :GOSUB 13000:GET #2,K:GOTO 1350
FK 11110 ? "ß":? "ERROR ";PEEK(195);".  P
   RESS ANY KEY.":GOSUB 13000:GET #2,K:GO
   TO 1350
CG 11200 POP :GOSUB 13030:? "ßERROR ";PEE
   START+7680,7680):NEXT N
WE 8240 FOR N=0 TO 571 STEP 11:A=USR(ADR(
   MOVE$),START+OFF,ADR(DIR$)+N,11)
GD 8250 OFF=OFF+40:NEXT N
BL 8260 SECT=1:FOR N=1 TO 572 STEP 126:BU
   F$=DIR$(N,N+126):BUF$(127)=CHR$(SECT+1
   ):BUF$(128)="♥"
CN 8270 GOSUB 10050:SECT=SECT+1:NEXT N
BI 8280 RETURN
QQ 9000 REM NEWSROOM PUT
UK 9001 GOTO 9080
IX 9002 IF FROM=3 THEN COLL=0:ROWT=0:COLR
   =88:ROWB=52:GOTO 9030
BL 9005 A=USR(ADR(MOVE$),ADR(CURSOR$),153
```

```
   6,LEN(CURSOR$)):A=USR(1560,0,0,191,319
   )
US 9010 D=4:FLAG=0:COLL=0:ROWT=0:COLR=319
   :ROWB=191:POKE 764,33:GOSUB 500
IA 9020 A=USR(1560,0,0,0,0)
FB 9030 LBYTE=INT(COLL/8):RBYTE=INT(COLR/
   8):SHIFT=COLL-LBYTE*8:C=COLR-COLL:SHIF
   T2=7-(C-INT(C/8)*8):WIDE=INT(C/8)+1
NT 9040 IF SHIFT2=0 THEN 9060
SW 9050 COLOR 0:FOR N=1 TO SHIFT2:PLOT CO
   LR+N,ROWT:DRAWTO COLR+N,ROWB:NEXT N
UX 9060 START=PEEK(88)+256*PEEK(89):IF SH
   IFT=0 THEN 9075
ZX 9070 FOR N=1 TO SHIFT:A=USR(ADR(ROL$),
   START+7680,7680):NEXT N
BW 9075 RETURN
AY 9080 GOSUB 9002:START=START+INT(COLL/8
   )+ROWT*40:R=ROWB-ROWT+1:SIZE=R*(INT(C/
   8)+1)
TR 9090 DIR$(1)="♥":DIR$(7680)="♥":DIR$(2
   )=DIR$
LJ 9100 GOSUB 400:SIZE=PEEK(205)+256*PEEK
   (206)-ADR(DIR$)
EM 9110 BUF$(1)="♥":BUF$(129)="♥":BUF$(2)
   =BUF$:BUF$(1,17)="SSI CLIP NEWSROOM":B
   UF$(26,28)="├┼┼"
EP 9120 BUF$(32,LEN(FN$)+32)=FN$:SECT=1:G
   OSUB 10050
HU 9130 BUF$(1)="♥":BUF$(129)="♥":BUF$(2)
   =BUF$:BUF$(1)=CHR$(60):BUF$(2)=CHR$(0)
   :BUF$(3)=CHR$(0):BUF$(4,6)="▌▐▌"
XJ 9140 SECT=13:GOSUB 10050:SECT=60:BUF$(
   1)="♥":BUF$(129)="♥":BUF$(2)=BUF$
QD 9145 BUF$(1)=CHR$(ROWT+1):BUF$(2)=CHR$
   (ROWB+1):IF FROM=3 THEN COLL=8:COLR=96
WR 9150 BUF$(3)=CHR$(COLL):BUF$(4)=CHR$(C
   OLR):BUF$(5)=DIR$:GOSUB 10050:IF SIZE<
   125 THEN 9200
QD 9160 FOR N=125 TO SIZE STEP 128:BUF$(1
   )="♥":BUF$(129)="♥":BUF$(2)=BUF$:BUF$=
   DIR$(N):SECT=SECT+1:GOSUB 10050
HX 9170 NEXT N
AL 9200 RETURN
NK 10000 BUF$(1)="":BUF$(129)="":BUF$(2)=
   BUF$:POKE 770,82
EL 10010 A=ADR(BUF$):POKE 773,INT(A/256):
   POKE 772,A-INT(A/256)*256
IL 7001 GOSUB 6010:CLOSE #3:RETURN
CM 8000 REM PRINTSHOP PUT
YN 8001 A=USR(ADR(MOVE$),ADR(CURSOR$),153
   6,LEN(CURSOR$)):A=USR(1560,70,116,122,
   204)
ZQ 8010 D=0:FLAG=0:COLL=116:ROWT=70:POKE
   764,33:GOSUB 500
HZ 8020 A=USR(1560,0,0,0,0)
MK 8201 BUF$="PRINT SHOP:CLK!┘▀":BUF$(18)
   ="♥":BUF$(129)="♥":BUF$(19)=BUF$(18):B
   UF$(33,33)="▌▌"
VP 8202 BUF$(78,82)="├▌▐▐▐"
DP 8210 SECT=361:GOSUB 10050
KB 8220 BUF$=FN$:BUF$(LEN(BUF$)+1)="
                "
SZ 8230 BUF$(17,22)="├♥♥x@ ▌":BUF$(23)="♥"
   :BUF$(129)="♥":BUF$(24)=BUF$(23):SECT=
   362:GOSUB 10050
TK 8232 DIR$(1)="♥":DIR$(7680)="♥":DIR$(2
   )=DIR$
DX 8235 START=PEEK(88)+256*PEEK(89):I=INT
   (COLL/8):OFF=ROWT*40+I:SHIFT=COLL-8*I:
   IF SHIFT=0 THEN 8240
AM 8236 FOR N=1 TO SHIFT:A=USR(ADR(ROL$),
```

LISTING 2: BASIC

```
SA 5 LINE=100
GZ 10 FOR N=1 TO 1092 STEP 26:ST=0:FOR S=
   1 TO 26:READ D:ST=ST+D:T=T+D:NEXT S
LR 20 READ X:IF X<>ST THEN ? "ERROR IN LI
   NE #";LINE:STOP
SO 30 ? "LINE ";LINE;" OK!"
QN 40 LINE=LINE+10:NEXT N
PE 50 IF T<>112481 THEN ? "CHECKSUM ERROR
   ":END
DL 60 ? "▌▐▌INSERT FORMATTED DISK & PRESS
    ANY KEY":POKE 764,255
YH 65 IF PEEK(764)=255 THEN 65
TN 70 ? :? "WRITING FILE "D:TEMP.LST"";:O
   PEN #1,8,0,"D:TEMP.LST":RESTORE :POKE
   764,255
AS 80 FOR N=1 TO 1092 STEP 26:FOR X=1 TO
   26:READ D:PUT #1,D:NEXT X:READ D:POKE
   755,( NOT (PEEK(755)))*2:NEXT N
IH 90 CLOSE #1:? :? :? "DONE":END
OL 100 DATA 57,50,49,32,67,48,77,80,36,61
   ,34,104,104,133,204,141,230,6,104,133,
   203,141,229,6,104,104,2537
AB 110 DATA 141,233,6,141,235,6,104,141,2
   32,6,104,141,231,6,104,133,206,104,133
   ,205,160,0,140,237,6,140,3295
AE 120 DATA 236,6,177,203,141,234,6,24,16
   5,203,105,40,133,203,165,204,105,0,133
   ,204,206,235,6,208,24,173,3539
GL 130 DATA 233,6,141,235,6,238,229,6,208
   ,3,238,230,6,34,155,57,50,50,32,67,48,
   77,80,36,40,56,2561
FR 140 DATA 49,41,61,34,173,229,6,133,203
   ,173,230,6,133,204,173,231,6,208,17,17
   3,232,6,208,9,173,237,3348
SC 150 DATA 6,208,3,32,199,6,96,206,232,6
   ,206,231,6,177,203,170,173,237,6,208,3
   8,173,234,6,208,6,3276
SG 160 DATA 238,237,6,76,161,6,236,234,6,
   240,11,173,234,6,145,205,32,192,6,76,4
   3,6,169,1,141,236,3116
HK 170 DATA 6,141,237,6,76,43,34,155,57,5
   0,51,32,67,48,77,80,36,40,49,54,49,41,
   61,34,6,238,1768
MM 180 DATA 236,6,208,15,206,236,6,32,199
   ,6,238,236,6,140,237,6,76,115,6,236,23
   4,6,240,3,32,199,3160
JP 190 DATA 6,76,43,6,230,205,208,2,230,2
   06,96,169,0,145,205,32,192,6,173,236,6
   ,145,205,32,192,6,3052
XY 200 DATA 173,234,6,145,205,32,192,6,14
   0,237,6,140,236,6,96,0,0,0,0,0,0,0,0,0
   ,0,34,1888
VL 210 DATA 155,57,50,52,32,85,78,67,79,7
   7,80,36,61,34,104,104,133,206,141,186,
   6,104,133,205,141,185,2591
DA 220 DATA 6,104,104,141,189,6,141,191,6
   ,104,141,188,6,104,141,187,6,104,104,1
   41.190.6.104,133,204.104,2855
AH 230 DATA 133,203,160,0,140,184,6,173,1
   84,6,240,1,96,32,83,6,138,240,6,32,113
   ,6,34,155,57,50,2478
HR 240 DATA 53,32,85,78,67,79,77,80,36,40
   ,54,49,41,61,34,76,45,6,32,83,6,142,19
   2,6,32,83,1569
MR 250 DATA 6,32,113,6,206,192,6,208,248,
   76,45,6,172,190,6,177,203,72,238,190,6
   ,16,16,169,0,141,2740
VR 260 DATA 190,6,238,10,3,208,3,238,11,3
```

```
   ,32,83,228,104,170,96,173,187,6,208,12
   ,173,188,34,155,57,2816
DN 270 DATA 50,54,32,85,78,67,79,77,80,36
   ,40,49,50,49,41,61,34,6,208,4,238,184,
   6,96,206,188,2098
IB 280 DATA 6,206,187,6,160,0,138,145,205
   ,206,191,6,208,27,238,185,6,208,3,238,
   186,6,173,185,6,133,3258
QP 290 DATA 205,173,186,6,133,206,173,189
   ,6,141,191,6,76,183,6,24,165,205,105,4
   0,133,205,165,206,105,0,3233
PS 300 DATA 133,206,96,0,0,0,0,0,0,0,0,
   0,34,155,57,51,48,32,67,65,76,76,36,61
   ,34,1227
NO 310 DATA 104,32,83,228,96,34,155,57,52
   ,48,32,77,79,86,69,36,61,34,104,104,13
   3,204,104,133,203,104,2452
KL 320 DATA 133,206,104,133,205,104,133,2
   08,104,133,207,160,0,165,207,208,7,165
   ,208,208,1,96,198,208,198,207,3906
PH 330 DATA 177,203,145,205,200,208,4,230
   ,204,230,206,144,230,176,228,34,155,57
   ,53,48,32,79,82,65,36,61,3492
RJ 340 DATA 34,104,104,133,204,104,133,20
   3,104,133,206,104,133,205,104,133,208,
   104,133,207,160,0,165,207,208,7,3540
DU 350 DATA 165,208,208,1,96,198,208,198,
   207,177,205,17,203,145,205,200,208,4,2
   30,204,230,206,144,228,176,226,4497
NY 360 DATA 34,155,57,54,48,32,67,85,82,8
   3,79,82,36,61,34,0,0,0,0,0,0,0,0,0,0
   ,989
IV 370 DATA 0,0,0,0,0,128,64,32,16,8,4,2,
   1,104,104,104,141,6,6,104,141,8,6,104,
   141,7,1231
WA 380 DATA 6,104,104,141,9,6,104,141,11,
   6,104,141,10,6,169,0,141,12,6,32,80,6,
   32,80,6,32,1489
SY 390 DATA 80,6,32,80,6,162,5,189,6,6,32
   ,0,6,202,16,247,96,174,12,6,189,0,6,32
   ,179,6,1775
WS 400 DATA 174,34,155,57,54,49,32,67,85,
   82,83,79,82,36,40,57,49,41,61,34,12,6,
   232,189,0,6,1796
SR 410 DATA 141,14,6,232,189,0,6,141,15,6
   ,232,142,12,6,32,114,6,96,173,14,6,41,
   7,170,189,16,2006
TQ 420 DATA 6,141,13,6,162,3,78,15,6,110,
   14,6,202,208,247,24,173,14,6,101,88,13
   3,205,165,89,105,2320
TW 430 DATA 0,133,206,160,0,162,192,177,2
   05,77,13,6,145,205,24,165,205,105,40,1
   33,205,165,206,105,0,133,3167
BI 440 DATA 206,202,208,233,96,141,34,155
   ,57,54,50,32,67,85,82,83,79,82,36,40,4
   9,56,49,41,61,34,2312
ZZ 450 DATA 14,6,169,40,141,15,6,162,8,16
   9,0,133,203,133,204,10,38,203,14,15,6,
   144,8,24,109,14,1988
IG 460 DATA 6,144,2,230,203,202,208,237,2
   4,101,88,133,205,165,203,101,89,133,20
   6,169,32,141,13,6,160,39,3240
WW 470 DATA 173,13,6,81,205,145,205,136,1
   6,246,96,34,155,57,55,48,32,82,79,82,3
   6,61,34,104,104,133,2418
ET 480 DATA 204,104,133,203,104,133,206,1
   04,133,205,160,0,24,165,205,208,7,165,
   206,208,1,96,198,206,198,205,3781
QH 490 DATA 177,203,106,145,203,200,208,2
   ,230,204,144,231,176,229,34,58,82,79,7
   6,36,61,34,104,104,133,204,3463
FM 500 DATA 104,133,203,104,133,206,104,1
   33,205,160,0,24,165,205,208,7,165,206,
   208,1,96,198,206,198,205,177,3754
MT 510 DATA 203,42,145,203,152,208,2,198,
   204,136,144,230,176,228,34,155,0,0,0,0
   ,0,0,0,0,0,0,2460
```

LISTING 3: BASIC

```
MD 10 GOSUB 1000
AW 20 FOR R=ROWT TO ROWB:FOR S=LC TO S+WI
   DE:GET #3,X:POKE S,X:NEXT S:LC=LC+40:N
   EXT R
TD 30 IF COLR=RBYTE*8+7 THEN 50
BB 40 FOR N=COLR TO RBYTE*8+7:PLOT N,ROWT
   :DRAWTO N,ROWB:NEXT N
WC 50 IF FLAG THEN PLOT 0,ROWT:DRAWTO 0,R
   OWB
KV 53 LC=START:FOR N=ROWT TO ROWB:A=USR(A
   DR(EOR$),LC,WIDE+1):LC=LC+40:NEXT N
JE 55 GRAPHICS 8+32:POKE 710,0:POKE 709,1
   4:? "SAVING"
ZH 60 CLOSE #3:OPEN #3,8,0,FN2$
JP 70 POKE 882,11:POKE 884,PEEK(88):POKE
   885,PEEK(89):POKE 888,0:POKE 889,30:A=
   USR(ADR("h█0 V▣♦"))
MK 80 CLOSE #3
WS 90 GRAPHICS 2+16:POSITION 8,6:? #6;"DO
   NE"
LG 100 GOTO 100
VI 999 STOP
DY 1010 DIM FN1$(15),FN2$(15),CALL$(5)
FJ 1020 ? CHR$(125):POSITION 2,10:? "USE
   DEVICE ID AND EXTENTERS.":POSITION 2,1
   :? "ENTER LOAD FILENAME:";:INPUT FN1$
AX 1030 ? :? "ENTER SAVE FILENAME:";:INPU
   T FN2$
IH 1040 D=47:IF FN1$(3,4)="BN" OR FN1$(4,
   5)="BN" THEN FLAG=1:D=31
BV 1050 OPEN #3,4,0,FN1$
YH 1060 FOR N=1 TO 8:GET #3,X:NEXT N
RX 1070 GET #3,ROWT:GET #3,ROWB:GET #3,CO
   LL:GET #3,COLR:IF FLAG THEN COLL=1:COL
   R=239
GJ 1080 FOR N=1 TO D:GET #3,X:NEXT N
MI 1090 LBYTE=INT(COLL/8):RBYTE=INT(COLR/
   8):WIDE=RBYTE-LBYTE
FC 1100 GRAPHICS 8+16:POKE 710,0:POKE 709
   ,14:COLOR 1:START=PEEK(88)+256*PEEK(89
   )+40*ROWT+LBYTE:LC=START
AE 1110 RETURN
```

LISTING 4: BASIC

```
DL 60 ? "███INSERT FORMATTED DISK & PRESS
    ANY KEY":POKE 764,255
YH 65 IF PEEK(764)=255 THEN 65
TN 70 ? :? "WRITING FILE "D:TEMP.LST"";:O
   PEN #1,8,0,"D:TEMP.LST":RESTORE :POKE
   764,255
HN 80 FOR N=1 TO 92:READ D:PUT #1,D:NEXT
   N
IH 90 CLOSE #1:? :? :? "DONE":END
QP 100 DATA 49,48,48,48,32,68,73,77,32,69
   ,79,82,36,40,52,52,41,58,69,79,82,36,6
   1,34,104,104
YZ 110 DATA 133,204,104,133,203,104,133,2
   06,104,133,205,160,0,24,165,205,208,7,
   165,206,208,1,96,198,206,198
GL 120 DATA 205,177,203,73,32,145,203,200
   ,208,2,230,204,144,230,176,228,34,58,6
   9,79,82,36,40,51,51,44
ET 130 DATA 51,51,41,61,67,72,82,36,40,50
   ,53,53,41,155
```

continued from page 15

You can either blow into the mouthpiece for dynamic control over volume and tone or set a switch to use the keys like a keyboard with on/off operation. The *DH-100* is battery operated, has a two-octave chromatic scale, contains its own built-in speaker, and has a MIDI out connection.

The Casio *DH-100 Digital Horn* is both easy to learn and fun to play. You guessed it: I bought one.

You always find the unusual at CES and Sanyo had it. Highlighting their activity in solar energy, Sanyo displayed a solar-powered golf cart. Since the vehicle does not run on normal fuel it is very efficient and pollution free. The thing looked more like a vehicle from *Blade Runner* than what you might see roaming the links on a Saturday morning.

Another unusual product seen was the *XPRES'R* electronic-reader board. This 2-inch by 12-inch electronic message display is designed to be mounted in the rear window of a car. The unit is remotely controlled by a handset resembling a cellular telephone with an illuminated LCD screen. The user can select from any of the 198 built-in messages or create up to 64 custom messages. The built-in messages run the gamut from emergency calls for help, safety, traffic tools to humor and animation.

The *XPRES'R* can also be used outside of the car with the optional AC adapter. Messages on the electronic display can be set to scroll, blink, pop-up or drop down, and the messages stay in memory even if the power is removed from the unit. The *XPRES'R* will be available in auto-parts stores by the time you read this. Price is yet to be announced.

At the last CES I saw a product called the *Mail-Call*, a device that used a solar-powered transmitter to alert you by means of an audible noise and flashing light that your mail had arrived. The product is meant to overcome the problem of making trips to your mail box to find that the mail has not yet arrived.

At this show I saw the *Letter Sledder*. Don't laugh—this product extends a curbside mailbox (30 inches) to your car window by remote control. Yes, you can avoid the hassle of having to get out of your car to get the mail with the *Letter Sledder*. The originator, Todd Powers, says the idea came to him when he noticed his wife making ruts in the lawn while trying to retrieve mail from her car. She kept banging the side mirror and car door into the mail so he thought up the solution.

If you want a *Letter Sledder*, it will cost you $200 for an AC model or $250 for a solar-powered version. Oh, yes, the post and mail box cost extra. It can be ordered from Todd Powers Associates. I guess technology is not *always* the answer to every problem. Okay, you can laugh now. Todd, ever think you might have too much free time on your hands?

It would be hard to imagine what life would be like without our modern technology. And we all get much satisfaction from the audio, video, entertainment and personal electronics products that technology has made available to us. The Consumer Electronics Show is where all the "new stuff" is shown before it becomes available to the consumer.

My definition of "neat things" has changed since I was a youngster. Now neat things are. . .well. . .what I see each year at CES. I'm fortunate both to be able to see all of these exciting products firsthand and to later be able to afford some of them. I'm also fortunate that I can share these neat things with you.

Arthur Leyenberger is a human factors psychologist and freelance writer living in New Jersey. Consumer electronics is his life.

Companies Mentioned in This Article

Brother International
8 Corporate Place
Piscataway, NJ 08854
(201) 981-0300

Canon, Inc.
One Canon Plaza
Lake Success, NY 11042
(516) 488-6700

Cobra Electronics Group
Dynascan Corp.
6500 West Cortland Street
Chicago, IL 60635
(312) 889-8870

Hewlett-Packard Company
Inquiries Manager
1000 N.E. Circle Blvd.
Corvallis, OR 97330
(503) 757-2000

Mitsubishi International Corporation
Communication Equipment Sales Division
879 Supreme Drive
Bensenville, IL 60106
(312) 860-4200

NEC America, Inc.
Mobile Radio Division
4910 W. Rosecrans Ave.
Hawthorne, CA 90250
(213) 973-2071

Opsin Inc.
18550 Firlands Way N.
Seattle, WA 98133
(206) 542-7871

Polyglot
P.O. Box 521603
Miami, FL 33152
(800) 634-4692

Reflection Technology
171 Third Street
Cambridge, MA 02141
(617) 890-5905

Sharp Electronics Corp.
Sharp Plaza
Mahwah, NJ 07430
(201) 529-8200

Sony Corporation of America
9 West 57th Street
New York, NY 10019
(212) 418-9427

Todd Powers Associates, Inc.
825 Harper Drive
Algonquin, IL 60102

Toshiba America, Inc.
82 Totowa Rd.
Wayne, NJ 07470
(201) 628-8000

XPRES Communications, Inc.
755-601 West broadway
Vancouver, B.C.
Canada V5Z 4C2
(604) 873-1749

Is DAT DEAD?

According to some sources, Digital Audio Tape (DAT) is the most eagerly awaited audio product since the compact disc. Being able to make very high-quality recordings has excited consumers since the technology was introduced a couple of years ago.

Unfortunately, the wait for DAT in the U.S. is not over yet.

DAT recorders were evident at almost every major audio and video electronics company booth. However, the recorders were not so prominently displayed or hyped as in previous shows. One reason is that consumers still can't purchase DAT recorders on the open U.S. market because manufacturers have been scared off by the threatened lawsuit from the Recording Industry Association of America (RIAA).

The recording industry is concerned that since DAT can make near-perfect recordings of CDs, there will be large-scale piracy of CDs and millions of dollars in lost revenue. The hardware manufacturers, mostly Japanese, counter with an analogy to the LP and cassette markets. They say, DAT is to the CD as the cassette is to the LP: People make copies for their own use and that's fine.

If you look hard enough and are willing to pay premium prices, you can buy "gray market" DAT decks that have been imported from Japan. However, prices range between $1-2,000 dollars, at least three times the price level that they would be at if DAT was as common a consumer product as the CD player. Several companies have introduced DAT players for car use, but there are few prerecorded DAT tapes, and the inability to make your own recordings limits the usefulness of a car DAT player.

Another threat of "premature DAT death" comes from the emerging recordable, erasable compact disc technology that will arrive in the early 1990s. Several companies are now working on an erasable CD which promises the ability to make the same high-quality recordings that can be done with DAT.

Radio Shack's parent company, Tandy Corporation, is probably the most widely known manufacturer doing research in recordable CD technology. Their system, announced last year, is called THOR-CD (Tandy High-Intensity Optical Recording) and is said to allow the user to re-record CDs up to 40 times. Further, once recorded, the blue THOR-CD discs are completely compatible and playable on any current CD player.

Even if DAT becomes stillborn in the U.S. market, the recordable CD products will probably face the same reaction from the RIAA. Whether it's DAT, THOR or whatever, the manufacturers and recording industry need to resolve the issue and many people feel that it should be resolved in the courts by having a hardware company introduce a DAT (or recordable CD) product and let the RIAA sue them. As it stands now, consumers are not benefiting from a technology that can significantly improve the quality of their recorded music.

Still Waiting For Still Video

It is true that still-video photography is a combination of standard still-photography and video. As such, the new technology contains some of the advantages and disadvantages of each technology. One of the major obstacles to the consumer success of still-video photography lies with how the user will get printed photos. Both Sony and Canon offer products that will print a video image but the cost of the machines is high (about $2-4,000) and the quality is poor.

Another (perhaps the only viable) alternative is to have photo finishers like your friendly neighborhood Fotomat provide a video-printing service. Professional video printers that cost upwards of $15,000 are affordable to commercial businesses. It doesn't seem too unlikely that you'll drop off your 2-inch still-video disks at the convenience store in the morning and pick up your prints on the way home from work.

As likely as this sounds, the Catch-22 is that photo finishers will be unwilling to provide such a service until there is sufficient demand from consumers. While at the same time, consumers may not embrace the new technology until they have some place to get a hard copy of their images.

What Is The CEBus

CEBus is not a product but a voluntary standard which is close to completion. Manufacturers will follow the CEBus standard when they build and supply you with home-automation products allowing products from different manufacturers to work together. Each product will be smart enough to pick up messages from other products and carry out their instructions.

CEBus-equipped products will be able to provide a variety of functions. For example, interconnected appliances like a hot water heater and dishwasher will be able to communicate with each other so that the dishwasher will not turn on until the water heater sends a message saying that the water is hot enough to wash the dishes. Other examples include the ability to preset your lights, TV and furnace to turn on and off at times you specify.

BASIC TRAINING

by Clayton Walnum

Programming a computer can be a rewarding hobby. It puts into your hands the ability to make your machine do what *you* want. Further, programming is a good mental exercise; getting a program to run properly is a lot like trying to complete the Sunday crossword puzzle—with the difference that you have something to show for your efforts, something you can use.

And, contrary to what you may have heard, programming a computer is not that difficult. It *can* be difficult. It all depends upon the level of expertise you wish to obtain. Simple programs can be written after only an hour or two of study. Of course, programming a fast-action arcade game may not be possible without years of programming experience.

For the most part, though, you can quickly develop the skills required to write useful programs. This column is dedicated to getting those of you with no programming experience up to speed as fast as possible. We'll cover all the topics necessary to make you into a proficient BASIC programmer, starting at the very beginning and working our way through to the more difficult topics only when you are ready for them.

What Is a Program?

To write a program we first have to understand what it is. Those of you who have just purchased your first computer (and even some of you who have owned a computer for a while) may have a lot of misconceptions about what a program is. You probably think of a program as sheet after sheet of high-tech hieroglyphics that can make sense only to "techies." The fact is that a program is nothing more than a list of instructions. These instructions are placed in the order necessary to accomplish a particular task. A program, in fact, is not unlike a recipe.

Think about the steps you must complete to bake a cake. Now imagine that you must tell someone else how to bake that cake. You might tell them to go to the pantry and get out the cake mix, place the cake mix into the bowl with the other ingredients, mix the ingredients, pour the batter into a pan, then place the pan in the oven which has been preheated to the correct temperature.

A computer program is much the same, but instead of telling someone how to bake a cake, you're telling the machine how to perform whatever task it is that you want it to do. For example, let's say that we want the computer to take two numbers from the user, total them and then print the results. The steps we need to accomplish might look like this:

```
STEP 1: CLEAR THE SCREEN
STEP 2: ASK FOR THE FIRST
        NUMBER
STEP 3: GET THE FIRST NUMBER
STEP 4: ASK FOR THE SECOND
        NUMBER
STEP 5: GET THE SECOND
        NUMBER
STEP 6: ADD THE NUMBERS
STEP 7: PRINT THE TOTAL
```

This all looks very logical, right? There's nothing "high-tech" about any of the above instructions. All we have to do to convert the preceding list of instructions into a computer program is to write them in a way that the computer can understand. For example, we can't tell the computer "Clear the screen" because it doesn't understand our language. We need to use a language that it does understand, and one of those languages is BASIC.

In BASIC, one of the instructions to clear the screen is "GRAPHICS 0." Here is the list of instructions translated into BASIC:

```
10 GRAPHICS 0
20 PRINT "ENTER THE FIRST NUMBER"
30 INPUT NUMBER1
40 PRINT "ENTER THE SECOND NUMBER"
50 INPUT NUMBER2
60 TOTAL=NUMBER1+NUMBER2
70 PRINT TOTAL
```

If you type the above program into your computer (go ahead and do it, if you like) and then run it, you would get the following on your screen (items typed by the user are shown in italics):

ENTER THE FIRST NUMBER
?12
ENTER THE SECOND NUMBER
?13
25

As you can see, a computer program is not very different from any other set of instructions. In fact, even without knowing BASIC, you can generally see what is going on in the program. It's just a matter of translating each step into the language the computer can understand. Of course, in order to do that translation, you have to have a "dictionary" of words that the computer knows. Each of the words in the BASIC language is called a "keyword" or a "reserved word," and a list of them is shown in Table 1. Take a look at this list, but don't worry if you can't figure out what some of the words mean. We'll get to them all sooner or later.

ABS	GOTO	PUT
ADR	GRAPHICS	RAD
AND	IF	READ
ASC	INPUT	REM
ATN	INT	RESTORE
BYE	LEN	RETURN
CLOAD	LET	RND
CHR$	LIST	RUN
CLOG	LOAD	SAVE
CLOSE	LOCATE	SETCOLOR
CLR	LOG	SGN
COLOR	LPRINT	SIN
COM	NEW	SOUND
CONT	NEXT	SQR
COS	NOT	STATUS
CSAVE	NOTE	STEP
DATA	ON	STICK
DEG	OPEN	STRIG
DIM	OR	STOP
DOS	PADDLE	STR$
DRAWTO	PEEK	THEN
END	PLOT	TO
ENTER	POINT	TRAP
EXP	POKE	USR
FOR	POP	VAL
FRE	POSITION	XIO
GET	PRINT	
GOSUB	PTRIG	

Table 1.

The list of words in Table 1 may seem large, but consider that they're all you need to do practically anything you want with your computer. That's pretty amazing, don't you think?

Three Modes

If you own an XL or XE computer, the BASIC language is built into your machine. To access BASIC, all you have to do is turn on the machine. You should then see a blue screen with the word "READY" displayed. READY means that BASIC is standing by, ready to process your next instruction. The only way to get rid of BASIC is to hold down the Option key when turning on the computer.

If you own an older model of Atari—a 400 or 800—BASIC comes on a cartridge. To access BASIC, you must plug the cartridge into the machine before you turn it on.

The BASIC language operates in three modes: immediate, deferred and execution. You are using the immediate mode whenever you type an instruction that isn't prefaced with a line number. For example, type the following line and press Return:

PRINT "HI THERE!"

As soon as you pressed Return, the computer performed your instruction, printing the words "HI THERE!" on the screen. You are in the deferred mode whenever you type instructions that begin with a line number. The number tells BASIC that the instruction is not to be performed right away, but is instead to be stored in the computer's memory for later use. Type in the line above again, but this time put a line number in front of it like this:

10 PRINT "HI THERE!"

When you press Return this time, nothing happens on the screen. The computer has stored the line in memory and is waiting for you to tell it when you want it performed.

Whenever you type the BASIC keyword "RUN," the machine enters the execution mode; that is, it begins to perform whatever instructions you may have previously stored in the machine's memory. Type RUN now and see what happens. The words "HI THERE!" should appear on your screen as the computer performs the instruction you entered previously in the deferred mode.

Line Numbers

What's all this talk about line numbers? Remember that a program is a series of instructions that must be performed in a certain order. The computer knows the general order of the instructions by the line numbers included with each instruction. That's not to say that every BASIC program is performed starting with the first line and working its way one line at a time to the last line. This isn't true because there are ways to change the order in which the instructions are executed. But the line numbers do organize the way the program will work in a general way. One thing is for sure: the first line of your program will always be the first line executed.

Line numbers can be any number from 0 to 32727. As you write your program, though, you'll want to leave "space" between each line number so that you can insert lines between existing ones. For example, take a look at this program:

```
0 GRAPHICS 0
1 PRINT "ENTER FIRST NUMBER"
2 PRINT "ENTER SECOND NUMBER"
3 INPUT NUMBER2
4 TOTAL=NUMBER1+NUMBER2
5 PRINT TOTAL
```

Do you see something wrong here? If you compare this program to the similar one above, you'll see that we forgot the line "INPUT NUMBER1." Because that line is missing, the program won't run properly. How can we fix it? Outside of manually changing the numbers for each line, we can't. But suppose

we had written the program without using consecutive numbers, like this:

```
0 GRAPHICS 0
10 PRINT "ENTER FIRST NUMBER"
20 PRINT "ENTER SECOND NUMBER"
30 INPUT NUMBER2
40 TOTAL=NUMBER1+NUMBER2
50 PRINT TOTAL
```

To fix the program, now all we have to do is type the following line:

15 INPUT NUMBER1

Because of the line numbers, the computer knows exactly where this line belongs in the program, and it'll put it there for us, giving us this program:

```
0 GRAPHICS 0
10 PRINT "ENTER FIRST NUMBER"
15 INPUT NUMBER1
20 PRINT "ENTER SECOND NUMBER"
30 INPUT NUMBER2
40 TOTAL=NUMBER1+NUMBER2
50 PRINT TOTAL
```

You can see now how important it is to number a program in such a way as to allow changes later on. Almost all the programs you'll write will require some changes and polishing to get them to do exactly what you want.

Constants and Variables

Before we can understand exactly what's going on in our program, we have to learn about two types of data: constants and variables. A constant is a piece of program data that never changes. An example from our program is the 0 that follows the command GRAPHICS and the words "ENTER FIRST NUMBER" that follow our first PRINT statement. The former is a "numerical" constant, and the latter is a "string" constant. (A string is defined as a series of characters.) Nothing we can do in the program can change these constants; the only way to change them is to edit the program itself.

A variable is a piece of data that *can* be changed within our program. If you've taken algebra, you've already had some experience with variables. Basically (no pun intended), they are holding places for data that are identified by a name. For example, in line 15 of our example program we have a variable named NUMBER1. You can think of NUMBER1 as a box with enough space to hold a number—any number at all. In this case, NUMBER1 will hold whatever number the user types in (you'll see why later). We don't know in advance what that number might be, and we don't care. NUMBER1 is a numerical variable. There are string variables too, but we won't talk about them just yet.

What's Going On?

Now that we know a little about how BASIC works, let's take a look at the program we just wrote and see what it is doing.

Line 0 in the program tells the computer to go into graphics mode 0. The Atari computers have many different graphics modes, each with its own number, but we won't talk about many of them for a while yet. It's enough to know that Graphics 0 is Atari's standard "text" mode; that is, the mode that is used strictly for printing words or numbers on the screen, rather than drawings. When you first turn on your machine, it automatically comes up in graphics mode 0; in fact, when we ran the above program, we were already in graphics 0. So why did we have to instruct the computer to go to that mode again? Actually, we didn't. The program would have run fine without that line, with the exception that the screen would not have cleared first. That's one of the side effects of the GRAPHICS command. It always clears the screen.

Line 10 in the program prints our first "prompt." When programming you have to remember to tell your program's user what you expect him to do. Our program would work fine without printing the prompts. But without the prompts, all we'd see on the screen would be the question marks (refer back to the figures in the "What is a program?" section to see the question marks I'm referring to); we'd have to know ahead of time what the computer was waiting for.

The PRINT command is a very sophisticated instruction, one that allows many variations. The form shown in this line is the simplest one. The text within the quotation marks is displayed on the screen, and the cursor is moved to the next line. (The cursor always marks the place where the next PRINT statement will print.)

Line 15 uses the INPUT command to tell the computer to get some information from the keyboard. It prints a question mark, then waits for the user to type something. Whatever number the user types will be stored in the variable that follows the word "INPUT." In our case, the variable is the numerical variable NUMBER1. (If you type something other than a number, you'll get an error. Numerical constants can only hold numbers, nothing else.)

Line 20 prints our next prompt.

Line 30 does the same thing as line 15, but this time the number that is typed is stored in the numerical variable NUMBER2.

Line 40 takes the two numbers stored in NUMBER1 and NUMBER2 and places their sum in the numerical variable TOTAL. You'll notice that there are no BASIC keywords, or commands, here. This line is an example of a mathematical expression. In BASIC, an expression is calculated working from right to left. In other words, first NUMBER1 and NUMBER2 are added together; then the result is placed in the variable TOTAL. Expressions like this can be confusing to those who are familiar with algebra because the equals sign in this case doesn't mean "equals"; rather it should be interpreted as "gets." In English, line 40 would read "the numerical variable TOTAL gets the sum of the numerical variables NUMBER1 and NUMBER2." This type of expression is called an "assignment" statement because we are assigning a value to a variable.

There are five arithmetic operators you can use. They are:

+	addition
−	subtraction
×	multiplication
/	division
∧	exponentiation

Line 50 uses our handy print statement to display the value of TOTAL on the screen. Notice that this time we haven't used any quotation marks after the word "PRINT." Leaving out the quotation marks tells the computer that the word after the PRINT is not a string constant, but rather a variable whose value we want to know. The number stored in TOTAL is printed on the screen. Confused? Let's say that the user typed in 5 and 35 as NUMBER1 and NUMBER2. TOTAL would then be given the value 40. Contrast the following examples of the PRINT statement:

```
PRINT "TOTAL" <------- You type this
TOTAL <--------------- You get this
PRINT TOTAL <--------- You type this
40 <------------------ You get this
```

See the difference the quotation marks make?

Wrap Up

That's enough information for you to ponder this month. Study what you've learned, and next month we'll continue our exploration of BASIC programming.

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_____

NAME_____

ADDRESS_____

CITY_____ STATE_____ ZIP_____

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

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

Offer expires July 28, 1989 CJRWA

M/L EDITOR

For use in machine-language entry.

by Clayton Walnum

The two-letter checksum code preceding the line numbers here is *not* a part of the BASIC program. For more information, see the "BASIC Editor II" elsewhere in this issue.

M/L Editor provides an easy method to enter our machine-language listings. It won't allow you to skip lines or enter bad data. For convenience, you may enter listings in multiple sittings. When you're through typing a listing with M/L Editor, you'll have a complete, runnable object file on your disk.

There is one hitch: It's for disk users only. My apologies to those with cassette systems.

Listing 1 is M/L Editor's BASIC listing. Type it in and, when it's free of typos, save a copy to disk, then run it.

On a first run, you'll be asked if you're starting a new listing or continuing from a previously saved point. Press S to start, or C to continue.

You'll then be asked for a filename. If you're starting a new listing, type in the filename you want to save the program under, then press RETURN. If there's already a file by that name on the disk, you'll be asked if you wish to delete it. Press Y to delete the file, or N to enter a new filename.

If you're continuing a file, type in the name you gave the file when you started it. If the program can't find the file, you'll get an error message and be prompted for another filename. Otherwise, M/L Editor will calculate where you left off, then go on to the data entry screen.

Each machine-language program in ANALOG Computing is represented by a list of BASIC data statements. Every line contains 16 bytes, plus a checksum. Only the numbers following the word DATA need to be considered.

M/L Editor will display, at the top of the screen, the number of the line you're currently working on. As you go through the line, you'll be prompted for each entry. Simply type the number and press Return. If you press Return without a number, the default is the last value entered.

This feature provides a quick way to type in lines with repetitions of the same number. As an added convenience, the editor will not respond to the letter keys (except Q for "quit"). You must either enter a number or press Return.

When you finish a line, M/L Editor will compare the entries' checksums with the magazine's checksum. If they match, the screen will clear, and you may go on to the next line.

If the checksums *don't* match, you'll hear a buzzing sound. The screen will turn red, and the cursor will be placed back at the first byte of data. Compare the magazine listing byte by byte with your entries. If a number is correct, press RETURN.

If you find an error, make the correction. When all data is valid, the screen will return to gray, and you'll be allowed to begin the next line.

Make sure you leave your disk in the drive while typing. The data is saved continuously.

You may stop at any time (except when you have a red screen) by entering the letter *Q* for byte 1. The file will be closed, and the program will return you to BASIC. When you've completed a file, exit M/L Editor in the same way.

When you've finished typing a program, the file you've created will be ready to run. In most cases, it should be loaded from DOS via the L option. Some programs may have special loading instructions; be sure to check the program's article.

If you want the program to run automatically when you boot the disk, simply name the file AUTORUN.SYS (make sure you have DOS on the disk.).

LISTING 1: BASIC LISTING

```
AZ 10 DIM BF(16),N$(4),A$(1),B$(1),F$(15)
   ,F1$(15)
LF 11 DIM MOD$(4)
BN 20 LINE=1000:RETRN=155:BACK$P=126:CHK$
   UM=0:EDIT=0
GO 30 GOSUB 450:POSITION 10,6:? "Start or
    Continue? ";:GOSUB 500:? CHR$(A)
ZG 40 POSITION 10,8:? "FILENAME";:INPUT F
   $:POKE 752,1:? " "
FE 50 IF LEN(F$)<3 THEN POSITION 20,10:?
   "  ";:GOTO 40
NF 60 IF F$(1,2)<>"D:" THEN F1$="D:":F1$(
   3)=F$:F$=F1$:GOTO 80
KL 70 F1$=F$
TN 80 IF CHR$(A)="S" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$:TRAP 110
HQ 100 FOR X=1 TO 16:GET #2,A:NEXT X:LINE
   =LINE+10:GOTO 100
WM 110 CLOSE #2:OPEN #2,9,0,F1$:GOTO 170
VT 120 TRAP 160:OPEN #2,4,0,F1$:GOSUB 440
   :POSITION 10,10:? "FILE ALREADY EXISTS
   !!";:POKE 752,0
ZU 130 POSITION 10,12:? "ERASE IT? ";:GOS
   UB 500:POKE 752,1:? CHR$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
    CLOSE #2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
   HEN 130
BH 160 CLOSE #2:OPEN #2,8,0,F1$
IE 170 GOSUB 450:POSITION 10,1:? "NOW ON
   LINE: ";LINE:CHK$UM=0
GH 180 L1=3:FOR X=1 TO 16:POSITION 13+(X(
   10)+12*(X)9),X+2:POKE 752,0:? "BYTE #"
   ;X;": ";:GOSUB 310
KH 190 IF EDIT AND L=0 THEN BYTE=BF(X):GO
   TO 210
FY 200 BYTE=VAL(N$)
OZ 201 MOD$=N$
BU 210 POSITION 22,X+2:? BYTE;"  "
YZ 220 BF(X)=BYTE:CHK$UM=CHK$UM+BYTE*X:IF
    CHK$UM>9999 THEN CHK$UM=CHK$UM-10000
MS 230 NEXT X:CHK$UM=CHK$UM+LINE:IF CHK$U
   M>9999 THEN CHK$UM=CHK$UM-10000
IG 240 POSITION 12,X+2:POKE 752,0:? "CHEC
   K$UM: ";:L1=4:GOSUB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
SY 270 POSITION 22,X+2:? C;"   "
IL 280 IF C=CHK$UM THEN 300
DI 290 GOSUB 440:EDIT=1:CHK$UM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GOSUB 500:IF (A=ASC("Q") OR A=ASC(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACK$P AND (A<4
   8 OR A>57) THEN 320
DX 331 IF A=RETRN AND N$="" THEN N$=MOD$
TD 335 IF A=RETRN AND L=0 AND X>1 THEN 35
   0
JR 340 IF ((A=RETRN AND  NOT EDIT) OR A=B
   ACK$P) AND L=0 THEN 320
DW 350 IF A=RETRN THEN POKE 752,1:? " ":R
   ETURN
GG 360 IF A<>BACK$P THEN 400
SA 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
AS 380 N$=""
RE 390 ? CHR$(BACK$P);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WK 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHICS 0:END
YT 430 GOSUB 440:POSITION 10,10:? "NO SUC
   H FILE!";:FOR X=1 TO 1000:NEXT X:CLOSE
   #2:GOTO 30
FD 440 POKE 710,48:SOUND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:SOUND 0,0,0,0:RETURN
MY 450 GRAPHICS 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 STEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 STEP 2:POKE DL+X,0
   :NEXT X
ZW 480 POKE DL+41,65:POKE DL+42,PEEK(560)
   :POKE DL+43,PEEK(561):POKE 87,0
AC 490 POSITION 2,0:? "analog ml editor":
   POKE 559,34:RETURN
HZ 500 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1
   :RETURN
```

WE'RE BACK!

SAN JOSE COMPUTER, 'THE ATARI STORE', is the largest seller of ATARI products and now we're back in the 8-Bit market to serve you better.

1200XL • 64K RAM • XE COMPATIBLE **$49.95**
Reconditioned, 90 day warranty

1025 PRINTER $89.95
Reconditioned, 90 day warranty

COMPUTERS

800 .....$69.95
400 .....$29.95
600XL $49.95

Reconditioned, 90 day warranty

MISCELLANEOUS

850 INTERFACE $89.95
MONO MONITOR ... $39.95
TRACKBALL $9.95
JOYSTICK 20' EXT. $ .99

Reconditioned, 90 day warranty

PRINTERS

Reconditioned

825 .... $79.95
1029 .. $129.95

Reconditioned, 90 day warranty

Brand new in box!

1027. Letter Quality Direct Connect ...$89.98
1020.. Printer Plotter ...$18.98
1020 plotter pins $ .98

CARTRIDGES FOR 800, XL, XE

BASIC CARTRIDGE	$4.95	ROBOTRON	$19.98	NECROMANCER	$19.98
QIX	$4.95	TENNIS	$19.98	RESCUE ON FRACTALUS	$19.98
TURMOIL	$4.95	FINAL LEGACY	$19.98	BALLBLAZER	$19.98
PAC-MAN (no box)	$4.95	MARIO BROS.	$19.98	BLUE MAX	$19.98
DONKEY KONG (no box)	$4.95	DONKEY KONG JR.	$19.98	STAR RAIDERS II	$19.98
GORF (400,800)	$4.95	JUNGLE HUNT	$19.98	DAVID'S MIDNIGHT MAGIC	$19.98
DEMON ATTACK (400,800)	$4.95	MOON PATROL	$19.98	ARCHON	$19.98
DELUXE INVADERS	$4.95	BATTLEZONE	$19.98	GATO	$24.98
JOURNEY TO THE PLANETS	$4.95	FOOD FIGHT	$19.98	ACE OF ACES	$24.98
MATH ENCOUNTER	$7.98	HARDBALL	$19.98	LODE RUNNER	$24.98
SKY WRITER	$14.95	FIGHT NIGHT	$19.98	BARNYARD BLASTER (XE gun)	$24.98
FOOTBALL	$14.95	ONE ON ONE BASKETBALL	$19.98	ATARI LAB LIGHT MODULE	$29.98
DEFENDER	$14.95	DESERT FALCON	$19.98	ATARI LAB STARTER KIT	$39.98

DISK SOFTWARE FOR 800, XL, XE

DAVID'S MIDNIGHT MAGIC	$4.98	CROSSCHECK	$4.98	SPIDERMAN	$4.98
ZORRO	$4.98	MOLECULE MAN	$4.98	HULK	$4.98
BANDITS (48K 400,800)	$4.98	CRYSTAL RAIDER	$4.98	VISICALC	$24.98
PROTECTOR II	$4.98	DESPATCH RIDER	$4.98	BOOKKEEPER W/ numeric keypad	$29.98
CLAIM JUMPER	$4.98	MISSION ASTEROID	$4.98	GET RICH	$39.98
SYNTREND	$4.98				

810 DISK DRIVES **$129.**

BASIC TUTOR .. (2 BOOKS) .. $4.98
SERIAL 8-BIT I/O CABLE $5.95

SAN JOSE COMPUTER

THE ATARI STORE

Sunrise Plaza 640 Blossom Hill Rd. San Jose, CA 95123
(408) 224-8575 • BBS (408) 224-0952

5.25" DISKS
20 CENTS EA.*

QTY.	PRICE
10	$4.00
100	$29.95
*1000	$200

MAJORITY ARE UNNOTCHED CONTAINING OLD SOFTWARE

SHIPPING: ADD $5.00 TO ALL ORDERS. AIR AND INTERNATIONAL SHIPPING EXTRA. THAT'S IT.
WARRANTY: 90 DAY WARRANTY ON ALL ITEMS. **TAX:** CALIFORNIA RESIDENTS ADD 7% SALES TAX.
PREPAYMENT: USE VISA, MASTERCARD, MONEY ORDER, CASHIER'S CHECK OR PERSONAL CHECK.
PERSONAL CHECK MUST CLEAR PRIOR TO SHIPMENT. **C.O.D.:** CASH, CASHIER'S CHECK OR M.O. ONLY.
Prices subject to change without notice. Brand and/or product names are trademarks or registered trademarks of their respective holders.
Ad produced on an ATARI ST.

CIRCLE #105 ON READER SERVICE CARD.

End User

Spring is here in some parts of the country, and, in many respects, Atari is also entering its springtime. Computers are more plentiful around the country. Dealers seem more optimistic with Atari's new attitude. Mail-order sales have been stopped for quite a while, allowing retailers a chance to make a decent profit on the computers they sell. In other words, retailers are no longer asked to provide support to users who have purchased their computers elsewhere.

And users seem generally more optimistic too. Atari has started listening to users and responding to their concerns. To many people, Atari no longer appears to be a humbug. On major telecommunications services like CompuServe, a direct line of communication has been established to Atari President Sam Tramiel and Vice President Sig Hartmann. Advertising by Atari is said to be right around the corner, now that the increased supply of product can feed the additional demand for computers. 1989 may indeed be the year that Atari turns things around.

Desktop-Accessory Accessory

Occasionally a program comes along that fulfills a need simply and elegantly. These niche programs often come from individuals or small companies and are created by user/programmers that initially wanted to create a solution for their own needs. Such is the case with a new utility program from CodeHead Software called *MultiDesk*.

Normally, when you boot the ST, GEM loads all desktop-accessory programs it finds (that is, programs with a filename extension of .ACC). You can have up to six desktop accessories loaded at once when the computer boots up. Actually, only the first six .ACC files are loaded into GEM, and the rest are ignored. If you want to access an accessory that was not originally loaded, you must ensure that the one you want is one of the six accessory files and then reboot the machine.

MultiDesk allows you to use any desktop-accessory program on your ST at any time without rebooting the computer. The size of

the ST's memory places the only limit on the number of accessories that can be loaded at one time. Actually, *MultiDesk* works both as a desktop accessory and as a regular GEM program.

As a desktop accessory, MULTIDESK.ACC is loaded in the usual manner at boot-up with up to five other accessories. Clicking on *MultiDesk* from the Desk menu calls up the *MultiDesk* dialog box where you decide which accessories you want to be loaded into *MultiDesk* itself. Up to 32 accessories can be named and included with *MultiDesk*.

The program reserves a portion of memory for all of the accessories you have chosen, and once your selections have been made, just enough memory is set aside: no more and no less. You can create a configuration file from within *MultiDesk* so that the next time you boot the machine, your selected accessory list will be automatically loaded. Also, to make using *MultiDesk* even more convenient, you can choose an option that will bring the mouse pointer directly to the list of accessories when you click on the MultiDesk accessory.

If for some reason, you want to be able to load more than 32 desktop accessories at once, *MultiDesk* will let you do it. By renaming MULTIDESK.ACC to other names (retaining the .ACC extension), you can have a maximum of 192 accessories available simultaneously (6 accessory slots times 32 each). I can't imagine anyone needing access to that many accessories, but some power users may find it necessary, assuming they have the megabytes to spare. You can even have *MultiDesk* nested as an accessory inside another version of *MultiDesk*—up to a reported 130 levels. Sheesh!

As a regular program, *MultiDesk* is also useful. Just renaming the program to MULTIDESK.PRG and double-clicking on it from the desktop, gives you access to any desktop accessory, whether it has already been loaded or not. When returning to the desktop, the accessories become unavailable again. Being able to execute any and all desktop accessories from the desktop without having to first install them is one of *MultiDesk*'s best features.

CodeHead Software is the company that brought us *G+Plus*, the excellent replacement for Atari's ineffective and memory-hungry GDOS. For $32, *MultiDesk* is an excellent tool for both programmers and normal users. In fact, I would call it the ultimate accessory program. I'm looking forward to future innovative and useful programs from CodeHead.

Regent News

Regent Software, makers of *Regent Word II* and *Regent Base II* for the ST has recently announced a new *Student Edition* of *Regent Word II*. This word-processing program is a new version of the GEM-based *Regent Word II* and is aimed at the educational and small-business market. According to Regent Software, *Student Edition* is one of the few ST word-processors that will work on a standard 512K ST computer.

The program has several new features including double-column printing, 40,000 word spelling checker (expandable up to 100,000 words) and support for dozens of printers. *Regent Word Student Edition* sells for $25, is not copy protected and comes with a complete user manual.

At the same time, Regent announced that *Regent Word III* will be appearing sometime this year. The new advanced word-processing program will feature the company's own graphics operating system, RDOS, and will allow the use of Macintosh and GEM (GDOS) fonts in a MacWrite-style word-processing environment. In addition, the program will allow graphics from *DEGAS Elite* and other drawing programs to be included in its documents. More details on *Regent Word III* will be available later in the year.

Regent Software is also selling a new, inexpensive product called *Megatouch*. The product consists of a set of springs or "keyboard stiffeners" that make the 520 or 1040ST keyboards feel like a Mega ST keyboard.

Megatouch is easy to install. Just pop off each keycap on your ST keyboard, place the spring in position and re-insert the keycap. If you have not memorized the QWERTY layout of your keyboard, it's best to do one key at a time so that you'll be able to put the keycaps back in the correct places.

Megatouch works surprisingly well. I've performed this modification to my 520ST and am impressed with the results. The keyboard still doesn't feel like an IBM keyboard (the best I've used), but it is a definite improvement over the mushy feel of the original ST keyboard. For $12, *Megatouch* is worth the price and highly recommended.

For information, Regent Software can be reached at (213) 439-9664, or write to Regent Software, P.O. Box 14628, Long Beach, CA 90803.

Not Another Joystick

Since my early days as an Atari Video Computer System (VCS) owner nearly ten years ago, I have been on a quest for the ultimate joystick. I know many true gamers share this obsession and, like me, can probably attest to a collection of joysticks that numbers in the dozens.

Wico has been the king of the joystick companies for many years. Although not all of their joystick creations are still in production, they can still be readily found in toy and game stores. Wico sticks are classics. They are big, rugged and solidly built. Some have bat handles, others have knob handles and some have interchangeable ones.

Although I admire Wico sticks and own just about every one they ever made. I always felt that they were a little on the large size, especially for my average-sized hands. Sure, they had a fire button at the top of the handle as well as on the base, but I often had difficulty using the sticks when the going got rough. It seemed that response-time suffered as a result.

One of my favorite sticks is the Suncom *StarFighter*. What I like most about it is that it is small. It has both a small base that is easy to grab and a short stick that allows fast action. Unfortunately, it has a "cheap plastic" feel to it that comes from the construction and lack of tactile feedback. Electrical contacts rather than microswitches are used in this stick so no click is heard when the handle is moved. Also, I have worn out two of these babies over the years. Another problem

by Arthur Leyenberger

with the Suncom is that, after prolonged use, my gripping hand feels fatigued.

Another favorite stick is the Epyx *500XJ*. This too is a small stick, but the base conforms to the shape of your hand and is less fatiguing. There is also a groove for the thumb on the base, and the trigger button is strategically located under the tip of the index finger. This stick is made to be held with the left hand while the right hand's fingers perform the action. The Epyx *500XJ* has a sturdy design and feels solid to the grip.

A while back, I came across a new joystick and have been using it on and off since. It's the *Ergostick* from Wico. The *Ergostick* looks like an Epyx *500XJ* but with something different added. That difference is the covering of the base—a slightly sticky, soft covering—with a pleasing feel to it.

The *Ergostick* appears to be a quality design and has held up well under occasional heavy use. Microswitches have been used which provide the all important feedback and the shaft has a steel center. The skin-like material covering the base feels a little odd at first, but I soon came to appreciate its gripping characteristics.

One of the best features of the *Ergostick* is that my left hand did not become fatigued after prolonged use. I can play longer with this stick because I don't need to grip it as tightly. This, I presume, is due to the soft covering. In addition, moving the shaft to the diagonal positions does not require extra attention as it does with some other joysticks.

With some other joysticks, such as the Suncom mentioned above, moving to the diagonal positions doesn't seem as precise as when moving the handle to the horizontal and vertical positions. This is frustrating and slows the overall response of the stick. The *Ergostick* feels comfortable when moving to all directions, and the handle has a short "throw," which I prefer.

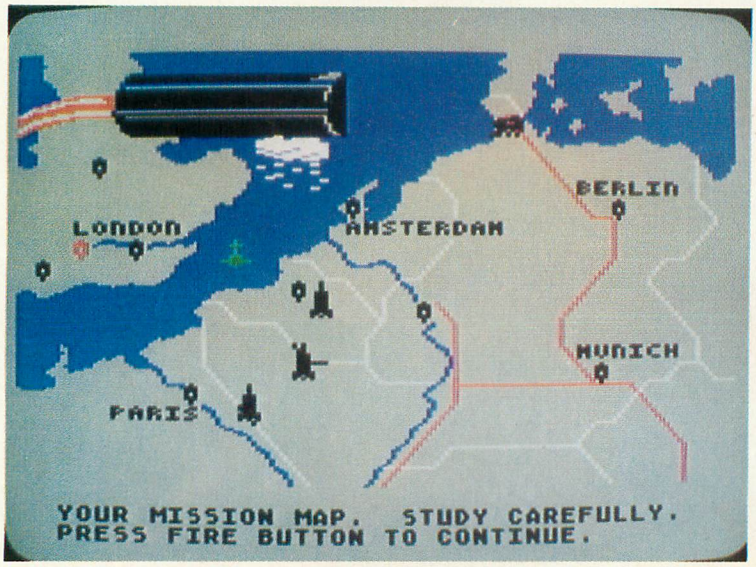
The *Ergostick* sells for $25 and is available from Wico Corporation, 6400 West Gross Point Rd., Niles, IL 60648. Phone (312) 647-7500.

Arthur Leyenberger is a human factors psychologist and freelance writer living in New Jersey.

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

```
THE A.N.A.L.O.G. #72 DISKETTE CONTAINS 18
MAGAZINE FILES.  THEY ARE LISTED BELOW.

SIDE 1:

FILENAME.EXT   LANG.    LOAD   COMMENTS
------------   -----    ----   --------

BBKCP    .OBJ  ML       (#4)   COMMAND PROCESSOR
CLOWN    .OBJ  ML       (#3)   CRAZY CLOWN JUMPER
CONVERT1.BAS   BASIC    LOAD   GRAPHICS CONVERTER
CONVERT2.BAS   BASIC    LOAD   GRAPHICS CONVERTER
INVADERS.BAS   BASIC    LOAD   GAME DES. WORKSHOP


SIDE 2:

FILENAME.EXT   LANG.    LOAD   COMMENTS
------------   -----    ----   --------

BBKCP    .M65  MAC/65   LOAD   COM. PROC. SOURCE
CLOWN    .AMA  AMAC     (#2)   CLOWN SOURCE 1
BDATA    .LNK  AMAC     (#2)   CLOWN SOURCE 2
PAGESIX  .LNK  AMAC     (#2)   CLOWN SOURCE 3
LINK1    .LNK  AMAC     (#2)   CLOWN SOURCE 4
PLATFORM.LNK   AMAC     (#2)   CLOWN SOURCE 5
VBISUB   .LNK  AMAC     (#2)   CLOWN SOURCE 6
SCORE    .LNK  AMAC     (#2)   CLOWN SOURCE 7
VBILINK  .LNK  AMAC     (#2)   CLOWN SOURCE 8
DOLLAR   .LNK  AMAC     (#2)   CLOWN SOURCE 9
LEVEL    .LNK  AMAC     (#2)   CLOWN SOURCE 10
FALLING  .LNK  AMAC     (#2)   CLOWN SOURCE 11
PMINT    .LNK  AMAC     (#2)   CLOWN SOURCE 12


TO LOAD YOUR A.N.A.L.O.G. DISK
------------------------------

1) INSERT BASIC CARTRIDGE (NOT REQUIRED FOR XL
   OR XE COMPUTERS)
2) TURN ON DISK DRIVE AND MONITOR
3) INSERT DISK IN DRIVE
4) TURN ON COMPUTER (XL AND XE OWNERS DO NOT
   HOLD DOWN OPTION KEY!)

WARNING: BEFORE YOU RUN A PROGRAM, READ THE
         APPROPRIATE ARTICLE IN THE MAGAZINE.


NOTE: ONLY PROGRAMS WITH THE ".BAS" OR ".OBJ"
      EXTENTION MAY BE RUN FROM THE MENU.
      OTHER PROGRAMS SHOULD BE LOADED AS
      INSTRUCTED IN THE LOADING NOTES AND MAY
      REQUIRE ADDITIONAL SOFTWARE AS LISTED
      BELOW.  HOWEVER, YOU SHOULD NOT ASSUME
      THAT EVERY FILE WITH THE PROPER FILE
      EXTENSION WILL RUN FROM THE MENU. YOU
      MAY HAVE TO MOVE CERTAIN PROGRAMS TO A
      DIFFERENT DISK TO OBTAIN CORRECT RESULTS.


 EXT   DESCRIPTION
 ---   -----------

.M65   REQUIRES THE OSS MAC/65 ASSEMBLER
.AMA   REQUIRES THE ATARI MACRO ASSEMBLER
.ASM   REQUIRES THE ATARI ASSEMBLER/EDITOR
.ACT   REQUIRES THE OSS ACTION! CARTRIDGE
.LGO   REQUIRES THE ATARI LOGO CARTRIDGE
.SYN   REQUIRES THE SYNAPSE SYN ASSEMBLER
.STB   REQUIRES ST BASIC

LOADING NOTES
-------------

LOAD BASIC PROGRAM:    LOAD "D:FILENAME.EXT"
ENTER BASIC PROGRAM:   ENTER "D:FILENAME.EXT"
LOAD MAC/65 PROGRAM:   LOAD #D:FILENAME.EXT
ENTER ASM/ED PROGRAM:  ENTER #D:FILENAME.EXT
LOAD LOGO PROGRAM:     LOAD "D:FILENAME.EXT"
LOAD SYN/AS PROGRAM:   LOAD "D:FILENAME.EXT"


#1: SEE ACTION! MANUAL.
#2: SEE ATARI MACRO ASSEMBLER MANUAL.
#3: MAY ALSO BE LOADED FROM DOS USING THE "L"
    OPTION OF THE DOS MENU.
#4: THIS FILE SHOULD BE TRANSFERRED TO ANOTHER
    DISK AND RENAMED "AUTORUN.SYS".
#5: SEE ST BASIC MANUAL.
```



ACE OF ACES

by Accolade
Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
XL/XE Cartridge $34.95

Reviewed by Matthew J.W. Ratcliff

Ace of Aces puts you in the cockpit of a De-Havilland Mosquito, in the middle of World War II. Take on the role of a would-be ace pilot, a member of Britain's Royal Air Force, in one or more of four scenarios.

Ace of Aces provides a simulation of assaulting the Nazi U-boats (submarines), outrunning V-1 buzz bombs, chasing down bombers, or stopping POW trains before reaching enemy lines. The user may begin with practice or a full mission. To practice, any one scenario may be chosen and played immediately. If a mission is enabled, one or more scenarios may be chosen. Each option presents an intelligence report including weather, expected adversaries and recommended armaments. Next, the user must choose the appropriate cluster of bombs, machine-gun ammo, rockets and spare fuel tanks. It seems that the only trade-offs encountered here are between bombs and rockets.

Once all mission, fire power and fuel options have been enabled, the air raid siren goes off as a black-and-white photo sequence is displayed, prepping the aircraft for takeoff. This is a nice graphic sequence, coupled with a very obnoxious sound effect. The fire button mercifully sends control to the flight screen.

It's always cloudy in *Ace of Aces*. The pilot will never see the ground, not even moments before he crashes. In fact, the only time the ground or ground targets are seen is when the bomb-bay doors are open and the aircraft is on the perfect course for enemy interception. If the plane is off course, only the gray of cloud cover is seen.

The main play screen is split, the top displaying mostly clouds and the blue sky above, while the bottom is occupied by the cockpit. A hash mark appears on the compass, located at the left of the display, indicating the proper direction for the next target. When the hash mark is centered on the compass, the proper course is set. An artificial horizon is at the center of the cockpit, with the yoke displayed below. The yoke moves in response to the joystick—a nice effect.

At any time, en route to the next target, enemy ME109s may attack. They are difficult to kill due to sluggish joystick response. The enemy seems to jump about the screen and sometimes vanishes from the display. It's difficult to line up an enemy plane in the gun sights and shoot it down. Evasive maneuvers are generally a wasted effort. The clouds look nice, but a simpler display would have sufficed for more responsive game play.

There are five displays the user may view during a mission, which may be selected by pressing the keys 1 through 5. Using a clever fire button/joystick move, these screens may be viewed without the use of a keyboard. A navigational map may be viewed, which displays the current location of the pilot's plane, the ever-present pair of storm clouds (that you apparently fly in all the time), and locations of the next target(s).

Throttle, fire extinguisher, flaps, gear, and other controls are found in the engine room, which is available from either of the side views. At the bottom left of each view is an icon of the plane, called the intercom. When the center flashes red, it is time to move to the bomb-bay screen to attack a ground target. Other flashing areas on the intercom indicate problems, such as a fire, that must be attended to.

When attacking a POW train or U-boat, the target is seen through the open bomb-bay doors. It seems illogical to bomb a POW train, for fear that the prisoners, your allies, may be killed in the process. The goal is to bomb the cars marked with iron, and not red, crosses. It is a wasted effort to attempt bombing the tracks in front of the locomotive, since the game automatically breaks off the attack before this can be achieved. However, this approach would seem to be the most logically safe way to stop the train from taking prisoners across the border. The scenarios seem contrived, stifling creative thinking. The U-boats are sitting ducks on the surface of the water. They do not spot the Mosquito until its bomb-bay doors open, then they begin to dive. There are only precious moments to get off a successful bombing run before they are safely submerged.

The ultimate goal of this game is to fly all four scenarios in a single mission and return to base safely to become *Ace of Aces*. The graphics in this game are excellent, to the detriment of playability. A little less graphic effects in favor of a more responsive joystick in the air-to-air scenarios could have made this a champion game instead of a lackluster one. This author would much prefer to see *F-15 Strike Eagle*, from MicroProse, packaged in a cartridge.

MASTER MEMORY MAP PART X

by Robin Sherer

How to Read the Memory Map

Beginning users: Read the text that is printed in bold type only. These memory locations will be the easiest for you to use and usually don't involve assembly language.

Advanced users: Read everything! Many areas of memory are not of any practical use, but you can learn a lot about how a computer works by reading the boring parts.

The Character Set

The data for the regular Atari character set is stored in Locations 57344 through 58367. There are eight bytes for each character and 128 characters in all (for a grand total of 1024 bytes). But wait a minute. Doesn't the Atari have 256 characters? Yes, but the information for the regular characters is all the Atari needs to know to print inverse characters, so that's why there are only 128 character descriptions.

For lots of information on how the bytes are used to describe a character, and on the order of the characters within the character set, see CHBAS at Location 756.

The following program will use the character descriptions to put text on the screen in graphics mode 8:

```
100 GRAPHICS 8
105 SCRMEM=PEEK(88)+PEEK(89
)*256
110 DIM TEXT$(120)
120 PRINT "Start text in wh
at column (0-39)";
130 INPUT COL
140 PRINT "In what row (0-1
52)";
150 INPUT ROW
160 PRINT "Type in the text
 you want to print:"
170 INPUT TEXT$
180 CHSET=PEEK(756)*256
190 FOR CHAR=1 TO LEN(TEXT$
)
200 ATASC=ASC(TEXT$(CHAR,CH
AR))
210 NOINV=ATASC-128*(ATASC>
127)
220 INTRNL=NOINV-32*(NOINV<
96)+96*(NOINV<32)
230 FOR BYTE=CHSET+INTRNL*8
 TO CHSET+INTRNL*8+7
240 POKE SCRMEM+ROW*40+COL,
ABS(255*(ATASC>127)-PEEK(BY
TE))
250 ROW=ROW+1
260 NEXT BYTE
270 ROW=ROW-8
280 COL=COL+1
290 IF COL=40 THEN COL=0:RO
W=ROW+8
300 NEXT CHAR
310 PRINT
320 GOTO 120
```

Vectors and Vector Tables

What are vector tables? You remember that a vector is a pair of memory locations that holds the address of a routine. A vector *table* is, quite simply, a table of vectors. Thus, Locations 58368 through 58533 hold the addresses of various routines, mostly having to do with I/O or interrupts.

EDITRV
58368-58383 E400-E40F

This is the vector table for the screen-editor handler. For a description of its contents, along with the contents of the next four vector tables, see HATABS at Locations 794 through 831 (where we called it a "handler address table").

SCRENV
58384-58399 E410-E41F

The vector table for the display handler. See the note at EDITRV.

KEYBDV
58400-58415 E420-E42F

The vector table for the keyboard handler. See the note at EDITRV.

PRINTV
58416-58431 E430-E43F

The vector table for the printer handler. See the note at EDITRV.

CASETV
58432-58447 E440-E44F

The vector table for the cassette handler. See the note at EDITRV.

You will notice that the following 16 vectors are three bytes long rather than two. Why the extra byte? The first byte is a 6502 JMP instruction, while the address is in the second two bytes.

The purpose of these vectors may not be obvious to you (it wasn't to me). Atari knew that it would probably need to make changes to the OS at some point. It also wanted to make sure that old programs would still be able to work with these newer versions of the OS, even though some of the addresses would be different. The solution was to use vectors. That way, even though the addresses in the vectors would change, the addresses *of* the vectors would remain the same, and programs using these addresses would still work. The reason that some programs don't work with Version B of the OS is that these programs didn't use the vectors.

DISKIV
58448-58450 E450-E452

DISKIV is the initialization vector for the disk handler. It points to Location 60906.

DISKINV
58451-58453 E453-E455

This is the entry vector for the disk handler. It points to Location 60912.

CIOV
48454-48456 E456-E458

CIOV is the entry vector for CIO (Central Input/Output).

You can see CIO yourself by first setting up an IOCB (see Locations 832 through 959), and then using the following routine:

```
100 DIM ML$(7)
110 GOSUB 10000
120 CIO=USR(ADR(ML$),IOCB*1
6)
130 END
10000 FOR BYTE=1 TO 7
10010 READ INSTR
10020 ML$(BYTE,BYTE)=CHR$(I
NSTR)
10030 NEXT BYTE
10040 RETURN
10050 DATA 104,104,104,170,
32,86,228
```

The data is for this machine-language routine:

```
68      PLA
68      PLA
68      PLA
AA      TAX
2056E4 JSR  $E456
```

CIO expects the number of the IOCB you want to use, *times 16*, in the X register. That's why we have IOCB*16 in the preceding program. You should substitute the IOCB number you are using for IOCB. Remember to OPEN the IOCB first.

CIOV points to 58564.

SIOV
58457-58469 E459-E45B

This is the entry vector for SIO (Serial Input/Output).

SIOV points to 59737.

SETVBV
58460-58462 E45C-E45E

SETVBV is the entry vector for a routine that serves two purposes. First of all, as we saw at VVBLKD (548,549), it will set up VVBLKI and VVBLKD for us. Second, as we saw at CDTMA1 (550,551), it will also set up the vectors for the system timers. See VVBLKD and CDTMA1 for more information.

SETBV points to 59666 in Version A of the OS, and to 59629 in Version B.

SYSVBV
58463-58465 E45F-E461

This is the entry vector for the Stage 1 VBLANK routine. Unless you have your own routine, VVBLK1 (546,547) normally points here. See VVBLK1 and VVBLKD (548,549) for more information on VBLANK.

SYSVBV points to 59345 in Version A of the OS, and to 59310 in Version B.

XITVBV
58466-58468 E462-E464

XITVBV is the exit vector for the VBLANK routine. This is what VVBLKD points to unless you've changed it.

Use XITVBV to return to where the computer left off from when the VBLANK interrupt occurred. It points to 59710 in Version A of the OS, and to 59653 in Version B.

The following four vectors are designed for use by the OS only.

SIOINV
58469-58471 E465-E467

This is the initialization vector for SIO.

SENDEV
58472-58474 E468-E46A

SENDEV is the vector for the "send-able" routine.

INTINV
58475-58477 E46B-E46D

This is the initialization vector for the interrupt-handler routine.

CIOINV
58478-58480 E46E-E47O

CIOINV is the initialization vector for CIO.

BLKBDV
58481-584483 E471-E473

This is the entry vector for the blackboard mode, which is more commonly known as the "Atari memo pad" mode. Type BYE from BASIC, or turn on the computer with no cartridges or disk drives to see what I mean. This mode lets you type things on the screen without anybody caring what you type. In other words, you can press Return and nothing will happen. To get back to BASIC, press System Reset (this won't erase your BASIC program).

BLKBDV points to 61987.

WARMSEV
58484-58486 E474-E476

WARMSV is the entry vector for the warmstart routine. The OS jumps through here when System Reset is pressed.

WARMSV points to 61723.

In case these locations don't seem useful to you, try this:

X = USR(58484)

What you have just done is told the computer to go to 58484, which contains a machine-language instruction to go to the address in the next two memory locations. Since this routine is for what's called warmstart, the computer will now act just like you pressed System Reset. You use the other locations in this section just like this. Try it!

COLDSV
58487-58489 E477-E479

This, appropriately, is the entry vector for the coldstart routine. Whereas going through WARMSV only initializes the OS RAM, going through COLDSV initializes *all* RAM, meaning that any programs in memory will be erased. See COLDST at Location 580 for a way to hook COLDSV up to System Reset rather than WARMSV.

COLDSV points to 61733.

The following two vectors are designed for use by the OS only.

RBLOCKV
58490-58492 E47A-E47C

RBLOKV is the entry vector for the cassette "read-block" routine.

CSOPIV
58493-58495 E47D-E47E

This is the vector for the cassette "OPEN-for-input" routine.

VCTABL
58496-58533 E480-E4A5

VCTABL is a table of the initial values for the OS RAM vectors.

Now we're into the final part of the OS, which consists mostly of the various built-in handlers, interrupt routines, and so forth. What follows is a list of addresses for some of these routines, which can be useful to you in one of several ways. If you're a beginner, the list will provide you with an idea of exactly what the OS does. If you're a machine-language programmer then, along with the OS listing, the list will help you find the various routines so that you can see exactly how things are done. By studying the routines, you can also pick up on programming techniques (don't be afraid of the OS listing; it's really not that difficult to understand). Finally, if you really know what you're doing, you can rewrite the routines and put them in your own programs, customizing them to your own needs.

Most of the routines will *not* work without some kind of previous setup, so make sure you check the OS listing before you attempt to use them.

Please note that all the following addresses are for the original OS only. Some of them may be different in the newer versions. At the time of this writing, however, the OS listing is for the original version, and that is why these addresses are used.

CIO Routines

CIOINT
58534 E4A6

CIO's initialization routine.

CIO
58564 E4C4

The main CIO routine (includes the following routines).

CIOPEN
58633 E509

OPEN routine.

CICLOS
58675 E533

CLOSE routine.

CISTSP
58702 E54E

STATUS and special requests routine.

CIREAD
58729 E569

GET routine (GET character *and* GET record).

CIWRIT
58825 E5C9

PUT routine (PUT character *and* PUT record).

CIRTN1
58907 E61B

Return from CIO with the status in the Y register.

CIRTN2
58909 E61D

Return from CIO with the status in ICSTAZ (35).

COMENT
58941 E63D

Compute the handler entry point using HATABS (794) and COMTAB (59081).

GOHAND
59017 E689

Jump indirectly to the device handler. An indirect jump, in this case, means fooling the 6502 into thinking that the address you want to jump to is actually the one you want to RTS to. This involves playing with the stack and is a pretty neat trick you may want to look at.

DEVSRC
59038 E69E

Find a particular device in the handler address table.

COMTAB
59081 E6C9

This is a table of offsets into the handler entry point table for the desired device. It is used to find the correct vector for the given command.

Interrupt Handler Routines

IHINIT
59093 E6D5

Initialize the interrupt handlers.

PIRQ
59123 E6F3

Jump to the main IRQ handler routine through VIMIRQ (534,535). Unless you've changed it, VIMIRQ points to SYIRQ.

SYIRQ
59126 E6F6

This is the system's IRQ handler routine.

PNMI
59316 E7B4

This is the system's NMI handler routine.

System VBLANK Routines

SYSVBL
59345 E7D1

This is the immediate vertical blank routine (Stage 1 VBLANK).

SYSVB3
59400 E808

This is the Stage 2 VBLANK routine.

SETVBL
59666 E912

This routine can be used to set up vectors for your own VBLANK routines, and also for the system timers. See SETVBV at 58460.

XITVBL
59710 E93E

Exit from vertical blank.

SIO Routines

SIOINT
59716 E944

SIO's initialization routine.

SIO
59737 E959

The main SIO routine (includes the following routines).

RETURN
59917 EAOD

Return from SIO.

WAIT
59930 EA1A

Wait for the device to finish what it has been told to do.

SEND
60011 EA6B

Send a buffer of bytes to a device.

ISRODN
60048 EA90

This is the "serial-output data needed" interrupt routine. See SER-OUT at Location 53773.

ISRTD
60113 EAD1

This is the "transmission done interrupt" routine. See POKMSK at Location 16.

RECEIV
60130 EAE2

Receive a bunch of bytes from a device and store them in a buffer.

ISRSIR
60177 EB11

This is the "serial-input data needed" interrupt routine. See SERIN at Location 53773.

CASENT
60292 EB84

Read or write a record to cassette (SIO handles the cassette differently than other devices).

BEGIN
60692 ED14

Figure out the baud rate for the next record. See CBAUDL/H at Locations 750 and 751.

POKTAB
60882 EDD2

This is a table of values used in the preceding baud-rate routine.

Disk Interface Routines

DINIT
60906 EDEA

The disk interface's initialization routine.

DSKIF
60912 EDFO

The main disk interface routine.

Printer Handler Routines

PRNORG
61048 EE78

This is the beginning of the printer handler. See HATABS at Location 794 for a list of routines in this or any handler.

Cassette Handler Routines

CASORG
61249 EF41

This is the beginning of the cassette handler. See the note at PRNORG.

BEEP
61528 FO58

The cassette handler uses this routine to make the keyboard speaker beep when you type CLOAD or CSAVE.

Monitor Routines

RESET
61723 F11B

This is the start of the System Reset routine.

PWRUP
61733 F125

The start of the coldstart routine.

ZERORM
61752 F138

Clear all the RAM locations.

ZOSRAM
61792 F160

Clear the OS RAM only (for warmstart).

BLACKB
61994 F22A

The blackboard routine (memo pad mode).

SPECL
62015 F23F

Check to see how much RAM there is.

HARDI
62081 F281

Initialize the hardware locations.

OSRAM
62100 F294

Initialize the OS RAM locations.

BOOT
62159 F2CF

Boot the disk if it's so desired (i.e., the disk drive is hooked up and turned on).

CSBOOT
62386 F3B2

Boot the cassette if it's so desired (i.e., the Start button was held down when the computer was turned on).

Display Handler Routines

DOPEN
62454 F3F6

OPEN the display handler (set up a graphics mode).

GETCH
62867 F593

GET a character from the screen.

OUTCH
62903 F5B7

PUT a character on the screen.

OUTPLT
62944 F5E0

PLOT a point on the screen.

Screen Editor Routines

EGETCH
63038 F63E

INPUT a logical line from the keyboard and print it to the screen. Remember that a logical line ends either when you press Return or fill three rows on the screen.

EOUTCH
63140 F6A4

PRINT a character on the screen, making sure that control characters are processed instead of just printing (i.e., a CTRL-arrow will move the cursor rather than printing an arrow).

Keyboard Handler Routines

KGETC2
63197 F6DD

GET a character from the keyboard.

ESCAPE
63353 F779

Process all the various control characters.

BELL
63754 F90A

Ring the bell.

More Display Handler Routines

CONVRT
63815 F947

Take the row number and column number that the cursor is on and figure out what memory location that corresponds to.

INATAC
64306 FB32

Convert an internal character value to its ATASCII value.

CLRLIN
64411 FB9B

Clear the line that the cursor is currently on.

SCROLL
64428 FBAC

Scroll the screen.

DRAW
64764 FCFC

Draw a line from OLDROW,OLDCOL to TWOCRS,COLCRS (Locations 90 through 92 and 84 through 86).

Tables, Tables and More Tables

Locations 65093 through 65469 are various tables for use with the display handler. Check the OS listing for more details and to find out which routines use them (use the cross-reference table at the end of the listing).

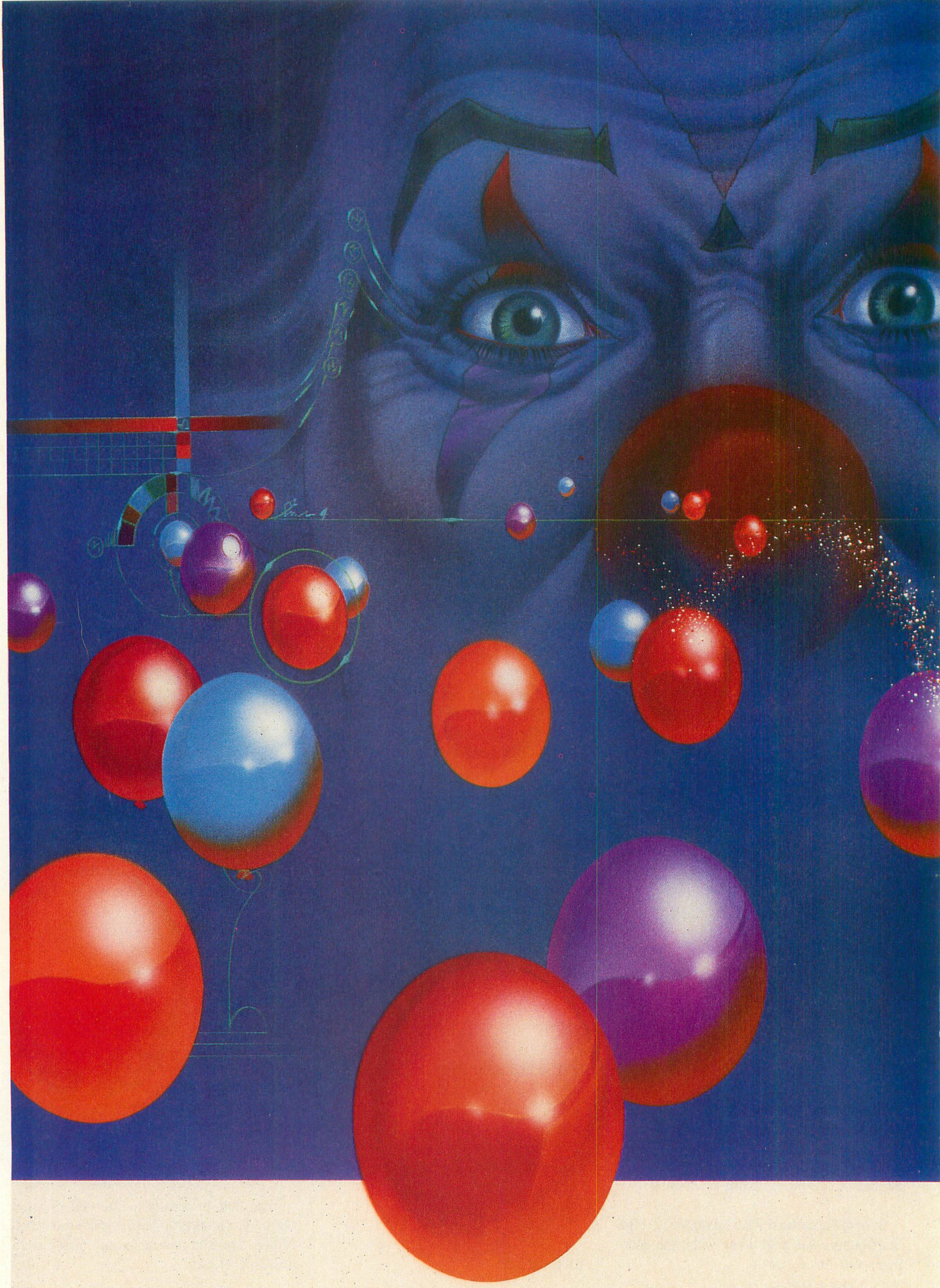
One More Keyboard Routine

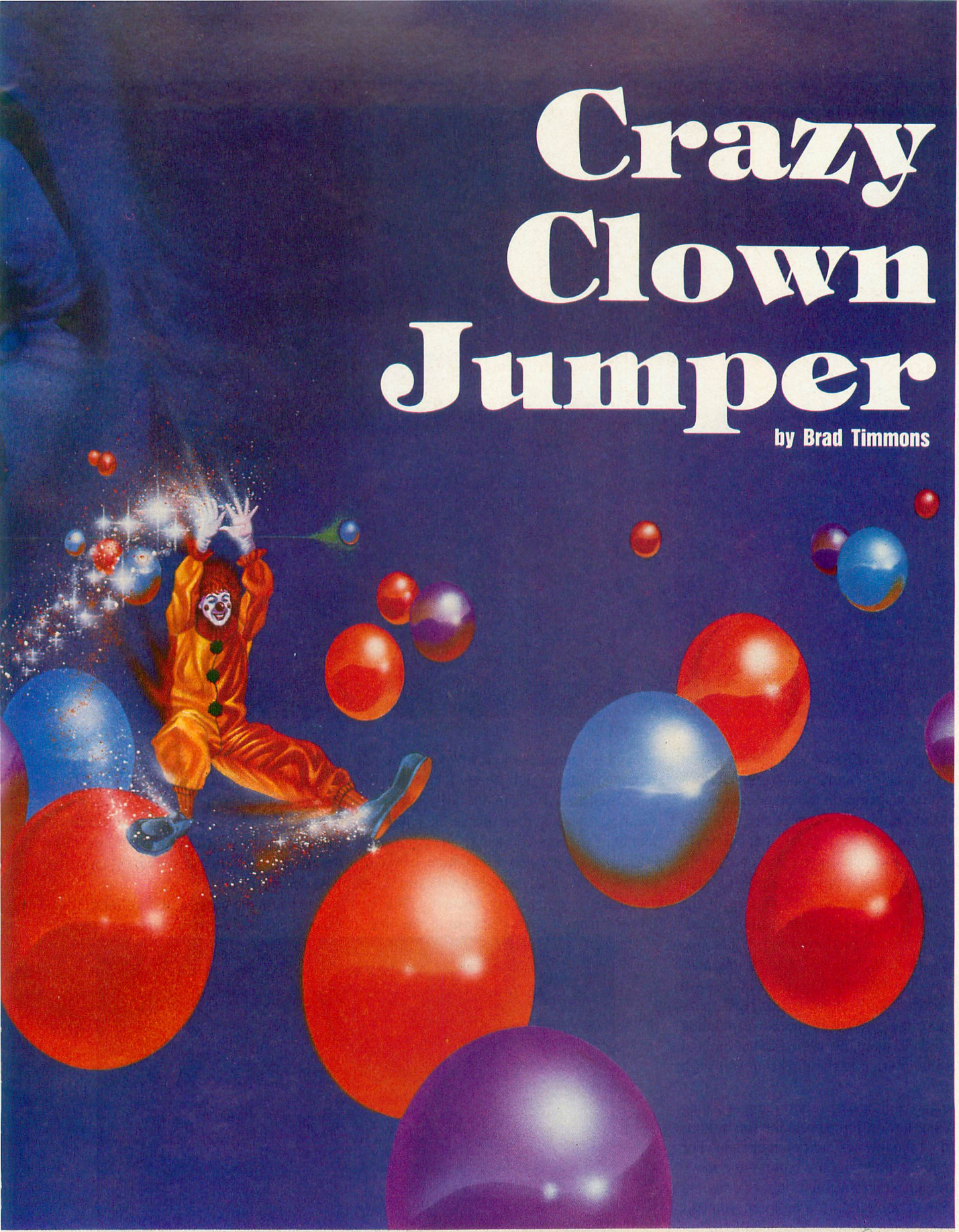
PIRQQ
65470 FFBE

The IRQ in this location's name should tip you off to the fact that this is the interrupt routine for the keyboard. It debounces the keys, checks for CTRL-1 (pause) and sets SSFLAG accordingly (767), stores the key value in CH (764) and CH1 (754), and clears ATRACT (77).

That's All Folks

Yup, that brings us to the end of Atari memory. Thanks for bearing with me for all of this. You can now relax and take a well-deserved break!





Crazy Clown Jumper

by Brad Timmons

It's audition night at one of the toughest circuses in the world. You've heard the circus is looking for a clown, but not just any clown, they want someone who is also an acrobatic expert! The audition is simple enough.

Standing on a 100-foot-high platform, you must jump across numerous rising balloons to get to a platform on the other side of the circus tent, and when you get there, you have to try to make it back again—but don't fall, because this circus doesn't use nets.

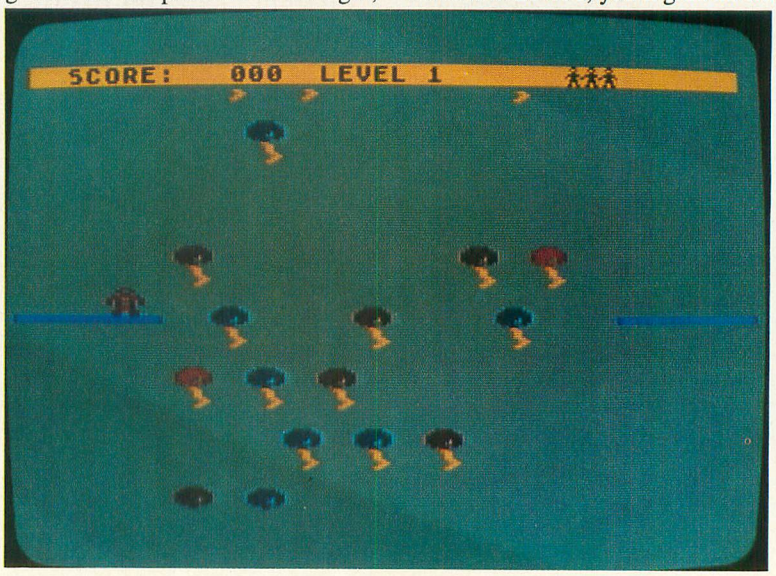
The game starts with you, the clown, standing on the platform at the left side of the screen. You must jump from balloon to balloon to get to the platform on the right side of the screen, then make it back to the platform on the left again. You control the clown with a joystick. To move left or right, simply move the joystick in the corresponding direction. To jump, press the fire button, and move the joystick in the direction you wish to jump.

You have a choice of a short jump or a long jump. For a short jump press the fire button, and move the joystick either left or right. For a long jump, press the fire button, and move the joystick either diagonal right or diagonal left. When you land on a balloon, you can jump off, drop off the side to a balloon below you or just stand there and enjoy the ride to the top of the screen.

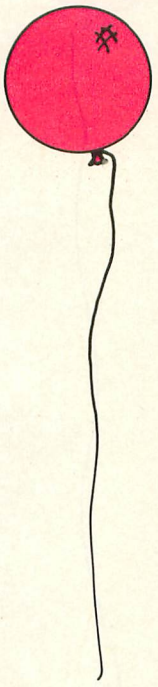
A status line at the top of the screen displays your score, the level you are currently on, and the number of clowns you have remaining. You get 500 points for each level you complete. Also, a bonus dollar sign will occasionally rise up on one of the balloons. Touching it gives you a 100-point bonus.

Crazy Clown Jumper has nine levels of difficulty. You'll advance in level every time you make it completely around. That is, jumping from the left platform to the right, then back again. At Level 2, the platforms will begin to slowly rise and fall. At Level 3, they'll also begin to expand and contract. The speed of the balloons and platforms will increase as you progress to higher levels, and the number of balloons appearing on screen will decrease.

You begin the game with four men. If you make it to Level 6, you'll gain an extra man.



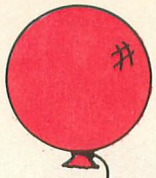
Crazy Clown Jumper



LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,64,171,76,32,136,6
68,32,147,68,162,143,169,0,4652
1010 DATA 149,0,232,208,251,32,181,68,
,32,219,69,169,10,141,200,2,6346
1020 DATA 169,130,141,196,2,169,10,141
1,198,2,169,68,141,197,2,169,6680
1030 DATA 248,141,199,2,160,2,169,0,14
45,88,32,129,74,32,109,66,2346
1040 DATA 32,20,75,32,228,67,169,16,14
41,111,2,169,0,141,8,210,4201
1050 DATA 169,3,141,15,210,169,16,133,
,137,169,80,133,138,169,9,133,6770
1060 DATA 175,32,18,73,169,0,32,105,69
9,165,196,240,11,198,175,208,9729
1070 DATA 7,169,9,133,175,32,71,73,165
5,180,240,3,32,252,71,165,7732
1080 DATA 191,240,3,32,89,66,165,190,2
240,20,201,4,208,5,169,0,5332
1090 DATA 76,80,67,201,8,208,5,169,1,7
76,80,67,208,6,165,188,5730
1100 DATA 240,2,208,34,230,192,32,131,
,65,169,2,32,254,68,165,192,8160
1110 DATA 201,60,240,2,208,179,169,0,1
141,0,210,141,1,210,133,165,8315
1120 DATA 32,91,72,76,94,64,165,188,20
01,4,240,216,165,165,208,9,167
1130 DATA 165,192,201,5,144,3,32,51,66
6,165,162,240,18,169,3,32,3735
1140 DATA 254,68,169,0,133,192,133,162
2,173,148,75,201,41,144,181,165,259
1150 DATA 196,240,13,165,190,240,9,169
9,1,32,254,68,169,0,133,192,7533
1160 DATA 173,132,2,240,72,173,120,2,2
201,15,208,3,76,99,64,201,5650
1170 DATA 7,208,27,32,131,65,165,188,2
240,5,169,3,32,254,68,173,7371
1180 DATA 147,75,201,205,240,36,169,2,
,32,105,69,76,99,64,201,11,3898
1190 DATA 208,24,32,131,65,165,188,240
0,5,169,3,32,254,68,173,147,7970
1200 DATA 75,201,45,240,5,169,1,32,105
5,69,76,99,64,173,120,2,2819
1210 DATA 201,11,208,10,169,32,133,194
4,169,9,133,204,208,92,201,7,8258
1220 DATA 208,8,169,0,133,194,133,204,
,240,80,201,10,208,10,169,128,9061
1230 DATA 133,194,169,27,133,204,208,6
66,201,6,208,10,169,64,133,194,8708
1240 DATA 169,18,133,204,208,52,76,99,
,64,162,0,160,0,200,208,253,9420
1250 DATA 165,180,240,7,134,149,32,252
2,71,166,149,165,196,240,7,134,402
1260 DATA 149,32,71,73,166,149,232,224
4,9,208,226,96,24,165,132,105,9199
1270 DATA 20,133,132,165,133,105,0,133
3,133,96,169,3,133,195,164,204,9206
1280 DATA 185,191,75,133,206,200,185,1
191,75,133,207,200,185,191,75,133,2671
1
1290 DATA 208,165,196,240,3,32,71,73,1
165,180,240,3,32,252,71,165,8274
1300 DATA 207,32,254,68,173,147,75,201
1,45,208,3,76,158,64,173,147,7812
1310 DATA 75,201,205,208,3,76,158,64,1
165,208,32,105,69,165,190,240,397
1320 DATA 18,201,4,208,5,169,0,76,80,6
67,201,8,208,19,169,1,3449
1330 DATA 76,80,67,165,188,240,10,165,
,165,208,3,32,51,66,76,192,5978
1340 DATA 64,165,191,240,3,32,89,66,19
98,206,208,165,230,204,230,204,5295
1350 DATA 230,204,198,195,208,136,76,9
99,64,169,1,133,165,169,180,141,9882
1360 DATA 0,210,169,175,141,1,210,133,
,177,169,15,133,176,96,169,1,7459
1370 DATA 133,164,169,250,133,178,141,
,2,210,169,239,141,3,210,96,32,8906
1380 DATA 72,66,32,8,72,169,100,133,15
52,169,0,133,153,133,191,32,6732
1390 DATA 223,72,96,169,1,133,166,169,
,0,133,169,133,167,165,170,240,1694
1400 DATA 252,169,0,133,170,169,1,133,
,169,165,170,240,252,169,1,133,1163
1410 DATA 167,169,0,133,166,169,44,141
1,198,2,160,0,185,138,76,145,7086
1420 DATA 88,200,192,34,208,246,173,13
32,2,208,251,169,0,133,167,133,785
1430 DATA 161,32,228,67,32,18,73,96,32
2,8,72,169,1,133,161,173,4400
1440 DATA 135,75,201,1,208,249,169,1,1
133,166,133,169,169,0,133,200,9797
1450 DATA 133,199,165,199,201,4,208,25
50,169,1,133,171,165,199,201,5,829
1460 DATA 208,250,169,0,133,171,165,19
99,201,23,208,250,169,0,133,166,1288
1470 DATA 133,169,169,1,133,167,160,0,
,169,3,141,141,75,141,189,75,7017
1480 DATA 169,130,141,149,75,141,150,7
75,169,180,141,152,75,169,0,133,7964
```

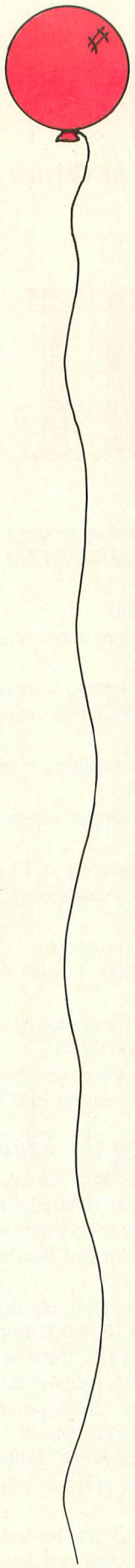
Crazy Clown Jumper



```
1490 DATA 128,169,102,133,129,162,2,32
2,204,69,169,1,133,173,141,177,8176
1500 DATA 75,32,193,74,169,17,141,186,
,75,169,32,141,188,75,169,0,6374
1510 DATA 133,196,133,185,133,186,141,
,176,75,169,100,141,153,75,169,255,216
61
1520 DATA 141,184,75,141,185,75,169,10
0,141,154,75,169,0,133,156,133,7448
1530 DATA 157,133,158,76,148,66,170,16
69,0,133,192,138,240,22,169,3,7518
1540 DATA 205,190,75,240,3,76,233,64,2
230,197,208,0,169,2,141,190,9632
1550 DATA 75,76,233,64,169,2,205,190,7
75,240,3,76,233,64,230,197,1091
1560 DATA 165,197,201,2,240,2,208,36,1
169,0,133,197,238,186,75,173,625
1570 DATA 186,75,201,26,208,5,206,186,
,75,208,17,32,174,67,32,13,3827
1580 DATA 68,169,244,133,152,169,1,133
3,153,32,223,72,169,3,141,190,8503
1590 DATA 75,76,233,64,160,0,185,11,76
6,145,88,200,192,40,208,246,569
1600 DATA 206,186,75,160,17,173,186,75
5,145,88,169,1,133,161,32,8,4715
1610 DATA 72,169,15,133,198,160,0,162,
,0,200,208,253,232,208,250,198,6013
1620 DATA 198,208,246,238,186,75,169,0
0,133,161,160,0,185,227,75,145,82
1630 DATA 88,200,192,40,208,246,172,18
87,75,173,186,75,145,88,173,189,2157
1640 DATA 75,240,15,162,0,160,30,169,3
3,145,88,200,232,236,189,75,610
1650 DATA 208,247,96,173,186,75,201,17
7,240,42,201,18,208,5,169,1,6587
1660 DATA 133,196,96,201,19,208,13,206
6,141,75,169,1,133,185,133,186,9539
1670 DATA 32,105,68,96,201,20,208,4,32
2,105,68,96,201,21,208,4,4442
1680 DATA 32,105,68,96,201,22,208,16,3
32,105,68,206,141,75,238,189,9266
1690 DATA 75,238,188,75,32,248,67,96,2
201,23,208,3,32,105,68,201,6475
1700 DATA 24,208,4,32,105,68,96,201,25
5,208,3,32,105,68,96,56,2868
1710 DATA 173,184,75,233,25,141,184,75
5,141,185,75,56,173,153,75,233,127
1720 DATA 5,141,153,75,24,173,154,75,1
105,10,141,154,75,96,162,96,6331
1730 DATA 169,12,157,66,3,32,86,228,96
6,162,96,169,3,157,66,3,4134
1740 DATA 169,12,157,74,3,169,0,157,75
5,3,169,178,157,68,3,169,5427
1750 DATA 68,157,69,3,32,86,228,96,83,
,58,155,169,0,133,128,169,6937
1760 DATA 80,133,129,169,0,133,130,169
9,224,133,131,160,0,162,2,177,8066
1770 DATA 130,145,128,200,208,249,230,
,129,230,131,202,208,242,169,8,133,440
08
1780 DATA 128,169,80,133,129,160,0,185
5,58,75,145,128,200,192,16,208,9362
1790 DATA 246,162,0,189,98,75,145,128,
,200,232,224,8,208,245,169,80,2724
1800 DATA 141,244,2,96,201,0,240,101,2
201,1,240,15,201,2,240,50,7992
1810 DATA 201,3,208,6,169,1,133,163,20
08,1,96,172,148,75,162,12,6349
1820 DATA 177,139,136,145,139,200,200,
,202,208,246,24,173,148,75,105,10,9555
5
1830 DATA 168,162,4,177,141,136,145,14
41,200,200,202,208,246,206,148,75,4726
6
1840 DATA 208,84,24,173,148,75,105,11,
,168,162,12,177,139,200,145,139,9387
1850 DATA 136,136,202,208,246,24,173,1
148,75,105,13,168,162,4,177,141,8422
1860 DATA 200,145,141,136,136,202,208,
,246,238,148,75,208,41,240,39,201,3270
0
1870 DATA 0,240,35,201,1,240,5,201,2,2
240,15,96,206,147,75,173,8711
1880 DATA 147,75,141,0,208,141,1,208,2
208,12,238,147,75,173,147,75,9333
1890 DATA 141,0,208,141,1,208,165,163,
,240,5,169,0,133,163,96,32,6963
1900 DATA 131,65,169,0,133,190,133,188
8,173,13,208,133,190,173,5,208,774
1910 DATA 133,188,162,0,189,8,208,240,
,4,133,191,208,5,232,224,4,9974
1920 DATA 208,242,169,0,141,30,208,96,
,169,0,133,128,169,100,133,129,8404
1930 DATA 162,2,160,0,169,0,145,128,20
00,208,251,230,129,202,208,246,6422
1940 DATA 96,160,11,162,70,169,7,32,92
2,228,173,48,2,133,128,173,6869
1950 DATA 49,2,133,129,160,6,169,0,145
5,128,200,145,128,160,8,169,8190
1960 DATA 38,145,128,200,192,29,208,24
49,165,88,133,132,165,89,133,133,980
1970 DATA 96,165,167,240,3,76,249,71,2
206,140,75,208,27,169,0,133,8059
1980 DATA 77,173,141,75,141,140,75,169
9,1,133,147,133,162,230,202,165,2096
1990 DATA 202,201,8,240,6,141,5,212,76
6,194,71,32,166,65,32,166,6178
2000 DATA 65,165,132,133,135,165,133,1
133,136,169,0,133,202,141,5,212,9560
2010 DATA 32,166,65,160,0,162,0,177,13
32,145,135,200,192,20,208,247,1968
2020 DATA 24,165,135,105,20,133,135,16
65,136,105,0,133,136,32,166,65,5978
2030 DATA 160,0,232,224,22,208,224,165
5,171,240,2,208,82,165,166,240,4191
2040 DATA 94,165,169,208,49,160,0,166,
,200,189,62,76,201,128,240,7,9737
2050 DATA 145,135,200,230,200,208,240,
,192,0,208,10,169,0,168,145,135,354
2060 DATA 200,192,20,208,249,230,199,2
230,200,165,199,201,7,208,4,169,3212
2070 DATA 1,133,170,76,186,71,160,0,16
69,0,145,135,200,192,20,208,9279
2080 DATA 249,230,199,165,199,201,20,2
208,234,169,1,133,170,208,228,160,4556
6
2090 DATA 0,185,124,76,145,135,200,192
2,14,208,246,230,199,208,212,206,7254
2100 DATA 135,75,240,3,76,100,71,173,1
142,75,16,19,169,4,133,160,5218
2110 DATA 169,16,133,159,173,142,75,73
3,128,141,142,75,76,9,71,169,6079
2120 DATA 5,133,160,169,17,133,159,173
3,142,75,73,128,141,142,75,162,8832
2130 DATA 0,169,0,157,0,6,232,224,40,2
208,248,164,160,173,10,210,1696
2140 DATA 205,153,75,176,54,173,10,210
0,41,3,170,189,143,75,153,0,6379
2150 DATA 6,169,130,153,20,6,169,0,153
3,40,6,165,180,208,28,165,6950
2160 DATA 161,208,24,173,10,210,205,15
54,75,176,16,185,155,75,133,182,281
2170 DATA 169,1,133,180,132,146,32,52,
,72,164,146,200,200,196,159,208,2987
2180 DATA 188,169,0,133,201,169,3,141,
,135,75,166,201,160,0,189,0,7524
2190 DATA 6,145,135,200,232,192,20,208
8,245,24,165,201,105,20,133,201,1335
2200 DATA 206,137,75,208,59,169,5,141,
,137,75,166,203,160,0,189,66,8425
2210 DATA 75,145,137,232,200,192,8,208
8,245,173,134,75,201,0,240,18,596
2220 DATA 165,203,201,24,240,7,24,105,
,8,133,203,16,19,206,134,75,5908
2230 DATA 240,14,165,203,240,7,56,233,
,8,133,203,16,3,238,134,75,7711
2240 DATA 165,88,133,132,165,89,133,13
33,165,165,240,28,198,176,240,14,1835
2250 DATA 165,177,41,240,5,176,141,1,2
210,133,177,76,226,71,169,0,8474
```

Crazy Clown Jumper

```
2260 DATA 141,0,210,141,1,210,133,165,
,165,164,240,19,56,165,178,233,2540
2270 DATA 10,133,178,141,2,210,208,7,1
169,0,133,164,141,3,210,76,7619
2280 DATA 98,228,165,147,240,28,198,18
81,165,181,201,40,208,21,169,0,9638
2290 DATA 133,180,169,0,133,128,169,99
9,133,129,160,0,152,145,128,200,71
2300 DATA 208,251,96,169,13,141,175,75
5,164,181,169,0,133,128,169,99,9495
2310 DATA 133,129,32,253,73,169,0,133,
,147,96,169,0,133,128,169,99,8034
2320 DATA 133,129,160,208,132,181,162,
,0,189,51,76,145,128,200,232,224,3218
2330 DATA 11,208,245,164,182,162,4,152
2,157,3,208,200,200,202,208,247,5905
2340 DATA 96,172,148,75,162,0,189,122,
,75,145,139,200,232,224,12,208,2347
2350 DATA 245,169,175,141,1,210,169,0,
,133,168,32,131,65,169,2,32,5024
2360 DATA 254,68,230,168,165,168,141,0
0,210,173,148,75,201,250,208,234,6057
2370 DATA 169,0,172,188,75,145,88,206,
,188,75,206,189,75,173,189,75,1724
2380 DATA 201,255,208,13,169,0,141,0,2
210,141,1,210,133,191,32,178,9161
2390 DATA 66,169,65,141,147,75,56,173,
,149,75,233,12,141,148,75,169,9046
2400 DATA 0,133,197,133,192,169,3,141,
,190,75,133,183,169,45,141,151,389
2410 DATA 75,32,194,69,32,232,74,32,20
0,75,169,0,141,1,210,141,5775
2420 DATA 0,210,133,191,96,169,0,133,1
150,133,151,162,15,24,6,152,5899
2430 DATA 38,153,248,165,150,101,150,1
133,150,165,151,101,151,133,151,216,30
067
2440 DATA 202,16,235,24,248,165,150,10
01,158,133,158,165,157,101,151,133,188
81
2450 DATA 157,165,156,105,0,133,156,21
16,172,139,75,136,162,0,200,181,840
2460 DATA 156,72,41,240,74,74,74,74,9,
,16,145,88,104,41,15,9,954
2470 DATA 16,200,145,88,232,224,3,208,
,229,172,139,75,162,3,177,88,283
2480 DATA 201,16,208,8,169,0,145,88,20
00,202,208,242,96,206,182,75,3007
2490 DATA 208,86,169,0,133,128,169,102
2,133,129,173,184,75,141,182,75,286
2500 DATA 173,176,75,240,37,238,149,75
5,173,149,75,201,200,208,7,169,2071
2510 DATA 0,141,176,75,240,234,24,173,
,149,75,105,3,168,32,11,74,5039
2520 DATA 165,185,240,3,32,25,74,76,16
62,73,206,149,75,173,149,75,8297
2530 DATA 201,50,208,7,169,1,141,176,7
75,208,197,172,149,75,32,253,1230
2540 DATA 73,165,185,240,3,32,25,74,20
06,183,75,240,1,96,173,185,9733
2550 DATA 75,141,183,75,169,0,133,128,
,169,103,133,129,173,177,75,240,1810
2560 DATA 35,238,150,75,173,150,75,201
1,200,208,7,169,0,141,177,75,9533
2570 DATA 240,234,24,173,150,75,105,3,
,168,32,11,74,165,186,240,3,7239
2580 DATA 32,77,74,96,206,150,75,173,1
150,75,201,50,208,7,169,1,7475
2590 DATA 141,177,75,208,199,172,150,7
75,32,253,73,165,186,240,3,32,9755
2600 DATA 77,74,96,174,175,75,177,128,
,136,145,128,200,200,202,208,246,6363
2610 DATA 96,162,6,136,177,128,200,145
5,128,136,136,202,208,246,96,206,5121
2620 DATA 180,75,240,1,96,173,178,75,1
141,180,75,165,183,240,20,238,2759
2630 DATA 151,75,173,151,75,201,46,208
8,6,169,0,133,183,240,236,141,2496
2640 DATA 2,208,96,206,151,75,173,151,
,75,201,19,208,242,169,1,133,839
2650 DATA 183,208,216,206,181,75,240,1
1,96,173,179,75,141,181,75,165,1359
2660 DATA 184,240,20,206,152,75,173,15
52,75,201,179,208,6,169,0,133,9567
2670 DATA 184,240,236,141,3,208,96,238
8,152,75,173,152,75,201,201,208,4422
2680 DATA 242,169,1,133,184,208,216,16
69,96,141,7,212,169,3,141,29,8540
2690 DATA 208,169,62,141,47,2,169,0,13
33,128,169,99,133,129,160,0,6875
2700 DATA 162,6,152,145,128,200,208,25
51,230,129,202,208,246,169,0,133,4984
2710 DATA 139,169,100,133,140,169,0,13
33,141,169,101,133,142,169,70,141,9981
1
2720 DATA 192,2,169,132,141,193,2,169,
,0,133,128,169,102,133,129,172,9821
2730 DATA 149,75,162,3,169,255,145,128
8,200,202,208,250,230,129,172,150,6825
5
2740 DATA 75,162,3,145,128,200,202,208
8,250,165,173,240,1,96,172,148,3634
2750 DATA 75,162,0,189,106,75,145,139,
,200,232,224,12,208,245,136,136,4090
2760 DATA 189,106,75,145,141,200,232,2
224,16,208,245,169,0,141,0,208,1536
2770 DATA 141,1,208,141,2,208,141,3,20
08,96,173,147,75,141,0,208,9146
2780 DATA 141,1,208,173,151,75,141,2,2
208,173,152,75,141,3,208,169,320
2790 DATA 3,141,10,208,141,11,208,169,
,118,141,194,2,141,195,2,96,8275
2800 DATA 60,126,255,255,251,243,126,2
24,24,24,48,48,96,192,96,48,6388
2810 DATA 24,48,24,48,24,48,96,48,24,1
12,24,12,24,12,6,12,6164
2820 DATA 24,12,12,12,6,3,6,12,24,36,2
24,126,153,60,36,102,491
2830 DATA 0,24,36,24,126,231,189,165,6
60,60,102,0,0,102,102,0,2961
2840 DATA 0,24,165,153,255,102,60,36,6
60,60,102,0,1,1,0,5,8959
2850 DATA 0,8,3,3,128,1,65,1,193,65,10
08,130,130,45,180,100,5751
2860 DATA 10,0,0,0,0,80,88,96,104,112,
,120,128,136,144,152,160,8270
2870 DATA 168,176,0,0,0,13,0,1,10,10,1
10,10,255,255,255,255,8686
2880 DATA 17,22,32,3,3,4,1,2,8,0,2,4,2
2,2,4,1,3383
2890 DATA 1,8,0,1,4,2,1,12,1,2,8,0,2,1
12,2,2,3419
2900 DATA 12,1,1,8,0,1,12,2,1,0,0,51,3
35,47,50,37,6131
2910 DATA 26,0,0,0,0,0,0,0,0,0,44,37,54,
,37,44,0,0,5528
2920 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,0,0,2920
2930 DATA 0,0,0,0,0,0,0,0,10,10,10,10,0,
,44,37,54,37,5802
2940 DATA 44,0,0,0,35,47,45,48,44,37,5
52,37,36,0,10,10,6700
2950 DATA 10,10,0,0,0,0,0,0,0,0,24,24,60
0,102,48,24,12,6754
2960 DATA 102,60,24,24,0,0,0,0,35,50,3
33,58,57,0,35,44,7194
2970 DATA 47,55,46,128,128,0,0,0,0,0,0
0,0,42,53,45,48,7148
2980 DATA 37,50,128,128,0,0,0,0,0,0,0,
,0,0,34,57,128,7392
2990 DATA 128,0,0,0,0,34,50,33,36,0,52
2,41,45,45,41,46,7890
3000 DATA 51,128,0,0,0,0,0,39,33,45,37
7,0,47,54,37,50,7495
3010 DATA 0,0,0,0,0,48,50,37,51,51,0,5
52,40,37,0,38,7183
3020 DATA 41,50,37,34,53,52,52,47,46,0
0,52,47,0,51,52,33,8297
3030 DATA 50,52,226,2,227,2,0,64,0,0,0
0,0,0,0,0,0,5529
```



BASIC Editor II

by Clayton Walnum

BASIC Editor II is a utility to help you enter BASIC program listings published in ANALOG Computing. To simplify the identification of errors, each program line is evaluated immediately after it's typed, eliminating the need for cumbersome checksum listings. When you've finished entering a program using BASIC Editor II, you can be certain it contains no typos.

An option is provided for those who wish to use standard BASIC abbreviations. Also, the program retains all Atari editing features. Finally, for those who prefer to type programs the conventional way, using the built-in editor, a post-processing mode is available. It allows you to check typing after the entire listing has been entered.

Typing in the Editor

To create your copy of BASIC Editor II, follow the instructions below— exactly.

Disk version:

(1) Type in Listing 1, then verify your work with Unicheck (see Issue 39).

(2) Save the program to disk with the command *SAVE "D:EDITORLI.BAS"*.

(3) Clear the computer's memory with the command *NEW*.

(4) Type in Listing 2, then verify your work with Unicheck.

(5) Run the program (after saving a backup copy) and follow all the on-screen prompts. A data file will be written to your disk.

(6) Load Listing 1 with the command *LOAD "EDITORLI.BAS"*.

(7) Merge the file created by Listing 2 with the command *ENTER "D:ML.DAT"*.

(8) Save the resultant program with the command *LIST "D:EDITORII.LST"*.

Cassette version:

(1) Type in Listing 1 and verify your work with Unicheck.

(2) Save the program to cassette with the command *CSAVE*. (Do not rewind the cassette.)

(3) Clear the computer's memory with the command *NEW*.

(4) Type in Listing 2 and verify your work with Unicheck.

(5) Run the program and follow the on-screen prompts. A data file will be written to your cassette.

(6) Rewind the cassette.

(7) Load Listing 1 with the command *CLOAD*.

(8) Merge the file created by Listing 2 with the command *ENTER "C:"*.

(9) On a new cassette, save the resultant program with the command *LIST "C:"*.

Using the Editor

Take a look at one of the BASIC program listings in this issue. Notice that each program line is preceded by a two-letter code. This code is the checksum for that line; it's not a part of the program.

To enter a program listing from the magazine, load BASIC Editor II with the *ENTER* command, and run it. You'll be asked if you wish to allow abbreviations (see your BASIC manual). If you do, type *Y* and press *RETURN*. Otherwise, type *N*.

Note: If you set BASIC Editor II to allow abbreviations, the program will run slightly slower.

Your screen will now be divided into two "windows." The upper window will display each line after it's processed, as well as the checksum generated for that line. The lower window is where program lines are typed and edited.

When the program's waiting for input, the cursor will appear at the left margin of the typing window. Type a program line and press *RETURN*. The line will be evaluated and reprinted in the message window, along with the checksum generated.

If the checksum matches the one in the magazine, then go on to the next program line. Otherwise, enter the command *E* (edit) and press *RETURN*. The line you just typed will appear in the typing window, where you may edit it. When you think the line has been corrected, press *RETURN*, and it'll be reevaluated.

Note: You may call up any line previously typed, with the command *E* followed by the number of the line you wish to edit. For example, *E230* will print Line 230 in the typing window. *Do not attempt to edit any program lines numbered 32600 and higher.* These lines fall within the BASIC Editor II program.

If you're using BASIC abbreviations, the two versions of the command *E* work slightly differently. The command *E*, without a line number, will call up the line exactly as you typed it. When you append the line number, the line will be printed in its expanded (unabbreviated) form.

Leaving the Editor

You may leave BASIC Editor II at any time, by entering either *B* (BASIC) or *Q* (quit). If you type *B*, the Editor will return you to BASIC. Enter *LIST* to review your work, if you wish. Note that lines 32600 and above are the Editor program. Your work will appear before these lines. To return to the Editor, type *GOTO 32600*.

Type *Q*, and you'll be asked if you really want to quit. If you type *Y*, the Editor program will be erased from memory, and you may then save your work in any manner you like. If you type *N*, the *Q* command will be aborted.

Large listings

If the program you're entering is particularly long, you may need to take a break. When you want to stop, type *Q* and press *RETURN*, then save your work to disk or cassette. When you're ready to start again, load the program you were working on, then load BASIC Editor II with the *ENTER* command. Type *GOTO 32600*, and you're back in business.

The post-processor

Many people may not want to use BASIC Editor II when entering a program listing, preferring, instead, the Atari's built-in editor. For that reason, BASIC Editor II will allow you to check and edit your programs after they've been typed.

To take advantage of this option, type any magazine program in the conventional manner, then save a copy to disk or cassette (just in case). With your typed-in program still in memory, load BASIC Editor II with the *ENTER* command, then type *GOTO 32600.*

Respond with *N* to the "abbreviations" prompt. When the Editor appears on your screen, enter the command *P* (post-process), and the first program line will appear in the typing window. Press *RETURN* to enter it into the Editor.

The line will be processed, and the checksum, along with the program line, will be printed in the upper window. If the checksum matches the one in the magazine, press *RETURN* twice, and the next line will be processed.

If you find you must edit a line, enter the command *E*, and the line will be moved back to the typing window for editing.

When the entire listing has been checked, you'll be asked if you wish to quit. Type *Y* and press *RETURN*. The Editor program will be removed from memory, and you may then save the edited program in any manner you wish.

Murphy's Law

Anyone who's been associated with computing knows this is the industry Murphy had in mind. You may find that, after typing a program with BASIC Editor II, it still won't run properly. There are two likely causes for this.

First, it may be that you're not following the program's instructions properly. Always read the article accompanying a program *before* attempting to run it. Failure to do so may present you with upsetting results.

Finally, though you can trust BASIC Editor II to catch your typos, it can't tell you if you've skipped some lines entirely. If your program won't run, make sure you've typed all of it. Missing program lines are guaranteed trouble.

One last word: Some people find it an unnecessary and nasty chore to type REM lines. I don't condone the omission of these lines, since they may be referenced within the program (a bad practice, but not unheard of). If you want to take chances, BASIC Editor II is willing to comply.

When you've finished entering a program using BASIC Editor II, you can be certain it contains no typos.

Listing 1.
BASIC listing.

```
32600 IF FL THEN 32616
32602 DIM L$(115),SV$(115),C2$(2),B$(1
15),M$(119),S$(98),E$(69),A$(1):FL=1:S
TMTAB=PEEK(136)+PEEK(137)*256
32604 GRAPHICS 0:POKE 710,0:P=0:ABR=0:
? "ALLOW ABBREVIATIONS";:INPUT A$:IF A
$="Y" OR A$="y" THEN ABR=1
32606 B$(1)=" ":B$(115)=" ":B$(2)=B$
32616 OPEN #17,4,0,"E:":L$=" ":GOSUB 3
2662:START=0
32618 POKE 766,1:POKE 83,39:POSITION 1
,3:IF LEN(L$)<39 THEN ? L$:GOTO 32624
32620 IF LEN(L$)<77 THEN ? L$(1,38):?
L$(39,LEN(L$)):GOTO 32624
32622 ? L$(1,38):? L$(39,76):? L$(77,L
EN(L$))
32624 POKE 752,0:POKE 766,0:POKE 559,3
4:POKE 82,1:POKE 83,38:POSITION 0,10:?
 " ";:INPUT #17;L$:POKE 766,1
32626 IF (L$="P" OR L$="p") AND START=
0 THEN P=1:L$=""
32628 IF L$="E" OR L$="e" THEN E=1:POS
ITION 1,10:? SV$ GOTO 32624
32630 IF L$="Q" OR L$="q" THEN 32690
32632 IF L$="" AND P=1 THEN 32686
32634 IF L$="" THEN 32624
32636 IF L$="B" OR L$="b" THEN GRAPHIC
S 0:? "TYPE 'GOTO 32600' TO CONTINUE":
END
32638 IF L$(1,1)="E" OR L$(1,1)="e" TH
EN E=1:TRAP 32624:EL=VAL(L$(2)):POSITI
ON 1,9:LIST EL:GOTO 32624
32640 SV$=L$:TRAP 32624:X=VAL(L$)
32642 START=1:IF P AND  NOT E THEN 326
52
32644 GOSUB 32674:IF  NOT ABR OR P THE
N 32652
32646 POKE 766,0:? CHR$(125):POSITION
0,3:L=VAL(L$):LIST L:? :? :? "CONT":L$
=B$
32648 POSITION 0,0:POKE 842,13:STOP
32650 POKE 842,12:A=USR(ADR(S$),ADR(L$
),4):L$=L$(1,A)
32652 CHKSUM=USR(ADR(M$),ADR(L$),LEN(L
$)):CHKSUM=CHKSUM+PEEK(1542)*65536
32654 CHK=CHKSUM-(INT(CHKSUM/676)*676)
:HI=INT(CHK/26):LO=CHK-(HI*26):C2$(1)=
CHR$(HI+65):C2$(2)=CHR$(LO+65)
32656 IF  NOT P OR E THEN E=0:GOSUB 32
662:IF  NOT P THEN 32660
32658 POKE 83,39:POKE 752,1:FOR X=3 TO
 5:POSITION 1,X:? B$(1,38):POSITION 1,
X+7:? B$(1,38):NEXT X:POKE 83,38
32660 POKE 766,1:POKE 83,38:POSITION 6
,7:? C2$:POKE 752,0:GOTO 32618
32662 GOSUB 32702:POKE 766,0:POKE 752,
1:? "R":POKE 82,1:DL=PEEK(560)+256*PEE
K(561)+4
32664 POKE DL-1,70:POKE DL+2,6:POKE DL
+3,112:POKE DL+4,112:POKE DL+5,112:POK
E DL+13,112:POKE DL+14,112
32666 POKE DL+22,112:POKE DL+23,112:PO
KE DL+24,65:POKE DL+25,PEEK(560):POKE
DL+26,PEEK(561):POKE 83,39
32668 POSITION 20,0:? "  basic editor
II  ":POSITION 0,7:? "                TYP
ING WINDOW                ":
32670 POSITION 0,1:? "                MES
SAGE WINDOW                ":POSITION 1,7
:? "CODE:";
32672 POKE 559,34:RETURN
32674 GRAPHICS 0:POKE 559,0:POKE 766,1
:POKE 82,0:POKE 83,39:POSITION 0,3:? L
$:? :? :? :? "CONT":POSITION 0,0
32676 POKE 842,13:STOP
32678 POKE 842,12:TRAP 32682:A=USR(ADR
(E$),VAL(L$)):IF A=4 THEN POP :GOTO 32
682
32680 RETURN
32682 GOSUB 32662:SOUND 0,75,10,8:FOR
X=1 TO 20:NEXT X:SOUND 0,0,0,0:POSITIO
N 1,3:? "SYNTAX ERROR!":POKE 766,1
32684 POKE 83,38:POSITION 1,10:? SV$:G
OTO 32624
32686 LINE=PEEK(STMTAB)+PEEK(STMTAB+1)
*256:IF LINE>32599 THEN 32690
32688 OFS=PEEK(STMTAB+2):STMTAB=STMTAB
+OFS:POSITION 1,9:LIST LINE:GOTO 32624
32690 POKE 766,0:POSITION 1,10:? "READ
Y TO QUIT";:INPUT A$:IF A$<>"Y" THEN P
OSITION 1,10:? B$(1,38):GOTO 32624
32692 GRAPHICS 0:? :? :? :FOR X=32600
TO 32636 STEP 2:? X:NEXT X:? "CONT":PO
SITION 0,0:POKE 842,13:STOP
32694 POKE 842,12:GRAPHICS 0:? :? :? :
FOR X=32638 TO 32674 STEP 2:? X:NEXT X
:? :? "CONT":POSITION 0,0
32696 POKE 842,13:STOP
32698 POKE 842,12:GRAPHICS 0:? :? :? :
FOR X=32676 TO 32702 STEP 2:? X:NEXT X
:? :? "POKE 842,12":POSITION 0,0
32700 POKE 842,13:STOP
32702 POKE 16,112:POKE 53774,112:RETUR
N
```

CHECKSUM DATA.
(see issue 39's ***Unicheck****)*

```
32600 DATA 6,665,923,757,809,171,225,8
98,532,499,910,267,912,144,735,8453
32638 DATA 97,358,230,693,706,878,317,
127,36,597,238,258,182,430,168,5315
32668 DATA 864,953,472,385,887,724,72,
687,908,736,625,612,672,184,891,9672
32698 DATA 8,856,85,949
```

Listing 2.
BASIC listing.

```
10 DIM L$(120),ML$(119),A$(1)
20 GRAPHICS 0:POKE 710,0:? "DISK OR BA
SSETTE";:INPUT A$:IF A$<>"C" AND A$<>"
D" THEN 20
30 IF A$="C" THEN 50
40 ? "PLACE FORMATTED DISK IN DRIVE":?
 "THEN PRESS RETURN":INPUT L$:OPEN #1,
8,0,"D:ML.DAT":GOTO 60
50 ? :? "READY CASSETTE, PRESS RETURN"
;:INPUT L$:OPEN #1,8,0,"C:"
60 L$="32608 M$(1)=":L$(13)=CHR$(34)
70 N=119:GOSUB 130:L$(14)=ML$(1,58):L$
(LEN(L$)+1)=CHR$(34):? #1;L$
80 L$(1)="32610 M$(59)=":L$(14)=CHR$(3
4):L$(15)=ML$(59):L$(LEN(L$)+1)=CHR$(3
4):? #1;L$
90 L$(1)="32612 S$=":L$(10)=CHR$(34)
100 ML$="":N=98:GOSUB 130:L$(11)=ML$:L
$(LEN(L$)+1)=CHR$(34):? #1;L$
110 L$(1)="32614 E$=":L$(10)=CHR$(34)
120 ML$="":N=69:GOSUB 130:L$(11)=ML$:L
$(LEN(L$)+1)=CHR$(34):? #1;L$:END
130 FOR X=1 TO N:READ A:ML$(X)=CHR$(A)
:NEXT X:RETURN
140 DATA 104,104,133,204,104,133,203,1
04,104,133,205,169,0,141,3,6,141,2,6,1
41,4,6,141,5,6
150 DATA 141,6,6,238,3,6,32,68,218,172
,2,6,177,203,133,212,32,170,217,32,182
,221,32,68,218
160 DATA 173,3,6,133,212,32,170,217,32
,219,218,32,210,217,165,212,141,0,6,16
5,213,141,1,6,24
170 DATA 173,0,6,109,4,6,141,4,6,173,1
,6,109,5,6,141,5,6,144,3,238,6,6,238,2
180 DATA 6,172,2,6,196,205,208,176,173
,4,6,133,212,173,5,6,133,213,96
190 DATA 104,104,133,204,104,133,203,1
04,104,41,255,6,169,0,133,213,216,165
,88,133,205,165,89,133,206
200 DATA 174,255,6,24,165,205,105,40,1
33,205,144,2,230,206,202,208,242,160,0
,177,205,201,64,144,18
210 DATA 201,96,144,19,201,128,144,18,
201,192,144,6,201,224,144,7,176,8,24,1
05,32,144,3,56,233
220 DATA 64,145,203,200,192,114,240,2,
208,215,177,203,201,32,208,3,136,208,2
47,200,132,212,96
230 DATA 104,104,141,254,6,104,141,253
,6,169,0,133,213,216,165,136,133,205,1
65,137,133,206,160,0,177
240 DATA 205,205,253,6,208,8,200,177,2
05,205,254,6,240,15,160,2,177,205,24,1
01,205,133,205,144,228
250 DATA 230,206,176,224,160,4,177,205
,201,55,240,4,160,0,240,0,132,212,96
```

CHECKSUM DATA.
(see issue 39's ***Unicheck****)*

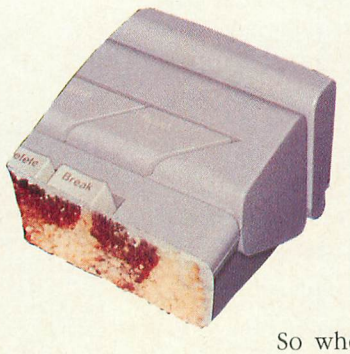
```
10 DATA 203,265,465,844,294,973,652,27
0,978,797,278,275,835,209,301,7639
50 DATA 355,94,254,420,935,840,580,41
,974,564,5435
```

You Own an Atar

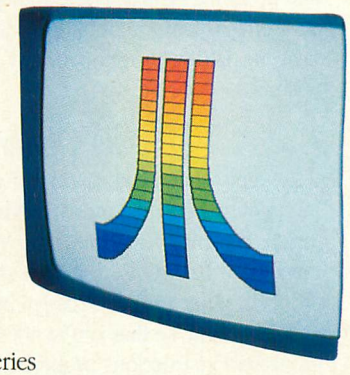
Why Be Forced to Read a Magazine that

YOUR ATARI RESOURCE CENTER

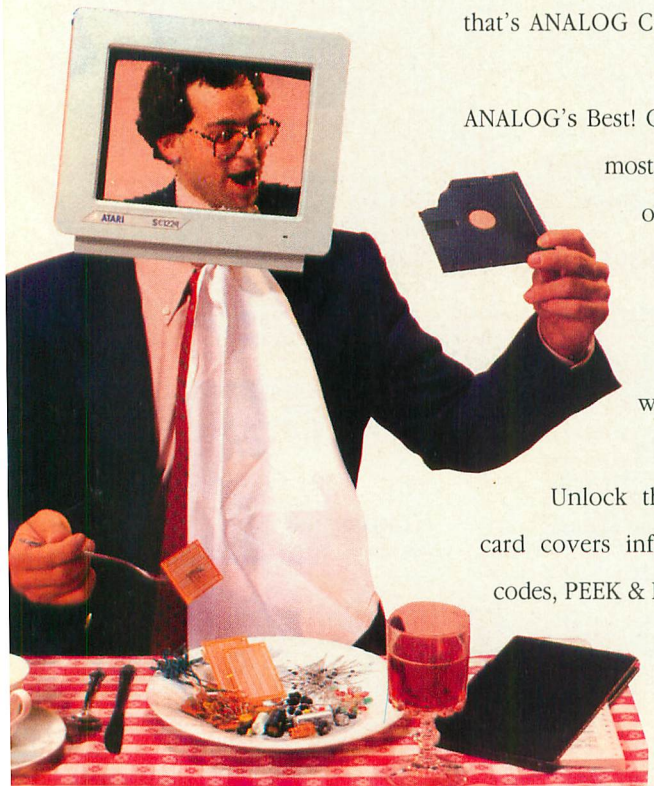
ANALOG Computing continues to offer exciting products for you and your Atari Computer. And we're the only magazine for the Atari 8-bit computer line that hasn't allowed its content to be virtually taken over by coverage of the Atari ST. We include only a minimal amount of ST material so that you can stay informed of what's happening with the 8-bit computer's brother.



Whether you own a reliable ol' 400 or 800, a shiny XL, new XE or even an XE Game Machine. . . we offer usable utilities, entertaining educational software, dynamite disk programs and great graphics and games. In fact, our readers still use ANALOG programs that were published over five years ago!



So when software companies turn their heads to other computers, you can turn yours to the one that supports your 8-bit Atari. And that's ANALOG Computing.

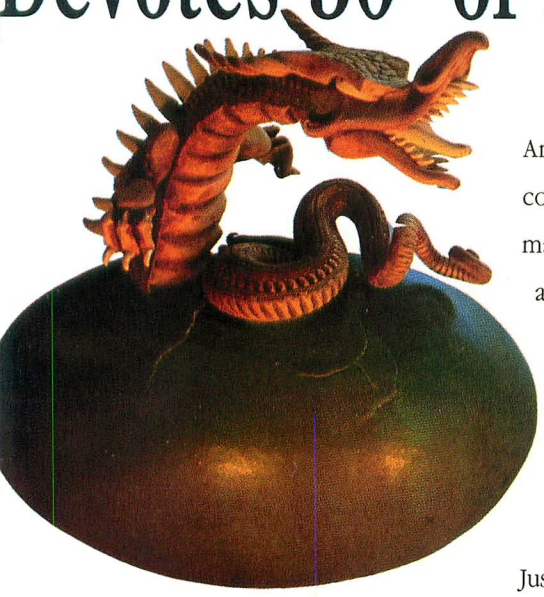


ANALOG's Best! Over 88 of ANALOG Computing's best and most requested programs are now available on this series of ten diskettes. The programs are all ready to run and come with complete documentation on the flip side of each floppy diskette. Select from Graphics, Educational, Utilities 1, Utilities 2, Disk Utilities and Games Disks 1, 2, 3, 4 and 5. Only $9.95 each (plus $1.50 shipping per order). Specify disk title when ordering.

Unlock the secrets of your Atari Computer! This handy 16-page pocket reference card covers information you need when programming your 8-bit. Error codes, internal codes, PEEK & POKE locations, machine-language aids, graphic mode specs and BASIC commands with abbreviations are only some of the helpful items at your fingertips. The ANALOG Computing Pocket Reference Card, only $7.95 each! (Plus $1.50 shipping and handling.)

i 6502 Computer.

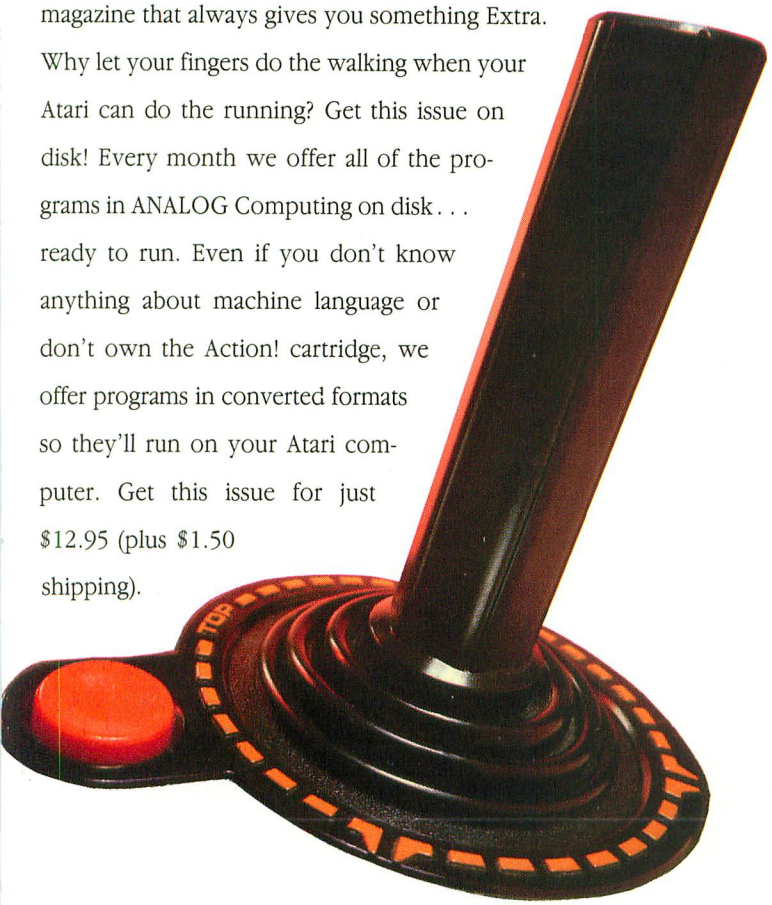
Devotes 50% of Its Pages to the Atari ST?



An Atari 8-bit Extra. While other "Atari" 8-bit magazines just make claims on how they cover your machine, we come through! Over 130 pages of new, never before published material. Programs like Easy Type, Dragon Chase, Pastels, Display List Mod, Tactics, Trivia and Create-a-base are all documented and ready to type in and run. . . all for just $8.95! (Add $1.50 for shipping.)

Get the Extra on disk! This special offer for Extra owners gets you all of the programs in an Atari 8-bit Extra on disk. Avoid typing errors, hours of tedious typing and frustration. Just plug in the disk and you are ready to roll! Two, ready-to-run double-sided floppies, $24.95. (Disks only. Atari 8-bit Extra sold separately. Please add $1.50 for shipping.) From the magazine that always gives you something Extra.

Why let your fingers do the walking when your Atari can do the running? Get this issue on disk! Every month we offer all of the programs in ANALOG Computing on disk. . . ready to run. Even if you don't know anything about machine language or don't own the Action! cartridge, we offer programs in converted formats so they'll run on your Atari computer. Get this issue for just $12.95 (plus $1.50 shipping).



ANALOG COMPUTING

ANALOG COMPUTING OFFICIAL ORDER FORM

Use this coupon to order the most complete up-to-date products specifically designed for your ATARI PC!

ANALOG'S BEST—Graphics Disk	$ 9.95	$ _____
ANALOG'S BEST—Educational Disk	$ 9.95	$ _____
ANALOG'S BEST—Utilities #1	$ 9.95	$ _____
ANALOG'S BEST—Utilities #2	$ 9.95	$ _____
ANALOG'S BEST—Disk Utilities	$ 9.95	$ _____
ANALOG'S BEST—Games #1	$ 9.95	$ _____
ANALOG'S BEST—Games #2	$ 9.95	$ _____
ANALOG'S BEST—Games #3	$ 9.95	$ _____
ANALOG'S BEST—Games #4	$ 9.95	$ _____
ANALOG'S BEST—Games #5	$ 9.95	$ _____
ANALOG COMPUTING—POCKET REFERENCE CARD	$ 7.95	$ _____
ANALOG COMPUTING—8-bit EXTRA	$ 8.95	$ _____
ANALOG COMPUTING—8-bit EXTRA (on disk)	$24.95	$ _____
ANALOG MAGAZINE ON DISK (please specify issue)	$12.95	$ _____
SHIPPING AND HANDLING—add $1.50 for each product ordered		$ _____
TOTAL ORDER		$ _____

☐ Payment Enclosed Charge My ☐ VISA ☐ Master Card

Card # _____ Exp. _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Make checks payable to: **LFP, Inc. P.O. Box 67068, Los Angeles, CA 90067.**
Your order will arrive in 4 to 6 weeks — WATCH FOR IT! ZIHYY

California residents add 6.5% sales tax on all orders except back issues.

Some Call It A Refreshing Change

We Call It The FA-ST Hard Drive

We named our drive after its swift and aggressive behavior. But it's really not fair to limit this incredible peripheral to just one name.

Call it cool. Cool, calm and collected with its whisper-quiet fan to prevent heated situations.

Call it high-class. With refined style, its sleek design complements your Atari computer system. Quite simply, functional elegance under your monitor that's designed to adjust to your system and lift your sights for easy viewing.

Call it friendly. Our FA•ST Hard Drive welcomes a host of features like dual DMA ports which invite new devices. Our SCSI expansion is ready when you are. And inside, our drive can handle a partner like no others can. Have the time? The FA•ST drive does . . . the right time, everytime.

Call it durable. Unwavering dependability from a winning design. Only the best components are found inside our FA•ST Hard Drive. A full one year warranty and ICD's uncompromised reputation for quality should say it all.

Now, don't let the abundance of features scare you . . . FA•ST Hard Drives are available in all sizes *and* at prices you can afford. So, to be quite honest, we really don't care what you call our hard drive – as long as you call for it today. And get ready for the best thing that ever happened to your Atari ST.

Call or write for our free catalog today.

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

ICD

CIRCLE #108 ON READER SERVICE CARD.