How To Choose An Income Tax Program

COMPUTE!

$2.95
December
1985
Issue 67
Vol. 7, No. 12
$3.75 Canada
02193
ISSN 0194-357X

The Leading Magazine Of Home, Educational, And Recreational Computing

SpeedScript Revisited
Enhancements To *SpeedScript 3.0*

Balloon Crazy
Exciting Arcade Game For Commodore 64/128, Atari, Apple, IBM PC/PCjr, TI

Atari Fine Scrolling
Secrets Of Advanced Atari Graphics

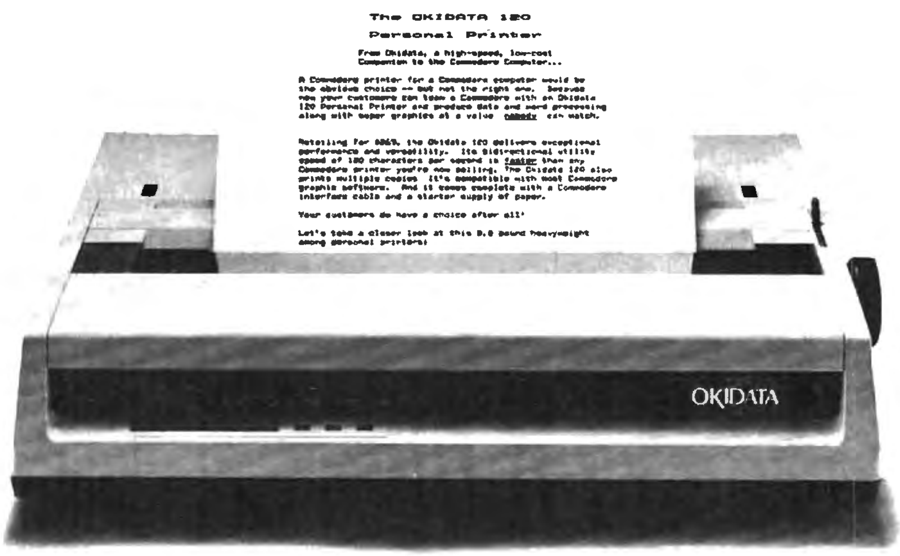
Commodore 128 Video
Design Your Own Custom Characters

Million-Color Palette For PC & PCjr
Create Extra Colors Without Extra Hardware

Apple Disk Booster
Increase Disk Capacity With This Short Program

12
0 71486 02193

Left Brain.



Rational. Functional. Precise.
Introducing the OKIDATA 120, the logical printer for your Commodore* computer.

Get results fast. With a utility mode that zips through letters and reports at twice the speed of any Commodore printer.

Switch to the enhanced mode. And print your most important ideas with typewriter clarity. Or illustrate your rationale with the 120's bit image graphics for high resolution charts, graphs and drawings.

Stay on target. With a self-inking "Clean Hands" ribbon cartridge. And Okidata's famous full year warranty on parts, labor and printhead.

The OKIDATA 120. At $269*, it's the only Commodore-compatible printer that makes sense.

For more information, call 1-800-OKIDATA (in New Jersey 609-235-2600). Mt. Laurel, NJ 08054.

OKIDATA®
an OKI AMERICA company
We put business on paper.

Get the OKIDATA 120 at these fine stores:

Advantage Computer Accessories (Canada)
Childworld/Children's Palace
Consumers Distributing
David Weis
Electronics Boutique/ Games & Gadgets
Federated
Fred Meyer
Lionel/Kiddie City/ Lionel Playworld
Montgomery Ward (at participating stores)
S.E. Nichols
Service Merchandise
Toys 'R Us

MS-8348

*Manufacturer's suggested retail price.

SYLVIA PORTER'S PERSONAL FINANCIAL PLANNER DOES MORE THAN MANAGE YOUR MONEY

IT PLANS YOUR FINANCIAL FUTURE TOO

Sylvia Porter, and the editors of Sylvia Porter's Personal Finance Magazine, now combine with all the computer tools you'll ever need to help manage your money on a day-to-day basis and plan your financial future, too. In Sylvia Porter's style, without complicated financial jargon or "computerese".

Volume 1

Your Personal Financial Planner:

Helps you track your day-to-day financial data, then combines this information with your future financial objectives to produce the most comprehensive and easily-understood financial planning program available.

For Your Day-to-Day Affairs:

- Maintains your electronic checkbook and credit card system.
- Writes your checks and balances your checkbook. (We even built in a calculator and memo pad for you.)
- Prepares and monitors your budget.
- Classifies and tracks your taxable income and expenses.
- Calculates your net worth and generates customized personal financial statements.
- Tracks your financial assets – and your insurance policies.
- Graphically generates supplemental data, such as percentages, ratios and charts.
- You get our Toll-Free Hotline and our Customer Technical Support Team at no charge.
- You get Timeworks' Money Back Guarantee. (Details in each package.)

For Your Financial Future:

- You'll be led step-by-step through a series of questions regarding your life and lifestyle, your financial goals, and your current financial condition. Your answers will enable a computer to determine and print a summary of the amounts you must save each year to meet your financial objectives - in both real and inflated dollars.
- Helps you plan for protection against major medical adversities and other financial setbacks.
- Each program interfaces with others in this series. Your information can be incorporated into letters and reports produced by Timeworks' Word Writer.
- Everything is integrated. You need to enter data only once.

Available for Apple, IBM and Commodore computers.

Moderately Priced - from your favorite Dealer or contact Timeworks for the Dealer closest to you.

Next in this integrated series:
Your Personal Investment Manager.

Other Timeworks Programs: The Evelyn Wood Dynamic Reader • Word Writer with Spell Checker • Data Manager 2 • SwiftCalc with Sideways • Business Systems • Swiftax • Cave of the Word Wizard • Wall Street

From America's #1 Financial Adviser

TIMEWORKS

More power for your dollar.

TIMEWORKS, INC., 444 Lake Cook Rd., Deerfield, IL 60015, 312-948-9200

COMPUTE!

DECEMBER 1985
VOLUME 7
NUMBER 12
ISSUE 67

FEATURES

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

AP Apple, **Mac** Macintosh, **AT** Atari, **ST,** Atari ST, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **128** Commodore 128, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **AM** Amiga. *General interest

TOLL FREE Subscription Order Line
800-334-0868 (In NC 919-275-9809)

COMPUTE! Publications,Inc. abc
One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019
Address all inquiries to:
P.O. Box 5406, Greensboro, NC 27403

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. POSTMASTER: Send address changes to: **COMPUTE!** Magazine, P.O. Box 10955, Des Moines, IA 50950. Second class postage paid at Greensboro, NC 27403 and additional mailing offices.

ISSN 0194-357X.

Editor's Notes

In this month's guest editorial, Apple Applications Editor and Assistant Book Editor Gregg Keizer takes a poke or two at Senior Editor Richard Mansfield. Is he left with a mouse? You decide.

—Robert Lock, Editor In Chief

Mouseketeer. 1. *One who wears large, black ears—usually found in Southern California, Florida, or Tokyo.* 2. *One who uses a small, hand-controlled device (see* mouse*) to direct a computer's actions.*

Last month's Editor's Notes raised some interesting points concerning the two methods of "talking" to computers currently in vogue. Senior Editor Richard Mansfield argued that entering direct commands through the keyboard—such as DIR (DIRectory) or CLS (CLear Screen)—is more desirable than using a mouse. *Mouseketeers,* he claimed, may have the advantage in learning to use mouse-based software, but in the long run sacrifice power and flexibility.

Not all of us agree. The mouse and its system of pull-down menus, dialog boxes, and pictorial symbols (icons) are here to stay; not only here to stay, but pushing keyboard commands out the window.

Ease of Use. The Macintosh established a new standard in making computers easy to use. And it's no accident that newer machines, like the Atari ST and Commodore Amiga, are adopting similar systems. The reasons are obvious.

There's no doubt that mouse-driven operation is easier than typing in commands. From the first time you turn on the computer, managing an operating system and using applications software are far simpler with a mouse and its environment. *Intuitive* is a key word here. Pointing and pressing a single button, selecting and reading, are intuitive. If a child wants something, he or she points to it. Adults haven't forgotten how to do that.

Probably the best test of ease of use is how fast you can get moving in a new program. I recently began using *PageMaker,* a page-layout and design program for the Macintosh. It's definitely an advanced application. Yet, because I was familiar with mouseketeering, I was able to produce and print a page in less than an hour—without more than a glance at the thin manual. Everything was intuitive. Rulers and guides were pulled into place, words typed just where they belonged, and graphics taken from files and cropped to the right size. Contrast that with a more traditional program like *WordStar,* the quintessential keyboard-based word processor. For what it does, *WordStar* is just as sophisticated as *PageMaker.* Yet there are commands I have to look up when I'm using *WordStar,* even though I've written thousands of words with it over the last three years. Few of us can remember two or three dozen commands for *every* program we use.

Know one, know all. If you were simply dealing with the computer's operating system—the way the machine handles such tasks as deleting or renaming files—mouse and keyboard might be more comparable. But most of us don't spend that much time with the operating system. We use the computer to run programs for a specific task. A spreadsheet one time, a word processor the next. With a mouse-driven computer and well-written software, it's as easy to learn and use one program as another. The knowledge base is there. Knowing how to make a menu choice in *Multiplan* means you know how to do the same in Microsoft *Word.* You don't have to spend time learning the basics over and over.

What it does, not how it does it. Given these aspects of mouseketeering, why would anyone want to use keyboard commands? The usual reason is that you can get inside the computer, controlling it more directly. *Power user* is a term that often crops up.

Yet even the IBM PC is succumbing to mouseketeers. Operating environments like *Topview,* Microsoft *Windows,* and *GEM,* all which use Macintosh-like control, are having an impact. One of the bestselling accessories for the IBM PC is a mouse. Popular software like *Sidekick* uses extensive menus.

More people are interested in *doing* something easily and quickly with the computer than in trying to remember *how to do* something easily and quickly.

This is a key to pulling more people into computing. Most people won't stand for complex directions on a computer any more than they'll tolerate thick manuals for a microwave or VCR. We want to do something with our tools, and we want to do it *now,* not in three days. That may be instant gratification, but computer manufacturers must realize it's vital to their success.

That's why the introduction of the ST and Amiga, and the continuing sales of the Macintosh, show so much promise for computing. Computer intimidation will be long forgotten once the A> prompt becomes history. Mouseketeering is no Mickey Mouse concept—it's the preferred gateway to a computer.

Next month, by popular demand, COMPUTE! is kicking off the new year with an exciting new service: the COMPUTE! DISK. Now you can get all the programs for your computer without hours of typing. Each quarterly disk will contain every program published for your machine in the current and two previous issues, ready to load and run. The first disk, for the Commodore 64/128, has all the programs in the January 1986 issue—including the professional-quality spreadsheet, *SpeedCalc*—and all the programs from the November and December 1985 issues. As a special bonus, the January 1986 disk also includes *SpeedScript 3.2,* an updated version of COMPUTE!'s popular word processor. The Apple COMPUTE! DISK debuts in February 1986, followed by the Atari COMPUTE! DISK in March and the IBM COMPUTE! DISK in April. The Apple and Atari disks also will feature *SpeedCalc* and *SpeedScript.*

Each COMPUTE! DISK costs only $12.95 plus $1 shipping and handling. Or you can order a year's subscription for $39.95 (a $12 savings). Call TOLL FREE 1-800-334-0868 (in NC call 1-919-275-9809) 8:30 a.m.–4:30 p.m. eastern standard time. To order by mail, send check or money order to *COMPUTE! Publications, Inc., P.O. Box 5058, Greensboro, NC 27403 USA.* Readers outside the U.S. and Canada add $2 shipping and handling. All orders must be prepaid in U.S. funds.

Coming In Future Issues

***SpeedCalc:* Professional Spreadsheet With *SpeedScript* Integrator For Commodore 64/128, Apple, Atari**

Making Music With MIDI Plus A MIDI Buyer's Guide

Solitaire For Commodore 64/128, Plus/4, 16, Atari, Apple, IBM PC/PCjr, TI

Disassembler 64

Apple Keyboard Customizer

Atari RESET Controller

IBM Advanced Function Key Techniques

Formatted Printouts For Commodore 64, 128, VIC-20, Plus/4, 16

COMPUTE! Publications, Inc. publishes

COMPUTE!
COMPUTE!'s GAZETTE
COMPUTE! Books
COMPUTE!'s GAZETTE DISK

Corporate office:
324 West Wendover Avenue
Suite 200
Greensboro, NC 27408 USA

Mailing address: COMPUTE!
Post Office Box 5406
Greensboro, NC 27403 USA

Telephone: 919-275-9809

Subscription Orders

COMPUTE!
P.O. Box 10955
Des Moines, IA 50950

TOLL FREE Subscription Order Line
800-334-0868
In NC 919-275-9809

COMPUTE! Subscription Rates (12 Issue Year):

US	(one yr.)	$24
	(two yrs.)	$45
	(three yrs.)	$65
Canada and Foreign Surface Mail		$30
Foreign Air Delivery		$65

Advertising Sales

1. New England
Jonathan M. Just
Regional Manager
212-315-1665

2. Mid Atlantic
John Saval
Eastern Advertising Manager
212-315-1665

3. Southeast & Foreign
Harry Blair
919-275-9809

4. Midwest
Gordon Benson
312-362-1821

5. Northwest/ Mountain/Texas
Phoebe Thompson
408-354-5553

6. Southwest
Ed Winchell
213-378-8361

Director of Advertising Sales
Ken Woodard

COMPUTE! Home Office 919-275-9809.

Address all advertising materials to:
Patti W. Stokes
Advertising Production Coordinator
COMPUTE! Magazine
324 West Wendover Avenue,
Greensboro, NC 27408

The COMPUTE! subscriber list is made available to carefully screened organizations with a product or service which may be of interest to our readers. If you prefer not to receive such mailings, please send an exact copy of your subscription label to: COMPUTE! P.O. Box 10955, Des Moines, IA 50950. Include a note indicating your preference to receive only your subscription.

This is the most interesting football game I have seen.
Howard Boldebuck
Lombard, IL
Police

Thank you for the great game, with printer options.
Daniel H. Antolec
Monona, WI
Police

SUPER BOWL SUNDAY is a very enjoyable game.
Charles M. Bolton
Jacksonville, Arkansas
US Air Force

Excellent game. Best sports simulation I've seen.
Richard G. Miller
Duluth, Georgia
Engineer

The game is unbelievable! A++!
Matt Ritchie
Cisco, TX
Student

Great game, very realistic.
James Antonaccio
Somers, New York
Architect

SUPERBOWL SUNDAY is the "BEST" football game I ever played.
Michael Cwirko
Monroe, NY
Student

Good games — would like good baseball game!
Scott Peterson
Vernon Hills, IL
Sales Manager

"Nice Animation!"
Kevin McGill
Spring, Texas
Student

Excellent game! Can you expand it (more plays, more teams?)
James Heironimus
Enon, Ohio
Doctor

SUPERBOWL SUNDAY is a great game.
Rick Bray
Eugene, Oregon
Clerk

Very good game.
Mike Trimarco
Franklin Park, IL

This program is fantastic!!
Chris Calkins
Seffner, FL
Computer Technician

Excellent quality — OK.
Michael Zoyrko
Neah Bay, WA
USAF

$10,000 REWARD

if you can prove that any of the testimonials shown here are not true or have been altered in any way!

Finally a good graphics, statistical football program. Would like more printer options.
John Sievila
Waukesha, WI
Photo Engraver

This game is fun to get into!
Taylor Thompson
Arlington, Texas
Student

The best game you've come up with yet.
R. Michael Lesher
Whitehall, PA
Engineer

Make a diskette for "SUPER BOWL SUNDAY" that allows me to create teams.
Rick Dakesian
Lincoln Park, MI

Great game!
Chuck Shea
Las Vegas, NV

More disks! — Coaches disk to create teams; great teams from past — not Super Bowl; college teams.
Greg Addy
Elgin, IL
Credit Approver

Very realistic, would like other sports.
N. Sideratos
Brooklyn, New York
Student

A great game.
Derek Aiello
Warminster, PA
Student

One of the very best sports games!
Jeffrey D. Malley
Philadelphia, PA
Insurance

I was wondering if you sell 1985 teams, & if you update the disk.
Rick Rosenthal
Peabody, MA
Student

Very impressive game. Good concept and implementation.
Patrick K. Moriarty
Richmond, VA
Programmer

Very realistic simulation with excellent graphics.
Jay Hertel
Sterling, Illinois
Student

These type of games are great to play with your kids (sharpens the mind).
Rod Phillips
Craig, CO
Jeweler

Your game is great, just keep on coming out with more.
Joe Yovino
Cincinnati, Ohio
Student

SUPER BOWL SUNDAY

Available NOW on diskette for the Commodore® 64/128 and IBM® PC, PCjr. Coming soon for Apple® family... **$35.00**
Season disks (where you can re-create an entire season) available separately.

Available at: **Toys "R" Us/Lionel Kiddie City/Caldor/Games and Gadgets/Software City Stores/K-Mart Stores/Sears Stores/Montgomery Ward Stores/Warehouse One Stores/Target Stores** and **Better GAME/COMPUTER STORES everywhere.**

microcomputer games DIVISION
The Avalon Hill Game Company
4517 HARFORD ROAD, BALTIMORE, MD. 21214

For Ordering Information
Call TOLL FREE: 1-800-638-9292

Readers Feedback

The Editors and Readers of COMPUTE!

If you have any questions, comments, or suggestions you would like to see addressed in this column, write to "Readers' Feedback," COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Due to the volume of mail we receive, we regret that we cannot provide personal answers to technical questions.

Uploading Files

I work in a publications department of a large company that uses IBM PC-compatible computers. I also have a Commodore 64 at home. Is there any way to convert my Commodore word processing files so they can be read by an IBM PC? I have seen programs that convert Apple files to IBM, but not for Commodore. Is it that it cannot be done, or just that no one has done so yet?

Merton Backlund

With the right setup, you can transfer a file between any two computers. We do it almost every day here at COMPUTE!*—text files received from outside authors on floppy disk or over the phone lines are uploaded directly into our editorial/typesetting computer system.*

In general, the easiest way to transfer files between normally incompatible computers is to link them together over the phone lines with modems. That means each computer must be equipped with its own modem and terminal software (a program that makes the telecommunications link possible). In addition to exchanging word processing files this way, you can also transfer programs—although they'll need to be translated by a programmer before they'll run on the other computer, of course.

If both computers are in the same room, or nearby, sometimes you can avoid the expense of equipping each computer with a modem by using a null modem cable. *This is a special cable which links the computers together by connecting to their interface ports (usually the RS-232 serial port). If this is done properly, each computer thinks it is talking to the other via modem, even though no modems are involved. However, null modem cables to fit every possible situation aren't easy to come by. Usually they must be custom-made by a technician familiar with both computers.*

In your particular case, modems are the solution, since you want to transfer files over a distance (home to office). If the computer at work will be unattended when you plan to transfer your files, you'll have to equip it with an autoanswer modem that can answer the phone and receive information automatically. Make sure the terminal programs you get allow uploading and downloading (the capability to send and receive files) and are otherwise compatible with the modems.

Before sending a word processing file, delete all special formatting commands from the document, such as those which trigger different printing styles, headers, footers, page numbers, centering, and so on. The other computer's word processor won't understand these formatting commands, and the control codes might interfere with the telecommunications link. The file you're preparing for transfer should be pure text.

One complication in your case is that Commodore and IBM computers use different codes to represent characters, although the codes for both are derivatives of ASCII (American Standard Code for Information Interchange). Though IBM ASCII is nearly identical to standard ASCII, Commodore ASCII is quite different. A good terminal program can convert Commodore ASCII to standard ASCII characters as it sends the file. So if your word processor stores characters as Commodore ASCII codes, you may able to send the files in their present form. However, many popular word processors for the 64 (including COMPUTE!'s SpeedScript*) store characters as screen codes, which are different from ASCII codes. Before sending such a file you must convert each screen code to its ASCII equivalent. Though it's too long to include here, a file converter program was published as part of the article* "SpeedScript 3.0: *All Machine Language Word Processor for Commodore 64" in* COMPUTE!*, March 1985. This program converts text files from Commodore screen codes to Commodore ASCII or standard ASCII, and Commodore ASCII files to screen codes.*

Your particular situation may require a little additional conversion. On Commodore computers, the code CHR$(13) performs both a carriage return *(moving the cursor back to the left margin) and a* line feed *(moving the cursor down one line). In IBM ASCII, these are separate functions: CHR$(10) performs a line feed and CHR$(13) does a carriage return. Here's a short IBM program that adds the line feeds:*

```
10 ON ERROR GOTO 60 'Add CHR$
   (10) to each CHR$(13) in t
   ext file
20 INPUT "Filename";N$:INPUT
   "Conversion filename";M$
30 OPEN N$ FOR INPUT AS #1:OP
   EN M$ FOR OUTPUT AS #2:WHI
   LE EOF(1)=0
40 C$=INPUT$(1,#1):PRINT #2,C
   $;:IF C$=CHR$(13) THEN PRI
   NT #2,CHR$(10);
50 WEND
60 CLOSE 1:CLOSE 2:ON ERROR G
   OTO 0
```

Apple Mousetext

I own an Apple IIc and have heard it has 32 special "Mousetext" characters built into ROM. How can I access these characters? Can I use them in my BASIC programs?

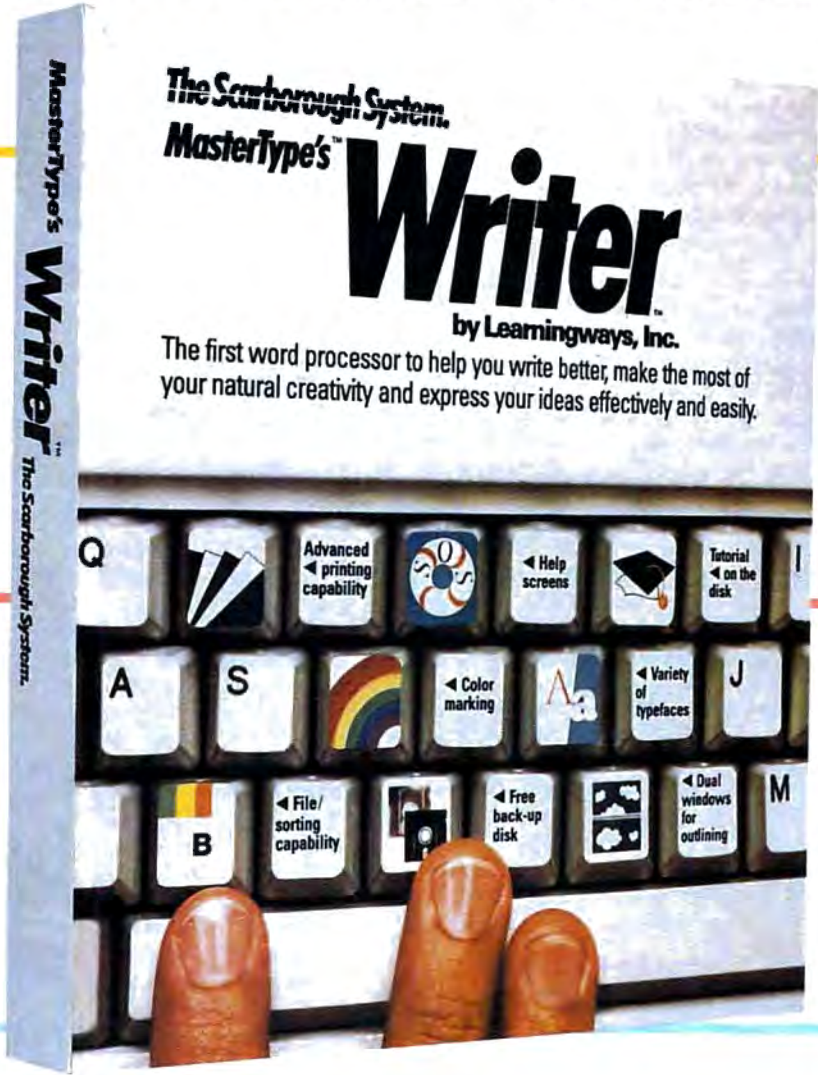
Murray Hanstead

Apple's Mousetext characters are designed especially for mouse-driven (Macintosh-style) software, but they can be used any time you need additional characters. Available on the enhanced IIe as well as the IIc, they include line segments, arrows, open and closed apples, cursors, and more. The Mousetext character set works in 40 or 80 columns and is easy to use from BASIC.

Apple II computers with Mousetext also contain enhanced video firmware, *a collection of screen routines in ROM (Read Only Memory) used in place of the original monitor routines. One advantage of the enhanced firmware is that it makes Mousetext much easier to use. When enhanced video is activated, the cursor is an inverse box instead of the normal flashing box. To turn on the firmware on the Apple IIc, press the ESC key, followed by the 4 key for 40 columns or the 8 key for 80 columns. On the extended IIe, type PR#3 and press RETURN (this also works on the IIc). PR#3 puts you in 80-column mode; use ESC-4 and ESC-8 to switch between 40 and 80 columns.*

Once the enhanced firmware is on, PRINT CHR$(27) replaces inverse mode uppercase characters with Mousetext.

"...makes turning ideas into text a delight."
—*Classroom Computer Learning*



"Head and shoulders above Bank Street Writer." —*Learning Lab*

"It's an exciting new word processor with great power, despite its ease of use." —*Publishers Weekly*

MasterType's Writer offers new features and ease of use that set it apart from every other comparably-priced word processor. Each and every feature has one overriding purpose: to help you express your ideas clearly and persuasively.

Whether you just bought your computer or are a veteran at the keyboard, with MasterType's Writer, you'll be able to write faster, more easily and more effectively than ever before.

And MasterType's Writer's advanced printing capabilities will help you get the most out of your Epson printer or whatever popular printer you have. And that means you'll be more productive at home—you'll have a jump on the competition, at work or in school.

MasterType's™ Writer is the first word processing program that actually helps you write better—to be more productive at home, succeed at school, get ahead at work.

Available for Apple II Family, Commodore 64/128. Atari 800XL/130XE coming soon.

The Scarborough System.

Now the INVERSE command turns Mousetext on and the NORMAL command turns it off. Uppercase characters printed in inverse mode appear as Mousetext, but inverse lowercase characters are still available. PRINT CHR$(24) restores the normal inverse uppercase characters without affecting Mousetext already on the screen. By using the INVERSE and NORMAL commands and PRINTing CHR$(27) and CHR$(24), it's possible to mix Mousetext, inverse uppercase, and normal uppercase on the screen at once. To see all the Mousetext characters, type in and run the following one-line program:

```
10 PRINT CHR$(4);"PR#3":PRINT
    CHR$(27):FOR I=64 TO 95:
   INVERSE:PRINT CHR$(I);:NOR
   MAL :PRINT" ";:NEXT :PRINT
    : FOR I=64 TO 95: PRINT C
   HR$(I);:NEXT
```

Atari Custom Characters

I know how to use CALL CHAR on the TI-99/4A computer to create custom characters, but how is this done on the Atari?

Marc Breaux

Atari BASIC lacks a command such as CALL CHAR to redefine characters in a single step, so you have to build a routine with PEEKs and POKEs instead. There are four steps involved, as demonstrated by the following program, which changes the exclamation point into an alien shape.

*First, line 10 lowers the top of memory to reserve a protected area for the new character set. This example lowers the top of memory by 2,048 (8*256) bytes, enough room for a full character set. You must declare a graphics mode after doing this to make the computer relocate screen memory just below the protected area. Next, line 20 copies the part of the original character set you'll need from ROM (Read Only Memory) into the protected memory. The ROM characters start at location 57344. Line 30 then POKEs the data for the new characters into memory. Finally, line 40 tells the computer where to find the new character set by POKEing the high byte of the new character set's address into location 756.*

Numerous articles describing these techniques in more detail have appeared in past issues of COMPUTE! *and are reprinted in such books as* COMPUTE!'s First Book of Atari Graphics *and* Second Book of Atari Graphics.

```
10 A=PEEK(106)-8:POKE 106
   ,A:GRAPHICS 0:CHBAS=25
   6*A:REM PROTECT 1024 B
   YTES OF MEMORY
20 FOR A=0 TO 2047:POKE C
   HBAS+A,PEEK(57344+A):N
   EXT A:REM COPY NORMAL
   CHARACTER SET TO RAM
30 FOR A=CHBAS+8 TO CHBAS
   +15:READ B:POKE A,B:NE
   XT A:REM PUT NEW CHARA
   CTER DEFINITION AT EXC
   LAMATION POINT
40 POKE 756,A:REM CHANGE
   CHARACTER POINTER
50 DATA 60,126,90,126,60,
   36,66,129
```

40 IBM Function Keys

I have an IBM PC and have written many BASIC programs. To increase speed and minimize typing errors, I usually reassign all 10 function keys. But sometimes 10 keys is not enough. I have seen programs like *Symphony* that allow as many as 20 function keys. Is there any way I can use the ALT key to assign additional function keys?

Ralph D'Angelo

As you've learned, IBM BASIC supports only 10 soft key assignments. Function keys 1–10 are called soft keys and can be reassigned with a statement like KEY 1, "CLS:FILES" + CHR$(13) in direct mode or in a program. The KEY() and ON KEY() GOSUB statements make it possible to trap as many as six additional keys (see "Readers' Feedback," September 1985), but that method can't provide a full extra set of function keys. However, you can get four sets of function keys—40 keys in all—by checking for extended scan codes. *When you press a single key, it generates a single scan code (a number in the range 0–255). Extended (two-number) scan codes are generated when you press ALT, CTRL, or SHIFT with another key. This program illustrates one keyboard scanning method that works on both the PC and PCjr; it detects F1, ALT-F1, CTRL-F1, and SHIFT-F1, displaying the scan codes for whatever keys you press.*

```
0 FOR J=1 TO 10:KEY J,"":NEXT
   J
1 DEF FNF1(X)=(X$=CHR$(0)+CHR
  $(59)):DEF FNALTF1(X)=(X$=C
  HR$(0)+CHR$(104))
2 DEF FNCTRF1(X)=(X$=CHR$(0)+
  CHR$(94)):DEF FNSHFF1(X)=(X
  $=CHR$(0)+CHR$(84))
3 X$=INKEY$:ON (FNF1(A)*-1)+(
  FNALTF1(A)*-2) GOSUB 6,7
4 ON (FNCTRF1(A)*-1)+(FNSHFF1
  (A)*-2) GOSUB 8,9
5 FOR J=1 TO LEN(X$):PRINT AS
  C(MID$(X$,J,1)):NEXT J:GOTO
   3
6 PRINT "Pressed F1":RETURN
7 PRINT "Pressed Alt-F1":RETU
  RN
8 PRINT "Pressed Ctrl-F1":RET
  URN
9 PRINT "Pressed Shift-F1":RE
  TURN
```

Lines 1–2 define user functions for the key combinations we want to detect. The INKEY$ statement in line 3 returns the scan codes in X$; and the ON-GOSUB statements in lines 3–4 transfer control to appropriate subroutines. Detecting additional key combinations is simply a matter of adding more user functions and appropriate subroutines. Page G-7 of the IBM BASIC Manual *and pages G-6–G-7 of the* PCjr BASIC Manual *list all the extended scan codes; note that certain key combinations don't generate extended codes.*

Arabian Atari?

I own an Atari 800 and have been trying to change the movement of the cursor so that I can type from right to left instead of left to right. I have looked at a large number of books without finding any answer. Is this possible in Atari BASIC?

Nour Abdullah Al-Rasheed
P.O. Box 2532
Hofuf, Al-Hassa 31982
Kingdom of Saudi Arabia

Every microcomputer that we've seen expects characters to be arranged in the left-to-right order common to Western languages. However, the arrangement of text is simply a convention, and some languages use different conventions. Arabic, Hebrew, and Japanese are read from right to left, Chinese is read from top to bottom, and so on.

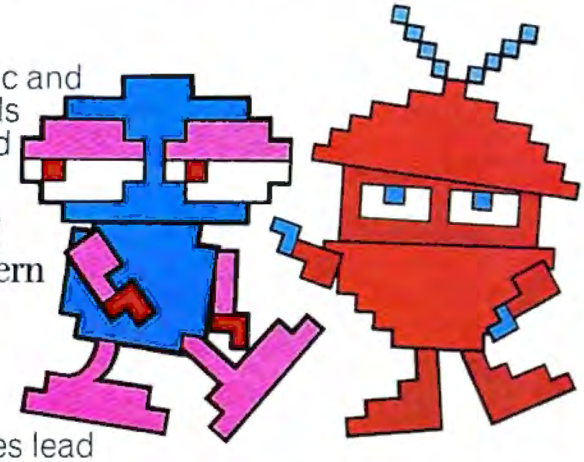
A true solution to your problem in Atari BASIC is next to impossible. To really make it work, you'd have to rewrite (in machine language) every part of BASIC and the operating system (OS) that manipulates text. Since BASIC and the OS are large, complex machine language programs, this project could take an expert programmer weeks or months. Then, to make the change permanent, you'd need to burn the modified BASIC and OS—as well as a new character set—onto PROM (Programmable Read Only Memory) or EPROM (Erasable PROM) chips and replace the machine's original chips. It's not impossible, but it involves far more labor than most people would be willing to expend. Also, there would probably be compatibility problems with commercial software.

If you don't mind a little inconvenience, there is a crude solution. Move your monitor to one side and turn it sideways, then mount a large mirror at an angle where the monitor used to be. Watch the mirror instead of the monitor, and each line appears to be typed from right to left. *Once that's done, you need to design a set of backward characters that will appear correct when viewed in the mirror. We have reprinted your address in case any of our foreign readers have a better solution.*

Making ML Loaders

Please give me a Commodore program that turns a machine language program into BASIC lines like the listings for

The computer age begins at four.



TINK!TONK!® Software. The computer classroom for ages 4 and up.

Mindscape's TINK!TONK! series provides the kind of stimulation for young minds that makes learning child's play.

Step-by-step you'll watch the smiles and chart new growth.

Take *Tonk In The Land of Buddy-Bots* and *Tink's Subtraction Fair* to see what we mean. Each has 5 separate learning games with 3 levels in addition to a fascinating adventure. Lively graphics, animated action, music and sound effects keep kids involved, attentive, and amused.

Watch TINK!TONK! teach math, logic, pattern recognition, and pre-reading skills.

TINK, TONK, and their friends are lively tutors. Their adventures lead kids to new levels of knowledge and understanding in math, reading basics, map reading, logic, and critical thinking. Each program focuses on different skills and activities.

An attention span that lasts for years.

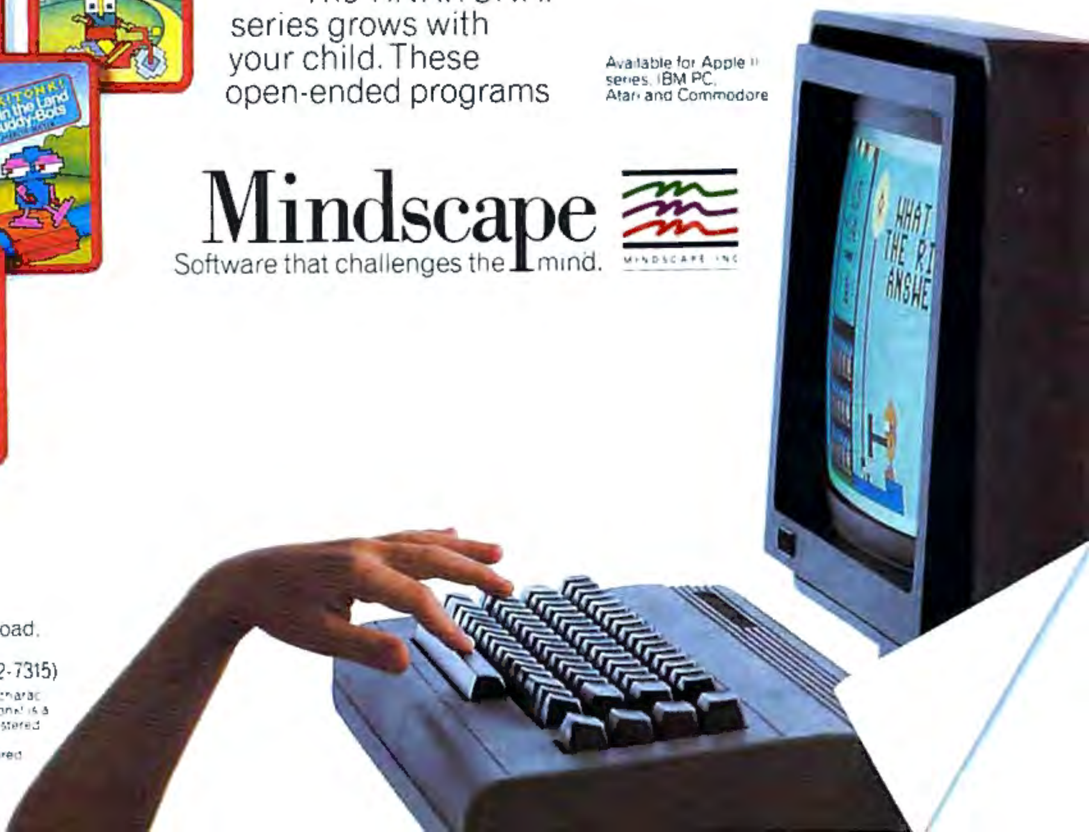
The TINK!TONK! series grows with your child. These open-ended programs let children progress at their own pace.

So open a new world of learning for your child. Cultivate basic skills. And begin their computer age with plenty of fun.

Just think what that can mean for the future.

Available for Apple II series, IBM PC, Atari and Commodore

Mindscape
Software that challenges the mind.



Mindscape, Inc. 3444 Dundee Road, Northbrook, Illinois 60062
1-800-221-9884. (In Illinois 1-800-942-7315)

Where to find and buy ColorMe: The Computer Coloring Kit.

National
Available at all Child World and Children's Palace Stores
Federated Group
K-Mart — Most larger stores
Montgomery Ward — Most larger stores
Sears — Most larger stores

Alabama
The Computer Shoppe
Huntsville

Arkansas
The Computer Shoppe
Little Rock

California
Access to Software
San Francisco
Alamo Computer Center
Cupertino
Alamo Electronic Components
San Jose
Boot's Camera Electronics
Fresno
Candy Computer
Elk Grove
Computermart
Diamond Bar
Crown Book & Software
Los Angeles, Santa Monica, Marina Del Rey, Studio City, Santa Ana, Redondo Beach, Hollywood, Glendale, Culver City, South Pasadena, Thousand Oaks, Torrance, Huntington Beach, Downey, Sherman Oaks, Encino, El Toro, Long Beach, Palos Verdes, Northbridge, Pasadena, Westminster, Woodland Hills, Ventura, Lakewood
Egghead Software
Los Angeles, Huntington Beach
Futurvision
Napa
H.T. Electronic, Inc.
Sunnyvale
Home Computing Centers
San Bruno, San Leandro
Learning Tree Computer
Santa Ana
Software First
San Rafael and Santa Rosa
Software Galeria
Orange
Software House
Fresno
Software Solution
Chico
Software Station
Costa Mesa
Software Supermarket
Lawndale, Los Angeles, Sherman Oaks
Software World
Redding

Colorado
Kazoo & Company
Denver

Connecticut
Caldor
West Norwalk, Old Saybrook, Enfield, Milford, W. Hartford, Vernon, Riverside, Norwalk, Brookfield, Trumbull, Mansfield, Norwich, Hamden, Waterbury, Manchester, Stamford, Wallingford, Avon, Rocky Hill, Ridgefield, Torrington, Branford, Bristol, Meddletown, Fairfield, Groton, New Britain, Derby, Southington

Connecticut (cont)
Crazy Eddie
Norwalk
Personal Computer Center
Norwich
Software City
Stamford, Orange
Software Kingdom
E. Hartford, E. Windsor
Video Connection
Bridgeport

District of Columbia
Crown Book & Software
(3 locations)

Florida
Games 'N Gadgets
Clearwater, Jacksonville, Tallahassee, Orange Park
Maxicat
Miami
Program Store
Clearwater, Pinellas Park

Georgia
Electronics Boutique
Atlanta — Lenox Square, Cumberland Mall
Games 'N Gadgets
Atlanta — North Lake Mall, Perimeter Mall

Hawaii
Software Library
Honolulu

Illinois
ABC Diskette
Chicago
Apprat Computers
Arlington Heights
Complete Computing
Lombard
ComputerLand
Niles
Computerworld
Chicago
Crown Book & Software
Orland Park, Niles, Wheaton, Evanston, Oaklawn, Oak Park, Chicago, Skokie, Deerfield, Downers Grove, Highland Park, Naperville, Hoffman Estates, Arlington Heights, Calumet City
Disk-N-Tech
Orland Park
F.D. Software & C.A.D.S.
Roselle
Software and Beyond
Schaumburg
Software City
Arlington Heights
Software Plus West
Hanover Park
Susie Software
Mount Prospect

Indiana
Burkat Computer Center
South Bend
Microcomputer, Inc
Indianapolis

Kentucky
Software Source
Louisville
The Computer Shoppe
Louisville (2 locations)
Videovisions
Louisville

Maryland
Buried Treasure
Rockville
Caldor
Timonium, Baltimore, Severna Park, Ellicott City

Maryland (cont)
Computer Kids
Rockville
Crown Book & Software
Wheaton, Columbia, Laurel, Kensington, Silver Springs, Bowie, Bethesda, Greenbelt, Gaithersburg, Annapolis, Rockville
Electronics Boutique
Baltimore
Games 'N Gadgets
Columbia, Baltimore, (White Marsh Mall, Security Square Mall, Eastern Ave.)
Software 'N Things
College Park

Massachusetts
Atlantic Computers
Waltham
Barnes & Noble Bookstore
Boston
Caldor
Chelmsford, Worcester, Westborough, Swansea, Pembroke, Westfield, Auburn, Malden, Norwell, Brockton, Canton, Brighton, Taunton, Salem, Chicopee, Stoneham, Springfield, Framingham, Northampton, Weymouth, Saugus, Methuen, Burlington
Computer Concepts
Hanover
Electronics Boutique
Holyoke
Software City
West Springfield
The Whiz
Westboro

Michigan
Micro-World
Livonia and Milford

Missouri
Software To Go
St. Louis and Clayton

Nevada
Century 23
Las Vegas

New Hampshire
Caldor
Bedford
Portsmouth Computer Center
Portsmouth

New Jersey
Bamberger's
Menlo Park, Cherry Hill, Paramus, Livingston, Wayne
Caldor
Eatontown, S. Plainfield, West Orange, Toms River, Watchung, Paramus, Morris Plain, Woodbridge, Brick Town, N. Brunswick, W. Paterson
Crazy Eddie
Paramus, E. Brunswick, Union, Totawa, Woodbridge
Electronics Boutique
Voohrhees, Rockaway, Woodbridge
Family Computer Centres
South Orange
Games 'N Gadgets
Livingston, Burlington, Wayne
Wolsten's Inc.
East Orange

New Mexico
Academy Computer
Albuquerque

New York
45th Electro
New York
47th St. Photo, Inc.
New York (3 locations)
Advanced Camera
New York
Annex Outlet
New York
Barnes & Noble Bookstore
New York (2 locations)
Big L Electronics
W. Hempstead, W. Babylon, Syosset, New Hyde Park, Patchogue
Binary Orchard Inc.
Hamburg, West Seneca
Broadway Computers
New York
Byte Shop
Merrick
Caldor
Lake Ronkonk, East Patchogue, Wappingers Falls, Riverhead, Rocky Point, Bridgehampton, Vails Gate, Glens Falls, Kingston, Middletown, Latham, Mahopac, Bedford Hills, Albany, Yorktown Heights, Peekskill, Pelham Manor, Schenectady, Nanuet, White Plains, Yonkers, Newburgh, Port Chester, Poughkeepsie, W. Babylon, Levittown, Coram, Clifton Park
Computer Discount of America
Huntington
Computerware
East Meadow
Computerworld
Bayshore
Crazy Eddie
Nesconset, Mass, New York, Syosset, Elmhurst, Brooklyn, Carle Place, Hartsdale, Bronx, Nanuet
East 33rd Typewriter
New York
Electronic Man
New York
Electronics Boutique
Albany
Focus Electronic
Brooklyn (2 locations)
Four Guys
New York
Games 'N Gadgets
Garden City, Huntington Station, Nanuet
Glossy Photo
New York
Great American Software
Flushing
Intercontinental
Flushing
J&R Computer Outlet
New York
J&S Electronic
New York
Lloyd's
New York
Lloyd Corner
New York
London Luggage
New York
Micro Electronics
Valley Stream
Montgomery Grant
New York
Park Avenue Video
New York
Photo Sound
New York
Programs Plus
Brentwood

New York (cont)
Quartz Electronics
New York
Quo Vadis
Ridgewood
Show & Tell Video
Bayshore
Software City
Brooklyn, Flushing
Staten Island Software
Staten Island
Wall Street Camera
New York
Willoughby's
New York
World Trade
Bayside

North Carolina
Games 'N Gadgets
Raleigh

North Dakota
Software House
Fargo

Ohio
Big Bytes Computers
Poland
Computer Center
Mentor
Computer Corner
Lorain
Computers Plus of Ohio
Mansfield
Disk Drive
Toledo
Dunn Computer Center
Lynhurst
Electronic Connexion
Kettering
Games 'N Gadgets
N. Randall
Local Computer Store
Chesterland
Microwave Magic
Fairfield
North Coast Programming
Willoughby and Solon
Software and More
Cincinnati
Swallens
Cincinnati
Video H.Q.
Cleveland

Oklahoma
Computer Software Center
Oklahoma City

Oregon
Link Three, Inc.
Beaverton
Software Galeria
Beaverton

Pennsylvania
Bamberger's
Whitehall, King of Prussia
Computer Renaissance
Monroeville
De Re Computers
Harrisburg
Electronics Boutique
Langhorne, King of Prussia, Whitehall, Exton
Games 'N Gadgets
Lancaster, West Mifflin, King of Prussia, Media, Philadelphia

Rhode Island
Caldor
Woonsocket, Warwick, Lincoln
Future Images
Woonsocket

South Carolina
Horizon Commodore Super Store
Greenville (2 locations)

Tennessee
Games 'N Gadgets
Memphis
Software Store
Memphis
The Computer Shoppe
Nashville, Knoxville, Memphis, Chattanooga, Madison

Texas
Babbages
Dallas, Houston, Plano, Mesquite, Fort Worth, Irving, Hurst
Compunite
Houston (2 locations)
Computer Experience
San Antonio
The Computer Store
San Angelo
Data Pro Computer Center
Wichita Falls
Edu-Tron
Fort Worth
Floppy Wizard
Houston
Software & Things
Austin
Software Store
San Antonio
Videoland
All locations

Utah
Software Center
Salt Lake City

Virginia
Crown Book & Software
Alexandria, Woodbridge, Reston, Springfield, Vienna, Manassas, Mclean, Falls Church, Fairfax
Games 'N Gadgets
Hampton, Virginia Beach
Program Store
Falls Church

Washington
Nibbles & Bites
Tacoma

Wisconsin
Computer Software Center
Milwaukee

Note: Not all formats available at all locations

If there isn't a dealer listed in your area, don't give up.
Give us a call. Phone 1-800-221-9884. In Illinois 1-800-942-7315.

MINDSCAPE

your "MLX" machine language entry utility. I want something that looks like this:

49152: 169,091,133,170,169,200,092

Darwin Clay

We frequently get letters from readers who want to generate MLX-style listings for their own machine language (ML) programs. However, while it's occasionally useful to convert an ML program into BASIC, there is no practical reason to list it in MLX format. An MLX listing is not a program—the only thing you can do with it is create a machine language program by running MLX and typing in the listing. Since you already have the ML program, that would be a fruitless exercise. What you seem to need is a BASIC *loader: a BASIC routine that creates a machine language program by reading values from DATA statements and POKEing them into the correct memory locations. BASIC loaders can be added as subroutines to other BASIC programs.*

Here are two short programs that construct loaders, using the dynamic keyboard technique to write the necessary DATA statements. The first program creates a loader from an ML program that's already in memory. Enter the starting and ending addresses of the ML program, followed by the beginning line number for the DATA statements (don't use lines 1–8). When the blinking cursor reappears, your BASIC loader is in memory, ready to be saved. The last line of the loader READs the DATA statements and POKEs the correct values into memory. Be sure to save this program before running it: Lines 1–8 erase themselves after the loader is complete.

Since this method uses string operations, it disrupts the contents of the highest memory addresses used by BASIC (just under location 40960 on the 64). That may cause problems if you're trying to make a loader for an ML program that resides in the same area. The solution is to lower the top of BASIC pointer: To protect a 2K area at the top of BASIC, type the statement POKE 56,PEEK(56)−8:CLR and press RETURN.

```
1 INPUT"START ADDRESS";AD:INPU
   T"END ADDRESS";E:INPUT"FIRST
    LINE NUMBER";L:SA=AD:rem 98
2 PRINT"{CLR}"L"DATA";:rem 123
3 PRINTMID$(STR$(PEEK(AD)),2);
   :NUM=NUM+1:AD=AD+1:IFAD>ETHE
   N7                    :rem 216
4 IFNUM<16THENPRINT",";:GOTO3
                          :rem 99
5 PRINT"{HOME}{2 DOWN}L="L"
   {LEFT}+1:AD="AD"{LEFT}:E="E"
   {LEFT}:SA="SA"{LEFT}:GOTO2"
                         :rem 182
6 POKE198,5:POKE631,19:POKE632
   ,13:POKE633,13:END    :rem 59
7 PRINT"{HOME}{2 DOWN}"L+1"FOR
   J="SA"TO"AD-1"{LEFT}:READQ:P
   OKEJ,Q:NEXT":FORJ=1TO8:PRINT
   J:NEXT                :rem 100
8 POKE631,19:POKE198,12:FORJ=0
   TO10:POKE632+J,13:NEXT:END
                         :rem 160
```

In some cases you may not know the ending address of the ML program, or it may be inconvenient to have the ML in memory while you're making a loader. With only slight modifications, this program can make a loader for an ML program stored on disk. First, in line 7 replace the statement FORJ=1TO8 with FORJ=0TO8. Then replace lines 1, 3, 5, and 8 with the lines shown here. This routine works like the first example, but gets the ML data from disk rather than memory. Again, remember to save the program before running it for the first time.

```
0 INPUT"FILENAME";F$:INPUT"FIR
   ST LINE NUMBER";L:OPEN2,8,2,
   "0:"+F$+",P,R"         :rem 48
1 GET#2,LO$:GET#2,HI$:SA=ASC(L
   O$+CHR$(0))+ASC(HI$+CHR$(0))
   *256:AD=SA             :rem 76
3 GET#2,X$:PRINTMID$(STR$(ASC(
   X$+CHR$(0))),2);:NUM=NUM+1:A
   D=AD+1:IFST<>0THEN7  :rem 142
5 PRINT"{HOME}{2 DOWN}L="L"
   {LEFT}+1:AD="AD"{LEFT}:SA="S
   A"{LEFT}:POKE152,1:GOTO2"
                          :rem 50
8 POKE631,19:POKE198,12:FORJ=0
   TO10:POKE632+J,13:NEXT:OPEN1
   5,8,15:CLOSE15:END   :rem 126
```

The Absent Printer Dilemma

I am trying to write a commercially salable Commodore program in BASIC and wish to make it as crashproof as possible. My problem is this: If the user selects printer output when the printer is disconnected or turned off, the program stops with a DEVICE NOT PRESENT error. Is there any way to detect this condition before the program crashes?

Daniel Henderson

This is a tough problem—so tough that many commercial programs don't even attempt a solution. In BASIC, as you've learned, a CMD or PRINT# command to the printer simply halts program execution with an error message if the device is absent (either turned off or disconnected). The following routine works with Commodore printers which do not require an external interface. Non-Commodore printers are another matter, as we'll explain in a moment.

```
10 DATA32,253,174,32,25,226,32
   ,192,255,162,4,32,201,255,1
   65,144                 :rem 67
20 DATA41,128,133,252,16,5,169
   ,4,32,195,255,32,204,255,96
                          :rem 42
30 SA=828:FORJ=SATOSA+30:READQ
   :POKEJ,Q:NEXT          :rem 39
40 SYS SA,4,4,7:IFPEEK(252)=0T
   HEN80                 :rem 246
50 PRINT"TURN PRINTER ON, PRES
   S ANY KEY"            :rem 202
60 GETA$:IFA$=""THEN60:rem 239
70 GOTO40                  :rem 4
80 PRINT#4,"THIS ROUTINE WORKS
   ":CLOSE4              :rem 148
```

Lines 10–30 place a short ML routine in locations 828–858 (the cassette buffer). You can relocate the ML by changing the value of SA in line 30: Replace 828 with the address where you want the routine to start. The SYS statement in line 40 takes the place of the BASIC statement OPEN 4,4,7 (don't forget the comma after SA). When using this routine, the first two numbers after the SYS must always *be 4. The third number sets the secondary address and may be changed as needed: SYS SA,4,4,6 does the equivalent of OPEN 4,4,6 in BASIC, and so on. Run the program when your printer is on; the printer should print THIS ROUTINE WORKS. If the printer is not ready, the program prints a warning message and lets you remedy the situation. Location 252 holds a zero when the printer is active, and 128 when it is not.*

Unfortunately, this method is limited to Commodore printers. Non-Commodore printers require an external interface between the computer and printer, and most such interfaces draw power whether or not the printer is turned on. Since the interface is always powered up, it responds with an "I am here" signal which convinces the computer that a printer is present—even when the printer is turned off.

Apple, IBM ML Addresses

Can you tell me how to find the beginning and ending addresses of a machine language program for the Apple IIe?

Bill Link

Ever since we told Commodore and Atari readers how to do this, owners of other machines have been asking for equivalent routines. Here are two routines for the Apple II (DOS 3.3 and ProDOS); we've thrown in an IBM PC/PCjr routine for good measure. The Apple II DOS 3.3 routine is listed first.

```
10 INPUT "FILENAME: ";N$
20 P$ = CHR$ (4): POKE 42954,
   0
30 PRINT P$;"OPEN ";N$
40 T = PEEK (46530): IF T < >
    132 AND T < > 4 THEN PRIN
   T "ERROR: "N$" IS NOT A BI
   NARY FILE": GOTO 90
50 PRINT P$;"READ ";N$
60 GET A$,B$,C$,D$: PRINT P$
70 PRINT "ADDRESS: "; ASC (A$
   ) + 256 * ASC (B$)
80 PRINT "LENGTH: "; ASC (C$)
    + 256 * ASC (D$)
90 PRINT : PRINT P$;"CLOSE ";
   N$
100 POKE 42954,127
```

Here is an equivalent routine for Apple II machines running ProDOS:

```
10 D$ = CHR$ (4): INPUT "FILE
   NAME: ";N$
20 IF LEN (N$) < 15 THEN N$ =
   N$ + " ": GOTO 15
30 PRINT D$;"PREFIX"
40 INPUT PN$
50 PRINT D$;"OPEN ";PN$;",TDI
   R"
60 PRINT D$;"READ ";PN$
70 INPUT F$: IF MID$ (F$,2,15
   ) < > N$ THEN 70
80 PRINT D$;"CLOSE ";PN$
90 IF MID$ (F$,18,3) < > "BIN
   " THEN PRINT "ERROR: "N$"
   IS NOT A BINARY FILE": END
100 D = 0:H$ = MID$ (F$,76,4)
    : FOR I = 1 TO 4:H = ASC
    ( MID$ (F$,75 + I,1)):D =
     D * 16 + H - 48 - 7 * (H
     > 57): NEXT
110 PRINT "ADDRESS: ";D
120 PRINT "LENGTH: "; MID$ (F
    $,67,5)
```

While the ProDOS CATALOG command provides the same information as the second example, this program also demonstrates how to open and read a disk file from BASIC, something that's not immediately apparent to many users. The final example finds starting and ending addresses on the IBM PC/PCjr:

```
10 INPUT "File name";A$
20 OPEN A$ FOR INPUT AS #1
30 IF INPUT$(1,1) <> CHR$(253
   ) THEN PRINT "Error: "A$"i
   s not a binary file":GOTO
   70
40 PRINT:GOSUB 90:PRINT "Star
   ting segment: ";S
50 GOSUB 90:PRINT "Starting a
   ddress: ";S
60 GOSUB 90:PRINT "File lengt
   h: ";S
70 CLOSE 1
80 END
90 S=ASC(INPUT$(1,1))+256*ASC
   (INPUT$(1,1)):RETURN
```

Borrowing ML From BASIC

How does a command like SYS 49152,1000,A$(1) work? I know the SYS command calls the routine, but how do you make SYS use the number and the string variable after 49152?

Tim Pickett

It's usually done by calling the same routines in ROM (Read Only Memory) that BASIC uses to accept information. Of course, this must be done from within the ML program called by SYS. Here's a short program for the Commodore 64 that shows one way to handle the statement you mentioned. You'll need a machine language assembler to type it in (the comments after the semicolons are optional):

```
      JSR  $AEFD  ; Check for comma.
      JSR  $AD8A  ; Get numeric expression.
      LDA  #$4E   ; (Put your
      JSR  $FFD2  ; code here.)
      JSR  $AEFD  ; Check second comma.
      JSR  $AD9E  ; Get any expression.
      BIT  $0D    ; Check string/numeric
      BPL  ERROR  ; flag in $0D.
      LDA  #$53   ; (Put your
      JSR  $FFD2  ; code here.)
      RTS
ERROR LDX  #$16   ; Output TYPE
      JMP  $A437  ; MISMATCH ERROR.
```

If you assemble this program at 49152, it accepts the statement SYS 49152,1000,A$(1), printing an N when it confirms that the first value is numeric and S when it determines that the second value is a string. (Of course, a working program would do something more useful than print N and S.) As long as you separate the expressions with commas, you can replace the number with any numeric expression (such as a numeric variable) and substitute any string expression for A$(1). For instance, SYS 49152,X,"HELLO" is also acceptable.

One advantage of using existing routines is that normal error-handling is preserved. This example detects missing or misplaced parameters as well as type mismatch errors (putting a string value where a number is expected or vice versa). The ROM routine at $AD8A looks for a numeric expression: If it finds a string instead, it automatically prints an error message and returns control to BASIC. $AD9E is BASIC's all-purpose evaluation routine: It accepts any expression and sets the flag in location $0D to show whether it's a string ($0D=$FF) or a number ($0D=0). BASIC's general error-handler ($A437) prints a BASIC error message determined by what value the X register holds when it is called.

Naturally, it's your job to do something useful with the information once it has been passed. The computer's ROM contains a host of other routines that can simplify that task as well. You may find detailed discussions of the ROM routines in the 64 and VIC-20 in Tool Kit: BASIC *and* Tool Kit: Kernal, *both available from COMPUTE! Books. Commented listings for the 64's ROM can be found in* Anatomy of the Commodore 64, *available from Abacus Software.* ©

The only software your home computer really needs.

13 of the most useful software programs on one disk for less than $50.

THE WORKS!™ gives you 13 useful home programs: Letter Writer for word processing, Typing Teacher, Family Finances, Calendar Pad, Address Book, Stock Portfolio, Loans & Investments, Music Composer, Graphics Painter, Calculator, Math Formulas, Weights & Measures and Math Races. Programs that, if bought separately, would cost you hundreds of dollars and hours of time to learn. THE WORKS! easy to read instructions plus "quick reference card" and on-screen help means you can start using the program with your computer *immediately*. No gimmicks, no games.

Easy, inexpensive and complete.

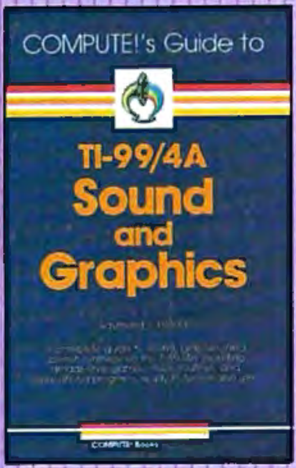
"There's nothing like THE WORKS!"

FIRST STAR SOFTWARE INC

For the Apple II series with 64K and Commodore 64/128. Printer and Second Disk Drive optional.

A special offer from COMPUTE! Books for owners and users of TI-99/4A computers.

Buy any two of these books and get 15% off.
Buy all four of these titles for only $39.95 (a savings of $13.85—over 25% off).



COMPUTE!'s Guide to TI-99/4A Sound and Graphics

Raymond J. Herold

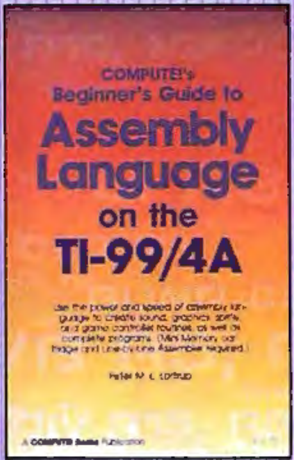
Using dozens of examples and clear, nontechnical explanations, *COMPUTE!'s Guide to TI Sound and Graphics* introduces you to the tremendous sound and graphics capabilities of your TI home computer. This book includes exciting arcade-style games, challenging educational programs, a versatile sprite editor, and many more useful, ready-to-type-in programs. Whether you're a beginning TI user or an experienced programmer, this is a book you'll refer to again and again.

$12.95
ISBN 0-942386-46-9

COMPUTE!'s TI Collection, Volume One

An anthology of *COMPUTE!*'s best games, applications, utilities, and tutorials for the TI-99/4A, this book includes more than 30 programs, most never before published. "SuperFont" is an exceptionally powerful and easy-to-use character editor. Other utilities, such as "Sprite Editor" and "Sound Shaper," make graphics and sound programming simple. Games like "Worm of Bemer" will provide hours of fun, and applications like "Mailing List" let you use your TI to organize your home.

$12.95
ISBN 0-942386-71-X



COMPUTE!'s Beginner's Guide to Assembly Language on the TI-99/4A

Peter M. L. Lottrup

A clearly written, step-by-step tutorial on machine language (ML) programming for the TI-99/4A. Using the Line-by-Line Assembler, the basic concepts of machine language programming are fully explained and illustrated. Many practical, easy-to-follow ML examples are included, from elementary keyboard, joystick, and data handling routines to sophisticated redefined character, sprite manipulation, and even high-resolution bitmap mode techniques.

$14.95
ISBN 0-942386-74-4

33 Programs for the TI-99/4A

Brian Flynn

Contains something for everyone: chapters on games ("Rings and Poles"), money management ("IRA Planner"), business ("Internal Rate of Return"), simple statistics ("Mean, Variance, and Standard Deviation"), and more. A wide variety of applications software, plus games. Thirty-three ready-to-type-in programs at a low cost.

$12.95
199 pages, split wire bound
ISBN 0-942386-42-6

Order today! This offer expires January 15, 1986.

Yes! I want to save money while enjoying COMPUTE! Books.

_____ COMPUTE!'s Guide to TI-99/4A Sound and Graphics $12.95
_____ COMPUTE!'s TI Collection, Volume One $12.95
_____ COMPUTE!'s Beginner's Guide to Assembly Language on the TI-99/4A $14.95
_____ 33 Programs for the TI-99/4A $12.95

All orders must be prepaid.
☐ Payment enclosed (check or money order).
☐ Charge: ☐ Visa ☐ MasterCard ☐ American Express
Acct. No. _____ Exp. Date _____
Signature _____
Name _____
Address _____
City _____ State _____ Zip _____

To order call toll-free, 1-800-334-0868 (in NC, call 919-275-9809) or mail this coupon with your payment to: COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403.

Please send me: ☐ 4 books for $39.95
☐ 2 books at 15% off

NC residents add 4.5% tax _____
Add $2.00 per book for shipping and handling _____
Total paid _____

Please allow 4–6 weeks for delivery.

75D1141

The Digital Diet

Staying In Shape With Your Computer

Selby Bateman, Features Editor

How would you like to have your own diet and exercise coach, someone to encourage you to lose those extra pounds, to help schedule your personal fitness program, and to set up a sensible eating plan? A growing number of software companies are taking advantage of the computer's interactive nature to provide these coaches-on-a-disk.

The winter months and holiday season are at hand, the time of year when many of us tend to add pounds that we've struggled to sweat off during the summer. We're all familiar with the drawbacks of being overweight and out of condition, but starting another diet or an exercise routine often seems to demand too much self-discipline. Calorie counting can be a numeric nightmare. And setting up a training schedule means reading more books and articles before getting started.

What if someone else took care of tallying up the calories, suggested alternate diets, planned a personalized workout routine, and even offered encouragement during occasional slips? You could sit down once a day, answer a few questions, and record your progress. Your coach would handle all the tedious calculations, offering you a rich variety of diet options and training schedules. Wouldn't that be easier?

"I wouldn't want to use the word *easy*. I wouldn't want to tell *anybody* that losing weight is easy. It isn't. That's why people need strong tools," says Dr. Leighton Read, designer of *The Original Boston Computer Diet*, a weight loss and diet counseling program available for the Commodore 64, Apple II series, and IBM PC/PCjr computers.

Read, an instructor of medicine at Harvard Medical School and a practicing internist, is involved with *decision support technologies*, computer-assisted research on medical decision-making. His medical and computer interests convinced him that a good interactive computer-based diet program could be far more helpful than the hundreds of diet books on the market. He was also concerned about the health consequences of the many fad diets promoted today.

"The computer can do the computations and the bookkeeping, and that's a big help," he says. "It can do the individualization which a diet book can't, and that's a *huge* help. But the real power of this new medium, for the developer, is to capture people's imagination to keep them coming back—because losing weight and keeping it off is a long-term issue."

SOME HISTORIC BREAKTHROUGHS DON'T TAKE AS MUCH EXPLAINING AS COMPUSERVE.

But then, some historic breakthroughs could only take you from the cave to the tar pits and back again.

CompuServe, on the other hand, makes a considerably more civilized contribution to your life.

It turns that marvel of the 20th century, the personal computer, into something useful.

Unlike most personal computer products you read about, CompuServe is an information service. It isn't software. It isn't hardware. And you don't even have to know a thing about programming to use it. You subscribe to CompuServe —and 24 hours a day, 7 days a week, it puts a universe of information, entertainment and communications right at your fingertips.

A few of the hundreds of things you can do with CompuServe.

COMMUNICATE

EasyPlex™ Electronic Mail lets even beginners compose, edit, send and file messages the first time they get online. It puts friends, relatives and business associates—anywhere in the country —in constant, convenient touch.

CB Simulator features 72 channels for "talking" with thousands of other enthusiastic subscribers throughout the country and Canada. The chatter is frequently hilarious, the "handles" unforgettable, and the friendships hard and fast.

More than 100 Forums welcome your participation in "discussions" on all sorts of topics. There are Forums for computer owners, gourmet cooks, veterinarians, pilots, golfers, musicians, you name it! Also, Electronic Conferencing lets businesses put heads together without anyone having to leave the shop.

Bulletin Boards let you "post" messages where thousands will see them. You can use our National Bulletin Board or the specialized Bulletin Boards found in just about every Forum.

HAVE FUN

Our full range of games includes "You Guessed It!", the first online TV-style game show you play for real prizes; and MegaWars III, offering the ultimate in interactive excitement. And there are board, parlor, sports and educational games to play alone or against other subscribers throughout the country.

Movie Reviews keep that big night at the movies from being a five star mistake.

SHOP

THE ELECTRONIC MALL™ gives you convenient, 24-hour-a-day, 7-day-a-week shopping for name brand goods and services at discount prices from nationally known stores and businesses.

SAVE ON TRIPS

Travelshopper℠ lets you scan flight availabilities (on virtually any airline — worldwide), find airfare bargains and order tickets right on your computer.

Worldwide Exchange sets you up with the perfect yacht, condo, villa, or whatever it takes to make your next vacation *a vacation*.

A to Z Travel/News Service provides the latest travel news plus complete information on over 20,000 hotels worldwide.

Dr. Leighton Read, developer of The Original Boston Computer Diet.

To accomplish all that, Read developed *The Original Boston Computer Diet* with dietician Cris Carlin, psychiatrist Issac Greenberg, and obesity specialist Dr. George Blackburn. Together they used the latest information on dietetics, behavior modification, and obesity research.

In addition to food reporting, meal planning, and other dieting techniques in *The Original Boston Computer Diet,* there are also the counselors—George, Amy, and Shirley. Each has a different personality. You choose the counselor you want, and the character guides you through each session, offering suggestions, warnings, encouragement, and even disapproval.

George, for instance, reacts in a matter-of-fact, straightforward way. He can even be a bit stern when you don't follow your plan. Shirley is breezy and freewheeling, and Amy is the sweet counselor who'll help you, but never hurt your feelings. Each counselor follows the same medical and dietary guidelines, but they give different kinds of psychological responses.

"You need to have a long-term compact with a weight-control program," says Read. "If it's dry and uninteresting and strictly a calculator, people aren't likely to have enough exposure to it. So you capture their imagination just a little bit, get them engaged, stimulate their curiosity, and give them a sense that there's a reason to come back to the computer."

The 97-page manual includes readings on exercise, junk food, diet drugs, snacking, eating habits, setting goals, and other related topics. A second booklet contains detailed instructions on food reporting and meal planning. A database in the program, containing a large list of foods normally eaten by Americans, has room for you to add up to 300 additional foods as well. Designed for people who want to lose 10 to 40 pounds, the program also tracks and graphs the dieter's progress. Only one person can use the program at a time, but a second set of disks is available for $10.

Based on the success of *The Original Boston Computer Diet,* Read feels that he and other developers will be creating many more computer packages for self-improvement, including programs for exercising, managing stress, and quitting smoking.

The Original Boston Computer Diet is by no means the only program of its kind on the market. There are dozens of other packages available for the more than 80 million Americans who are overweight. They range from nutrition education programs for children to sophisticated trackers for adults on restricted diets.

One of the most popular diet books of the last decade, *The Complete Scarsdale Medical Diet,* has been adapted for the IBM PC and Apple II series. It uses the same Gourmet, International, Vegetarian, Money-Saver, and Basic diets found in the book. A meal-planning calendar helps people schedule their eating patterns. After users choose their menus, the program automatically compiles a shopping list for one or more people. As with the *Boston Diet,* the food directory is expandable. And the *Scarsdale* package also analyzes the caloric and nutritional values of the menus, offers guidance on balancing meals, and shows comparisons among different foods.

A good day on* The Boston Computer Diet *and this screen figure climbs to the top of the hill.

© 1985 LFL.

You've joined an elite Rescue Squadron, flying to the hostile planet Fractalus to confront the ruthless enemy Jaggies head on. The mission is a treacherous one for, as everyone knows, the cyanitric acid atmosphere on Fractalus is fatal and Jaggi saucers are cunning. You're needed to rescue Ethercorps pilots shot down and stranded on that brutal planet, and to help lead our forces to victory . . . for the merciless Jaggi onslaught must be stopped to preserve the future of our galaxy.

Rescue on Fractalus! is a rescue and space action game with realistic 3-D flight simulation. You pilot your Valkyrie Fighter through the canyons and around the mountain peaks of the planet Fractalus to rescue fellow pilots, do battle with enemy saucers and destroy enemy gun emplacements.

We supply the Long Range Scanner, Dirac Mirror Shield and Anti-Matter Bubble Torpedoes . . . YOU supply the skill and guts! Take the challenge: The perils of Fractalus await you.

	C64/128	ATARI	APPLE
Rescue on Fractalus!	✓	✓	✓

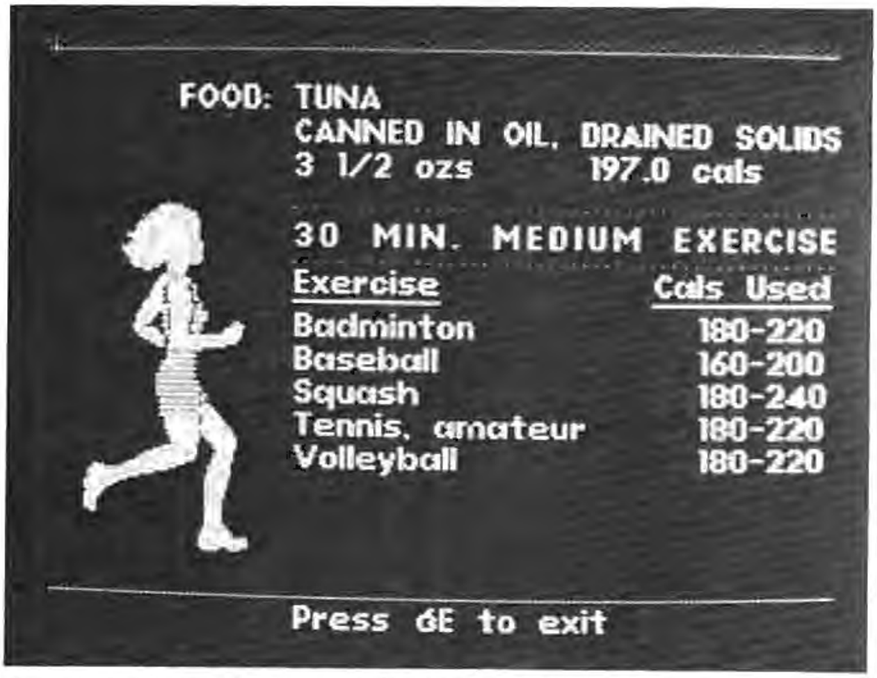
EPYX®
COMPUTER SOFTWARE
1043 Kiel Ct., Sunnyvale, CA 94089

Strategy Games for the Action-Game Player®

© 1985 LFL.

© 1985 LFL.

* See specially marked boxes for details. No purchase necessary. Sweepstakes ends Dec. 31, 1985. Official rules available at participating dealers.



A screen from the* Scarsdale Medical Diet *showing how much exercise it would take to work off a specific amount of food.

"This highly successful diet can be custom-tailored to an individual's lifestyle and fitness goals," says Kenzi Sugihara, director of Bantam Electronic Publishing, which sells *The Complete Scarsdale Medical Diet*. "It's an interactive diet management system that's like having one's own personal diet counselor." Bantam Books also published the paperback version of the book, which is included with the computer program.

Among the attractions of these programs is the custom-tailoring which Sugihara mentions. For example, both programs ask for your current weight, height, frame size, and sex to determine your ideal weight. *The Original Boston Computer Diet* goes even further, asking such questions as how much you eat, what kinds of foods you eat, how fast you eat, and so on, building a profile of your habits. And both programs have help screens or information windows to guide novice users.

A number of programs are available to teach children the fundamentals of nutrition. The approaches are as varied as the number of packages. For instance, *Nutrition Express*—designed for youngsters nine years or older—uses a game format. Available for the Apple II series, the program guides players through the imaginary land of FodaFoda. The Fodars ask questions about food and teach about the basic food groups.

Published by the Center for Science in the Public Interest, *Nutrition Express* teaches children to think about nutrition as they make everyday decisions about food. Consider this sample question: "Beer and soda ads use slim and healthy actors. Do you think the actors got that way by drinking beer or soda pop?"

Another nutrition program for children is *Snackmonster: A Nibbler's Dilemma*, also for the Apple II series, Commodore 64, and IBM PC/PCjr computers. This educational game tempts youngsters with snacks. If they choose the ones with the lowest calorie count, they win.

The Minnesota Educational Computing Consortium (MECC) offers *Food Facts* for the Commodore 64 and Apple II series. *Food Facts* lists the refined sugar content of 64 breakfast cereals, the ingredients in common foods, the time it takes to burn off calories in fast foods, and the percentage of the federal government's recommended daily allowances for eight nutrients in 64 common foods.

Taking yet another approach, Wholebody Health Management publishes *Calorie Calculation—Stress*, a package for adults that determines the number of calories you need and also helps uncover sources of stress in hypothetical situations. Available for the Commodore 64 and Apple II series computers, the two modules—dealing with calories and stress—can be purchased together or separately.

Closely related to computer diet programs are exercise and fitness packages. While still outnumbered by the diet programs, this software genre shows a strong potential for growth as the packages become more sophisticated and interactive.

Avant-Garde's *Be Your Own Coach*, for the Commodore 64, Apple II series, and IBM computers, helps joggers keep their own logs, whether it's just a couple of runs a week or training for a marathon. Developed by Robert Lee Smith, a successful marathon runner, triathlete, and coach, the program produces 14 different types of workouts, tailors each workout to your abilities, and prompts you to record mileage, speed, heart rate, weight, and even your feelings after each run. The software also graphs your progress and can forecast pace and mileage progressions.

MECA's *The Running Program*, subtitled "Your Personal Running Coach," tries a similar angle. It was developed by noted runner and writer James Fixx before his death. *The Running Program* evaluates your fitness level, sets personal

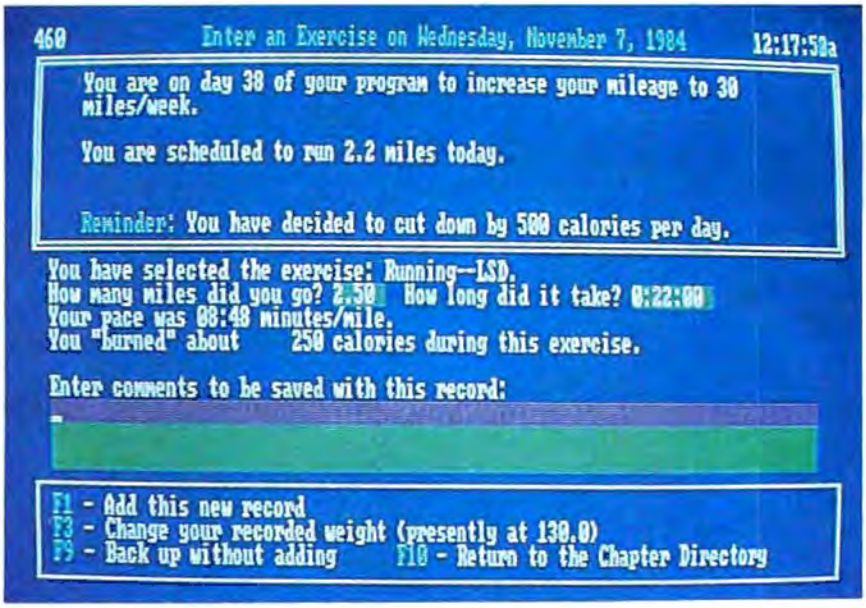
training goals, creates day-by-day training schedules, records and graphs statistics, and predicts your probable performance at different race distances. It is available for IBM computers.

As with the diet programs, the interactivity of the computer gives a more personalized approach to training and recording your workout efforts. Training schedules can be customized and are easily restructured as circumstances change. Should you injure yourself or find that you're overtraining, it's a simple matter to revise the schedules. In fact, both *Be Your Own Coach* and *The Running Program* offer guidance when such problems occur.

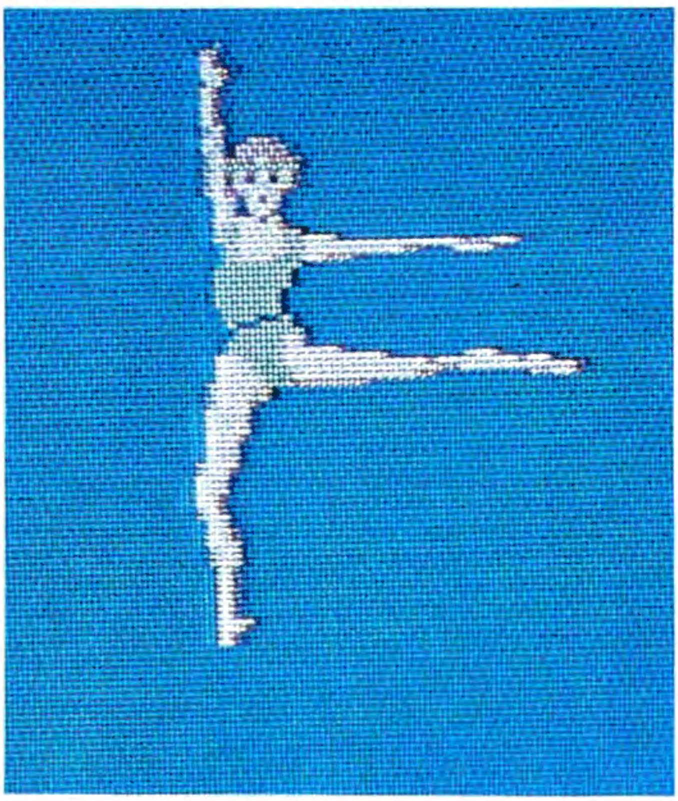
The popularity of aerobics has been captured in a workout program appropriately called *Aerobics*. Published by Spinnaker Software for the Atari and Commodore 64 computers, *Aerobics* is an overall fitness program with a variety of difficulty levels and a choice of musical backgrounds. An onscreen instructor takes you through all the exercises to the accompaniment of music. Different levels and intensities of aerobic exercises are built into the program. You can even choose your own exercises and the order in which you want to do them.

Finally, for the busy executive with an Apple II-series computer, Monument Computer Service publishes *Executive Fitness*. Suggested exercises are shown onscreen, and harried executives can follow along at their own pace.

As computers grow more powerful and software more sophisticated, diet counselors and fitness coaches on disks will become even more helpful, knowledgeable, and interactive. More and more doctors, coaches, and other health professionals are discovering that computers can become amazingly helpful instructors and guides. But no matter what the goal, says Dr. Leighton Read, most of the effort has to come from you. "The critical issue is the motivation, the planning and building it into your life," says Read. And even the best computers and programs can go only so far to help you reach those goals.



A training log from* The Running Program*, showing the day, the distance, the calories expended, and a diet reminder.



***An onscreen instructor guides you through exercises in Spinnaker's* Aerobics.**

TEMPLE OF APSHAI TRILOGY™ BIGGER. MEANER. AND RICHER THAN EVER.

You know Temple of Apshai.

The classic. Best-seller for over *four years.*

You may have friends trapped forever in its dark recesses.

Players have dropped from sight for weeks at a time, searching for the treasures of Apshai.

Well now we've raised the stakes.

Introducing the *new* Apshai Trilogy. The *combined* wrath of the world famous Temple of Apshai®, Upper Reaches of Apshai®, and Curse of Ra®. All on a single disk. Twelve levels. 568 rooms to explore. More choices. More chances. Best of all, there's faster game play.

The graphics and sounds are new. The challenge of the dungeons is timeless.

Are you ready for the most involving role-playing game ever designed?

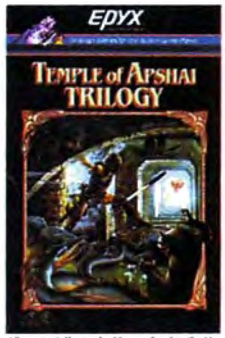
Temple of Apshai is waiting. Silently lurking. Patiently waiting. *For you.* At your nearest Epyx dealer.

	APPLE II	MAC	ATARI	IBM PC	C64/128
Temple of Apshai Trilogy	✓	✓	✓	✓	✓

EPYX™ COMPUTER SOFTWARE

1043 Kiel Court, Sunnyvale, CA 94089

Strategy Games for the Action-Game Player



*See specially marked boxes for details. No purchase necessary. Sweepstakes ends December 31, 1985. Official rules available at participating retailers. T-shirts available while supplies last.

Products and companies mentioned in this article:

Aerobics
Spinnaker Software Co.
1 Kendall Square
Cambridge, MA 02139
Atari 400/800, XL, XE
Commodore 64
$34.95

Be Your Own Coach
Avant-Garde
37-B Commercial Blvd.
Novato, CA 94947
Commodore 64, $39.95
IBM PC/PCjr (128K RAM), $49.95
Apple II series (64K RAM), $49.95

Calorie Calculation—Stress
Wholebody Health Management
18653 Ventura Blvd.
Suite 137
Tarzana, CA 91356
Apple II series (48K RAM)
Commodore 64
$19.95 for Stress disk
$16 for Calorie Calculation disk
$25 for both

The Complete Scarsdale Medical Diet
Bantam Electronic Publishing
Bantam Books, Inc.
666 Fifth Avenue
New York, NY 10103
Apple II series (48K RAM)
IBM PC/PCjr (128K RAM)
$39.95

Executive Fitness
Monument Computer Service
Village Data Center
P.O. Box 603
Joshua Tree, CA 92252
Apple II series (48K RAM)
$19.95

Food Facts
MECC
3490 Lexington Avenue N.
St. Paul, MN 55112
Apple II series (48K RAM)
Commodore 64
$45

Nutrition Express
Center for Science in the Public Interest
1501 16th St. N.W.
Washington, D.C. 20036
Apple II series (48K RAM)
$39.95

The Original Boston Computer Diet
Scarborough Systems, Inc.
55 S. Broadway
Tarrytown, NY 10591
Commodore 64, $49.95
Apple II series (64K RAM), $79.95
IBM PC/PCjr (128K RAM), $79.95

The Running Program
MECA
285 Riverside Avenue
Westport, CT 06880
IBM PC (128K RAM), PCjr (256K RAM), $79.95

Snackmonster: A Nibbler's Dilemma
The Learning Seed Co.
21250 N. Andover Road
Kildeer, IL 60047
Apple II series (48K RAM)
IBM PC/PCjr (64K RAM)
Commodore 64
$49

HOTWARE: Software Best Sellers

This Month	Last Month	Title	Publisher	Remarks	Apple	Atari	Commodore	IBM	Macintosh
Entertainment									
1.	3.	*Gato*	Spectrum Holobyte	Submarine simulation	•			•	
2.	1.	*F-15 Strike Eagle*	MicroProse	Air combat simulation	•	•	•	•	
3.	4.	*Karateka*	Brøderbund	Action karate game	•		•		
4.	2.	*Flight Simulator II*	SubLogic	Aircraft simulation	•	•	•		
5.	5.	*Flight Simulator*	Microsoft	Aircraft simulation				•	
Education									
1.	2.	*Math Blaster!*	Davidson	Introductory math program, ages 6-12	•	•	•	•	
2.	1.	*Typing Tutor III*	Simon & Schuster	Typing instruction program	•		•	•	•
3.	3.	*New Improved MasterType*	Scarborough	Typing instruction program	•	•	•	•	•
4.	5.	*Music Construction Set*	Electronic Arts	Music composition program	•	•	•		
5.		*Sky Travel*	Commodore	Astronomy learning program			•		
Home Management									
1.	1.	*Print Shop*	Brøderbund	Do-it-yourself print shop	•	•	•		
2.	2.	*The Newsroom*	Springboard	Do-it-yourself newspaper	•		•	•	
3.		*Print Shop Graphics Library II*	Brøderbund	Upgraded graphics library	•	•	•		
4.	3.	*Print Shop Graphics Library*	Brøderbund	100 additional graphics	•	•	•		
5.	4.	*Print Master*	Unison World	At-home print shop				•	

The Japanese have a word for it...

(EEEE-YAHHH!!!)

You are the star of a Martial Arts movie so real, you'll feel it like a kick in the ribs.

KARATEKA, you have learned well the disciplines of karate...but now it is time to put your skills to the test. Your village has been ransacked, your friends and family scattered to the winds, your bride-to-be, Princess Mariko, kidnapped and cruelly imprisoned by the evil warlord Akuma. If you ever hope to see her again, Karateka, you know what you must do.

Scale the mighty cliffs that lead to Akuma's fortress. There, you will encounter the first of many palace guards. Kick! Thrust!

Parry! At every turn you will face yet another warrior, each stronger than the last.

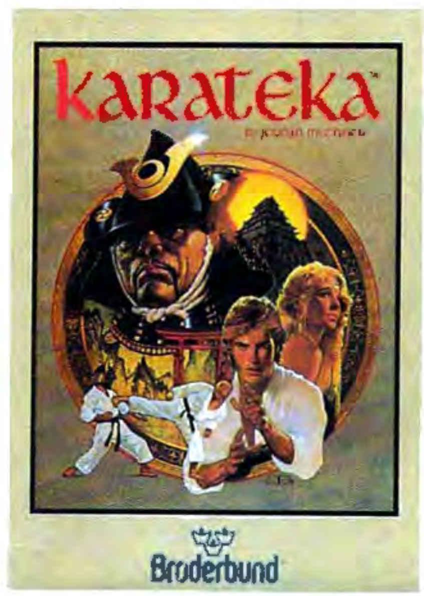
Finally, Karateka, you will come face-to-face with Akuma himself. Here your fate will be decided. Either eternal happiness or instant death.

THE MAKING OF KARATEKA.

"Karateka" designer Jordan Mechner is a karate enthusiast and a stickler for realism. He used film clips of karate masters as a guide for the moves used in the game.

The carefully detailed, animated figures perform all the moves of real martial arts combat with stunning realism.

Beautiful scrolling, hi-res backgrounds, an intricate story line and fast-paced karate action make "Karateka" a great way to get your kicks.



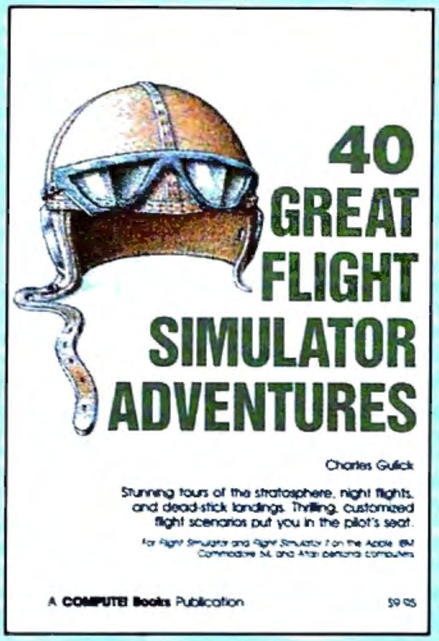
Brøderbund®

KARATEKA is available for Apple, Commodore 64 and Atari personal computers. Look for it at your favorite Brøderbund Software dealer. For more information about Brøderbund products, please write us at 17 Paul Drive, San Rafael, California 94903-2101. Apple, Commodore and Atari are registered trademarks of Apple Computer, Inc., Commodore Electronics, Ltd. and Atari Corporation respectively.

Discover the thrill of *flying!*

with COMPUTE! Books'
40 Great Flight Simulator Adventures

- Maneuver around the towers of the World Trade Center.
- Sightsee the Hudson River.
- Practice night flying and aerobatics.



40 Great Flight Simulator Adventures
Charles Gulick
Forty exciting, customized flight simulator scenarios put you in the pilot's seat as you fly over bridges, around skyscrapers, and land at mysterious airports. *Flight Simulator* (IBM PC) and *Flight Simulator II* (Apple II, Commodore 64, Atari) are two of the most popular games/simulations for personal computers. With this book, you can experience flight adventures from the moment you load the program. Parameters set up each flight and a running commentary describes what you'll see (and where to look to see it). Ranging from the simple and straightforward to the advanced and even mystical, these 40 flights will open a new dimension to an already outstanding program.
$9.95 ISBN 0-87455-022-X

Please send me ________ copies of *40 Great Flight Simulator Adventures* at $9.95 each.
(ISBN No. 0-87455-022-X)

All orders must be prepaid in U.S. funds.

Subtotal ________
NC residents add 4.5% tax ________
$2.00 shipping and handling charge per book. ________
Total amount enclosed ________

To order this exciting adventure guide, mail the attached coupon with your payment to COMPUTE! Books, P.O. Box 5058, Greensboro, NC 27403. Or call toll free 800-334-0868 (In NC 919-275-9809.)

☐ Payment enclosed (check or money order)
☐ Charge ☐ Visa ☐ MasterCard ☐ American Express

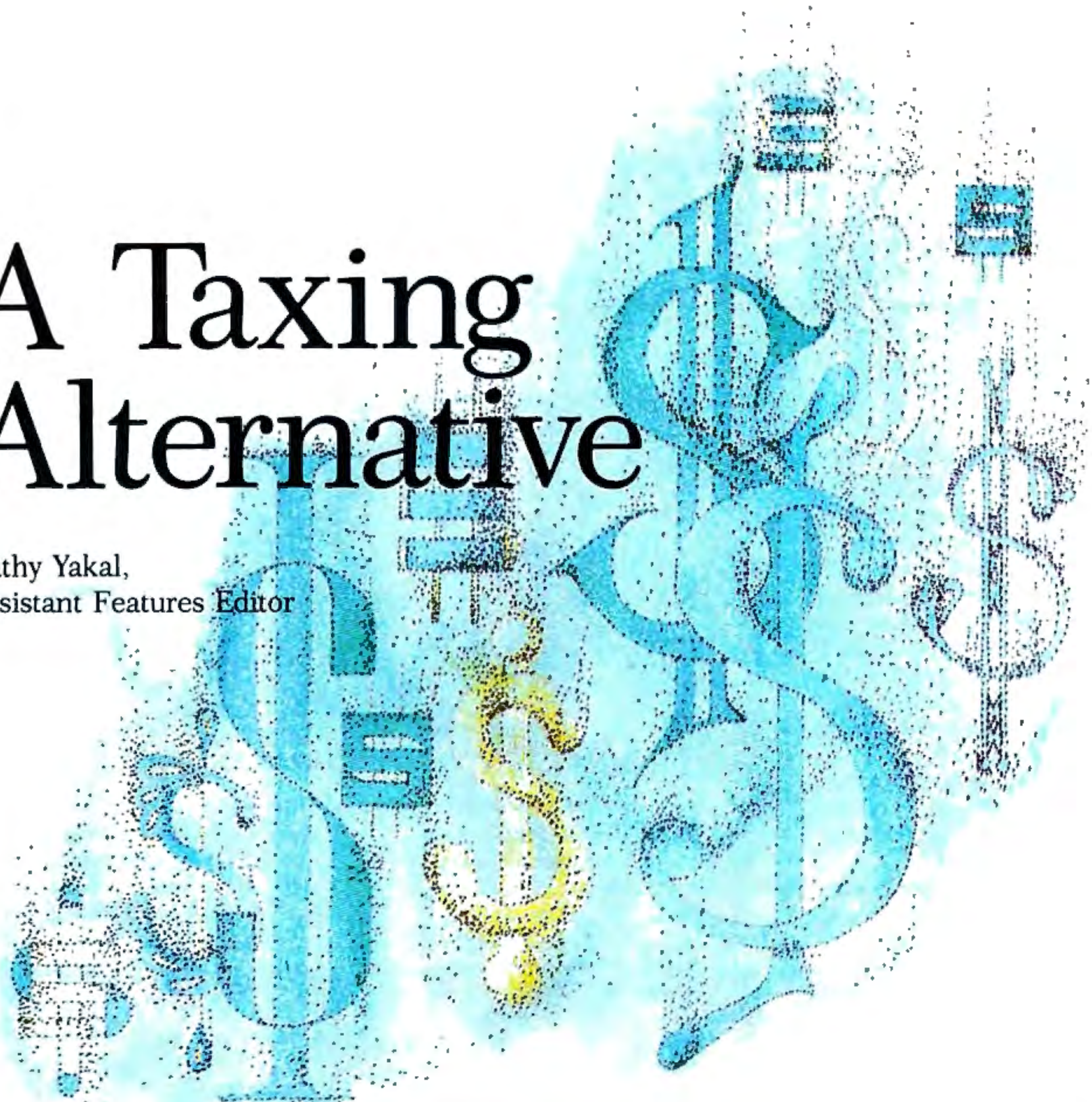
Name ____________________
Address ____________________
City ____________________ State ________ Zip ________

Please allow 4–6 weeks for delivery. 75D0221

COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies

COMPUTE! books are available in the U.K., Europe, the Middle East, and Africa from Holt Saunders, Ltd., 1 St. Anne's Road, Eastbourne, East Sussex BN21 3UN, England.

A Taxing Alternative



Kathy Yakal,
Assistant Features Editor

Americans probably dread no day more than April 15: income tax deadline. For many people, the annual tax filing ritual is a frustrating exercise in organizing hundreds of scattered financial records. Even if you're due a refund, it still means hours of poring over numbers to prove it. However, a growing number of software publishers are offering help via tax-preparation and tax-planning software.

Filing your income taxes seems so easy when you only have a part-time job in high school. You get one form from your employer, write a few figures on the short form, mail it in, and get back a check.

When you first start working full time, it's still pretty simple. The short form suffices for a few years, since you haven't yet reached a tax bracket that claims very much of your income. But your salary keeps inching upward, and you start to see the percentages shift. Less money for you, more for the government. Unless you buy a house. Or have children. Or invest in tax shelters like Individual Retirement Accounts (IRAs).

Whatever your situation, you've probably found tax preparation more troublesome each year. But there are options. You can buy books prepared by experts to help you legally find as many breaks as possible and streamline the process of filling out the forms. Or you can take all of your shoe boxes and file folders into a tax service and pay them a fee to sort through it all.

Home computer software offers another alternative. Help with tax preparation and planning is available in three different formats. Some personal finance programs contain a section for entering tax information that can be tallied at the end of the year. Some have companion tax programs that are integrated with the main package. And a few stand-alone products are devoted solely to income tax preparation.

There's no way to completely avoid some fundamental understanding of tax laws. Even if you go to a tax service, you still have to know what information your preparer will need. And all through the year, you'll need to keep track of pertinent paperwork.

A computer, however, offers some significant advantages over traditional tax preparation methods. For example, with a computer you can easily ask *what if?* Let's say that when you sat down to do your taxes last March, you found that you owed $1,000 on top of what you had been paying in all year.

NOW YOU CAN PUT YOUR COMPUTER TO WORK IN YOUR GARDEN

THE PERFECT PACKAGE FOR COMPUTER OWNERS WHO ARE GARDENERS...AND FOR GARDENERS WHO ARE COMPUTER OWNERS!

Introducing Computerized Gardening from ORTHO — the first and only personalized gardening program available on computer software. It works by detailing plant and shrubs by user zip codes to provide an individualized gardening and landscaping program. Planting requirements are listed according to shade or sun-loving plants, blooming or non-blooming plants, watering needs and plant color and others.

You'll find the software available for IBM, Apple II and Commodore Computers.

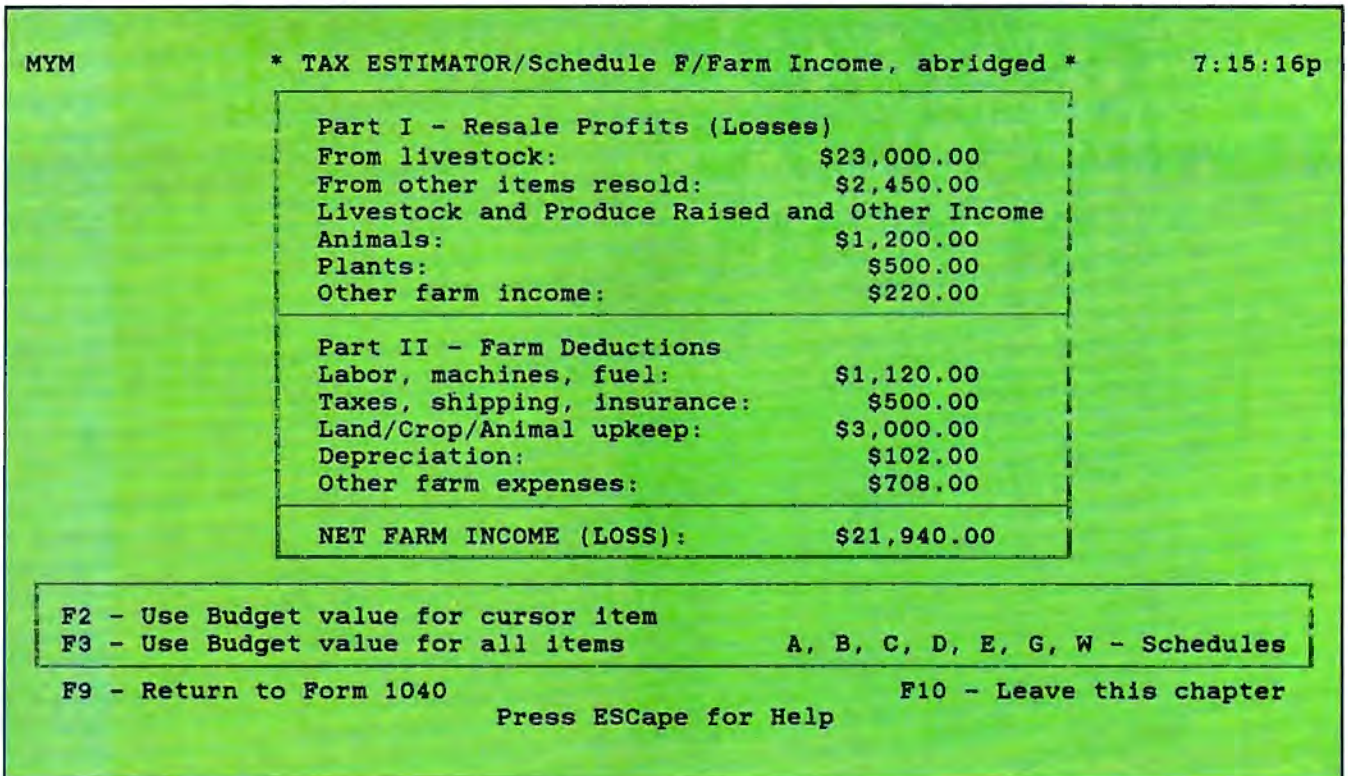
Plus, included in the package is a copy of the new ORTHO Book *Gardening Techniques* — an authoritative "how-to" that shows you in color and detail page after page of gardening procedure.

The ORTHO Computerized Gardening Package is available at software outlets, bookstores and lawn and garden centers where ORTHO books are sold.

Just $49^{95}

For more information write:
ORTHO Information Services
575 Market Street, San Francisco, CA 94105

ORTHO INFORMATION SERVICES



Managing Your Money, *designed by financial expert Andrew Tobias, is a very easy-to-use personal finance package. If you enter pertinent financial information regularly, it can help determine the most advantageous way to file your income tax return.*

Tax programs can help you make sure the additional payment is necessary by suggesting alternatives, and by re-figuring the taxes as many ways as are legally possible.

For example, what if you income-average? If your salary took a recent dramatic upswing in the last year but you have no more deductions to claim than you did a year ago, you're likely to be sending a sizeable check in with your return. *Plus* you'll be charged a penalty for underpaying during the year. But if you were to income-average, you could end up getting money back.

How about tax-deductible expenses? Did you move in the last year; have unusually large medical bills; or buy a new car and a new house and pay a lot of interest? Tax software alerts you to any possible deductions.

These programs can also remind you of income that must be claimed for tax purposes, such as capital gains, freelance work, or income tax refunds from the previous year. Again, if you're familiar with tax laws, you'll know these things already. If you're not, you'll learn. And you will be better equipped for the next tax year.

Income tax software cannot prevent the annual tax preparation marathon if you haven't kept good records throughout the year. Many personal finance software packages offer ongoing accounting, making it much simpler to gather the necessary information when it's tax time. *Managing Your Money*, designed by financial expert Andrew Tobias for the Micro Education Corporation of America, has that capability. You can either estimate your annual income and deductions, or pull up actual figures from elsewhere in the program. The program suggests different tax strategies to try—like filing singly instead of jointly—and can figure and refigure your taxes in seconds. It also prints out much of the documentation you'll need to submit to the IRS.

Your Personal Net Worth, from Scarborough Systems, and Timeworks' *Your Personal Financial Planner* (designed in consultation with Sylvia Porter) have similar features. If used regularly to keep track of income and expenses, a few keystrokes will sort out necessary tax information.

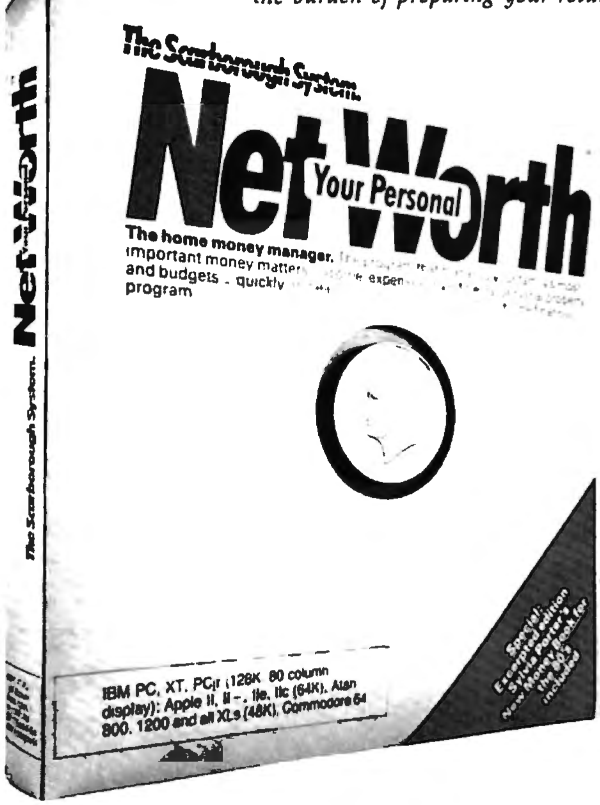
Also, some publishers offer companion programs to their personal finance software, specific tax programs that accept information from the main program to tally taxes. Arrays, Inc./Continental Software publishes *The Home Accountant* for general financial tracking, along with *The Tax Advantage*, for computing tax liability.

Swiftax, from Timeworks, is a stand-alone income tax preparation package. It guides you through the process and tells you which forms must be completed. It automatically checks for tax alternatives, and calculates the lowest possible tax. It supports the most commonly used schedules—A, B, C, D, W, SE, and Form 2441—and plugs this information into your Form 1040, 1040A, or 1040EZ. And when you're finished, the program can print out directly onto tax forms, as well as printing additional necessary information such as amortization schedules. Timeworks issues an annual updated program disk for *Swiftax*, incorporating revised tax law changes and tables.

How you choose to prepare your tax returns probably depends on a number of things. If you have only one source of income, no dependents, and own

PC/TaxCut, *from Best Programs of Alexandria, Virginia, can be used by individuals, professionals, or small businesses and is capable of handling very sophisticated tax forms. First introduced in 1982, the program has been updated annually.* PC TaxCut *can be used as a stand-alone tax planning and preparation package or with Best's personal financial management system,* PC/Professional Finance Program.

Scarborough Systems' Your Personal Net Worth *does not include a fixed set of categories for financial accounts. Instead, it lets you create accounts for your own individual expenses. Each time you make a financial transaction, the program asks if you want a tax record. If you do, it records the relevant parts of the transaction, flagging deductible expenses and interest on loans.* Your Personal Net Worth *also keeps track of all sales tax paid on nondeductible items. When tax time rolls around, the program gives you totals for each area to ease the burden of preparing your return.*



no property or other investments, your tax calculations will not be that difficult. Spending $100 for software that you will use once may not be money spent wisely.

But if you're at the point where expert help is necessary, here are some things to keep in mind when looking for tax software:

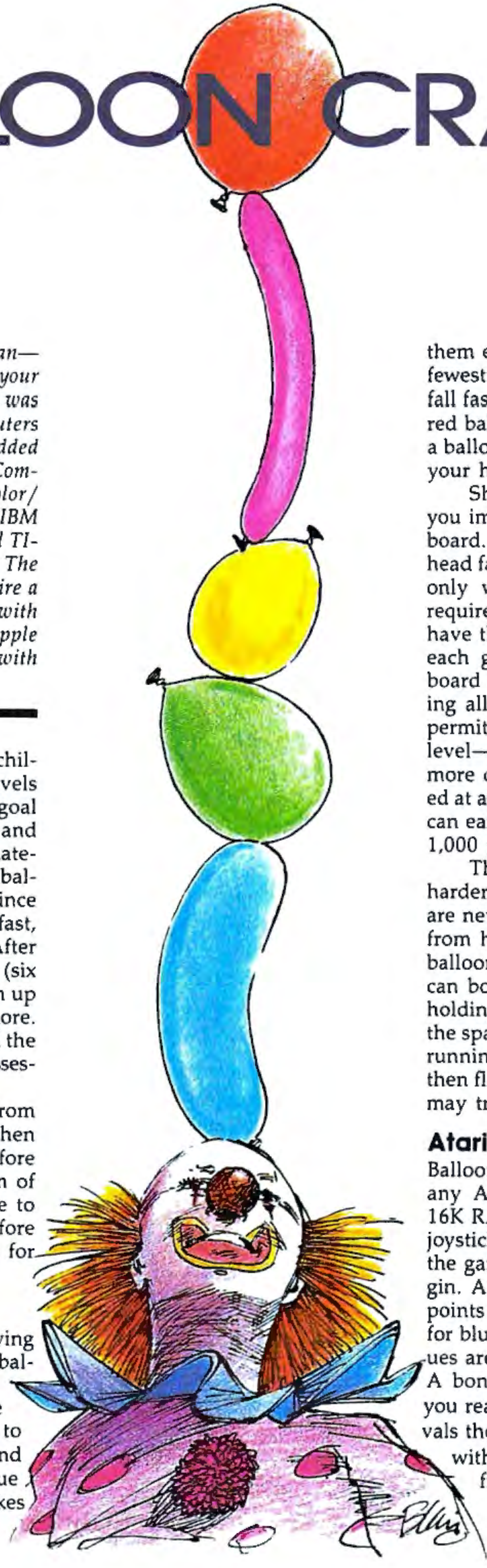
- **Does it contain all the forms you'll need to file?** Most programs have all of the standard, and some nonstandard, schedules. But if the program is missing a crucial one, it may be worthless to you.
- **Can it print directly onto legal tax forms?** Some do, but many just print final figures onto regular paper, requiring you to go back and transfer all the numbers to a form. This may not be terribly important to you, but it's a nice timesaver, and it prevents transcription errors.
- **Can you be reasonably sure that the publisher will supply updates over the next few years?** If using software for your taxes is just an experiment, or a temporary measure to get you through a bad year, this isn't so vital. But if you're looking for a permanent solution, you'll want to ascertain that the software will be kept current as laws change.
- **How extensive are the program's *what-if* capabilities?** This is key to the usefulness of a tax program. A good program will point out options that will, if possible, lower your tax bill.
- **How thorough and simple is the documentation?** As with any complicated software package, clear, complete instructions are essential. A program with bad documentation may actually add hours to the time it would take you to figure your taxes by hand. Good documentation will reinforce its software by alerting you to tax laws and alternatives.
- **Is it worth the expense?** Bottom line. If you don't expect to find enough hidden refund money to recover more than the cost of the package, it may not be worth it to you (unless you just want to computerize your tax records).

Though these programs can be very helpful to consumers in completing an often complicated annual task, they can be quite a headache for the developers and publishers who must insure their accuracy. After all, these software packages are being used to compile reports made to federal and state governments. Current, legally correct information is essential.

But in the end it's your name on the form. It doesn't matter to the Internal Revenue Service *how* a return is prepared, only that it is correct. "We don't take any position for or against the programs," says Ernest Acosta of the IRS. "Like with anything you do with a computer, the information that goes into it is what determines whether it's going to be successful or not. You should remember that you're the one who signs the tax return. Ultimately, it's your responsibility to make sure it's correct.

"On the whole, they're probably a good idea for people who know how to use them. You still need some tax knowledge to go through and know exactly what you're being asked to input. It's certainly not a substitute for following the instructions and keeping good records and all the other things you would have to do to keep up with your taxes." ©

BALLOON CRAZY



Joseph Russ

Catch as many balloons as you can—but be careful not to fall off your skateboard. This whimsical game was originally written for Atari computers with at least 16K RAM. We've added versions for the Apple II series, Commodore 64, IBM PC (with color/graphics adapter and BASICA), IBM PCjr (with Cartridge BASIC), and TI-99/4A (with Extended BASIC). The 64, IBM, and Atari versions require a joystick. A joystick is optional with the TI version. The Atari and Apple versions can also be played with paddles.

"Balloon Crazy" is a game that children can enjoy, yet its higher levels are a challenge for adults. The goal is simple: You must zip back and forth across the screen on a skateboard while catching falling balloons on top of your head. Since some of the balloons fall very fast, that's not as easy as it sounds. After you've caught enough balloons (six in most versions), you can reach up to pop them, then catch some more. If you miss just one, you lose all the balloons currently in your possession.

Type in Balloon Crazy from the listing for your computer, then save a copy of the program before you try to run it. Every version of the game is similar, so be sure to read the general game rules before referring to the specific notes for your computer.

Oodles Of Balloons

Each game begins by displaying several rows of multicolored balloons at the top of the screen. You are the skateboarder at the bottom. When a balloon begins to fall, move directly under it and catch it on your head. The blue balloons fall slowly, which makes them easy to catch (but worth the fewest points). The green balloons fall faster, but swiftest of all are the red balloons. As soon as you snare a balloon, it joins the pile on top of your head.

Should you miss a balloon, you immediately fall off the skateboard. All the balloons on your head fall and pop. Points are scored only when you have caught the required number of balloons. You have three players to work with in each game: Falling off the skateboard costs you one player. Clearing all the balloons from a screen permits you to advance to the next level—where everything becomes more difficult. Bonuses are awarded at appropriate intervals, and you can earn an extra player by scoring 1,000 points.

Though the balloons become harder to catch at higher levels, you are never helpless to prevent them from hitting the ground. Should a balloon miss the top of the pile, you can bounce it back into the air by holding down the fire button (or the space bar in some versions) and running into it. The balloon will then float back into the air, and you may try to catch it again.

Atari Version

Balloon Crazy (Program 1) runs on any Atari computer with at least 16K RAM (or 24K for disk). Plug a joystick into port 1 before you run the game, and press START to begin. At the first level, you score 5 points for each green balloon, 10 for blue, and 15 for red. These values are multiplied at higher levels. A bonus player is awarded when you reach 1,000 points and at intervals thereafter. Move left and right with the joystick, and press the fire button when you want to hit a balloon. You must hit

(YOU'RE WELCOME.)

When we released KAMPFGRUPPE™ in February 1985, we knew we had created one of the best tactical tank battle games ever. How could we do any less with the care, the detail, the features we put into this computer simulation of Eastern Front armored warfare.

As much success as we had predicted for KAMPFGRUPPE, we were not prepared for the overwhelming enthusiasm from the thousands of gamers that played it. Our customer response cards for KAMPFGRUPPE are invariably filled with words of praise and thanks. On a scale of 9, it has consistently been ranked in the 8's for playability, realism, and excitement.

Even the experts agree. The April/May 1985 issue of **Computer Gaming World** called KAMPFGRUPPE "a truly superior game." And in the Aug./Sept. issue, CGW's reader survey placed it as the highest rated wargame ever in the history of the magazine!

Not one to rest on our laurels, we've just released MECH BRIGADE™, the definitive simulation of Soviet-NATO armored combat. It lets you engage in tank warfare of the 1990's, using the latest, laser-guided, missile-armed juggernauts such as the Abrams M-1 and the T-80. Attack helicopters are also included.

From all preliminary indications, the same greatness and gratitude bestowed on KAMPFGRUPPE is destined for MECH BRIGADE.

So what are you waiting for? Rush on down to the nearest computer/software or game store and check out these two great tactical games today!

You can thank us later.

If there are no convenient stores near you, VISA & M/C holders can order these $59.95 games direct by calling **800-227-1617, x335 (toll free)**. In California, 800-772-3545, x335.

Please specify computer format and add $2.00 for shipping and handling.

KAMPFGRUPPE and MECH BRIGADE rulebooks with charts and diagrams to delight the wargamer.

To order by mail, send your check to: Strategic Simulations, Inc., 883 Stierlin Road, A-200, Mountain View, CA 94043. (California residents, add 7% sales tax.) All our games carry a "14-day satisfaction or your money back" guarantee.

WRITE FOR A FREE COLOR CATALOG OF ALL OUR GAMES TODAY.

KAMPFGRUPPE is on diskette for 48K APPLE II® series, ATARI® home computers, and the COMMODORE 64™. MECH BRIGADE is on diskette for APPLE and C-64.

APPLE, ATARI, and COMMODORE 64 are trademarks of Apple Computer, Inc., Atari, Inc., and Commodore Electronics, Ltd., respectively.

STRATEGIC SIMULATIONS, INC.

ANNOUNCING THE FAMILY SOFTWARE SPECTACULAR.™

BUY 3, GET 1 FREE.

CHOOSE FROM OVER 50 LEADING TITLES.

What do you call an offer that brings together, for the first time ever, five leading brands of home software?

The Family Software Spectacular!

And it's your opportunity to choose a *free* software program from the best titles on the market today. All you have to do is buy any combination of three titles from any participating brands during this special offer period. And we'll send you another one of your choice – *free!*

It's a one-of-a-kind value your family shouldn't miss.

FISHER-PRICE.® A NAME YOU CAN TRUST.

For years Fisher-Price has been known for helping children develop skills as they play. With Fisher-Price™ Learning Software, your child can develop skills in five key areas: math, language, creativity, basic learning and computer literacy.

Your child will love developing and testing language skills with the help of PETER RABBIT READING, a program that uses voice to teach sound and letter recognition in words.

WINDHAM CLASSICS.™ CLASSIC NOVELS COME TO LIFE.

Pick up Windham Classics, graphic interactive fiction based on some of the greatest adventures of all time.

They're great for kids, yet fun for all ages.

Enjoy "THE WIZARD OF OZ.™" You are Dorothy, and each of your decisions determines your adventure. There's more than one adventure to be had as you meet old friends and new in the land of Oz.

WORK BETTER WITH BETTER WORKING.™

Constantly erasing and retyping? Unsure of your spelling? WORD PROCESSOR WITH SPELLCHECKER is exactly what you need. This full featured word processor includes the 50,000 word AMERICAN HERITAGE DICTIONARY™ spellchecker which detects over 99% of misspelled words.

The full line also includes FILE & REPORT and SPREADSHEET – titles that represent the best values on the market today.

SPINNAKER® THE LEADER IN FAMILY LEARNING SOFTWARE.

From pre-school counting to high-school compositions, Spinnaker offers the high quality educational programs you have come to expect. For example, there's the HOMEWORK HELPER™ series which provides step-by-step methods for improving writing and math skills and helps kids tackle otherwise troublesome school assignments.

TELARIUM.™ INTERACTIVE FICTION AT ITS BEST.

Here's graphic interactive fiction by famous writers of science fiction, fantasy, and mystery like Ray Bradbury, Arthur C. Clarke and Erle Stanley Gardner. Each story unfolds with you in the center of the action – and completely in control. In PERRY MASON: THE CASE OF THE MANDARIN MURDER,™ you are the world famous criminal lawyer. Challenge the evidence. Cross-examine the witness. Only you can prove your client's innocence. And time is running out.

A SOFTWARE OFFER FOR ALL AGES.

Any software offer including something for everyone in the family is certainly fantastic. But combine that with a free product offer and what you have is something spectacular. Look for the specially marked boxes of The Family Software Spectacular at your retailer.

The Family Software Spectacular™

TURN PAGE FOR MORE INFORMATION ON THE FAMILY SOFTWARE SPECTACULAR. →

the balloon with the player's arm. Since the player has a limited reach, be careful not to knock a balloon completely away.

This game requires only horizontal movement, so you might find it easier to use a paddle instead of a joystick. If you have a set of paddle controllers, plug them into port 1 and make the following modification:

```
500 S=STICK(0):PP=255-PAD
    DLE(0):POKE 53248,PP:
    RETURN
1020 IF PTRIG(0)=0 THEN I
     F ABS(BY-178)<3 THEN
     1040
```

Commodore 64 Version

The 64 version of Balloon Crazy (Program 2) is written completely in machine language, and must be typed in with "The New MLX for Commodore 64" machine language entry program found elsewhere in this issue. Be sure to read the instructions for using MLX carefully before you start. You'll need to read the MLX article even if you've used the old version of MLX many times in the past, since this is a completely new version. *No previous version of MLX can be used to enter the data from Program 2.* After you finish entering the game, be sure to save a copy of the game before you play it. Here are the starting and ending addresses required for MLX:

Starting address: C000
Ending address: C81F

Plug a joystick into port 2, then load Balloon Crazy with:

LOAD "*name*",8,1 for disk

or

LOAD "*name*",1,1 for tape

(replace *name* with the filename used when you saved the program). Then type SYS 49152 and press RETURN.

You must collect six balloons on your head in order to score points. Blue, green, and red balloons are worth 10, 20, and 30 points, respectively. Use the joystick to move left and right, and press the fire button when you have missed a balloon and wish to bounce it upward. You must hit the balloon with the upper part of the player's body. Note that the player can wrap around from one side of the screen to the other, but the balloons cannot. As a bonus, you are awarded an extra player after completing level 5.

Apple Version

Apple II Balloon Crazy (Program 3) runs on Apple II-series computers with either DOS 3.3 or ProDOS. The listing must be entered using COMPUTE!'s "Apple MLX" machine language editor program found elsewhere in this issue. Be sure you understand the instructions for using Apple MLX before entering the data for Balloon Crazy. The MLX starting and ending addresses for the game are:

Starting address: 8000
Ending address: 8D97

After you've entered the game and saved a copy, start the game by entering:

BRUN *"name"*

where *name* is the filename you used when you saved Balloon Crazy.

You can play the game with a paddle on any Apple II computer: Move the paddle to control the player, and press the paddle button to bounce the balloon upward. Alternatively, keyboard controls can be used on the Apple IIc and Apple IIe: press the open-Apple key to move left, the closed-Apple key to move right, and the space bar to bounce.

Four balloons must be collected to score points. If you miss a balloon completely, all the balloons on your head drift off into space and disappear. There are nine game levels. Red balloons do not appear until the second level, but each higher level contains more red balloons. You may pause the game by pressing RETURN; resume play by pressing the space bar.

IBM PC/PCjr Version

IBM PC/PCjr Balloon Crazy (Program 4) requires a joystick and BASICA (if you have a PC) or Cartridge BASIC (PCjr). You may want to unlock the horizontal axis of the joystick. Before the game begins, you have an opportunity to adjust the joystick if needed: Press Y when prompted and follow the instructions on the screen. In this version, all balloons are red and are worth the same number of points.

The number of balloons you need to catch depends on how many rows of balloons are left on the screen: Only three are required at first, but this number increases each time you clear an entire row of balloons. When clearing the top row of balloons, you must catch seven balloons to score. There is no way to bounce a missed balloon back into play. After clearing an entire screen of balloons, you may advance to the next screen.

Your final score reflects the number of balloons caught (no bonus is awarded). You may adjust the difficulty of the game by changing the statement DF=10 in line 120. The variable DF controls how close you must be to a balloon to catch it. Changing DF to a higher value makes the game easier, and decreasing it makes the game more difficult.

TI-99/4A Version

Balloon Crazy for the TI (Program 5) requires Extended BASIC and is played with either keyboard controls or a joystick. Press the S key to move left and the D key to move right. You cannot bounce a balloon back up after missing it. When you catch a balloon, it turns the same color as the player and immediately increases your score. At higher levels, the balloons fall faster and are worth more points. The game ends when you have lost all three players.

Program 1: Atari Balloon Crazy

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
JE 10 GOSUB 4500:GOSUB 5000:
      GOSUB 4000:GRAPHICS 17
      :POKE 756,CHS/256:POKE
       77,0:POKE 559,62:REM
      INITIALIZATION
CH 20 GOSUB 3500:GOSUB 3000
KK 30 GOSUB 2500
LP 40 FOR BY=BL TO 220 STEP
      SL:PM$(P1+BY,D+BY)=B$:
      GOSUB 500:GOSUB 1000:S
      OUND 0,BY,10,8:NEXT BY
      :SOUND 0,0,0,0
FP 50 BAL=BAL-1:GOSUB 1500:L
      F=LF-1:HIT=0:IF LF=0 T
      HEN 4100
IO 60 SOUND 0,0,0,0:PM$(P1+B
      Y,D+BY)=N$:HIT=0:POKE
      PC,1:GOSUB 1005:IF BOH
      =6 OR BAL<1 THEN GOSUB
       2000:BB=169:BOH=0
KF 70 IF BAL<1 THEN GOSUB 30
      10
AE 80 GOTO 30
PF 499 REM MOVEMENT
EC 500 S=STICK(0):PP=PP+((S=
       7)-(S=11)+(PP<65)-(PP
       >200))*3:POKE 53248,P
       P:RETURN
```

```
DB 599 REM POPPING SOUND
60 600 FOR S=15 TO 0 STEP -1:SOUND 0,15,0,S:SOUND 1,16,0,S:NEXT S
16 610 SOUND 0,0,0,0:SOUND 1,0,0,0:RETURN
1A 999 REM CHECK FOR COLLISION
HK 1000 PEK=PEEK(53261):IF PEK=0 THEN RETURN
LO 1002 GOTO 1010
NL 1005 FOR I=25 TO 10 STEP -5:SOUND 0,I,4,8:SOUND 1,I+2,2,8:NEXT I:SOUND 0,0,0,0:SOUND 1,0,0,0:RETURN
LM 1009 REM DID BALLOON HIT HEAD?
BM 1010 IF BY-BB<3 THEN PM$(H+BB,P0+BB)=B$:BB=BB-11:BAL=BAL-1:BOH=BOH+1:BAL(BOH)=C(I-1,BR/3):POP :GOTO 60
IL 1019 REM DID ARM HIT BALLOON?
HJ 1020 IF STRIG(0)=0 THEN IF ABS(BY-178)<3 THEN 1040
PE 1030 A=A+((A>PP)-(A<=PP))*3:POKE 53249,A:POKE PC,1:RETURN
NM 1040 P=A:IF HIT THEN P=PP+((P>PP)-(P<=PP))*3
AB 1050 FOR K=BY TO BL STEP -3:GOSUB 500:SOUND 0,K,10,8:PM$(P1+K,F+K)=B$:POKE 53249,P
PE 1060 P=P+(((P>PP)-(P<=PP))*(P>65 AND P<200)*(PEEK(53261)=0)):POKE PC,1
06 1070 NEXT K:SOUND 0,0,0,0:HIT=1:POP :GOTO 40
6D 1499 REM MAN MISSED BALLOON
PN 1500 PM$(P1+BY,P1+BY+100)=N$
BE 1510 PM$(PM,PB)=P$(1,37):PM$(P0+190,PB)=MDR$:POKE 53250,PP-8:FOR I=200 TO 0 STEP -10:SOUND 0,I,2,10:NEXT I
J0 1520 SOUND 0,0,0,0:IF BOH<1 THEN 1550
PH 1530 FOR I=158 TO BB STEP -11:FOR J=I TO 175
ED 1540 PM$(P0+J,P0+J+12)=B$:NEXT J:PM$(P0+J,P0+J+12)=POB$:GOSUB 600:FOR K=1 TO 9:NEXT K:PM$(P0+J,P0+J+12)=N$:NEXT I
OE 1550 FOR I=1 TO 100:NEXT I:POKE 53250,0:COLOR 32:PLOT LF,23:SOUND 0,0,0,0
AL 1560 PM$(PT,PB)=P$:POKE 53248,PP:BOH=0:BB=169:PP=120:RETURN
OP 1999 REM POP BALLOONS AND TALLY POINTS
IG 2000 IF BOH<1 THEN RETURN
OH 2010 FOR I=1 TO BOH:PM$(PM+1,PD+1)=H$
MM 2020 PM$(P0+158,PM)=POB$:GOSUB 600:FOR V=1 TO 15:NEXT V:PM$(P0+158,PM)=N$:BOH=BOH-1:PM$(PM,PB)=P$(100,137)
HF 2030 SC=SC+BAL(I)*5*LVL:POSITION 7,0:? #6;SC:IF SC>BONUS THEN LF=LF+1:BONUS=BONUS+1000*LVL:COLOR 72:PLOT LF,23
```

Player/missile graphics animate the player and balloons in the Atari version of "Balloon Crazy."

```
JC 2040 IF BOH<1 THEN 2060
EF 2050 FOR K=1 TO BOH:FOR J=158-K*11 TO 168-K*11:PM$(P0+J,P0+J+12)=B$:NEXT J:NEXT K:NEXT I
BA 2060 PM$(PT,PM)=N$:RETURN
E5 2499 REM CHOOSE BALLOON TO FALL
JN 2500 BR=3+INT(RND(0)*6)*3:FOR I=5 TO 2 STEP -1:GOSUB 500:LOCATE BR,I,BT:IF BT=32 THEN NEXT I:GOTO 2500
6E 2510 POKE 705,PEEK(707+C(I-1,BR/3)):A=24*(BR/3+2):BL=32+I*8:POKE 53249,A
IB 2520 PM$(P1+BL,D+BL)=B$:COLOR 32:PLOT BR,I
JC 2530 BS=(C(I-1,BR/3)+2)/2:SL=INT((LVL/3+1)/2+BS):RETURN
KP 2999 REM DRAW SCREEN
NL 3000 RESTORE 3000:DIM F(3),C(4,6),BAL(6):FOR I=1 TO 3:READ A:F(I)=A:NEXT I:DATA 0,32,128
NL 3010 PP=120:POKE 53248,PP:BAL=24:LVL=LVL+1:IF LVL>10 THEN LVL=10
JN 3020 POSITION 11,23:? #6;"LEVEL:";LVL:FOR I=1 TO LF:COLOR 72:PLOT I,23:NEXT I
DL 3030 FOR X=3 TO 18 STEP 3:FOR Y=2 TO 5:A=INT(RND(0)*3)+1:COLOR 79+F(A):C(Y-1,X/3)=A:PLOT X,Y:NEXT Y:NEXT X:RETURN
FA 3500 DIM T$(13):T$="BALLOON CRAZY":POKE 708,134:POKE 709,198:POKE 710,86
NF 3510 FOR I=4 TO 16:POSITION 1,I:? #6;T$(I-3,I-3):NEXT I
EE 3520 POSITION 1,0:? #6;"score:";SC
KL 3530 RETURN
EB 3999 REM SET UP VARIABLES AND STARTING DISPLAY
EA 4000 PP=120:BB=169:SC=0:LVL=0:LF=3:PC=53278:H=P0-12:PT=P0+69:PB=P0+206
58 4010 PM=PB-38:D=P1+12:F=D+3:BT=PB-10:PD=PM+23:BONUS=1000
JN 4020 GRAPHICS 18:POKE 708,134:POKE 709,198:POKE 710,84:POKE 756,CHS/256
SC 4030 POSITION 3,4:? #6;"BALL□oN crazy!":POSITION 4,6:? #6;"[illegible]"
AD 4040 DIM S$(10):S$="Oo□Oo□Oo□O":FOR I=0 TO 10 STEP 10:POSITION I,0:? #6;S$:POSITION I,10:? #6;S$:NEXT I
1A 4050 IF PEEK(53279)<>6 THEN 4050
CK 4060 RETURN
FB 4100 POKE 53248,0:POKE 53249,0:GRAPHICS 17:POKE 708,134:POKE 709,198:POKE 710,84
NI 4110 POSITION 6,4:? #6;"game over":POSITION 8,6:? #6;"SCORE:"
JF 4120 POSITION 9,8:? #6;SC:POSITION 5,10:? #6;"[illegible]":POSITION 4,12:? #6;"[illegible]"
MD 4130 IF PEEK(53279)<>6 THEN 4130
HC 4140 CLR :GOTO 10
JA 4499 REM REDEFINE CHARACTERS
KE 4500 GRAPHICS 17:POKE 559,0:DIM C$(2):C$="OH"
AA 4510 CHS=(PEEK(106)-8)*256:CHO=57344
ML 4520 IF PEEK(CHS+9)<>0 THEN RETURN
BA 4530 FOR I=0 TO 511:POKE CHS+I,PEEK(CHO+I):NEXT I
CK 4540 RESTORE 4560:FOR I=1 TO 2:CHP=CHS+(ASC(C$(I))-32)*8:FOR J=0 TO 7:READ A:POKE CHP+J,A:NEXT J:NEXT I
FF 4550 FOR I=32 TO 39:POKE CHS+I,255-PEEK(CHO+I):NEXT I:RETURN
MC 4560 DATA 28,58,125,125,121,62,28,8
CD 4570 DATA 0,16,56,124,56,60,56,0
CA 4999 REM STE UP P/M
KO 5000 DIM PM$(4096),P$(255),N$(12),B$(15),POB$(12),MDL$(16),MDR$(16),H$(22)
DB 5010 FOR I=1 TO 12:N$(I)=CHR$(0):NEXT I
DO 5020 A=ADR(PM$):PMB=INT(A/2048)*2048:IF PMB<A THEN PMB=PMB+2048
MA 5030 S=PMB-A:POKE 54279,PMB/256:POKE 53277,3
EP 5040 PM$=CHR$(0):PM$(4096)=CHR$(0):PM$(2)=PM$:P$=PM$
IP 5050 P0=S+1024:POKE 704,54:RESTORE 5070
PO 5060 FOR I=100 TO 137:READ A:P$(I)=CHR$(A):NEXT I:PM$(P0+69,P0+206)=P$
FC 5070 DATA 0,16,16,56,56,124,124,56,56,60,60,56,56,16,16,56,56,124,124,186,186,185,185,120,120,40,40,40,40,40,40
```

Another Great Simulation from Sid Meier – Author of F-15 Strike Eagle

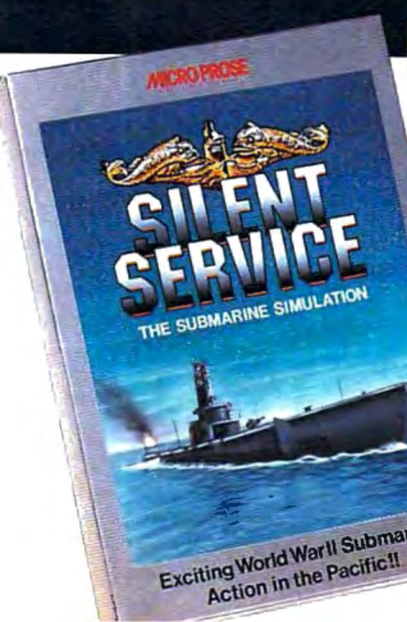
Now he takes you from the cold, thin air and limitless space of F-15 Strike Eagle down into the dark depths of the Pacific Ocean inside an American World War II submarine for a realistic, action-filled simulation —

SILENT SERVICE

Thrill to the initial sighting of the enemy's strike force in your periscope as their ships come into your range. But watch out — the enemy's escorts have just sighted you. You're the **hunter** — but suddenly — you've become the **hunted!**

As Commander, you must sink their ships and keep your submarine from being destroyed — if you can. Will you select a quiet patrol sector in the Marianas Islands or choose the dangerous waters off the coast of Japan? Is a submerged daylight periscope attack best or do you charge in on the surface at night using only radar bearings to guide you? Do you fire a spread of your precious torpedoes or can you close the range and pick off the enemy with a single torpedo shot? These decisions and many more are yours to make as you take your place among the elite ranks of the **SILENT SERVICE!**

It's **exciting** — and it's **fun.** It's another great Micro Prose simulation — and it's called **SILENT SERVICE.** Look for it now on your dealer's shelves.



FIVE AUTHENTIC BATTLE STATION SCREENS

Silent Service is available for Commodore 64/128, Apple II Family, Atari XL/XE, IBM PC/PC Jr. computers for a suggested retail of only $34.95. Available soon for Macintosh for a suggested retail of only $39.95. Call or write for more information or MC/VISA orders.

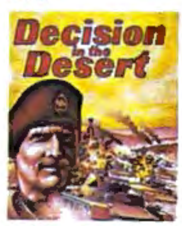
COMMODORE, APPLE, ATARI, IBM, and MACINTOSH are registered trademarks of Commodore Business Inc., Apple Computer Inc., Atari Inc., and International Business Machines Inc., respectively.

Photo courtesy Baltimore Maritime Museum

Try These Other Real Life Simulations

Daring Attack Helicopter Action in the AH-64 Apache!

Thrilling Decathalon of aerobatic events in your own personal jet!!



You are in command — North Africa 1940-1942

MICRO PROSE
SIMULATION • SOFTWARE

120 LAKEFRONT DRIVE • HUNT VALLEY, MD 21030 • (301) 667-1151

```
FH 5080 DATA 40,40,255,255,6
        6,66,0
JL 5090 P1=P0+256:POKE 705,1
        32
LA 5100 FOR I=1 TO 15:READ A
        :B$(I)=CHR$(A):NEXT
        I:FOR I=1 TO 12:READ
         A:POB$(I)=CHR$(A):N
        EXT I
DO 5110 DATA 0,0,0,0,28,58,1
        25,125,121,62,28,8,0
        ,0,0
LM 5120 DATA 0,0,0,0,42,0,12
        9,16,145,0,66,16
6K 5130 P2=P1+256:POKE 706,7
        0
LC 5140 FOR I=1 TO 16:READ A
        :MDL$(I)=CHR$(A):NEX
        T I:PM$(P2+190,P2+20
        6)=MDL$
KL 5150 DATA 0,0,0,0,1,1,0,0
        ,24,24,36,36,67,67,2
        55,255
NK 5160 FOR I=1 TO 16:READ A
        :MDR$(I)=CHR$(A):NEX
        T I:FOR I=1 TO 22:RE
        AD A:H$(I)=CHR$(A):N
        EXT I:RETURN
BB 5170 DATA 0,0,132,132,254
        ,254,32,32,32,32,34,
        34,247,247,255,255
CI 5180 DATA 18,18,57,57,125
        ,125,57,57,61,61,57,
        57,17,17,58,58,124,1
        24,184,184,184,184
```

Program 2: Commodore 64 Balloon Crazy

Version by Kevin Mykytyn, Editorial Programmer

For instructions on entering this listing, please refer to "The New MLX for Commodore 64" elsewhere in this issue.

```
C000:20 56 C0 20 DE C1 20 5A D9
C008:C1 4C D6 C5 20 DE C1 20 D4
C010:5A C1 A5 A5 0A 85 A7 0A FD
C018:85 A6 C6 03 D0 0A A5 04 0D
C020:38 E5 A7 85 03 20 C8 C0 6F
C028:C6 F9 D0 0D A5 FA 38 E5 E5
C030:A6 85 F9 20 7A C4 20 3F 0E
C038:C3 AD 8D 02 D0 FB A0 46 D6
C040:88 D0 FD 4C 12 C0 98 48 CB
C048:A0 32 20 B3 EE 88 D0 FA 1C
C050:CA D0 F5 68 A8 60 A9 1B E5
C058:8D 11 D0 A9 7F 8D 0D DC C2
C060:A9 70 8D 14 03 A9 C0 8D 93
C068:15 03 A9 81 8D 1A D0 60 59
C070:A9 01 8D 19 D0 AD 8D 02 A4
C078:D0 3C A4 F8 F0 14 AD 3C 05
C080:03 18 69 05 99 3D 03 AD 7C
C088:46 03 69 00 99 47 03 88 93
C090:D0 EC A0 07 A2 0E A9 00 DA
C098:85 B7 B9 3C 03 9D 00 D0 25
C0A0:B9 50 03 9D 01 D0 B9 46 52
C0A8:03 4A 26 B7 CA CA 88 10 21
C0B0:E9 A5 B7 8D 10 D0 A9 FA 72
C0B8:8D 12 D0 AD 0D DC 29 01 A9
C0C0:F0 03 4C 1C C3 4C BC FE 8E
C0C8:20 C1 C2 AD 00 DC 4A 4A 50
C0D0:4A B0 2F AE 46 03 D0 13 67
C0D8:AE 3C 03 E0 1E D0 0C A9 25
C0E0:3B 8D 3C 03 A9 01 8D 46 CD
C0E8:03 D0 13 48 AD 3C 03 38 A3
C0F0:E9 01 8D 3C 03 AD 46 03 7B
C0F8:E9 00 8D 46 03 68 A2 FF 84
C100:86 05 4A B0 2F AE 46 03 20
C108:F0 13 AE 3C 03 E0 3C D0 47
C110:0C A9 1F 8D 3C 03 A9 00 02
C118:8D 46 03 F0 13 48 AD 3C B4
C120:03 18 69 01 8D 3C 03 AD 79
C128:46 03 69 00 8D 46 03 68 B0
```

"Balloon Crazy" for the Commodore 64 uses sprites as well as character graphics.

```
C130:A2 01 86 05 4A B0 22 AD 6D
C138:1E D0 29 01 F0 1B AD 03 86
C140:D0 C9 D2 90 14 C9 DC B0 34
C148:10 20 9E C2 A5 C4 49 FF AE
C150:18 69 01 85 C4 A5 05 85 FE
C158:C3 60 A9 93 20 D2 FF 20 B0
C160:D6 C1 A9 03 85 FB A9 05 99
C168:85 FC AD 1B D4 29 03 C9 70
C170:03 F0 F7 A8 B9 7D C7 48 D6
C178:A9 03 38 E5 FB 0A 0A 0A 1D
C180:85 02 A9 05 38 E5 FC 18 38
C188:65 02 A8 68 99 1F C8 8D 43
C190:86 02 A9 71 A0 C7 20 1E A6
C198:AB C6 FC 10 CD 20 D6 C1 A2
C1A0:C6 FB 10 C2 A0 03 A9 3F 58
C1A8:99 64 03 88 10 FA A9 00 BA
C1B0:85 F7 85 F8 18 A2 18 A0 51
C1B8:0D 20 F0 FF A9 EA A0 C7 EA
C1C0:20 1E AB 20 31 C5 A9 96 DD
C1C8:8D 3C 03 A9 00 8D 46 03 E2
C1D0:A2 14 20 46 C0 60 A9 8A 78
C1D8:A0 C7 20 1E AB 60 A2 07 AF
C1E0:A0 00 A9 00 85 FB 85 FD 0F
C1E8:A9 38 85 FC A9 D0 85 FE 6A
C1F0:78 A9 33 85 01 B1 FD 91 36
C1F8:FB B9 31 C6 99 C0 3E B9 81
C200:71 C6 99 00 3F 88 D0 ED CE
C208:E6 FC E6 FE CA D0 E6 A9 1E
C210:37 85 01 58 A0 1F B9 11 3E
C218:C6 99 D8 38 88 10 F7 A9 24
C220:1E 8D 18 D0 A9 D8 8D 16 0A
C228:D0 A0 01 8C 25 D0 88 8C 31
C230:21 D0 88 8C 1C D0 8C 15 A6
C238:D0 A9 01 8D 25 D0 A0 07 3E
C240:8C 26 D0 88 8C 27 D0 C8 A3
C248:A9 00 99 50 03 B9 BD C7 1D
C250:99 F8 07 88 10 F2 A9 DC C6
C258:8D 50 03 A9 0A 85 04 85 A7
C260:03 A9 00 85 A3 85 A4 85 2C
C268:A5 A9 03 85 A8 A0 17 A9 83
C270:00 99 00 D4 88 10 FA A9 CD
C278:0F 8D 18 D4 A9 0C 8D 03 D4
C280:D4 A9 05 8D 05 D4 A9 0F 32
C288:8D 01 D4 A9 80 8D 12 D4 7D
C290:8D 0F D4 A9 0A 8D 0C D4 49
C298:A9 08 8D 0A D4 60 20 B2 62
C2A0:C2 AD A7 02 8D A9 02 A9 C8
C2A8:11 8D 0B D4 A9 00 8D AA DB
C2B0:02 60 A9 10 8D 0B D4 EE B6
C2B8:AA 02 A9 00 8D 08 D4 60 DF
C2C0:60 AD 00 DC 29 0F C9 0F D7
C2C8:D0 16 A9 10 8D 04 D4 A9 42
C2D0:00 8D AB 02 8D AC 02 8D FF
C2D8:01 D4 A9 08 8D AD 02 60 51
C2E0:AD AB 02 D0 19 A9 01 8D 74
C2E8:AB 02 A9 10 8D 04 D4 A9 CA
C2F0:0F 8D 05 D4 A9 00 8D 06 BD
C2F8:D4 A9 11 8D 04 D4 18 AD 9F
C300:AC 02 69 23 8D AC 02 AD BE
C308:AD 02 69 00 8D AD 02 AD E8
C310:AC 02 8D 00 D4 AD AD 02 DA
C318:8D 01 D4 60 AD AA 02 D0 34
C320:12 AD A9 02 F0 10 18 AD 17
C328:A9 02 6D A8 02 8D A9 02 D8
C330:8D 08 D4 4C 31 EA AD A7 18
C338:02 8D A9 02 4C 31 EA A5 1C
C340:F7 F0 03 4C FD C3 20 9E 03
C348:C2 E6 F7 A9 00 85 C3 A9 CB
C350:FF 8D F9 07 A9 01 85 C4 0C
C358:A9 00 A0 03 19 64 03 88 E1
C360:10 FA C9 00 D0 1B A5 F8 1F
C368:F0 03 20 B4 C4 A5 A5 C9 4A
C370:04 D0 04 E6 A8 D0 02 E6 90
C378:A5 20 5A C1 68 68 4C 12 D1
C380:C0 A0 03 AD 1B D4 29 07 51
C388:C9 06 B0 F7 AA BD C5 C7 AB
C390:39 64 03 D0 05 88 10 F5 9B
C398:30 E7 BD C5 C7 59 64 03 B5
C3A0:99 64 03 BD D3 C7 8D 3D 60
C3A8:03 BD D9 C7 8D 47 03 B9 22
C3B0:DF C7 8D 51 03 98 0A 0A 79
C3B8:0A 85 02 8A 18 65 02 84 6E
C3C0:02 A8 B9 1F C8 29 07 8D 23
C3C8:28 D0 A8 0A 0A 8D A7 02 26
C3D0:B9 83 C7 85 FA 8D A8 02 C8
C3D8:20 9E C2 A4 02 8A 85 02 02
C3E0:0A 0A 18 65 02 18 69 07 93
C3E8:85 02 98 0A 18 69 02 AA 7C
C3F0:A4 02 18 20 F0 FF A9 CB F6
C3F8:A0 C7 4C 1E AB A5 C4 30 DB
C400:0E 18 6D 51 03 8D 51 03 4D
C408:C9 E6 90 1A 4C 6B C5 49 C8
C410:FF 18 69 01 85 02 AD 51 BD
C418:03 38 E5 02 C9 82 B0 06 CD
C420:A5 02 85 C4 D0 03 8D 51 F8
C428:03 A5 C3 30 25 18 6D 3D B9
C430:03 AE 47 03 F0 0F C9 3C 93
C438:90 0B A5 C3 49 FF 18 69 A1
C440:01 85 C3 D0 34 8D 3D 03 86
C448:AD 47 03 69 00 8D 47 03 39
C450:10 27 49 FF 18 69 01 85 C2
C458:02 AD 3D 03 38 E5 02 AE 32
C460:47 03 D0 0A C9 1E B0 06 37
C468:A5 02 85 C3 D0 0B 8D 3D 3D
C470:03 AD 47 03 E9 00 8D 47 B1
C478:03 60 AD 1E D0 29 02 F0 53
C480:32 AD 03 D0 A4 F8 F9 E3 DC
C488:C7 C9 03 90 04 C9 FB 90 A1
C490:22 E6 F8 A4 F8 B9 E3 C7 8C
C498:99 51 03 AD 28 D0 29 07 5C
C4A0:99 28 D0 C6 F7 A9 FF 99 87
C4A8:F9 07 A5 F8 C9 06 D0 03 40
C4B0:20 B4 C4 60 A9 00 85 A9 18
C4B8:A0 00 8C 51 03 A6 A9 D0 10
C4C0:25 A9 FC 8D F8 07 B9 29 40
C4C8:D0 29 07 09 08 A2 FF E8 2A
C4D0:DD 7D C7 D0 FA BD 80 C7 46
C4D8:18 65 A3 85 A3 A5 A4 69 FA
C4E0:00 85 A4 20 31 C5 A9 FE 55
C4E8:99 FA 07 20 F4 C5 A2 05 E9
C4F0:20 46 C0 A9 00 99 52 03 DC
C4F8:A6 A9 D0 05 A9 FD 8D F8 04
C500:07 A2 04 20 46 C0 C0 05 F5
C508:F0 17 C8 84 06 A2 01 BD AD
C510:E3 C7 99 52 03 E8 C8 C4 E9
C518:F8 90 F4 A4 06 C4 F8 D0 33
C520:9C 88 C8 A9 00 99 51 03 DB
C528:C0 07 D0 F6 A9 00 85 F8 B0
C530:60 98 48 A9 94 A0 C7 20 8C
C538:1E AB 18 A2 00 A0 0A 20 A1
C540:F0 FF A5 A4 A6 A3 20 CD 15
C548:BD 18 A2 00 A0 17 20 F0 9F
C550:FF A9 00 A6 A5 E8 20 CD 8F
C558:BD 18 A2 00 A0 24 20 F0 E3
C560:FF A9 00 A6 A8 20 CD BD DF
C568:68 A8 60 CE F9 07 20 F4 6C
C570:C5 C6 F7 A2 04 20 46 C0 A7
C578:A9 00 8D 51 03 A9 FB 8D E3
C580:F8 07 A5 F8 F0 07 A9 01 86
C588:85 A9 20 B8 C4 A2 09 20 B3
C590:46 C0 A9 FD 8D F8 07 C6 A9
C598:A8 20 31 C5 A5 A8 D0 4F C3
C5A0:A9 03 8D 15 D0 A9 FE 8D 7D
C5A8:F9 07 20 81 C3 A9 00 A0 74
C5B0:03 19 64 03 88 10 FA 48 83
C5B8:A2 02 20 46 C0 68 D0 EA B2
C5C0:A9 01 8D 15 D0 68 68 18 75
C5C8:A2 0B A0 0F 20 F0 FF A9 DB
C5D0:04 A0 C8 20 1E AB 18 A2 14
```

COMPUTER MAIL ORDER

THE CMO ADVANTAGE

- THE BEST PRICES!
- Next day shipping on all in stock items.
- Free easy access order inquiry.
- Orders from outside Pennsylvania save state sales tax.
- Free technicial support with our factory trained technical staff.
- There is no limit and no deposit on C.O.D. orders.
- There's no extra charge for using your credit card. Your card is not charged until we ship.
- No waiting period for cashiers checks.
- We accept purchase orders from qualified corporations. Subject to approval.
- Educational discounts available to qualified institutions.
- FREE CATALOG MEMBERSHIP

ORDER LINE

CALL TOLL-FREE

1-800-233-8950

CUSTOMER SERVICE & TECH SUPPORT

1-717-327-1450

DEPT. A212

MAILING ADDRESS

Computer Mail Order

DEPT. A212

477 East Third Street

Williamsport, PA 17701

MEMBER DIRECT MARKETING ASSOCIATION

CREDIT CARDS

SHIPPING

Add 3%, minimum $5.00 shipping and handling on all orders. Larger shipments may require additional charges.

All items subject to availability and price change.

Returned shipments may be subject to a restocking fee.

CANADIAN ORDERS

1-800-268-3974
Ontario/Quebec

1-800-268-4559
Other Provinces

1-416-828-0866
In Toronto

TELEX: 06-218960

2505 Dunwin Drive,
Mississauga, Ontario
Canada L5L1T1

All prices shown are for U.S.A. orders.
Call The Canadian Office for Canadian prices.

HOME COMPUTERS

ATARI

130XE (128K)	**CALL**
800XL 64K	CALL
1010 Recorder	$49.99
1050 Disk Drive	$144.00
1025 Dot Matrix Printer	$199.00
1027 Letter Quality Printer	$269.00
1030 Direct Connect Modem	$59.99
520ST (512K)	**CALL**

BOARDS FOR ATARI

Axlon 32K	$39.99
Axlon 48K (400)	$69.99
Axlon 128K	$199.00
Bit 3 Full View 80	$169.00

Software Specials - While They Last!

8036 Atari Writer	$29.99
Star Raiders	$4.99
Missile Command	$4.99
Defender	$4.99
Galaxian	$4.99
Asteroids	$4.99
Centipede	$4.99
Robotron	$4.99
Eastern Front	$4.99

APPLE

APPLE IIe	CALL
APPLE IIc	CALL
MacINTOSH	CALL
IIc LCD Display	CALL

ASSIMULATION

Mac Port Adapter	$59.99

HAYDEN

Art Grabber	$31.99
Home Design	$49.99
Media Works	$63.99

PALADIN

Crunch 512	$189.00

Macintosh Software

Lotus Jazz	CALL
Microsoft Excel	CALL
Microsoft Business Pak	$375.00
Living Videotext ThinkTank 512	$159.00
Manhatten Ready, Set, Go	$79.99
Creighton Development Mac Spell	$69.99
Monogram Dollars & Sense	$99.99
Peachtree Back to Basics - GL	$109.00
PFS File & Report (New Version)	$129.00
Silicon Beach Airborn	$25.99

commodore

C128 Computer	**$299.00**
C1571 (Disk Drive for C128)	**$NEW**
C1902 (RGB 13" Monitor for C128)	**$NEW**
C1670 (Modem for C128)	**$NEW**
Commodore Plus 4	$199.00
CBM 64	$149.00
C1541 Disk Drive	$199.00
C1530 Datasette	$39.99
M-801 Dot Matrix Printer	$169.00
MCS 803 Dot Matrix	$179.00
C1702 Color Monitor	$189.00
C1660 Auto Modem	$59.99
DPS 1101 Daisy Printer	$339.00

Professional Software

Fleet System II w/Spell	$49.99
Trivia Fever	$29.99
Word Pro 4 Plus/5 Plus each	$239.00
Info Pro	$179.00

BRODERBUND

The Print Shop	$29.99

pfs

File (64)	$59.99
Report (64)	$49.99

BATTERIES INCLUDED

PaperClip w/Spell Pack	$79.99
The Consultant DBMS	$69.99
Bus Card II	$139.00
80 Col Display	$129.00

MODEMS

ANCHOR

Volksmodem	$59.99
Volksmodem 300/1200	$189.99
Signalman Express	$299.00
Lightning 2400 Baud	$399.00

DIGITAL DEVICES

AT300 - 300 Baud (Atari)	$99.99

Hayes

Smartmodem 300	$139.00
Smartmodem 1200	$389.00
Smartmodem 1200B	$359.00
Smartmodem 2400	$629.00
Micromodem IIe	$149.00
Smart Com II	$89.99
Chronograph	$199.00
Transet 1000	$309.00

AST

Reach 1200 Baud Half Card	$399.00

MPP MICROBITS

MPP-1064 AD/AA (C-64)	$69.99

Novation

Smart Cat Plus	$319.00
J-Cat	$99.99
Novation 2400	CALL
Apple Cat II	$229.00
212 Apple Cat II	$379.00
Apple Cat 212 Upgrade	$229.00
Macmodem	$319.00

QUADRAM

Quadmodem II	
300/1200	$339.00
300/1200/2400	$499.00

TELELEARNING

C64 300 Baud .... (Closeout)	$39.99

GRAPHICS

Koala

IBM	$89.99
Apple/Franklin	$79.99

Polaroid

Palette	$1299.00

PORTABLE COMPUTERS

HEWLETT PACKARD

41CV	$189.99
41CX	$249.99
HP 71B	$419.99
HP 11C	$62.99
HP 12C/15C/16C	$89.99
HP 75D	$949.00
HPIL Module	$98.99
HPIL Cassette or Printer	$359.99
Card Reader	$143.99
Extended Function Module	$63.99
Time Module	$63.99

We stock the full line of HP calculator products

NEC

PC-8401	CALL
PC-8201 Portable Computer	$289.00
PC-8231 Disk Drive	$599.00
PC-8221A Thermal Printers	$149.00
PC-8281A Data Recorder	$99.99
PC-8201-06 8K RAM Chips	$105.00

SHARP

PC-1350	$159.99
PC-1261	$159.99
PC-1260	$109.99
PC-1500A	$165.99
PC-1250A	$88.99
CE-125 Printer/Cassette	$128.99
CE-150 Color Printer Cassette	$171.99
CE-161 16K RAM	$134.99

DISKETTES

maxell

3½" SS/DD	$39.99
3½" DS/DD	$54.99
5¼" MD-1 w/Hardcase	$14.99
5¼" MD-2 w/Hardcase	$19.99
5¼" MD-2-HD for AT	$44.99

Verbatim.

5¼" SS/DD	$19.99
5¼" DS/DD	$24.99
Disk Analyzer	$24.99

Dennison

Elephant 3½" SS/DD	$29.99
Elephant 5¼" SS/SD	$13.99
Elephant 5¼" SS/DD	$15.99
Elephant 5¼" DS/DD	$16.99
Elephant Premium DS/DD	$22.99

IBM

5¼" DS/DD Floppy Disks (Box of 10)	$25.99

DISK HOLDERS

INNOVATIVE CONCEPTS

Flip-in-File 10	$3.99
Flip-in-File 50	$17.99
Flip-in-File 50 w/lock	$24.99
Flip-in-File (400/800 ROM)	$11.99
Flip-in-File 100	$24.99

AMARAY

50 Disk Tub 5¼"	$9.99
30 Disk Tub 3½"	$8.99

DRIVES

HARD

ALLOY

PC Stor	CALL

μ-SCI

10 meg Internal (IBM)	$429.00
20 meg Internal (IBM)	$569.00

I-MEGA

10 meg Bernoulli Box	$1999.00
20 meg Bernoulli Box	$2599.00
5 meg "MacNoulli"	$1499.00

TALLGRASS TECHNOLOGIES

12, 25, 35, 50, 80 meg (PC)	from $1499.00

FLOPPY

INDUS

Atari GT	$219.00
C-64 GT	$229.00

μ-SCI

A1.5 Apple	$179.00
A2 Apple	$179.00

MSD

SD1 C-64 Single	$219.00
SD2 C-64 Dual	$469.00

Tandon

320K 5¼" (PC)	$109.00

TEAC

320K 5¼"	$109.00

CALL TOLL-FREE

```
C5D8:ØD AØ ØD 2Ø FØ FF A9 ØF AØ
C5EØ:AØ C8 2Ø 1E AB AD ØØ DC C5
C5E8:29 1Ø DØ F9 4C ØC CØ A9 84
C5FØ:ØØ 85 F8 6Ø 2Ø B2 C2 A9 FD
C5F8:8Ø 8D Ø4 D4 A9 Ø5 8D Ø5 77
C6ØØ:D4 A9 ØØ 8D Ø6 D4 A9 ØF 21
C6Ø8:8D Ø1 D4 A9 81 8D Ø4 D4 FØ
C61Ø:6Ø ØØ ØØ ØØ ØF 3F FF FF 43
C618:FF ØØ ØØ ØØ CØ FØ FC DC 46
C62Ø:DC FF FF 3F 3F ØF Ø3 ØØ 4C
C628:ØØ DC 7C FØ FØ CØ ØØ ØØ 16
C63Ø:ØØ ØC ØØ 3Ø FF FF FF Ø1 C4
C638:45 ØØ Ø1 Ø4 ØØ Ø2 Ø8 ØØ EØ
C64Ø:Ø2 Ø8 ØØ Ø2 2Ø ØØ Ø2 8Ø 76
C648:ØØ Ø2 AA 85 Ø2 55 95 Ø2 96
C65Ø:AA 85 ØØ ØØ ØØ ØØ ØØ ØØ 94
C658:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ E5
C66Ø:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ED
C668:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ F5
C67Ø:FE Ø1 14 ØØ Ø4 55 ØØ 1Ø C5
C678:55 ØØ Ø4 14 ØØ Ø2 AA ØØ CF
C68Ø:ØØ AA 8Ø ØØ AA 1Ø ØØ AA Ø9
C688:Ø4 ØØ AA 1Ø ØØ AA 4Ø ØØ 99
C69Ø:AA ØØ ØØ 82 ØØ ØØ 82 ØØ AØ
C698:ØØ 82 ØØ ØØ 82 ØØ ØØ 82 5D
C6AØ:ØØ ØØ 41 ØØ Ø1 41 4Ø FF E3
C6A8:FF FF ØC ØØ 3Ø ØØ ØØ ØØ 39
C6BØ:ØØ ØØ 14 ØØ ØØ 55 ØØ ØØ 16
C6B8:55 ØØ ØØ 14 ØØ ØØ AA ØØ 87
C6CØ:Ø2 AA 8Ø Ø4 AA 1Ø 1Ø AA AA
C6C8:Ø4 Ø4 AA 1Ø Ø1 AA 4Ø ØØ E2
C6DØ:AA ØØ ØØ 82 ØØ ØØ 82 ØØ EØ
C6D8:ØØ 82 ØØ ØØ 82 ØØ ØØ 82 9D
C6EØ:ØØ ØØ 41 ØØ Ø1 41 4Ø FF 24
C6E8:FF FF ØC ØØ 3Ø ØØ ØØ ØØ 79
C6FØ:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ 7E
C6F8:ØØ ØØ Ø8 8Ø ØØ 2Ø ØØ ØØ 1Ø
C7ØØ:Ø2 Ø8 ØØ 2Ø 8Ø ØØ 88 ØØ A9
C7Ø8:ØØ ØØ 88 ØØ 88 Ø8 ØØ Ø2 ØF
C71Ø:ØØ ØØ 2Ø 2Ø ØØ ØØ 8Ø ØØ A6
C718:Ø2 ØØ ØØ ØØ ØØ ØØ ØØ ØØ A8
C72Ø:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ AF
C728:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ B7
C73Ø:35 ØØ ØØ ØØ ØØ ØØ ØØ ØØ 5A
C738:ØØ ØØ ØA 8Ø ØØ 2A AØ ØØ FA
C74Ø:AA A8 ØØ AA 98 ØØ AA 98 AC
C748:ØØ AA 98 ØØ AA 68 ØØ 2A B6
C75Ø:AØ ØØ 2A AØ ØØ 3A 8Ø ØØ A8
C758:Ø2 ØØ ØØ ØØ ØØ ØØ ØØ ØØ E8
C76Ø:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ EF
C768:ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ F7
C77Ø:35 5B 5C 11 9D 9D 5D 5E 8A
C778:91 1D 1D 1D ØØ ØA ØD ØE DD
C78Ø:1E 14 ØA ØØ ØØ 14 ØØ ØØ B5
C788:19 1E ØD 11 2Ø 2Ø 2Ø 2Ø CØ
C79Ø:2Ø 2Ø 2Ø ØØ 13 1D 1D 1D AØ
C798:9E 12 53 43 4F 52 45 1D Ø6
C7AØ:1D 1D 1D 1D 1D 1D 1D 1D 3Ø
C7A8:4C 45 56 45 4C 1D 1D 1D FC
C7BØ:1D 1D 1D 1D 1D 4C 49 56 8E
C7B8:45 53 92 9F ØØ FD FF FF Ø4
C7CØ:FF FF FF FF FF Ø1 Ø2 Ø4 5C
C7C8:Ø8 1Ø 2Ø 2Ø 2Ø 11 9D 9D 84
C7DØ:2Ø 2Ø ØØ 5Ø 78 AØ C8 FØ 46
C7D8:19 ØØ ØØ ØØ ØØ ØØ Ø1 42 39
C7EØ:52 62 72 DB CC CØ B4 A8 89
C7E8:9C 9Ø Ø5 42 1C 41 9F 4C 21
C7FØ:9C 4C 1E 4F 1F 4F 9E 4E 5C
C7F8:2Ø Ø5 43 1C 52 9F 41 9C 34
C8ØØ:5A 1E 59 ØØ 9E 47 41 4D 53
C8Ø8:45 2Ø 4F 56 45 52 ØØ 9E A5
C81Ø:48 49 54 2Ø 46 49 52 45 E5
C818:42 55 54 54 4F 4E ØØ ØØ A3
```

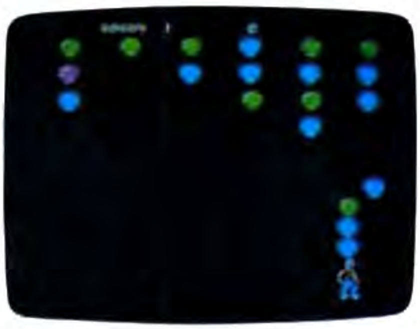
Program 3: Apple Balloon Crazy

Version by Tim Victor, Editorial Programmer

For instructions on entering this listing, please refer to the "Apple MLX" article elsewhere in this issue.

START ADDRESS: 8ØØØ
END ADDRESS: BD97

```
8ØØØ: 2Ø BB 8C 2Ø ØA 8D 2Ø AF ØA
8ØØ8: 88 A9 ØØ 8D 85 8D A9 2Ø 66
8Ø1Ø: 85 E6 2Ø F2 F3 2C 57 CØ 3Ø
8Ø18: 2C 52 CØ 2C 54 CØ 2C 5Ø EC
8Ø2Ø: CØ A9 4Ø 85 E6 2Ø F2 F3 DD
8Ø28: A9 ØØ 8D 93 8D 8D 94 8D 42
8Ø3Ø: A9 2Ø 85 E6 A9 4Ø 8D 74 ØB
8Ø38: 8D A9 ØØ A2 Ø4 9D 51 8E 5C
8Ø4Ø: CA 1Ø FA A9 Ø1 8D 56 8E 1E
8Ø48: A9 Ø3 8D 86 8D 2Ø EB 87 45
8Ø5Ø: A9 Ø1 8D 9Ø 8D 2Ø 4B 88 2D
8Ø58: EE 9Ø 8D A9 Ø3 CD 9Ø 8D 3F
8Ø6Ø: BØ F3 A9 14 8D 76 8D A9 38
8Ø68: ØØ 8D 77 8D A9 ØØ 8D 6A 67
8Ø7Ø: 8D A9 Ø1 8D 69 8D AØ ØØ 5E
8Ø78: AE 56 8E 2Ø 49 87 DD C8 27
8Ø8Ø: 8A 9Ø ØB DD D1 8A 9Ø Ø3 Ø7
8Ø88: A9 Ø3 2C A9 Ø2 2C A9 Ø1 54
8Ø9Ø: 99 35 8E C8 CØ 18 DØ EØ F2
8Ø98: A9 ØØ 8D 8Ø 8D 8D 81 8D 5B
8ØAØ: A9 Ø2 8D 82 8D A9 4Ø 8D F1
8ØA8: 7F 8D AC 8Ø 8D B9 35 8E B6
8ØBØ: 8D 7E 8D 2Ø 6A 87 EE 8Ø 9B
8ØB8: 8D 38 AD 7F 8D E9 12 8D Ø2
8ØCØ: 7F 8D BØ E6 AD 81 8D 69 61
8ØC8: Ø6 C9 Ø7 9Ø Ø5 E9 Ø7 EE F5
8ØDØ: 82 8D 8D 81 8D AD 82 8D F5
8ØD8: 18 69 Ø6 8D 82 8D AD 8Ø FF
8ØEØ: 8D C9 18 DØ CØ A9 ØØ 8D 65
8ØE8: 87 8D A9 ØØ 8D 8Ø 8D A9 79
8ØFØ: 92 8D 84 8D A9 ØØ 8D 7C EC
8ØF8: 8D 2Ø 49 87 2Ø 62 89 2Ø 23
81ØØ: CA 86 AD 8Ø 8D FØ Ø3 4C 4A
81Ø8: B9 81 2Ø 49 87 4A 4A 4A 25
811Ø: 29 Ø7 C9 Ø6 BØ F4 ØA ØA 7A
8118: 8D 8Ø 8D A8 B9 35 8E DØ CE
812Ø: 2E C8 CØ 18 DØ Ø2 AØ ØØ D5
8128: CC 8Ø 8D DØ EF 2Ø 92 84 1A
813Ø: 2Ø 53 87 2Ø 62 89 2Ø EE 73
8138: 89 2Ø EB 87 EE 56 8E A9 95
814Ø: Ø9 CD 56 8E BØ Ø3 8D 56 F1
8148: 8E 2Ø EB 87 4C 76 8Ø 8D 5B
815Ø: 7D 8D 98 8D 8Ø 8D 29 Ø3 FØ
8158: ØA 8D 7F 8D ØA ØA ØA 6D 86
```



The player is just about to lose a collection of balloons in this game of Apple "Balloon Crazy."

```
816Ø: 7F 8D 49 FF 38 69 4Ø 8D 25
8168: 7F 8D A9 Ø2 8D 82 8D A9 1F
817Ø: ØØ 8D 81 8D AD 8Ø 8D 4A B4
8178: 4A A8 FØ 1E AD 81 8D 18 71
818Ø: 69 Ø6 C9 Ø7 9Ø Ø5 E9 Ø7 D6
8188: EE 82 8D 8D 81 8D AD 82 4E
819Ø: 8D 18 69 Ø6 8D 82 8D 88 Ø8
8198: DØ E2 AC 8Ø 8D B9 35 8E A6
81AØ: 8D 7E 8D 2Ø 6A 87 A9 ØØ 82
81A8: AC 8Ø 8D 99 35 8E AD 7D 2A
81BØ: 8D 8D 8Ø 8D A9 ØØ 8D 7A A9
81B8: 8D AD 7C 8D FØ 64 AD 7F 4A
81CØ: 8D C9 58 BØ ØB A9 ØØ 8D 9F
81C8: 7C 8D 8D 7A 8D 4C 22 82 2B
81DØ: AD 8Ø 8D ØA 49 FF 38 6D 45
81D8: 7F 8D 8D 7F 8D AD 82 8D 5E
81EØ: 2C 7C 8D 3Ø 1Ø AD 81 8D 95
81E8: 38 E9 Ø2 BØ 17 CE 82 8D 54
81FØ: 69 Ø7 4C Ø4 82 AD 81 8D 8F
81F8: 18 69 Ø2 C9 Ø7 9Ø Ø5 EE 82
82ØØ: 82 8D E9 Ø7 8D 81 8D AD 92
8208: 82 8D C9 26 9Ø ØE A9 FE 5C
821Ø: 4D 7C 8D 8D 7C 8D 2Ø EB AB
8218: 8A 4C DD 81 AD 7F 8D 4C 1C
822Ø: 2C 82 18 AD 8Ø 8D 6D 7F 4E
8228: 8D 8D 7F 8D CD 84 8D BØ 6C
823Ø: Ø3 4C FB 82 38 AD 76 8D 64
8238: ED 82 8D C9 FC 1Ø Ø3 4C 9D
824Ø: 9Ø 82 C9 Ø5 3Ø Ø3 4C 9D 7B
8248: 82 8D 7B 8D ØA ØA ØA 38 9E
825Ø: ED 78 8D 18 6D 77 8D 38 3A
8258: ED 81 8D 2C 7A 8D 3Ø 5Ø E3
826Ø: C9 F6 3Ø 2C C9 Ø9 1Ø 35 98
8268: 2Ø F3 8A 38 AD 84 8D E9 D3
827Ø: ØE 8D 84 8D AE 87 8D AD A5
8278: 8Ø 8D 9D 4D 8E E8 8E 87 66
828Ø: 8D EØ Ø4 DØ Ø8 2Ø 92 84 7C
8288: A9 92 8D 84 8D 4C 3C 83 9A
829Ø: AE 7A 8D FØ 15 2C 7B 8D 2A
8298: 3Ø 5C 4C D8 82 AE 7A 8D 35
82AØ: FØ Ø8 2C 7B 8D 1Ø 4F 4C F4
82A8: BC 82 8D 7B 8D 4C F6 82 24
82BØ: C9 EB 3Ø DC C9 16 1Ø E5 16
82B8: Ø9 ØØ 1Ø 1C AD 76 8D 18 8Ø
82CØ: 69 Ø3 8D 82 8D AC 77 8D 8Ø
82C8: C8 CØ Ø7 DØ Ø5 AØ ØØ EE E9
82DØ: 82 8D 8C 81 8D 4C EF 82 24
82D8: AD 76 8D 38 E9 Ø3 8D 82 8Ø
82EØ: 8D AC 77 8D 88 1Ø Ø5 CE FC
82E8: 82 8D AØ Ø6 8C 81 8D AD 3A
82FØ: 82 8D C9 26 8Ø 18 A9 FF 6F
82F8: 8D 7A 8D AD 7F 8D C9 AE 64
83ØØ: 8Ø ØC AD 8Ø 8D 8D 7E 8D 4D
83Ø8: 2Ø 6C 87 4C 41 83 AD 8Ø E3
831Ø: 8D Ø9 8Ø 8D 7E 8D 2Ø 6C DF
8318: 87 2Ø DB 8A 2Ø 8C 85 A9 F6
832Ø: 92 8D 84 8D A9 ØØ 8D 87 2D
8328: 8D AD 86 8D 8D 9Ø 8D 2Ø F4
833Ø: 4B 88 CE 86 8D DØ Ø5 A9 A4
8338: FF 8D 85 8D A9 ØØ 8D 8Ø 15
834Ø: 8D 2Ø 4A 85 2Ø 53 87 2Ø 35
8348: 62 89 2Ø EE 89 AD 85 8D 71
835Ø: FØ Ø3 4C Ø7 89 AD ØØ CØ 4E
8358: C9 8D DØ Ø8 2C 1Ø CØ AD 13
836Ø: ØØ CØ 1Ø FB 2C 6B 8D 1Ø 93
8368: Ø3 4C 7E 83 2C 61 CØ 3Ø A4
837Ø: 39 2C 62 CØ 3Ø 55 AD ØØ A9
8378: CØ 3Ø 29 4C F9 8Ø A2 ØØ EC
838Ø: 2Ø 1E FB CC 6C 8D FØ 17 FD
8388: AD 6C 8D 8C 6C 8D 8D 78 29
839Ø: 8D 18 6D 6C 8D 6A CD 78 Ø3
8398: 8D FØ Ø4 BØ 5C 9Ø 57 2C 2E
83AØ: 61 CØ 1Ø Ø3 4C 45 84 4C 87
83A8: F9 8Ø A9 FF 8D 69 8D AD DC
83BØ: 77 8D 38 E9 Ø6 BØ ØE 69 F4
83B8: Ø7 CE 76 8D 1Ø Ø7 A9 ØØ 8E
83CØ: 8D 76 8D A9 ØØ 8D 77 8D 28
83C8: 4C F9 8Ø A9 Ø1 8D 69 8D BD
83DØ: AD 77 8D 18 69 Ø6 C9 Ø7 8D
83D8: 9Ø Ø5 E9 Ø7 EE 76 8D 8D 11
83EØ: 77 8D AC 76 8D CØ 25 9Ø 4E
83E8: Ø7 AØ 25 8C 76 8D A9 Ø1 47
83FØ: 8D 77 8D 4C F9 8Ø A2 FF 2A
83F8: 2C A2 Ø1 8E 69 8D 38 E9 A3
84ØØ: 3F BØ Ø2 A9 ØØ C9 82 9Ø 6C
84Ø8: Ø2 A9 81 8D 78 8D A9 ØØ D2
841Ø: 8D 79 8D 18 AØ Ø4 6D 78 D9
8418: 8D A2 Ø3 6A 6E 79 8D 18 24
842Ø: CA DØ F8 88 DØ FØ 2A 8D 96
8428: 76 8D AD 79 8D 4A 4A 4A 91
843Ø: 4A 29 ØE C9 Ø7 9Ø Ø8 E9 7B
8438: Ø7 EE 76 8D 4C 33 84 8D ED
844Ø: 77 8D 4C F9 8Ø AD 1Ø CØ 2D
8448: AD 8Ø 8D FØ 1Ø AD 76 8D BA
845Ø: CD 82 8D BØ Ø3 A9 Ø1 2C 8A
8458: A9 FF 8D 69 8D A9 Ø5 8D 29
846Ø: 6A 8D AD 7C 8D DØ 28 AD 2D
8468: 8Ø 8D FØ 23 AD 7F 8D C9 85
847Ø: AØ 9Ø 1C AD 76 8D 38 ED 94
8478: 82 8D C9 FB 3Ø 11 BØ Ø7 4D
848Ø: C9 Ø6 1Ø ØB A9 Ø1 2C A9 F5
8488: FF 8D 7C 8D 2Ø E3 8A 4C 4F
849Ø: F9 8Ø AE 87 8D DØ Ø1 6Ø 17
8498: 2Ø 4A 85 2Ø 53 87 A2 ØØ F4
84AØ: AØ ØØ CA DØ FD 88 DØ FA ØF
84A8: 2Ø 62 89 2Ø EE 89 2Ø 62 CD
84BØ: 89 A9 FF 8D 69 8D 2Ø CA 4E
84B8: 86 2Ø EB 87 A2 Ø3 AD 4D CC
```

```
84C0: 8E 18 7D 51 8E 9D 51 8E F7
84C8: C9 0A 90 12 E9 0A 9D 51 70
84D0: 8E CA 30 0A FE 51 8E BD 92
84D8: 51 8E C9 0A B0 EE 20 EB 75
84E0: 87 AD 4D 8E 09 80 8D 4D 5E
84E8: 8E 8D 4D 8E 20 4A 85 A9 0E
84F0: 16 85 FC A9 8C 85 FD A9 C0
84F8: 00 8D 6D 8D A9 15 8D 6E 17
8500: 8D AD 76 8D 8D 72 8D AD E3
8508: 77 8D 8D 73 8D A9 9C 8D F4
8510: 71 8D A9 00 8D 70 8D 20 D5
8518: 7F 89 20 53 87 20 DB 8A 7D
8520: A2 00 A0 00 CA D0 FD 88 AE
8528: D0 FA A2 00 BD 4E 8E 9D 90
8530: 4D 8E E8 EC 87 8D D0 F4 7A
8538: CE 87 8D 20 62 89 20 EE A8
8540: 89 20 62 89 20 CA 86 4C 82
8548: 92 84 A9 92 8D 89 8D A2 6C
8550: 00 8E 88 8D AE 88 8D EC 88
8558: 87 8D F0 2F BD 4D 8E 8D 69
8560: 7E 8D AD 76 8D 8D 72 8D 40
8568: AD 77 8D 8D 73 8D AD 89 69
8570: 8D 8D 71 8D A9 80 8D 83 9A
8578: 8D 20 85 87 AD 89 8D 38 62
8580: E9 0E 8D 89 8D EE 88 8D 15
8588: 4C 54 85 60 AD 87 8D D0 FC
8590: 01 60 A9 08 8D 8B 8D A9 49
8598: 06 8D 8A 8D AD 8B 8D CD B8
85A0: 76 8D 90 0A D0 26 AD 8A 02
85A8: 8D CD 77 8D B0 1E AD 8A 99
85B0: 8D 69 06 C9 07 90 05 E9 A8
85B8: 07 EE 8B 8D 8D 8A 8D AD AC
85C0: 8B 8D 18 69 06 8D 8B 8D 99
85C8: C9 24 D0 D0 AD 8A 8D 38 D3
85D0: E9 03 B0 05 69 07 CE 8B 88
85D8: 8D 8D 8A 8D AD 8B 8D 38 27
85E0: E9 03 8D 8B 8D 38 ED 76 AB
85E8: 8D 8D 78 8D 0A 0A 0A 38 CA
85F0: ED 78 8D 18 6D 8A 8D ED E2
85F8: 77 8D 8D 8C 8D 10 05 49 9D
8600: FF 38 69 00 18 69 51 8D DE
8608: 8D 8D A9 90 8D 8E 8D A9 E8
8610: 00 8D 88 8D AD 8E 8D 8D BA
8618: 89 8D AE 88 8D EC 87 8D 68
8620: D0 03 4C AA 86 BD 4D 8E DE
8628: 8D 7E 8D AD 76 8D 8D 72 9F
8630: 8D AD 77 8D 8D 73 8D AD 3A
8638: 89 8D C9 52 90 35 AD 8D 0E
8640: 8D 38 ED 89 8D 90 38 AE 46
8648: 8C 8D F0 33 10 05 49 FF 77
8650: 38 69 00 18 6D 73 8D 30 D9
8658: 0C C9 07 90 10 E9 07 EE EC
8660: 72 8D 4C 59 86 18 69 07 97
8668: CE 72 8D 90 F9 8D 73 8D AE
8670: 4C 7F 86 AD 8B 8D 8D 72 4F
8678: 8D AD 8A 8D 8D 73 8D AD E4
8680: 89 8D C9 0A B0 0A AE 88 23
8688: 8D CA 8E 87 8D 4C AA 86 D2
8690: 8D 71 8D A9 80 8D 83 8D DB
8698: 20 85 87 AD 89 8D 38 E9 BF
86A0: 0E 8D 89 8D EE 88 8D 4C 23
86A8: 1A 86 AD 8E 8D 38 E9 02 26
86B0: 8D 8E 8D C9 0A 90 12 20 4D
86B8: 53 87 20 62 89 20 EE 89 AF
86C0: 20 62 89 20 CA 86 4C 0F C1
86C8: 86 60 AD 76 8D 8D 72 8D 63
86D0: AD 77 8D 8D 73 8D 2C 69 B0
86D8: 8D 10 0B A9 AA 85 FC A9 BB
86E0: 8B 85 FD 4C EE 86 A9 56 D4
86E8: 85 FC A9 8B 85 FD A9 02 5F
86F0: 8D 6D 8D A9 1C 8D 6E 8D ED
86F8: A9 A0 8D 71 8D A9 00 8D 6C
8700: 70 8D 20 7F 89 AD 6A 8D 0C
8708: F0 3E CE 6A 8D AD 76 8D 3D
8710: 8D 72 8D AD 77 8D 8D 73 8F
8718: 8D 2C 69 8D 10 0B A9 0A 09
8720: 85 FC A9 8C 85 FD 4C 31 1D
8728: 87 A9 FE 85 FC A9 8B 85 C8
8730: FD A9 02 8D 6D 8D A9 04 BA
8738: 8D 6E 8D A9 B5 8D 71 8D 4A
8740: A9 00 8D 70 8D 20 7F 89 52
8748: 60 A5 4E 0A 0A 38 65 4E A5
8750: 85 4E 60 AD 74 8D C9 40 4A
8758: A9 00 2A AA BD 54 C0 AD 9A
8760: 74 8D 85 E6 49 60 8D 74 87
8768: 8D 60 18 24 38 A9 00 6A 6E
8770: 8D 83 8D AD 7F 8D 8D 71 72
8778: 8D AD 82 8D 8D 72 8D AD E1
8780: 81 8D 8D 73 8D A9 2B 85 BB
8788: FC A9 8C 85 FD A9 0D 8D A8
8790: 6E 8D A9 02 8D 6D 8D A9 76
8798: 00 8D 70 8D AD 7E 8D 29 9D
87A0: 7F F0 47 C9 02 90 17 D0 82
87A8: 10 AC 73 8D C8 C0 07 D0 5A
87B0: 05 A0 00 EE 72 8D 8C 73 AF
87B8: 8D A9 80 8D 70 8D 2C 7E 72
87C0: 8D 10 10 A9 52 85 FC A9 83
87C8: 8C 85 FD CE 71 8D A9 0F 50
87D0: 8D 6E 8D 2C 83 8D 10 03 2C
87D8: 4C 7F 89 20 BF 89 AD 74 15
87E0: 8D 49 60 8D 74 8D C5 E6 3A
87E8: F0 89 60 A9 00 8D 8F 8D 5C
87F0: AE 8F 8D BD 51 8E 0A 8D 2F
87F8: 78 8D 0A 18 6D 78 8D 69 3C
8800: 7F 85 FC A9 8C 85 FD 90 73
8808: 02 E6 FD A9 00 8D 6D 8D CC
8810: 8D 71 8D 8D 70 8D 8D 73 17
8818: 8D A9 06 8D 6E 8D E0 05 64
8820: D0 04 A9 0E D0 04 8A 18 74
8828: 69 07 8D 72 8D 20 BF 89 7E
8830: AD 74 8D 49 60 8D 74 8D 2B
8838: C5 E6 F0 B4 AE 8F 8D E8 07
8840: E0 06 F0 06 8E 8F 8D 4C DB
8848: F0 87 60 A9 02 8D 92 8D 53
8850: A9 14 8D 91 8D A9 00 8D A6
8858: 70 8D AD 90 8D 0A 0A 0A 76
8860: 0A 6D 92 8D C9 07 90 07 8F
8868: EE 91 8D E9 07 B0 F5 8D 1A
8870: 92 8D A9 07 8D 6E 8D A9 BE
8878: 02 8D 6D 8D AD 91 8D 8D D0
8880: 72 8D AD 92 8D 8D 73 8D 24
8888: A9 00 8D 71 8D A9 56 85 7C
8890: FC A9 8B 85 FD 20 BF 89 CD
8898: AD 74 8D 49 60 8D 74 8D 93
88A0: C5 E6 F0 CE 60 AD 54 C0 7C
88A8: AD 51 C0 20 58 FC 60 20 96
88B0: A5 88 A9 08 85 25 20 22 8F
88B8: FC A9 06 85 24 A2 00 BD 35
88C0: EC 88 20 ED FD E8 E0 1B BD
88C8: D0 F5 AD 10 C0 AD 00 C0 F3
88D0: C9 D0 F0 12 C9 F0 F0 0E 3C
88D8: C9 CB F0 04 C9 EB D0 ED AD
88E0: A9 00 8D 6B 8D 60 A9 FF 70
88E8: 8D 6B 8D 60 C9 CE D0 D5 54
88F0: D4 BA A0 A8 D0 A9 C1 C4 2F
88F8: C4 CC C5 AC A0 A8 CB A9 0C
8900: C5 D9 C2 CF C1 D2 C4 A2 47
8908: 00 A0 00 88 D0 FD CA D0 B0
8910: F8 20 A5 88 A9 08 85 25 82
8918: 20 22 FC A9 08 85 24 A2 3F
8920: 00 BD 48 89 20 ED FD E8 42
8928: E0 17 D0 F5 AD 10 C0 AD C7
8930: 00 C0 C9 D9 F0 12 C9 F9 A7
8938: F0 0E C9 CE F0 04 C9 EE 87
8940: D0 ED AD 10 C0 4C D0 03 C9
8948: 4C 09 80 D0 CC C1 D9 A0 A2
8950: C1 C7 C1 C9 CE BF A0 A8 6A
8958: D9 A9 C5 D3 AC A0 A8 CE C0
8960: A9 CF AD 74 8D C9 20 D0 DD
8968: 08 A9 95 A0 8D A2 00 F0 8E
8970: 06 A9 E5 A0 8D A2 01 8E 3F
8978: 75 8D 85 1E 84 1F 60 A0 3E
8980: 00 AD 71 8D 91 1E C8 AD 4A
8988: 72 8D 91 1E C8 AD 73 8D BD
8990: 91 1E C8 A5 FC 91 1E C8 9A
8998: A5 FD 91 1E C8 AD 6E 8D 79
89A0: 91 1E C8 AD 6D 8D 91 1E DA
89A8: C8 AD 70 8D 91 1E A5 1E E0
89B0: 18 69 08 85 1E 90 02 E6 A1
89B8: 1F AE 75 8D FE 93 8D AD 9D
89C0: 73 8D 09 60 8D 47 8A 09 BF
89C8: 08 8D 51 8A AC 6D 8D C8 15
89D0: 8C 6F 8D 20 60 8A 20 3C 63
89D8: 8A EE 71 8D A5 FC 38 6D F2
89E0: 6D 8D 85 FC 90 02 E6 FD F6
89E8: CE 6E 8D D0 E6 60 AE 75 49
89F0: 8D 8D 93 8D F0 45 A0 00 63
89F8: B1 1E 8D 71 8D C8 B1 1E 46
8A00: 8D 72 8D C8 B1 1E 8D 73 4B
8A08: 8D C8 B1 1E 85 FC C8 B1 91
8A10: 1E 85 FD C8 B1 1E 8D 6E 71
8A18: 8D C8 B1 1E 8D 6D 8D C8 44
8A20: B1 1E 8D 70 8D 20 BF 89 44
8A28: A5 1E 18 69 08 85 1E 90 54
8A30: 02 E6 1F AE 75 8D DE 93 02
8A38: 8D D0 BB 60 A0 00 A9 00 1E
8A40: 48 B1 FC AA 68 1D 00 61 49
8A48: 4D 70 8D 51 FE 91 FE BD E0
8A50: 00 69 C8 CC 6F 8D D0 E8 E1
8A58: 4D 70 8D 51 FE 91 FE 60 93
8A60: AD 71 8D 29 3F A8 B9 89 96
8A68: 8A 0D 74 8D 85 FF AD 71 66
8A70: 8D 29 08 F0 02 A9 80 18 76
8A78: 2C 71 8D 70 04 10 04 69 8A
8A80: 28 69 28 6D 72 8D 85 FE B3
8A88: 60 00 04 08 0C 10 14 18 AF
8A90: 1C 00 04 08 0C 10 14 18 95
8A98: 1C 01 05 09 0D 11 15 19 1D
8AA0: 1D 01 05 09 0D 11 15 19 A5
8AA8: 1D 02 06 0A 0E 12 16 1A 2D
8AB0: 1E 02 06 0A 0E 12 16 1A B5
8AB8: 1E 03 07 0B 0F 13 17 1B 3D
8AC0: 1F 03 07 0B 0F 13 17 1B C5
8AC8: 1F B0 90 70 60 50 40 30 A7
8AD0: 10 01 FF F7 D8 C6 B4 A2 9B
8AD8: 90 80 60 A9 FF 8D 19 8B F0
8AE0: 4C FB 8A A9 70 8D 19 8B 7E
8AE8: 4C FB 8A A9 71 8D 19 8B 8E
8AF0: 4C FB 8A A9 72 8D 19 8B 9E
8AF8: 4C FB 8A A9 01 8D 53 8B 8F
8B00: A0 00 A9 E0 8D 2C 8B A9 88
8B08: FF 8D 36 8B AD 36 8B 8D EC
8B10: 55 8B 4E 53 8B 90 0C B9 24
8B18: 00 70 C8 8D 54 8B A9 80 E1
8B20: 8D 53 8B 4E 54 8B 90 03 1E
8B28: AD 30 C0 A2 FF E8 D0 FD A7
8B30: 90 03 AD 30 C0 A2 FF E8 82
8B38: D0 FD EE 55 8B D0 D3 18 C9
8B40: AD 36 8B E9 01 8D 36 8B 02
8B48: AD 2C 8B 69 01 8D 2C 8B 6B
8B50: 90 BA 60 20 2E 20 80 BE 1E
8B58: 80 80 FF B1 80 AF 80 80 2C
8B60: E7 80 80 FB 80 80 8D 80 5D
8B68: 80 8E 80 A0 D5 80 B8 D1 71
8B70: 81 9C E4 81 8E D1 83 86 6D
8B78: C4 83 86 91 87 86 84 87 A3
8B80: 8E 91 8E CC AA 8C D0 AA B5
8B88: 98 C0 AA 80 C0 A2 81 C0 CD
8B90: 82 85 C0 82 85 D0 A0 81 BC
8B98: D0 A0 81 D0 A0 81 F0 E3 4E
8BA0: 87 FC FF 9F F0 80 87 F0 3E
8BA8: 80 87 C0 8F 80 F0 9F 80 7A
8BB0: C0 9E 80 E0 9C 80 E0 9B 32
8BB8: 80 C0 97 80 80 8E 80 A0 1B
8BC0: D5 80 B0 D1 83 F0 84 87 86
8BC8: 88 91 8E B8 84 8C 9C 91 1F
8BD0: 8C 9C 84 8C 8E 91 8E C6 4D
8BD8: AA 86 C3 AA 81 C0 AA 80 EE
8BE0: D0 A8 80 94 A8 80 94 A8 FC
8BE8: 80 D0 A0 81 D0 A0 81 D0 7D
8BF0: A0 81 FC F8 81 FF FF 87 7B
8BF8: 9C E0 81 9C E0 81 00 20 BD
8C00: 15 00 20 1F 00 20 01 00 1C
8C08: 60 07 55 00 00 5F 00 00 3B
8C10: 50 00 00 7C 00 00 06 0E 33
8C18: 06 83 07 07 07 03 07 0E C6
8C20: 1C 18 40 70 38 38 1C 1C 53
8C28: 0E 06 03 C0 AA 80 D0 AF DE
8C30: 81 F4 AB 85 F4 AA 85 D4 47
8C38: AA 85 DC AA 85 D4 AA 85 AB
8C40: D4 AA 85 D0 AA 81 D0 BA E3
8C48: 81 C0 AA 80 C0 AA 80 80 E1
8C50: 8A 80 C0 B0 80 D0 80 80 B7
8C58: D4 A0 81 F4 F0 85 D4 A8 73
8C60: 95 DC AC 8D D4 A8 95 94 F2
8C68: A8 85 80 80 85 80 AA B0 53
8C70: C0 AA 81 C0 AA 81 C0 BA 68
8C78: 80 C0 AA B0 80 8A 80 BC 4B
8C80: E6 F6 EE E6 BC 98 9C 98 31
8C88: 98 98 BC BC E6 B0 8C E6 71
8C90: FE BC E6 B0 E0 E6 BC B0 0D
8C98: B8 B4 FE B0 B0 FE 86 BE 73
8CA0: E0 E6 BC BC 86 BE E6 E6 2B
8CA8: BC FE E0 B0 98 8C BC BC D3
8CB0: E6 BC E6 E6 BC BC E6 E6 45
8CB8: FC B0 98 A9 00 85 EC 85 9F
8CC0: EE A9 60 85 ED A9 68 85 8C
8CC8: EF A0 00 8C 73 8D A9 00 EF
8CD0: 85 1E 98 8D 70 8D 0A AE 9C
8CD8: 73 8D F0 06 0A 26 1E CA 7D
8CE0: D0 FA 2C 70 8D 10 02 38 96
```

```
8CE8: 24 18 6A 91 EC A5 1E 2C E6
8CF0: 70 8D 10 02 09 80 91 EE 24
8CF8: C8 D0 D3 E6 ED E6 EF EE 6D
8D00: 73 8D AE 73 8D E0 07 D0 14
8D08: C5 60 A0 00 A9 80 99 00 B4
8D10: 70 99 01 70 99 00 71 99 3A
8D18: 01 71 99 04 71 99 05 71 F0
8D20: A9 00 99 02 70 99 03 70 C3
8D28: 99 04 70 99 05 70 99 06 DB
8D30: 70 99 07 70 99 02 71 99 23
8D38: 03 71 99 06 71 99 07 71 36
8D40: 98 18 69 08 A8 90 C5 A0 0F
8D48: 00 A9 80 99 00 72 A9 0B 9C
8D50: 99 01 72 A9 00 99 02 72 3E
8D58: 99 03 72 99 04 72 99 05 0B
8D60: 72 98 18 69 06 A8 90 E1 4A
8D68: 60 20 20 20 20 20 4F 20 02
8D70: 04 20 20 20 23 20 7F 20 54
8D78: 2D 20 20 20 58 41 20 20 60
8D80: 20 45 20 20 20 20 20 20 E4
8D88: 4C 2E 58 52 00 20 20 00 46
8D90: 20 52 42 20 41 45 4D 4F A3
```

Program 4: IBM PC/PCjr Balloon Crazy

Version by Charles Brannon, Program Editor

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
KL 100 'Balloon Crazy for IBM PC/PCjr requires BASICA, Color Graphics adapter, and one joystick
OK 110 DEFINT A-Z:RANDOMIZE TIMER:DIM BP$(3),MOBJ(452),X(49),Y(49)
OF 120 DF=10:' change df to a smaller number for a greater challenge
ID 130 SCREEN 1:COLOR 9,0:KEY OFF:STRIG ON::PLAY "mf":CLS
AD 140 GOSUB 280:X=100:HP=164:EY=Y+22
IG 150 TX=3:LX=134:SKEW!=2.27:SP!=4:LIVES=4
BE 160 GOSUB 740:PRINT"BALLOON C R A Z Y !":LOCATE 9,10:PRINT"/":LOCATE 8,11:PRINT"Do you need":LOCATE 9,11:PRINT"to adjust":LOCATE 10,11:PRINT"your joystick?":WHILE INKEY$<>"":WEND
EI 170 A$=INKEY$:IF A$="" AND STRIG(1)=0 THEN 170 ELSE IF (ASC(A$+CHR$(0)) OR 32)=121 THEN GOSUB 600
KE 180 CLS:PRINT"BALLOON C R A Z Y !":LOCATE 1,25:PRINT"Score:";SC!
BA 190 FOR I=158 TO 174 STEP 8:PUT(I,0),TINY:NEXT:GOSUB 350
KE 200 FOR ROW=20 TO 80 STEP 20:FOR COL=45 TO 255 STEP 15:PUT (COL,ROW),BALL:NEXT COL,ROW
BJ 210 PUT(X,Y),MAN:GET (X,Y)-(X+21,EY),MOBJ
CN 220 IF BP<0 THEN GOSUB 380:GOTO 180 ELSE BY=BP*20+20:T$=BP$(BP):PTR=.5+RND(1)*LEN(T$):BX=(ASC(MID$(T$,PTR))-64)*15+30:T$=LEFT$(T$,PTR-1)+MID$(T$,PTR+1):BP$(BP)=T$:IF T$="" THEN BP=BP-1
JG 230 PUT(BX,BY),BALL:BY=BY+SP!:PUT(BX,BY),BALL:IF BY>180 THEN 440
KI 240 IF ABS(BY-HP)<SP! THEN IF ABS((BX-7)-X)<DF THEN Y=Y-13:PUT(BX,BY),BALL:PUT(X+7,Y),BALL,PSET:SOUND 30000,1:GET(X,Y)-(X+21,EY),MOBJ:HP=HP-13:FLOATERS=FLOATERS+1:SP!=SP!+.5:IF FLOATERS=7-BP THEN GOSUB 380:GOTO 210 ELSE 220
FD 250 PUT(X,Y),MOBJ:NX=(STICK(0)-TX)*SKEW!:X=NX:IF NX<0 THEN X=0 ELSE IF NX>297 THEN X=297
HE 260 PUT(X,Y),MOBJ
DG 270 GOTO 230
PO 280 READ XS,YS:E=(4+INT((XS+7)/8)*YS)/2:DIM MAN(E):MAN(0)=XS:MAN(1)=YS:FOR I=2 TO E:READ MAN(I):NEXT:Y=200-YS:MY=Y
HF 290 READ XS,YS:E=(4+INT((XS+7)/8)*YS)/2:DIM POP(E):POP(0)=XS:POP(1)=YS:FOR I=2 TO E:READ POP(I):NEXT
HO 300 READ XS,YS:E=(4+INT((XS+7)/8)*YS)/2:DIM FALL(E):FALL(0)=XS:FALL(1)=YS:FOR I=2 TO E:READ FALL(I):NEXT
HE 310 READ XS,YS:E=(4+INT((XS+7)/8)*YS)/2:DIM BALL(E):BALL(0)=XS:BALL(1)=YS:FOR I=2 TO E:READ BALL(I):NEXT
KE 320 READ XS,YS:E=(4+INT((XS+7)/8)*YS)/2:DIM XBALL(E):XBALL(0)=XS:XBALL(1)=YS:FOR I=2 TO E:READ XBALL(I):NEXT
CL 330 READ XS,YS:E=(4+INT((XS+7)/8)*YS)/2:DIM TINY(E):TINY(0)=XS:TINY(1)=YS:FOR I=2 TO E:READ TINY(I):NEXT
MF 340 RETURN
LG 350 BP=3:FOR I=0 TO BP:BP$(I)="ABCDEFGHIJKLMNO":NEXT
NJ 360 RETURN
JH 370 GOSUB 380:GOTO 220
LE 380 FOR I=1 TO FLOATERS
LK 390 PUT(X,MY),POP,PSET:PUT(X+7,MY-13),BALL:PUT(X+7,MY-18),XBALL:FOR J=0 TO 5:SOUND 100+J,.5:NEXT:PUT(X+7,MY-18),XBALL
OD 400 IF I<FLOATERS THEN PUT(X+7,Y),BALL:PUT(X+7,MY-13),BALL
FL 410 PUT(X,MY),MAN,PSET:FOR W=1 TO 5:NEXT:SC!=SC!+10:LOCATE 1,31:PRINT SC!:Y=Y+13:NEXT
GH 420 Y=MY:GET (X,Y)-(X+21,EY),MOBJ:PUT (X,Y),MAN
PO 430 SP!=4:HP=164:FLOATERS=0:RETURN
FD 440 PUT(BX,BY),BALL:PUT(BX,BY-5),XBALL:FOR J=0 TO 5:SOUND 105-J,.5:NEXT:PUT(BX,BY-5),XBALL
EH 450 PUT(X,MY),MAN:PUT(X,MY),FALL
CC 460 IF FLOATERS=0 THEN 510
LD 470 FOR I=1 TO FLOATERS
OI 480 PUT(X+7,MY-13),BALL:PUT(X+7,MY-18),XBALL:FOR J=0 TO 5:SOUND 105-J,.5:NEXT:PUT(X+7,MY-18),XBALL
PF 490 IF I<FLOATERS THEN PUT(X+7,Y),BALL:PUT(X+7,MY-13),BALL
JM 500 FOR W=1 TO 5:NEXT:Y=Y+13:NEXT
FL 510 NY=MY:S=-6:FOR I=X+5 TO 291 STEP 5:PUT(I-5,NY),FALL:NY=NY+S:IF NY<MY-18 OR NY>MY THEN NY=NY-S:S=-S
BJ 520 PUT(I,NY),FALL:SOUND 100+NY,.5:NEXT
HD 530 NX=I-5:FOR I=NY-16 TO 0 STEP-16:PUT(NX,I+16),FALL:PUT(NX,I),FALL:SOUND 5000-I*5,.1:NEXT
LD 540 PUT(NX,I+16),FALL
JM 550 LIVES=LIVES-1:PUT(150+LIVES*8,0),TINY:IF LIVES>0 THEN PUT(X,MY),MAN:GOSUB 420:GOTO 210
OM 560 CLS:FOR I=0 TO 49:X(I)=4+15*INT(20*RND(1)):Y(I)=18*INT(10*RND(1)):PUT (X(I),Y(I)),BALL,PSET:NEXT
JA 570 FOR I=0 TO 49:PUT (X(I)-4,Y(I)),XBALL,PSET:SOUND 100+5*RND(1),.2:SOUND 30000,.2:PUT (X(I)-4,Y(I)),XBALL:NEXT
LL 580 LOCATE 12,16:PRINT"GAME OVER":LOCATE 13,17-LEN(STR$(SC!))/2:PRINT"Score:";SC!:LOCATE 14,15:PRINT"Press Button"
OJ 590 A$=INKEY$:IF INKEY$="" AND STRIG(1)=0 THEN 590 ELSE RUN
KP 600 GOSUB 740
JN 610 LOCATE 9,10:PRINT"/":LOCATE 8,11:PRINT"Move stick to":LOCATE 9,11:PRINT"far left,":LOCATE 10,11:PRINT"press button!"
JI 620 TX=STICK(0):IF STRIG(1)<>0 THEN 640
JI 630 FOR I=10 TO 0 STEP-1:C=-C*(C<3)+1:LINE (30+I,40)-(I,70),C:LINE-(30+I,100),C:NEXT:GOTO 620
OJ 640 PUT(50,63),POP,PSET:PUT(57,45),XBALL,PSET:FOR J=1 TO 15:SOUND 100+J,.5:NEXT
BG 650 CLS:PUT (50,63),MAN:LOCATE 9,10:PRINT"/":LOCATE 8,11:PRINT"Gimme another":LOCATE 9,11:PRINT"balloon!"
CJ 660 FOR J=1 TO 2000:NEXT
KN 670 PUT(57,0),BALL:FOR I=2 TO 50 STEP 2:WAIT &H3DA,8:PUT (57,I-2),BALL:WAIT &H3DA,8:PUT (57,I),BALL:NEXT
CE 680 LOCATE 8,11:PRINT"Move stick to ":LOCATE 9,11:PRINT"far right,":LOCATE 10,11:PRINT"press button!"
CB 690 LX=STICK(0):IF STRIG(1)<>0 THEN 710
BP 700 FOR I=10 TO 0 STEP-1:C=-C*(C<3)+1:LINE (289-I,40)-(319-I,70),C:LINE-(289-I,100),C:NEXT:GOTO 690
MK 710 FOR I=1 TO 5:PUT (57,45),XBALL,PSET:PUT(50,63),POP,PSET:FOR J=1 TO 4:SOUND 100+J,.5:NEXT:PUT(57,50),
```

The main character in IBM PC/PCjr "Balloon Crazy" is a humorous clown.

```
         BALL,PSET:PUT(50,63),MAN,
         PSET:FOR J=1 TO 100:NEXT:
         NEXT
NA 720 SKEW!=297/ABS(LX-TX)
MH 730 RETURN
PC 740 CLS:PUT(0,63),MAN:PUT (57
       ,0),BALL:FOR I=2 TO 50 ST
       EP 2:PUT (57,I-2),BALL:PU
       T (57,I),BALL:PUT (I-2,63
       ),MAN:PUT (I,63),MAN,PSET
       :NEXT:RETURN
HA 750 DATA &H2C,&H17,&H0,&H5,&H
       0,&H0,&H4015,&H0
FD 760 DATA &H0,&H5055,&H0,&H0,&
       H30CF,&H0,&H300,&HCC3
JJ 770 DATA &H0,&H300,&HACAA,&H0
       ,&H0,&HA0A0,&H0,&H0
SF 780 DATA &H802A,&H0,&H0,&HF,&
       H0,&HE00,&HEEEE,&HC0
EA 790 DATA &HFB00,&HBBEB,&HB0,&
       HC003,&HE0EE,&H3C,&HF,&HB
       03B
DM 800 DATA &HF,&HFF,&HC02E,&HF0
       0F,&H0,&H4015,&H0,&H0
6J 810 DATA &HA0AA,&H0,&H0,&HA0A
       A,&H0,&H200,&HABA0,&H0
FJ 820 DATA &H200,&HABA0,&H0,&HA
       00,&H2A80,&H0,&HA37,&H2AB
       0
AB 830 DATA &HC00D,&HDADD,&H3AC0
       ,&H7077,&H7737,&H1D40,&HC
       0DD,&HA800
5F 840 DATA &H2C,&H17,&H800,&H5,
       &H0,&HA000,&H4015,&H0
IG 850 DATA &H8003,&H5055,&H0,&H
       C003,&HC3,&H0,&H30F,&H3CC
       F
KH 860 DATA &H0,&H30F,&HACAA,&H0
       ,&HC003,&HA0A0,&H0,&HF003
KI 870 DATA &HB02A,&H0,&HFC00,&H
       F,&H0,&H2E00,&HEEEE,&HC0
CN 880 DATA &HB00,&HBBBB,&HBC,&H
       0,&HE0EE,&HFF,&H0,&H803B
EG 890 DATA &HF,&H0,&HC02E,&HC00
       3,&H0,&H4015,&HC003,&H0
6C 900 DATA &HA0AA,&HC000,&H0,&H
       A0AA,&H0,&H200,&HABA0,&H0
FI 910 DATA &H200,&HABA0,&H0,&HA
       00,&H2A80,&H0,&HA37,&H2AB
       0
AA 920 DATA &HC00D,&HDADD,&H3AC0
       ,&H7077,&H7737,&H1D40,&HC
       0DD,&HA800
PM 930 DATA &H38,&H16,&H0,&H0,&H
       0,&H0,&H0,&H0
EG 940 DATA &H0,&H0,&H1400,&H0,&
       H0,&H0.&H55.&H0
OB 950 DATA &H0,&H55F1,&H4F,&H0,
       &HF303,&HCF3C,&HC0,&HF00
HN 960 DATA &HF30C,&HF0F0,&H0,&H
       F3C,&HF0AA,&H3C,&H3C00,&H
       820E
HN 970 DATA &H3CB0,&H0,&H23F,&HB
       028,&HFC,&HF00,&HFFC0,&HF
       003
BL 980 DATA &H0,&HFB03,&HBFBB,&H
       0,&H0,&HEE3E,&HEC,&H0
QJ 990 DATA &H300,&H80BB,&H0,&H0
       ,&HEE00,&H0,&H7700,&H0
PS 1000 DATA &HBB,&HDD00,&HC01D,
        &H5500,&H300,&H774,&HA24
        0,&H8AAA
LG 1010 DATA &HD001,&HE201,&HAAA
        A,&H8BAA,&H40,&HAA7A,&HA
        AAA,&HAD
DG 1020 DATA &H1A00,&H820A,&HA4A
        0,&H0,&H4,&H0,&H10,&HA00
        2
MJ 1030 DATA &H14,&HD,&HAB02,&H2
        A00,&H80BE,&HAFAA,&HAAA0
        ,&HA0AF
MM 1040 DATA &HAFAA,&HAAA0,&HA0A
        F,&HAE2A,&H2A80,&H80AA,&
        HAA0A,&H200
MH 1050 DATA &HAB,&HA000,&H0,&H4
        0,&H1,&H200,&HAB
OF 1060 DATA &H26,&H12,&H2020,&H
        20,&H0,&H2800,&H0,&H0
NF 1070 DATA &H2828,&H800,&H2028
        ,&HAB,&H2A00,&HA002,&H28
        ,&H8202
OH 1080 DATA &H2080,&H202,&HAB0,
        &H2000,&H0,&HB,&HA000,&H
        A088
DB 1090 DATA &HB,&H8028,&H2880,&
        H0,&H88A2,&HAA0,&H200,&H
        B08A
FG 1100 DATA &HA000,&H2800,&H0,&
        H2000,&H0,&H200,&H8000,&
        H0
NN 1110 DATA &H80A,&HA0,&H400,&H
        2000,&H0,&H10,&H0,&H0
JE 1120 DATA &HE,&HA,&H1,&HC00F,
        &HC00E,&H3,&HB03B,&HCCCE
5D 1130 DATA &H1,&H800A,&HA028,&
        H1450,&H30CF
```

"Balloon Crazy" for the TI-99/4A can be played with the keyboard or a joystick.

Program 5: TI-99/4A Balloon Crazy

Version by Patrick Parrish, Programming Supervisor

```
90 REM REQUIRES EXTENDED
    BASIC
100 GOTO 140
110 CALL DELSPRITE(#2)::
    CALL MOTION(#1,0,0,#3
    ,0,0):: RETURN
120 CALL KEY(0,K,ST):: IF
     ST=0 THEN CALL JOYST
    (1,H,V):: H=SGN(H)ELS
    E H=(K=83)-(K=68)
130 CALL MOTION(#1,0,60*H
    ):: RETURN
140 DIM DROP(2),KOLOR(2):
    : RANDOMIZE :: CALL M
    AGNIFY(4)
150 CALL CHAR(136,"030303
    030103070B0B0B07020202
    20F0480C080B00080C0A0
    90808080808080F020"):: 
    REM SKATEBOARD MAN
160 FOR I=96 TO 112 STEP
    8 :: CALL CHAR(I,"003
    87C7C7C381000"):: NEX
    T I :: LEVEL,SC,SC2=0
     :: MEN=3 :: ROW=41 :
    : KHAR=100
170 CALL CLEAR :: CALL SC
    REEN(16):: A$=RPT$("'
    hp",9):: FOR I=1 TO 2
    4 STEP 23 :: DISPLAY
    AT(I,1):A$ :: NEXT I
180 DISPLAY AT(10,8):"B A
     L L O O N" :: DISPLA
    Y AT(13,9):"C R A Z Y
     !" :: A=3 :: B=5 ::
    C=7
190 FOR I=1 TO 50 :: CALL
     COLOR(9,A,1,10,B,1,1
    1,C,1):: TEMP=A :: A=
    B :: B=C :: C=TEMP ::
     IF I=30 THEN CALL SP
    RITE(#1,136,14,150,1,
    0,31)
200 NEXT I :: CALL DELSPR
    ITE(#1):: CALL CLEAR
    :: GOSUB 490
210 DROP(0)=15 :: DROP(1)
    =20 :: DROP(2)=25
220 CALL CLEAR :: LEVEL=L
    EVEL+1 :: BALL=24 ::
    GOSUB 570
230 DISPLAY AT(1,6):"LEVE
    L:";LEVEL :: DISPLAY
    AT(1,17):"SCORE:";SC
240 FOR R=3 TO 6 :: FOR C
    =4 TO 29 STEP 5 :: CA
    LL HCHAR(R,C,96+INT(R
    ND*3)*8):: NEXT C ::
    NEXT R
250 CALL HCHAR(24,1,122,3
    2):: CALL SPRITE(#1,1
    36,14,150,115,0,H)
260 BALL=BALL-1 :: IF BAL
    L<0 THEN 410
270 BR=6 :: BC=4+INT(RND*
    6)*5
280 GOSUB 120 :: CALL GCH
    AR(BR,BC,BT):: IF BT=
    32 THEN BR=BR-1 :: IF
     BR=2 THEN 270 ELSE 2
    80
290 POINT=(BT-96)/8 :: CA
    LL HCHAR(BR,BC,32)::
    CALL SPRITE(#2,KHAR,K
    OLOR(POINT),ROW-(6-BR
    )*8,8*(BC-2)-2,DROP(P
    OINT),0)
300 GOSUB 120 :: CALL COI
    NC(#1,#2,15,C):: IF C
     THEN 340
310 CALL POSITION(#2,BROW
    ,BCOL):: IF BROW<155
    THEN 300
320 CALL POSITION(#1,MROW
    ,MCOL):: IF (BCOL-MCO
    L<16)*(BCOL-MCOL>-8)T
    HEN C=1 :: GOTO 340
330 GOSUB 110 :: MEN=MEN-
    1 :: CALL DELSPRITE(#
    3):: GOSUB 560 :: IF
    MEN=0 THEN 430 ELSE 4
    00
340 GOSUB 110 :: SC2=SC2+
    (POINT+1)*LEVEL*5 ::
    SC=SC+(POINT+1)*LEVEL
    *5
350 IF SC2>=1000 THEN MEN
    =MEN+1+(MEN=3):: SC2=
    0 :: GOSUB 570
360 IF C=0 THEN 400
370 CALL POSITION(#1,MROW
    ,MCOL):: CALL SPRITE(
    #3,100,14,118,MCOL)
380 FOR I=1 TO 50 :: NEXT
     I :: CALL SPRITE(#1,
    140,14,MROW,MCOL)
390 CALL SPRITE(#3,124,14
400 CALL HCHAR(1,3+MEN,32
    ):: DISPLAY AT(1,12):
    LEVEL;:: DISPLAY AT(1
    ,23):SC;:: GOTO 260
410 FOR G=300 TO 1200 STE
```

```
     P 100 :: CALL SOUND(8
     0,G,1):: NEXT G :: FO
     R I=0 TO 2 :: DROP(I)
     =DROP(I)+2 :: NEXT I
420  CALL DELSPRITE(ALL)::
      GOTO 220
430  CALL SCREEN(11):: IF
     SC>HS THEN HS=SC
440  CALL DELSPRITE(#1)::
     CALL CLEAR :: DISPLAY
      AT(8,5):"YOUR SCORE:
      ";SC :: DISPLAY AT(1
     1,5):"HIGH SCORE: ";H
     S
450  DISPLAY AT(16,5):"PLA
     Y AGAIN (Y/N)? " :: A
     CCEPT AT(16,24)BEEP V
     ALIDATE("NYny")SIZE(1
     ):REP$
460  IF REP$="N" THEN STOP
470  CALL SCREEN(16):: MEN
     =3 :: LEVEL,SC,SC2=0
     :: GOTO 210
480  REM SET COLORS
490  CALL COLOR(9,5,1,10,3
     ,1,11,7,1,12,13,1,13,
     14,1)
500  FOR J=0 TO 2 :: READ
     KOLOR(J):: NEXT J
510  DATA 5,3,7
520  CALL CHAR(100,"000000
     000000000000030707070
     301000000000000000000
     0080C0C0C0800000"):: 
     REM BALLOON
530  CALL CHAR(124,"000000
     000000000001000401040
     001000000000000000000
     0000400040000000"):: 
      REM BALLOON POPPING
540  CALL CHAR(140,"030303
     030103070B0B0B0702020
     20F0490D0909010A0C080
     8080808080080F20"):: R
     EM MAN POPPING BALLOO
     N
550  CALL CHAR(128,"383C38
     3810387CBA",122,"4949
     494949494949"):: RETU
     RN
560  FOR F=0 TO 25 STEP 5
     :: CALL SOUND(-200,-5
     ,F):: NEXT F :: RETUR
     N
570  CALL HCHAR(1,3,128,ME
     N):: RETURN
```

©

To receive additional information from advertisers in this issue, use the handy reader service cards in the back of the magazine.

MEMO Diary

Jim Butterfield, Associate Editor

Keep track of important dates, holidays, and personal events with this simple, easy to use BASIC program. It was originally written for Commodore computers (with at least 8K RAM and a tape or disk drive), and modifications are included for the Atari 400/800, XL, and XE (with at least 16K RAM for tape or 24K RAM for disk); Apple II series (disk only, DOS 3.3 or ProDOS); IBM PC and Enhanced Model PCjr (disk only); and TI-99/4A with Extended BASIC (disk or tape).

"Memo Diary" helps you record and recall birthdays, holidays, appointments, or any other event worth remembering. The program maintains a data file with as many as 100 events whose dates can range from tomorrow to one year in the future. You can record two different types of dates: temporary, one-time events such as appointments which have no importance once they have passed; and permanent, recurring events such as birthdays and anniversaries. By routinely running Memo Diary each time you use your computer, you'll no longer have to worry about forgetting to mail a birthday card to a relative or finding an anniversary gift for a spouse.

The program always shows the correct day of the week when you enter a date, and you need to enter the year only once—the very first time you run the program. After that (for the next 99 years, anyway) Memo Diary keeps track of the year for you. Each time you run the program, it automatically shows all due and overdue events on the screen or printer, and erases one-time events from the calendar after they're displayed.

You can enter temporary or recurring new events and erase existing events whenever you wish. You can also examine all events from the current date forward, or search the entire calendar for events matching a given starting pattern. Finally, Memo Diary saves your calendar either on disk or tape.

Typing The Program

We've listed Memo Diary in the form of one main program that contains common routines (Program 1), followed by line changes for each different computer. No matter which computer you're using, you'll need to type in Program 1 plus the modifications for your machine. However, before typing anything, cross out every line in Program 1 that has the *same* line number as a line in the listing for your specific computer. The idea is to eliminate duplicate lines from the main program; they're replaced

commodore

FOR THE BEST AND LATEST IN ATARI & COMMORDORE EQUIPMENT AT THE LOWEST PRICES!

1541 DISK DRIVE COMMODORE
$159

• Lets you store & retrieve programs & other information you create on your computer.

1571 DISK DRIVE Now Available ...... **CALL**
BLUE CHIP DISK DRIVE **$169**
ENHANCER 2000 DISK DRIVE **$159.95**

ATARI 1030 MODEM
74.95

• Modem link program • Free time on Dow Jones • Direct connect

ATARI 1050 DISK DRIVE
CALL

Gives you fast convient information.

STAR SG10 PRINTER
$205.95

BLUE CHIP M150/15 PRINTER **269.95**
STAR SG15 PRINTER **359.95**

VIDEO TAPES RIOT

TDK T120
SCOTCH T120
JVC T120
PANASONIC T120

YOUR CHOICE 4.39 EACH

MIN. PURCHASE 10

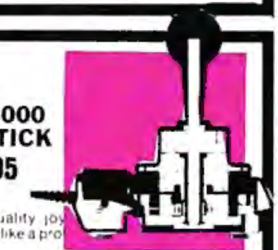
NASHUA DISKETTES

BOX OF 10 **8.95**

• Double density • Single sided

ELEPHANT DISKETTES
11.95

• Box of 10
• Double Density



PRO 5000 JOY STICK
11.95

Arcade quality joy stick scorelike a pro

WORLD TRADE
COMPUTER WHOLESALERS

MODEMS

Video 7	59.95
Mitey Mo (1 hour free compuserve time)	69.95
West Ridge	79.95

INTREFACES

Tymac Connection	69.95
G-Wiz	49.95
ApeFace	CALL
U-Print 16K	CALL
U-Print 64K	CALL
16K Centronics Buffer (For Use with Any Centronic Printer)	CALL

DISKETTES

Nashua (box of 10) DD/DS	12.95
Scotch (3M) (box of 10) DD/SS	10.95
Scotch Head Cleaner Kit	6.95

NIBBLE NOTCH

Turns your 5¼" Single Disk into A 2 Sided Disk **7.95**

DISK FILES

10 Disk Holder	6.00
60 Disk Holder (Stackable)	9.95

MONITORS

Sharp amber 12"	69.95

JOYSTICKS

Wico Boss	10.50
Wico Bat	18.00
Wico Threeway	21.00
Gim 1000-left/right firing button	9.95

PRINTERS

JUKI 6100 Daisy wheel letter quality	
Parallel	399.95
Serial	449.95
SG10-C w/built in Commodore Interface	215.00
Legende 880	239.95
Legende 1080	269.95
Legende 1380	300.95
Legende 1385	379.95

PAPER & RIBBON

Delivery on paper Included in price

	paper size
1000 x 3 boxes	9½-11"
Microperf	49.95
M120/M150	9.95
Ribbon SG10/SG15/ Okidata 82A	2.95
Ribbon 1101	19.95
Color Paper Yellow, Blue, Red 100 Each, 300 For only	12.95

Too Many Software Titles To List Call!

TO ORDER CALL
1-800-722-4002

Check, Money Order, MC or VISA accepted call for shipping & handling information. NYS residents add applicable sales tax. Prices and availability are subject to change without notice. Air factory fresh merchandise with manufacturer's warranty. Dealers welcome. Call for recent price reductions & new rebate information. No returns without authorization. For information & customer service call 1-718-895-2388

New Mindscape Inc.
THE HALLEY PROJECT
CALL
A mission to our solar system used with Atari & Commodore

NEW!
BIG 5 SOFTWARE
BOUNTY BOB STRIKES BACK
CALL
For use with both Atari & Commodore

by lines from the version for your computer. For example, if you're using an Atari, you would cross out line 150 in Program 1, because there's already a line 150 in the Atari listing (Program 3).

After crossing out duplicate lines in Program 1, type in the listing for your computer. Once that's done, type in every line of the Program 1 that's *not* crossed out. Be sure to save a copy of the program and read the instructions before running it.

The first time you run Memo Diary is special. *Do not start the program by entering RUN.* For every version except Atari you should type RUN 100 and press RETURN (or ENTER on the TI and IBM). Atari users should type CLR: GOTO 100 and press RETURN. *If you don't do this, the program will not work correctly.* When you start the program at line 100, Memo Diary lets you enter the correct year without looking for a previous file of events. Thereafter, start the program with RUN in the usual way.

On the first run you'll probably want to enter fixed holidays such as New Year's Day as well as birthdays and anniversaries. These are permanent events that you won't need to enter year after year. A holiday like Thanksgiving should be entered as a one-time event since it falls on a different date each year.

When Memo Diary asks you to enter today's date, you can type in the name of the month (such as OCTOBER) or its number (such as 10). In either case, be careful to enter it correctly. Memo Diary lets you enter any day of the month from 1 to 31, so it won't mind if you specify the date as February 30. Mistakes like these may confuse the calendar file. For instance, if you use the program on July 4 and the next day mistakenly give the date as June 5, the computer thinks you've let almost a whole year go by. To warn you of this, Memo Diary displays HAPPY NEW YEAR. If you see this message when a new year hasn't arrived, stop the program and start over, entering the correct date.

A Memory Jogger

Except for the very first run, Memo Diary always begins by reporting all due and overdue events ("You just missed your anniversary"). Take careful note of these events, since they'll soon be erased from the calendar (if they're temporary events) or moved ahead to next year (if they're permanent). To help jog your memory, Memo Diary also lets you make a copy of the list of events on your printer.

After disposing of due and overdue events, Memo Diary displays five options: You can see future events, add a new event, cancel an event, search for an event, or quit the program. You'll ordinarily want to look ahead to see what's coming in the next week or two. To do this, choose Option 1 (see future events) and supply an appropriate future date when requested. If you enter the current date when looking at future events, Memo Diary assumes you mean the same date *next year* and gives you everything on file.

When you want to make a new entry, select Option 2 (add new event). First Memo Diary asks whether the new event is one-time or permanent. Then it lets you enter the date and details. Again, the current date is understood as one year from today (it's assumed you don't need to record an event that's happening the same day).

To cancel an event (Option 3), you must know its date. When an event is entered, you're shown every item scheduled for that date, each with its own code number. To cancel an event, type in its code number when prompted.

Option 4 (search for event) lets you search for an event based on the first few letters of the entry. You may find many events in the course of a search. For instance, if the calendar file contains the events CLUB MEETING, CLUB CONFERENCE, and CLUB ELECTION, searching for CLUB displays all three events. In this case you would *not* see the entry CANADIAN CLUB, since CLUB is spotted only if it's in the first word of the entry. Thus, if you plan to search for certain keywords (BIRTHDAY, CHURCH, SOFTBALL, or whatever) keep them at the front of each calendar entry.

After you've finished an option, Memo Diary always returns you to the main menu. Sooner or later you'll be ready to use Option 5 (quit). The program knows when it's time to update the calendar file. If you've erased past and overdue events, added or deleted items, Memo Diary will—with your permission—proceed to update the data file on disk or tape.

The Time Pivot

A program that handles dates can encounter some subtle paradoxes. Does August come before April, or after it? The correct answer is *both*. Memo Diary could resolve this difficulty by adding a year designation to every event, but that complicates the handling of permanent events, which don't belong to a specific year. This is not a trivial problem: If you schedule a new event for August, the program must decide whether to add the event to the calendar ahead of an existing April event, or after it. Without a year designation, how can anyone tell?

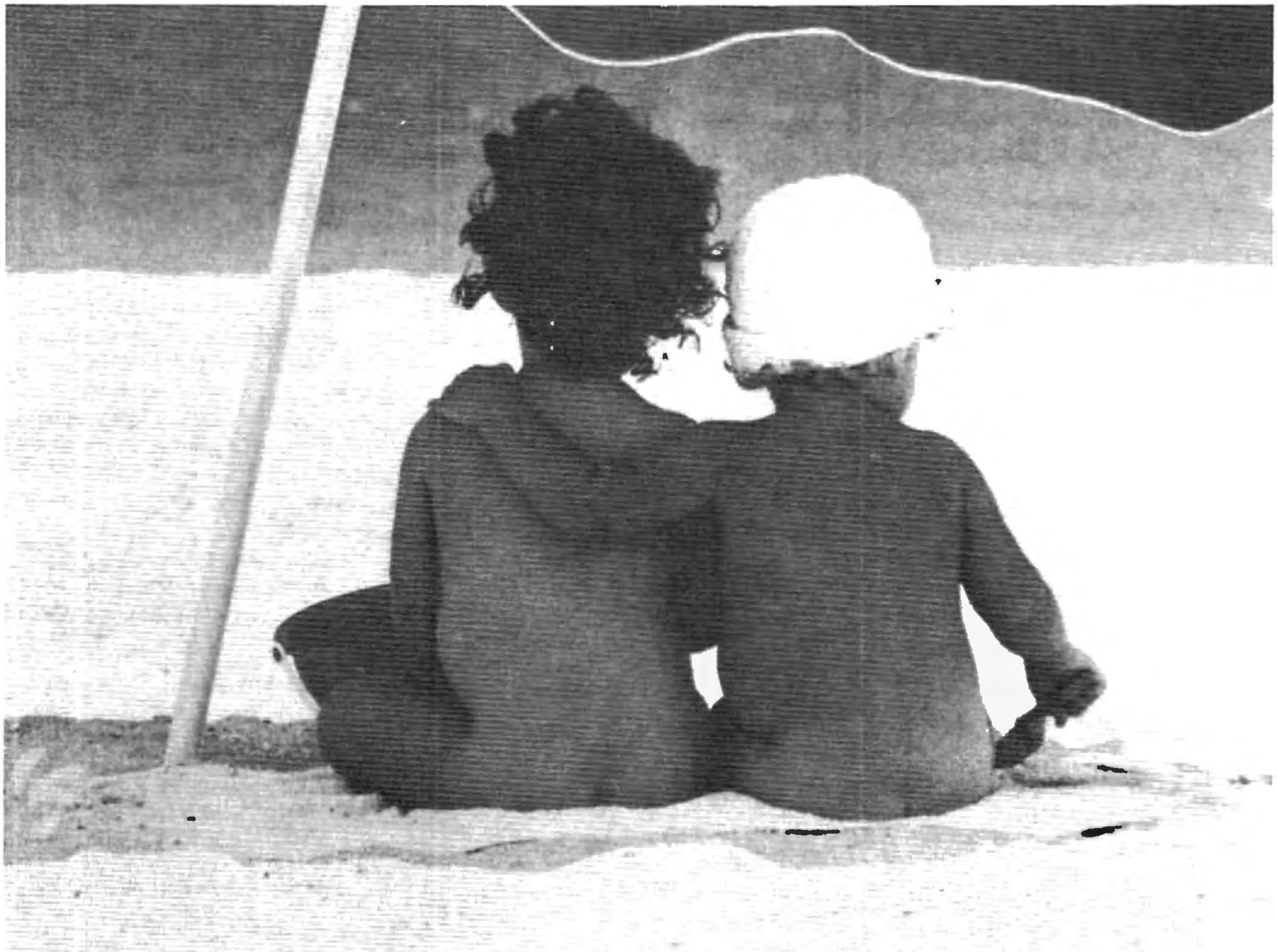
The problem is solved by using a *pivot* date, usually the same as the current date. If today is July 4, August does indeed come before April. On the other hand, if today is November 11, April comes before August. Since the calendar always looks one year into the future, everything is kept in order.

However, there's one case in which the pivot date can't be the current date. Each time the program begins, it must measure the time lapse since its last use. For example, say that you last used the program on August 20, 1985 and next use it on September 4, 1985. On the first run (August 20) Memo Diary uses August 20 as the pivot. That way an event dated September 1 is seen ahead of another item dated in October.

On the second run (September 4) the September 1 event is reported as past due and either erased from the calendar (if it's temporary) or moved ahead to September 1 of next year (if it's permanent). Once this is done, the pivot date moves forward to September 4, meaning that a September 1 event now belongs *after* an item dated in October. Don't worry if this sounds confusing: It works out more simply in practice than in theory.

The day of the week is worked out with a simple formula. If you haven't seen it before, here's a hint on how it works. The calendar is

The Source℠ Is Friendly.



Many online information services claim to be "user friendly." But only one really lives up to that promise.

The Source.

You see, The Source is specifically designed to save you time online. With new, shorter menus. Simpler commands. And a user's manual so well-written and easy to understand, it's setting an industry standard.

The Source is also the only service that gives you introductory lessons and assistance, free of on-line charges. So you get up-to-speed on our dime, not yours.

You get to the information you need in record time, without frustration. Everything from the hour's headlines to travel reservations. From special interest groups to online stock trading. So you don't waste your valuable time. Or money.

Call 1-800-336-3366, send the coupon, or visit your nearest computer dealer.

And make friends with America's friendliest online information network.

INFORMATION NETWORK
The Source℠
The most powerful resource any personal computer can have.

I'd like to make friends with The Source.
Please send more information to:

Name ______________________ 7826J18

Address ______________________

City, State ______________ Zip ________

Mail to: Source Telecomputing Corporation, 1616 Anderson Road, McLean, VA 22102

---

modified to make March 1 the first day of the "adjusted year." This way, leap year with its extra February 29 date doesn't break up the sequence of days: The extra leap day just gets pasted onto the year's end. Though the math is a bit convoluted, you may find it interesting to trace the logic of this routine (it starts at line 2150).

Expanding The Calendar

Memo Diary can keep track of a maximum of 100 events. In practice it's wise to limit the number to 80 or 90 to leave room for permanent events that move automatically from the front to the back of the list. If you need more than 100 events, change the L$ value in the DIM statement. Line 150 contains the value L$(100). You can increase the 100 to whatever number you like, but don't get carried away. Since Memo Diary (except the Atari version) uses string arrays, a very large value may cause garbage collection delays. There's no particular limit to the number of events allowed for a particular date.

Program Notes

Let's take a look at the program's major features. Line 90 prepares Memo Diary to read a file. The variable F is a *Boolean* (logical) variable that's defined as *true* here, to let you read the calendar file on a normal run. When you enter at line 100 on the first run, F is *false* (like every other undefined variable) and no file is read.

DATA statements in lines 110–140 hold the names of the months of the year and days of the week; the names are read into the arrays M$ and W$. Line 150 dimensions the L$ array for 100 items. Lines 230–250 call for a reading of the calendar file if appropriate. This is done in the subroutine at line 3010. When Memo Diary reads this file, it detects and reports the last date the file was used. Line 260 asks for today's date; the subroutine at line 1670 asks for and accepts the date.

Now it's time to search for due and overdue events. Using the previous date as a pivot, the subroutine at line 1960 scans for all events up to today's date. The program reports these events, erases them, or moves them ahead as needed, and proceeds to the main menu. Line 680 begins a main activity loop: It prompts with the menu, asks for a choice, then goes to the appropriate subroutine. Line 850 lets you see future events. Since the pivot date is now today, the program scans to the requested future date to see how many events fall into the today-to-future-date range.

Line 940 lets you add a new event. After asking ANNUAL OR ONE-TIME? the program requests the event's date and then asks for details. After adding a year designation to the date of one-time events, the new event is inserted into the proper sequence. Line 1210 lets you cancel an event. Memo Diary asks for a date and then lists all events that match that date. At line 1350, the program asks which event to delete. Note that the number you supply must be in the correct range.

Line 1450 begins the search-for-an-event routine. After it receives a search string (P$), the program looks for a match. When it scans through the calendar, it must look in different places depending on whether the event is one-time or permanent. That's because one-time events carry a year designation, making their dates three characters longer.

A Horrible Mistake?

Line 1570 handles the quit option; the flag F9 registers activity. If you haven't changed any of the data, there's no need to update the calendar file. Before scratching the old file and writing the new one, the program asks whether you're ready. That way, if you made some horrible mistake, you can cancel the file update.

The main loop ends at line 1580 and is followed by several subroutines. The routine starting at line 1590 writes a new calendar file when appropriate, and line 1670 begins the date input routine. The date is formed into a string (D8$) to allow for easy searches or entry. The subroutine at line 1930 reads the calendar file. The first item in the file is always the most recent date of use; the remaining data is events.

The subroutine at line 1960 scans all events to see which have dates between the pivot date (D9$) and a second date (D8$). There are three dates involved: event, pivot, and the second date, which makes the comparison a bit messy. Boolean variables keep everything in order. Eventually, the variable F0 indicates the date is in range, and the variable L0 indicates when the last event is found within the date range.

The routine starting at line 4020 displays the information, on the printer if desired. (TI users should change line 4070 to match their printer configuration.) The date is given complete with the day of the week, and events falling on the same day are grouped together. The weekday calculation begins at line 2150. The weekday variable, W, ranges from 0 to 6, so 0 means Sunday. As written, this routine is good for years ranging from 85 (1985) to 84 (2084). If you want to plan more than 99 years in advance, you'll need to modify the routine.

Program 1: Memo Diary Main Program

Please refer to instructions in the article before entering this listing

```
90 F=(1=1)
100 GOSUB 2250
110 DATA JAN,FEB,MAR,APR,MAY,J
    UN
120 DATA JUL,AUG,SEP,OCT,NOV,D
    EC
130 DATA SUNDAY,MONDAY,TUESDAY
    ,WEDNESDAY
140 DATA THURSDAY,FRIDAY,SATUR
    DAY
150 DIM M$(12),W$(6),L$(100)
160 FOR J=1 TO 12
170 READ M$(J)
180 NEXT J
190 FOR J=0 TO 6
200 READ W$(J)
210 NEXT J
220 PRINT "EVENT CALENDAR"
230 IF F=0 THEN 260
240 C=1
250 GOSUB 3010
260 PRINT "TODAY'S DATE:"
270 Y8=Y9
280 GOSUB 1670
290 M8=M
300 D8=D
310 IF M8>=M9 THEN 330
320 Y8=Y9+1
330 IF M8<>M9 OR D8>=D9 THEN 3
    50
340 Y8=Y9+1
350 IF Y8<=Y9 THEN 370
360 PRINT "HAPPY NEW YEAR"
370 IF F THEN 400
380 PRINT "YEAR";
390 INPUT Y8
400 D9$=RIGHT$(STR$(100+M9),2)
    +"/"
410 D9$=D9$+RIGHT$(STR$(100+D9
    ),2)
```

```
420 IF F THEN 440
430 D9$=D8$
440 F=(1=1)
450 GOSUB 1960
460 PRINT "PAST EVENTS: ";
470 IF L0>=0 THEN 500
480 PRINT "NONE"
490 GOTO 650
500 PRINT L0+1
510 GOSUB 4010
520 F9=-1
530 FOR J=0 TO L0
540 IF MID$(L$(J),6,1)="/" THEN 570
550 L$(L9)=L$(J)
560 L9=L9+1
570 NEXT J
580 L8=L0+1
590 FOR J=L8 TO L9-1
600 L$(J-L8)=L$(J)
610 NEXT J
620 L9=L9-L8
630 L8=0
640 L=L9
650 F=0
660 F9=0
670 D9$=D8$
680 L=L9-L8
690 IF L<>0 THEN 710
700 PRINT "NO FUTURE EVENTS"
710 IF L=0 THEN 730
720 PRINT L;" FUTURE EVENTS"
730 PRINT
740 PRINT "1. SEE FUTURE EVENTS"
750 PRINT "2. ADD NEW EVENT"
760 PRINT "3. CANCEL EVENT"
770 PRINT "4. SEARCH FOR EVENT"
780 PRINT "5. QUIT"
790 PRINT
800 PRINT "...YOUR CHOICE (1-5)";
810 INPUT A
820 PRINT
830 ON A GOTO 850,940,1210,1450,1570
840 GOTO 730
850 PRINT "AHEAD TO DATE:"
855 FL=1
860 GOSUB 1670
865 FL=0
870 GOSUB 1960
875 IF D8$=D9$ THEN L0=L9-1
880 IF L0<>-1 THEN 910
890 PRINT "NO EVENTS"
900 GOTO 920
910 GOSUB 4010
920 PRINT L9-L0-1;" OTHER FUTURE EVENTS"
930 GOTO 730
940 PRINT "ANNUAL OR ONE-TIME{SPACE}(A/O)";
950 INPUT P$
960 A=0
970 P$=LEFT$(P$,1)
980 IF P$="O" THEN 1010
990 A=1
1000 IF P$<>"A" THEN 730
1010 GOSUB 1670
1020 Y$="/"+RIGHT$(STR$(101+Y8),2)
1050 IF A<>1 THEN 1070
1060 Y$=""
1070 GOSUB 1960
1080 IF L9-1<L0+1 THEN 1120
1090 FOR J=L9-1 TO L0+1 STEP -1
1100 L$(J+1)=L$(J)
1110 NEXT J
1120 PRINT "DETAIL";
1130 INPUT LL$
1140 D8$=D8$+Y$
1150 D8$=D8$+" "
1160 L$(L0+1)=D8$+LL$
1170 L9=L9+1
1180 L=L9
1190 F9=-1
1200 GOTO 680
1210 PRINT "CHANGE WHICH DATE: "
1220 GOSUB 1670
1230 L0=-1
1240 FOR J=L8 TO L9-1
1250 IF D8$<>LEFT$(L$(J),5) THEN 1300
1260 L1=J
1270 IF L0<>-1 THEN 1290
1280 L0=J
1290 PRINT J;": ";L$(J)
1300 NEXT J
1310 IF L0<>-1 THEN 1340
1320 PRINT "NO EVENTS"
1330 GOTO 730
1340 PRINT
1350 PRINT " DELETE WHICH EVENT ABOVE";
1360 INPUT A
1370 IF A<L0 OR A>L1 THEN 730
1380 FOR J=A TO L9-1
1390 L$(J)=L$(J+1)
1400 NEXT J
1410 L9=L9-1
1420 F9=-1
1430 PRINT " ... DELETED"
1440 GOTO 680
1450 PRINT "SEARCH FOR";
1460 INPUT P$
1470 P=LEN(P$)
1480 FOR J=0 TO L9-1
1490 A=7
1500 IF MID$(L$(J),6,1)<>"/" THEN 1520
1510 A=10
1520 IF A+P-1>LEN(L$(J)) OR P$<>MID$(L$(J),A,P) THEN 1540
1530 PRINT L$(J)
1540 NEXT J
1550 PRINT "{4 SPACES}END OF SEARCH"
1560 GOTO 730
1570 IF F9<>0 THEN 1590
1580 END
1590 PRINT "READY TO WRITE NEW EVENTS FILE (Y/N)";
1600 INPUT P$
1610 IF LEFT$(P$,1)="Y" THEN 1630
1620 STOP
1630 D9$=D9$+"/"
1640 D9$=D9$+RIGHT$(STR$(Y8+100),2)
1650 C=2
1660 GOTO 3010
1670 M=0
1680 PRINT "MONTH";
1690 INPUT MM$
1700 M=VAL(MM$)
1710 MM$=LEFT$(MM$+"XX",3)
1720 IF M=0 THEN 1760
1730 IF M<1 OR M>12 THEN 1670
1740 PRINT M$(M)
1750 GOTO 1810
1760 FOR J=1 TO 12
1770 IF MM$<>M$(J) THEN 1790
1780 M=J
1790 NEXT J
1800 IF M<1 OR M>12 THEN 1670
1810 PRINT "DAY";
1820 INPUT D
1830 IF D<1 OR D>31 THEN 1670
1840 D8$=RIGHT$(STR$(100+M),2)+"/"
1850 D8$=D8$+RIGHT$(STR$(100+D),2)
1860 Y=Y8
1865 IFD8$=D9$ANDFL=1THEN1880
1870 IF D8$>=LEFT$(D9$,5) THEN 1890
1880 Y=Y8+1
1890 GOSUB 2150
1900 IF LEN(LL$)<=0 THEN 1920
1910 PRINT "(";W$(W);")"
1920 RETURN
1930 C=1
1940 GOSUB 3010
1950 RETURN
1960 LL$=CHR$(255)
1970 L0=-1
1980 IF L<>0 THEN 2000
1990 RETURN
2000 V$=D8$+LL$
2010 WW$=D9$
2030 WW$=D9$+LL$
2040 F1=(WW$>V$)
2050 FOR J=L8 TO L9-1
2060 F2=(L$(J)>WW$)
2070 F3=(V$>L$(J))
2080 F0=F2 AND F3
2090 IF F1=0 THEN 2110
2100 F0=F2 OR F3
2110 IF F0=0 THEN 2130
2120 L0=J
2130 NEXT J
2140 RETURN
2150 IF Y>=85 THEN 2170
2160 Y=Y+100
2170 M1=M+1
2180 M2=INT(1/M1+.7)
2190 M3=Y-M2
2200 M4=M1+12*M2
2210 N=INT(M4*30.6001)+INT(M3*365.25)+D
2220 M6=INT(N/7)
2230 W=N-7*M6
2240 RETURN
2250 PRINT CHR$(147)
2260 RETURN
3000 REM INPUT/OUTPUT ROUTINE
4000 REM PRINT ROUTINE
```

Program 2: Modifications For Commodore

For instructions on entering this listing please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
255 IF E=0 THEN 260                         :rem 164
256 F=0                                     :rem 80
1575 IFOTHENCLOSE15                         :rem 187
3010 F$="EVENTS"                            :rem 132
3020 PRINT"DISK OR CASSETTE (D/C)?"         :rem 4
3030 GETA$:IF((A$<>"C")AND(A$<>"D"))ORA$=""THEN3030
                                            :rem 227
3040 IFA$="D"THEN3060                       :rem 120
3050 D1=0:G$="":GOTO3070                    :rem 13
3060 F$="@0:"+F$:D1=1                       :rem 16
3070 IFC=2THEN3160                          :rem 4
3080 IFD1=1THENG$=",S,R"                    :rem 85
3090 OPEN1,1+7*D1,8*D1,F$+G$:GOSUB3220:IFETHENCLOSE1:GOTO3150
                                            :rem 93
3100 INPUT#1,LL$:D9$=LL$:IF LEN(LL$)<>8 THEN PRINT LL$;"?":GOTO 3140
                                            :rem 60
3110 M=VAL(LEFT$(LL$,2)):D=VAL(MID$(LL$,4,2)):Y0=VAL(MID$(LL$,7,2))
                                            :rem 245
3120 M9=M:D9=D:Y9=Y0:L=0:PRINT "LAST ACCESS: ";LL$
                                            :rem 181
3130 INPUT#1,L$(L):L=L+1:IF ST
```

```
      =0 THEN 3130          :rem 34
3140  CLOSE1:GOSUB3220  :rem 240
3150  L8=0:L9=L:RETURN   :rem 28
3160  IFD1=1THENG$=",S,W"
                            :rem 89
3170  OPEN1,1+7*D1,8*D1,F$+G$:G
      OSUB3220:IFETHENCLOSE1:CL
      OSE15:END            :rem 71
3180  PRINT#1,D9$;CHR$(13);
                           :rem 166
3190  FORJ=0TOL9-1:PRINT#1,L$(J
      );CHR$(13);:NEXTJ  :rem 50
3200  GOSUB3220:CLOSE1:GOSUB322
      0:IFOTHENCLOSE15  :rem 145
3210  END                  :rem 157
3220  IFD1=0THENRETURN    :rem 71
3230  IFO=0THENOPEN15,8,15:O=1
                           :rem 199
3240  INPUT#15,E,B$:IFETHENPRIN
      TB$:CLOSE15:O=0     :rem 33
3250  RETURN               :rem 170
4010  D$="":P=3            :rem 168
4020  INPUT "WANT EVENTS ON PRI
      NTER (Y/N)";P$      :rem 64
4030  IF LEFT$(P$,1)<>"Y" THEN
      {SPACE}4050         :rem 214
4040  P=4                  :rem 137
4050  OPEN 3,P             :rem 170
4060  FOR J=L8 TO L0      :rem 219
4070  IF D$=LEFT$(L$(J),5) THEN
       4150                  :rem 4
4080  D$=LEFT$(L$(J),5):rem 125
4090  M=VAL(LEFT$(D$,2))
                           :rem 241
4100  D=VAL(MID$(D$,4,2))
                           :rem 239
4110  Y=Y8:IF D$<=D9$ THEN Y=Y8
      +1                  :rem 234
4120  GOSUB 2150            :rem 15
4130  PRINT#3,W$(W);" ";
                           :rem 180
4140  PRINT#3,M$(M);D   :rem 102
4150  PRINT#3,"{3 SPACES}";MID$
      (L$(J),6)           :rem 20
4160  NEXT J                :rem 84
4170  CLOSE 3              :rem 117
4180  RETURN               :rem 173
```

Program 3: Modifications for Atari

For instructions on entering this listing please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
FG 150 DIM A$(10),DB$(9),D9$
       (9),M$(36),W$(63),L$(
       100*30),LL$(30),TE$(4
       ),MM$(10),P$(30),Y$(1
       0),V$(30),WW$(10)
EG 155 DIM D$(30),DE$(3),FN$
       (15):W$=" ":W$(63)=W$
       :W$(2)=W$:L$=" ":L$(3
       000)=L$:L$(2)=L$
DN 170 READ A$:M$((J-1)*3+1,
       J*3)=A$
EK 200 READ A$:W$(J*9+1,(J+1
       )*9)=A$
CP 400 TE$="":TE$=STR$(100+M
       9):D9$(1,2)=TE$(LEN(T
       E$)-1,LEN(TE$)):D9$(3
       )="/"
BO 410 TE$="":TE$=STR$(100+D
       9):D9$(4,5)=TE$(LEN(T
       E$)-1,LEN(TE$))
AA 540 IF L$(J*30+6,J*30+6)=
       "/" THEN 570
JF 550 L$(L9*30+1,(L9+1)*30)
       =L$(J*30+1,(J+1)*30)
MO 600 L$((J-L8)*30+1,(J-L8+
       1)*30)=L$(J*30+1,(J+1
       )*30)
KE 970 P$=P$(1,1)
CI 1020 Y$="/":TE$=STR$(101+
        Y8):Y$(2,3)=TE$(LEN(
        TE$)-1,LEN(TE$))
AI 1040 Y$(1,1)="/":TE$=STR$
        (100+Y8):Y$(2,3)=TE$
        (LEN(TE$)-1,LEN(TE$)
        )
KG 1060 Y$(1,3)="{3 SPACES}"
PF 1100 L$((J+1)*30+1,(J+2)*
        30)=L$(J*30+1,(J+1)*
        30)
FK 1130 INPUT LL$:IF LEN(LL$
        )>20 THEN LL$=LL$(1,
        20)
HG 1135 IF LEN(LL$)<20 THEN
        LL$(LEN(LL$)+1,20)="
        {20 SPACES}"
AL 1140 DB$(6,8)=Y$
HC 1150 DB$(9)=" "
GM 1160 L$((L0+1)*30+1,(L0+1
        )*30+9)=DB$:L$((L0+1
        )*30+10,(L0+1)*30+29
        )=LL$
LL 1250 IF DB$<>L$(J*30+1,J*
        30+5) THEN 1300
DL 1290 PRINT J;": ";L$(J*30
        +1,(J+1)*30)
AA 1390 L$(J*30+1,(J+1)*30)=
        L$((J+1)*30+1,(J+2)*
        30)
LC 1490 REM
KK 1500 REM
IK 1520 IF P$<>L$(J*30+A,J*3
        0+A+P-1) THEN 1540
PK 1530 PRINT L$(J*30+1,(J+1
        )*30)
GL 1610 IF P$(1,1)="Y" OR P$
        (1,1)="y" THEN 1630
PC 1630 D9$(LEN(D9$)+1,LEN(D
        9$)+1)="/"
OA 1640 P$=STR$(Y8+100):P=LE
        N(P$):P$=P$(P-1,P):D
        9$(LEN(D9$)+1,LEN(D9
        $)+2)=P$
JO 1710 MM$(LEN(MM$)+1,LEN(M
        M$)+2)="XX":MM$=MM$(
        1,3)
KG 1740 PRINT M$((M-1)*3+1,M
        *3)
CP 1770 IF MM$<>M$(J*3,J*3+2
        ) THEN 1790
MA 1840 DB$=STR$(100+M):DB$=
        DB$(LEN(DB$)-1,LEN(D
        B$)):DB$(3,3)="/"
HB 1850 P$=STR$(100+D):DB$(L
        EN(DB$)+1,LEN(DB$)+2
        )=P$(LEN(P$)-1,LEN(P
        $))
PP 1870 IF DB$>=D9$(1,5) THE
        N 1890
BM 1910 PRINT "(";W$(W*9+1,(
        W+1)*9);")"
AK 2000 V$=DB$:V$(LEN(V$)+1,
        LEN(V$)+LEN(LL$))=LL
        $
GO 2030 WW$=D9$:WW$(LEN(WW$)
        +1,LEN(WW$)+LEN(LL$)
        )=LL$
IC 2060 F2=(L$(J*30+1,(J+1)*
        30)>WW$)
CM 2070 F3=(V$>L$(J*30+1,(J+
        1)*30))
EB 2800 PRINT CHR$(125)
FC 3010 POKE 195,0:PRINT "
        {CLEAR}ENTER DEVICE
        AND FILENAME":PRINT
        "(i.e., D:EVENTS.DAT)
        :":INPUT FN$
DI 3020 TRAP 3070:IF C=2 THE
        N 3050
NG 3030 OPEN #1,4,0,FN$:INPU
        T #1;LL$:D9$=LL$:M=V
        AL(LL$(1,2)):D=VAL(L
        L$(4,5)):Y0=VAL(LL$(
        7,8))
LG 3040 M9=M:D9=D:Y9=Y0:L=0:
        PRINT "LAST ACCESS:
        ";LL$
OP 3042 INPUT #1;NE:IF NE TH
        EN FOR L=0 TO NE-1:I
        NPUT #1;LL$:L$(L*30+
        1,(L+1)*30)=LL$:NEXT
        L
IH 3044 TRAP 40000:CLOSE #1:
        L8=0:L9=L:RETURN
GB 3050 NE=L9:OPEN #1,8,0,FN
        $:PRINT #1;D9$:PRINT
        #1;NE
OF 3060 IF NE THEN FOR A=0 T
        O NE-1:PRINT #1;L$(A
        *30+1,(A+1)*30):NEXT
        A
KJ 3065 CLOSE #1:END
KC 3070 POKE 849,1:CLOSE #1:
        TRAP 40000:IF PEEK(1
        95)=0 THEN 3010
DL 3080 PRINT :PRINT CHR$(25
        3);"* ERROR ";PEEK(1
        95);" *":CLOSE #1
EI 3090 IF PEEK(764)<255 THE
        N POKE 764,255:GOTO
        3010
MJ 3100 GOTO 3090
JF 4010 D$="":DE$="E:"
MJ 4020 PRINT "WANT EVENTS O
        N PRINTER (Y/N)";
ML 4030 INPUT P$
LK 4040 IF P$(1,1)<>"Y" THEN
         4060
IB 4050 DE$="P:"
OJ 4060 OPEN #1,8,0,DE$
NM 4070 FOR J=L8 TO L0
FA 4080 IF D$=L$(J*30+1,J*30
        +5) THEN 4150
MJ 4090 D$=L$(J*30+1,J*30+5)
NI 4095 M=VAL(D$(1,2))
MI 4100 D=VAL(D$(4,5))
OK 4110 Y=Y8:IF D$<=D9$ THEN
         Y=Y8+1
AP 4120 GOSUB 2150
BD 4130 PRINT #1;W$(W*9+1,(W
        +1)*9);" ";
DA 4140 PRINT #1;M$((M-1)*3+
        1,M*3);" ";D
AO 4150 PRINT #1;"
        {3 SPACES}";L$(J*30+
        6,(J+1)*30)
FE 4160 NEXT J
JG 4170 CLOSE #1
KN 4180 RETURN
```

Program 4: Modifications For Apple

For instructions on entering this listing please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!

```
84 105  DD$ = CHR$ (4):I$ = CHR$
        (9)
51 2250 HOME
94 3010 F$ = "EVENTS"
76 3020 PRINT DD$;"OPEN ";F$
F8 3030 IF C = 2 THEN 3080
3C 3040 PRINT DD$;"READ ";F$: IN
      PUT LL$:D9$ = LL$: IF LE
        N (LL$) < > 8 THEN PRINT
        LL$;"?": GOTO 3080
FA 3050 M = VAL ( LEFT$ (LL$,2))
        :D = VAL ( MID$ (LL$,4,2
        )):Y0 = VAL ( MID$ (LL$,
        7,2)):M9 = M:D9 = D:Y9 =
        Y0:L = 0: PRINT "LAST A
      CCESS: ";LL$
DA 3060 INPUT L$(L): IF L$(L) <
        > "EOF" THEN L = L + 1:
```

```
        GOTO 3060
76 3070 L$(L) = "":L8 = 0:L9 = L
        : GOTO 3090
42 3080 PRINT DD$;"WRITE ";F$: P
        RINT D9$: FOR J = 0 TO L
        9 - 1: PRINT L$(J): NEXT
        J: PRINT "EOF"
6A 3090 PRINT DD$;"CLOSE ";F$: I
        F C = 2 THEN END
D5 3100 RETURN
F8 4010 PRINT :D$ = "": INPUT "W
        ANT EVENTS ON PRINTER (Y
        /N) ";P$: IF LEFT$ (P$,1
        ) < > "Y" THEN 4030
E4 4020 PRINT DD$;"PR#1": PRINT
        I$;"80N"
6D 4030 FOR J = L8 TO L0: IF D$
        = LEFT$ (L$(J),5) THEN 4
        060
8A 4040 D$ = LEFT$ (L$(J),5):M =
        VAL ( LEFT$ (D$,2)):D =
        VAL ( MID$ (D$,4,2)):Y
        = Y8: IF D$ < = D9$ THEN
        Y = Y8 + 1
0A 4050 GOSUB 2150: PRINT W$(W);
        " ";M$(M);" ";D
B9 4060 PRINT "     "; MID$ (L$(J)
        ,6): NEXT J
D6 4070 PRINT : IF LEFT$ (P$,1)
        = "Y" THEN PRINT DD$;"PR
        #0"
F4 4080 RETURN
```

Program 5: Modifications For IBM PC/PCjr

For instructions on entering this listing please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
KL 105 WIDTH 80:KEY OFF:DEF SEG=
       0:POKE 1047,PEEK(1047) OR
       64
ND 2250 CLS
FD 3010 ON ERROR GOTO 3100
MA 3020 F$="EVENTS":INPUT "ENTER
        DRIVE # (IE., A): ";FF$
        :F$=FF$+":"+F$
EE 3030 IF C=2 THEN 3080
OK 3040 OPEN F$ FOR INPUT AS #1:
        INPUT #1,LL$:D9$=LL$:IF
        LEN(LL$)<>8 THEN PRINT L
        L$;"?":GOTO 3070
6E 3050 M=VAL(LEFT$(LL$,2)):D=VA
        L(MID$(LL$,4,2)):Y0=VAL(
        MID$(LL$,7,2)):M9=M:D9=D
        :Y9=Y0:L=0:PRINT "LAST A
        CCESS: ";LL$
FN 3060 INPUT #1,L$(L):L=L+1:IF
        EOF(1)=0 THEN 3060
PA 3070 CLOSE #1:ON ERROR GOTO 0
        :L8=0:L9=L:RETURN
CG 3080 OPEN F$ FOR OUTPUT AS #1
        :PRINT #1,D9$:FOR J=0 TO
        L9-1:PRINT #1,L$(J)
NB 3090 NEXT J:CLOSE #1:ON ERROR
        GOTO 0:END
EC 3100 CLOSE #1:PRINT "DISK ERR
        OR #";ERR;"OCCURRED.":PR
        INT "TRY AGAIN."
HH 3110 PRINT:PRINT "HIT A KEY T
        O CONTINUE"
FK 3120 A$=INKEY$:IF A$="" THEN
        3120
JA 3130 RESUME 3020
CJ 4010 ON ERROR GOTO 4090
BA 4020 D$="":INPUT "WANT EVENTS
        ON PRINTER (Y/N)";P$
PO 4030 IF LEFT$(P$,1)="Y" THEN
        OPEN "LPT1:" FOR OUTPUT
        AS #1 ELSE OPEN "SCRN:"
        FOR OUTPUT AS #1
CH 4040 FOR J=L8 TO L0:IF D$=LEF
        T$(L$(J),5) THEN 4080
BP 4050 D$=LEFT$(L$(J),5):M=VAL(
        LEFT$(D$,2)):D=VAL(MID$(
        D$,4,2))
JJ 4060 Y=Y8:IF D$<=D9$ THEN Y=Y
        8+1
GL 4070 GOSUB 2150:PRINT #1,W$(W
        );" ";:PRINT #1,M$(M);D
PL 4080 PRINT #1,"     ";MID$(L$(J
        ),6):NEXT J:CLOSE #1:ON
        ERROR GOTO 0:RETURN
IC 4090 CLOSE #1:PRINT "PRINTER
        ERROR #";ERR;"OCCURRED."
        :PRINT "TRY AGAIN."
HF 4100 PRINT:PRINT "HIT A KEY T
        O CONTINUE"
E6 4110 A$=INKEY$:IF A$="" THEN
        4110
KL 4120 RESUME 4020
```

Program 6: Modifications For TI-99/4A

```
330 IF (M8<>M9)+(D8>=D9)THE
    N 350
400 TE$=STR$(100+M9)
405 D9$=SEG$(TE$,LEN(TE$)-1
    ,2)&"/"
410 TE$=STR$(100+D9)
415 D9$=D9$&SEG$(TE$,LEN(TE
    $)-1,2)
540 IF SEG$(L$(J),6,1)="/"
    THEN 570
875 IF D8$<>D9$ THEN 880
876 L0=L9-1
970 P$=SEG$(P$,1,1)
1020 TE$=STR$(101+Y8)
1025 Y$="/"&SEG$(TE$,LEN(TE
     $)-1,2)
1040 TE$=STR$(100+Y8)
1045 Y$="/"&SEG$(TE$,LEN(TE
     $)-1,2)
1140 D8$=D8$&Y$
1150 D8$=D8$&" "
1160 L$(L0+1)=D8$&LL$
1250 IF D8$<>SEG$(L$(J),1,5
     )THEN 1300
1370 IF (A<L0)+(A>L1)THEN 7
     30
1500 IF SEG$(L$(J),6,1)<>"/
     " THEN 1520
1520 IF (A+P-1>LEN(L$(J)))+
     (P$<>SEG$(L$(J),A,P))T
     HEN 1540
1610 IF SEG$(P$,1,1)="Y" TH
     EN 1630
1630 D9$=D9$&"/"
1640 TE$=STR$(Y8+100)
1645 D9$=D9$&SEG$(TE$,LEN(T
     E$)-1,2)
1710 MM$=SEG$(MM$&"XX",1,3)
1730 IF (M<1)+(M>12)THEN 16
     70
1800 IF (M<1)+(M>12)THEN 16
     70
1830 IF (D<1)+(D>31)THEN 16
     70
1840 TE$=STR$(100+M)
1845 D8$=SEG$(TE$,LEN(TE$)-
     1,2)&"/"
1850 TE$=STR$(100+D)
1855 D8$=D8$&SEG$(TE$,LEN(T
     E$)-1,2)
1865 IF (D8$=D9$)*(FL=1)THE
     N 1880
1870 IF D8$>=SEG$(D9$,1,5)T
     HEN 1890
1960 LL$=CHR$(127)
2000 V$=D8$&LL$
2030 WW$=D9$&LL$
2080 F0=F2*F3
2100 F0=F2+F3
2250 CALL CLEAR
3010 F$="EVENTS"
3020 PRINT "DISK OR CASSETT
     E (D/C)?"
3030 CALL KEY(0,K,S)
3040 IF S=0 THEN 3030
3050 A$=CHR$(K)
3060 IF (A$<>"C")*(A$<>"D")
     THEN 3030
3070 IF A$="D" THEN 3100
3080 D$="CS1"
3090 GOTO 3110
3100 D$="DSK1."&F$
3110 IF C=2 THEN 3320
3120 OPEN #1:D$,INTERNAL,IN
     PUT ,FIXED
3130 INPUT #1:LL$
3135 D9$=LL$
3140 IF LEN(LL$)=8 THEN 317
     0
3150 PRINT LL$;"?"
3160 GOTO 3280
3170 M=VAL(SEG$(LL$,1,2))
3180 D=VAL(SEG$(LL$,4,2))
3190 Y0=VAL(SEG$(LL$,7,2))
3200 M9=M
3210 D9=D
3220 Y9=Y0
3230 L=0
3240 PRINT "LAST ACCESS: ";
     LL$
3250 INPUT #1:L$(L)
3260 IF L$(L)="EOF" THEN 32
     75
3270 L=L+1
3272 GOTO 3250
3275 L$(L)=""
3277 L=L-1
3280 CLOSE #1
3290 L8=0
3300 L9=L
3310 RETURN
3320 OPEN #1:D$,INTERNAL,OU
     TPUT,FIXED
3330 PRINT #1:D9$
3340 FOR J=0 TO L9-1
3350 PRINT #1:L$(J)
3360 NEXT J
3365 PRINT #1:"EOF"
3370 CLOSE #1
3380 END
4010 D$=""
4020 DE=1
4030 INPUT "WANT EVENTS ON
     PRINTER (Y/N) ":P$
4040 IF SEG$(P$,1,1)<>"N" T
     HEN 4070
4050 DE=0
4060 GOTO 4080
4070 OPEN #1:"RS232/2.BA=96
     00.DA=8.PA=N"
4080 FOR J=L8 TO L0
4090 IF D$=SEG$(L$(J),1,5)T
     HEN 4190
4100 D$=SEG$(L$(J),1,5)
4110 M=VAL(SEG$(D$,1,2))
4120 D=VAL(SEG$(D$,4,2))
4130 Y=Y8
4140 IF D$>D9$ THEN 4160
4150 Y=Y8+1
4160 GOSUB 2150
4170 PRINT #DE:W$(W);" ";
4180 PRINT #DE:M$(M);D
4190 PRINT #DE:"{3 SPACES}"
     ;SEG$(L$(J),6,LEN(L$(J
     )))
4200 NEXT J
4210 IF DE=0 THEN 4230
4220 CLOSE #DE
4230 RETURN
```

©

The New MLX
Enhanced Machine Language Editor For The Commodore 64

Ottis R. Cowper, Technical Editor

This significantly improved version of COMPUTE!'s *"MLX" utility will help you enter machine language program listings without typos. It's more foolproof than the old MLX and is easier to use, too—especially for beginners. The new MLX is required to enter all machine language programs published in* COMPUTE! *for the Commodore 64, starting with "Balloon Crazy" in this issue.*

Since its initial publication in the December 1983 issue of COMPUTE!, our "MLX" machine language editor has helped thousands of readers type in dozens of ML programs with a minimum of problems. MLX detects most common typing mistakes as they're made. However, your growing appetite for high-quality programs is leading us to publish longer and longer listings. Such programs demand a more efficient entry system, so this month we're introducing a new MLX with important enhancements:

- A much more compact format. With each line of a new MLX listing, you enter eight bytes of data with 18 keystrokes, as opposed to only six bytes of data in 21 keystrokes when using the original MLX. This means you can enter machine language programs with 40 percent less typing.
- A more sophisticated checksum scheme. Transposition errors that could slip past the original MLX are caught by this version. Typing mistakes are now virtually impossible.
- A buffer (reserved area of memory) that holds the data you enter instead of direct storage in memory. This means that you'll never again have to worry with those bothersome POKEs that were sometimes necessary to reconfigure memory before using the old MLX.

Hexadecimal Checksums

Type in and save a copy of the new MLX. You'll need it for all future machine language programs in COMPUTE!, as well as ML programs in our companion magazine, COMPUTE!'s GAZETTE, and COMPUTE! books.

When you're ready to enter an ML program, load and run the new MLX. It asks you for a starting address and ending address. These addresses appear in the article accompanying the MLX-format program listing you're typing. If you're unfamiliar with machine language, the addresses (and all other values you enter in MLX) may appear strange. Instead of the usual decimal numbers you're accustomed to, these numbers are in *hexadecimal*—a base 16 numbering system commonly used by ML programmers. Hexadecimal—hex for short—includes the numerals 0–9 and the letters A–F. But don't worry—even if you know nothing about ML or hex, you should have no trouble using the new MLX.

After you enter the starting and ending addresses, MLX offers the option of clearing the workspace. Choose this option if you're starting to enter a new listing. If you're continuing a listing that's partially typed from a previous session, don't choose this option.

It's not necessary to know more about this option to use MLX, but here's an explanation if you're interested: When you first run MLX, the workspace area contains random values. Clearing the workspace fills it with zeros. This makes it easier to find where you left off if you enter the listing in multiple sittings. However, clearing the workspace is useful only before you first begin entering a listing; there's no need to clear it before you reload to continue entering a partially typed listing. When you save your work with the new MLX, it stores the entire contents of the data buffer. If you clear the workspace before starting, the incomplete portion of the listing is filled with zeros when saved and thus refilled with zeros when reloaded. If you don't clear the workspace when first starting, the incomplete portion of the listing is filled with random data. Whether or not you clear the workspace before you reload, this random data will refill the unfinished part of the listing when you load your previous work. The rule, then, is to use the clear workspace feature before you begin entering data from a listing, and not bother with it afterward.

At this point, MLX presents a menu of commands:

Enter data
Display data
Load data
Save file
Quit

You no longer have to remember SHIFT command keys as in the original MLX. Instead, just press the letter of a menu option. These commands are available only while the menu is displayed. You can get back to the menu from most options by pressing RETURN with no other input.

Entering A Listing

To begin entering data, press E. You'll be asked for the address at which you wish to begin entering data. (If you pressed E by mistake, you can return to the command menu by pressing RETURN.) When you begin typing a listing, you should enter the starting address here. If you're typing in a long listing in multiple sittings, you should enter the address where you left off typing at the end of the previous session. In any case, make sure the address you enter corresponds to the address of a line in the MLX listing. Otherwise, you'll be unable to enter the data correctly.

After you enter the address, you'll see that address appear as a prompt with a nonblinking cursor. Now you're ready to enter data.

To help prevent typing mistakes, only a few keys are active while you're entering data, so you may have to unlearn some habits. The new MLX listings consist of nine columns of two-digit numbers—eight bytes of data and a checksum:

```
C000:A9 0C 8D 15 D0 A9 FF 8D 17
C008:3B 63 8D 3C 63 A9 01 8D C6
C010:01 58 A9 00 8D 33 63 20 7D
C018:0B C5 20 C1 CB A9 FF 8D 43
```

You *do not* type spaces between the columns; the new MLX automatically inserts these for you. You *do not* press RETURN after typing the last number in a line; the new MLX automatically enters and checks the line after you type the last digit. The only keys you need for data entry are 0–9 and A–F. Pressing most of the other keys generates a warning buzz.

To correct typing mistakes before finishing a line, use the INST/DEL key to delete the character to the left of the cursor. (The cursor-left key also deletes.) If you mess up a line really badly, press CLR/HOME to start the line over.

The RETURN key also is active, but only before any data is typed on a line. Pressing RETURN at this point returns you to the command menu. After you type a character of data, the new MLX disables RETURN until the cursor returns to the start of a line. Remember, you can press CLR/HOME to quickly get to a line number prompt.

Beep Or Buzz?

After you type the last digit in a line, MLX calculates a checksum of the line number and the first eight columns of data, then compares it with the value in the ninth column. The formula (found in lines 370–390 of the MLX program) catches almost every conceivable typing error, including the transposition of entire numbers that the original MLX could miss. If the values match, you'll hear a pleasant beep, the data is added to the workspace area, and the prompt for the next line of data appears (unless the line just entered was the last line of the listing—in which case you'll automatically advance to the Save option). But if MLX detects a typing error, you'll hear a low buzz and see an error message. Then MLX redisplays the line for editing.

To edit a line, move the cursor left and right using the normal cursor controls. (The INST/DEL key now works as an alternative cursor-left key.) You cannot move left beyond the first character in the line. If you try to move beyond the rightmost character, you'll reenter the line.

To make corrections in a mistyped line, compare the line on the screen with the one printed in the listing, then move the cursor to the mistake and type the correct key. During editing, RETURN is active; pressing it tells MLX to recheck the line. You can press the CLR/HOME key to clear the entire line if you want to start from scratch, or if you want to get to a line number prompt to use RETURN to get back to the menu.

Other MLX Functions

The Display data option lets you review your work. Unlike the original MLX, the new MLX calculates and displays checksums for each line. Thus, a quick way to check your typing is to compare the reverse video checksums on the screen with the data in the rightmost column of the printed listing. If the values match, you can be confident that the line is entered correctly.

When you select D, you'll be asked for a starting address. (As with the other menu options, pressing RETURN at this point takes you back to the command menu.) When entering an address, make sure it corresponds to the address of a line from the listing. Otherwise, the checksums will be meaningless. You can pause the scrolling display by pressing the space bar. (MLX finishes printing the current line before halting.) To resume scrolling, press the space bar again. The display continues to scroll until the ending address is reached, then the menu reappears. To break out of the display and return to the menu before the ending address is reached, press RETURN.

The Save and Load menu options are straightforward. First, MLX asks for a filename. (Again, pressing RETURN at this prompt without entering anything returns you to the command menu.) Next, MLX asks you to press either T or D for tape or disk. If you notice the disk drive starting and stopping several times during a load or save, don't panic; MLX opens and reads from or writes to the file instead of using the usual LOAD and SAVE commands, so this behavior is normal. Disk users should also note that the drive prefix 0: is automatically added to the filename (line 750), so this should *not* be included when entering the name. (This also precludes the use of @ with for Save-with-Replace, so remember to give each version you save a different name.)

Remember that MLX saves the entire workspace area from the starting address to the ending address, so the save or load may take longer than you might expect if you've entered only a small amount of data from a long listing. When saving a partially completed listing, make sure to note the address where you stopped typing so you'll know where to resume entry when you reload.

Error Alert

MLX reports any errors detected during the save or load. Tape users should bear in mind that the Commodore 64 is never able to detect errors when saving to tape. The new MLX also has three special load error messages:

- INCORRECT STARTING ADDRESS. This means the file you're trying to load does not have the starting address you specified when you ran MLX. If you feel certain you're trying to load the right file, exit and rerun MLX, being careful to enter the correct starting address.

- LOAD ENDED AT *address*. This means the file you're trying to load ends before the ending address you specified when you started MLX. If you feel certain that you've loaded the right file, exit and rerun MLX, being careful to enter the correct ending address.

- TRUNCATED AT ENDING ADDRESS. This means the file you're trying to load extends beyond the ending address you specified when you started MLX. If you feel certain that you've loaded the right file, exit and rerun MLX, being careful to enter the correct ending address.

The Quit menu option has the obvious effect—it stops MLX and enters BASIC at a READY prompt. Since the RUN/STOP key is disabled, Q lets you exit the program without turning off the computer. (Of course, RUN/STOP–RESTORE also gets you out.) If you choose this option, MLX asks for verification. Press Y to exit to BASIC, or any other key to return to the menu. After quitting, you can type RUN again and reenter MLX without losing your data, as long as you don't use the clear workspace option.

The Finished Product

When you've finished typing all the data for an ML program and saved your work, you're ready to see the results. Unlike the original MLX, this version keeps the data in a temporary holding area rather than in its final resting place in memory, so you must always save the finished program with MLX and then reload it from BASIC with a standard LOAD command.

The instructions for loading the finished product varies from program to program. Some ML programs are designed to be loaded and run like BASIC programs, so all you need to type is LOAD *"filename"*,8 for disk or LOAD *"filename"* for tape, and then RUN. (Such programs usually have 0801 as their MLX starting address.) Others must be reloaded to specific addresses with a command such as LOAD *"filename"*,8,1 for disk or LOAD *"filename"*,1,1 for tape, then started with a SYS to a particular memory address. (On the Commodore 64, the most common starting address for such programs is 49152, which corresponds to MLX address C000.) In either case, you should always refer to the article which accompanies the ML listing for information on loading and running the program.

An Ounce Of Prevention

By the time you finish typing in the data for a long ML program, you'll have several hours invested in the project. Don't take chances—use our "Automatic Proofreader" to type the new MLX, and then test your copy *thoroughly* before first using it to enter any significant amount of data. Make sure all the menu options work as they should. Enter fragments of the program starting at several different addresses, then use the Display option to verify that the data has been entered correctly. And be sure to test the Save and Load options several times to ensure that you can recall your work from disk or tape. Don't let a simple typing error in the new MLX cost you several nights of hard work.

The New MLX For Commodore 64

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
100 POKE 56,50:CLR:DIM IN$,I,J
    ,A,B,A$,B$,A(7),N$ :rem 34
110 C4=48:C6=16:C7=7:Z2=2:Z4=2
    54:Z5=255:Z6=256:Z7=127
                         :rem 238
120 FA=PEEK(45)+Z6*PEEK(46):BS
    =PEEK(55)+Z6*PEEK(56):H$="
    0123456789ABCDEF" :rem 118
130 R$=CHR$(13):L$="{LEFT}":S$
    =" ":D$=CHR$(20):Z$=CHR$(0
    ):T$="{13 RIGHT}" :rem 173
140 SD=54272:FOR I=SD TO SD+23
    :POKE I,0:NEXT:POKE SD+24,
    15:POKE 788,52    :rem 194
150 PRINT"{CLR}"CHR$(142)CHR$(
    8):POKE 53280,15:POKE 5328
    1,15              :rem 104
160 PRINT T$" {RED}{RVS}
    {2 SPACES}{8 @}{2 SPACES}"
    SPC(28)"{2 SPACES}{OFF}
    {BLU} MLX II {RED}{RVS}
    {2 SPACES}"SPC(28)"
    {12 SPACES}{BLU}" :rem 121
170 PRINT"{3 DOWN}{3 SPACES}CO
    MPUTE!'S MACHINE LANGUAGE
    {SPACE}EDITOR{3 DOWN}"
                         :rem 135
180 PRINT"{BLK}STARTING ADDRES
    S{4}";:GOSUB300:SA=AD:GOSU
    B1040:IF F THEN180:rem 113
190 PRINT"{BLK}{2 SPACES}ENDIN
    G ADDRESS{4}";:GOSUB300:EA
    =AD:GOSUB1030:IF F THEN190
                         :rem 173
200 INPUT"{3 DOWN}{BLK}CLEAR W
    ORKSPACE [Y/N]{4}";A$:IF L
    EFT$(A$,1)<>"Y"THEN220
                           :rem 9
210 PRINT"{2 DOWN}{BLU}WORKING
    ...";:FORI=BS TO BS+EA-SA+
    7:POKE I,0:NEXT:PRINT"DONE
    "                    :rem 139
220 PRINTTAB(10)"{2 DOWN}{BLK}
    {RVS} MLX COMMAND MENU
    {DOWN}{4}":PRINT T$"{RVS}E
    {OFF}NTER DATA"    :rem 62
230 PRINT T$"{RVS}D{OFF}ISPLAY
     DATA":PRINT T$"{RVS}L
    {OFF}OAD DATA"      :rem 19
240 PRINT T$"{RVS}S{OFF}AVE FI
    LE":PRINT T$"{RVS}Q{OFF}UI
    T{2 DOWN}{BLK}"    :rem 238
250 GET A$:IF A$=N$ THEN250
                         :rem 127
260 A=0:FOR I=1 TO 5:IF A$=MID
    $("EDLSQ",I,1)THEN A=I:I=5
                          :rem 42
270 NEXT:ON A GOTO420,610,690,
    700,280:GOSUB1060:GOTO250
                          :rem 97
280 PRINT"{RVS} QUIT ":INPUT"
    {DOWN}{4}ARE YOU SURE [Y/N
    ]";A$:IF LEFT$(A$,1)<>"Y"T
    HEN220               :rem 189
290 POKE SD+24,0:END    :rem 95
300 IN$=N$:AD=0:INPUTIN$:IFLEN
    (IN$)<>4THENRETURN :rem 31
310 B$=IN$:GOSUB320:AD=A:B$=MI
    D$(IN$,3):GOSUB320:AD=AD*2
    56+A:RETURN          :rem 225
320 A=0:FOR J=1 TO 2:A$=MID$(B
    $,J,1):B=ASC(A$)-C4+(A$>"@
    ")*C7:A=A*C6+B       :rem 143
330 IF B<0 OR B>15 THEN AD=0:A
    =-1:J=2              :rem 132
```

```
340 NEXT:RETURN            :rem 240
350 B=INT(A/C6):PRINT MID$(H$,
    B+1,1);:B=A-B*C6:PRINT MID
    $(H$,B+1,1);:RETURN:rem 42
360 A=INT(AD/Z6):GOSUB350:A=AD
    -A*Z6:GOSUB350:PRINT":";
                           :rem 32
370 CK=INT(AD/Z6):CK=AD-Z4*CK+
    Z5*(CK>Z7):GOTO390:rem 131
380 CK=CK*Z2+Z5*(CK>Z7)+A
                          :rem 168
390 CK=CK+Z5*(CK>Z5):RETURN
                          :rem 159
400 PRINT"{DOWN}STARTING AT{4}
    ";:GOSUB300:IF IN$<>N$ THE
    N GOSUB1030:IF F THEN400
                           :rem 75
410 RETURN                :rem 117
420 PRINT"{RVS} ENTER DATA ":G
    OSUB400:IF IN$=N$ THEN220
                           :rem 85
430 OPEN3,3:PRINT          :rem 34
440 POKE198,0:GOSUB360:IF F TH
    EN PRINT IN$:PRINT"{UP}
    {5 RIGHT}";             :rem 6
450 FOR I=0 TO 24 STEP 3:B$=S$
    :FOR J=1 TO 2:IF F THEN B$
    =MID$(IN$,I+J,1)  :rem 226
460 PRINT"{RVS}"B$L$;:IF I<24T
    HEN PRINT"{OFF}";  :rem 15
470 GET A$:IF A$=N$ THEN470
                          :rem 135
480 IF(A$>"/"ANDA$<":")OR(A$>"
    @"ANDA$<"G")THEN540
                          :rem 100
490 IF A$=R$ AND((I=0)AND(J=1)
    OR F)THEN PRINT B$;:J=2:NE
    XT:I=24:GOTO550    :rem 46
500 IF A$="{HOME}" THEN PRINT
    {SPACE}B$:J=2:NEXT:I=24:NE
    XT:F=0:GOTO440     :rem 66
510 IF(A$="{RIGHT}")ANDF THENP
    RINT B$L$;:GOTO540:rem 107
520 IF A$<>L$ AND A$<>D$ OR((I
    =0)AND(J=1))THEN GOSUB1060
    :GOTO470           :rem 232
530 A$=L$+S$+L$:PRINT B$L$;:J=
    2-J:IF J THEN PRINT L$;:I=
    I-3                    :rem 12
540 PRINT A$;:NEXT J:PRINT S$;
                            :rem 2
550 NEXT I:PRINT:PRINT"{UP}
    {5 RIGHT}";:INPUT#3,IN$:IF
     IN$=N$ THEN CLOSE3:GOTO22
    0                     :rem 106
560 FOR I=1 TO 25 STEP3:B$=MID
    $(IN$,I):GOSUB320:IF I<25
    {SPACE}THEN GOSUB380:A(I/3
    )=A                    :rem 81
570 NEXT:IF A<>CK THEN GOSUB10
    60:PRINT"{BLK}{RVS} ERROR:
     REENTER LINE {4}":F=1:GOT
    O440                  :rem 161
580 GOSUB1080:B=BS+AD-SA:FOR I
    =0 TO 7:POKE B+I,A(I):NEXT
                          :rem 245
590 AD=AD+8:IF AD>EA THEN CLOS
    E3:PRINT"{DOWN}{BLU}** END
     OF ENTRY **{BLK}{2 DOWN}"
    :GOTO700              :rem 207
600 F=0:GOTO440            :rem 84
610 PRINT"{CLR}{DOWN}{RVS} DIS
    PLAY DATA ":GOSUB400:IF IN
    $=N$ THEN220          :rem 146
620 PRINT"{DOWN}{BLU}PRESS:
    {RVS}SPACE{OFF} TO PAUSE,
    {SPACE}{RVS}RETURN{OFF} TO
     BREAK{4}{DOWN}"   :rem 241
630 GOSUB360:B=BS+AD-SA:FORI=B
    TO B+7:A=PEEK(I):GOSUB350:
    GOSUB380:PRINT S$;  :rem 56
640 NEXT:PRINT"{RVS}";:A=CK:GO
    SUB350:PRINT          :rem 144
650 F=1:AD=AD+8:IF AD>EA THENP
    RINT"{DOWN}{BLU}** END OF
    {SPACE}DATA **":GOTO220
                          :rem 170
660 GET A$:IF A$=R$ THEN GOSUB
    1080:GOTO220           :rem 65
670 IF A$=S$ THEN F=F+1:GOSUB1
    080                    :rem 28
680 ONFGOTO630,660,630:rem 224
690 PRINT"{DOWN}{RVS} LOAD DAT
    A ":OP=1:GOTO710    :rem 31
700 PRINT"{DOWN}{RVS} SAVE FIL
    E ":OP=0               :rem 32
710 IN$=N$:INPUT"{DOWN}FILENAM
    E{4}";IN$:IF IN$=N$ THEN22
    0                     :rem 229
720 F=0:PRINT"{DOWN}{BLK}{RVS}
    T{OFF}APE OR {RVS}D{OFF}IS
    K: {4}";               :rem 66
730 GET A$:IF A$="T"THEN PRINT
    "T{DOWN}":GOTO880   :rem 90
740 IF A$<>"D"THEN730   :rem 90
750 PRINT"D{DOWN}":OPEN15,8,15
    ,"I0:":B=EA-SA:IN$="0:"+IN
    $:IF OP THEN810       :rem 163
760 OPEN 1,8,8,IN$+",P,W":GOSU
    B860:IF A THEN220   :rem 66
770 AH=INT(SA/256):AL=SA-(AH*2
    56):PRINT#1,CHR$(AL);CHR$(
    AH);                  :rem 221
780 FOR I=0 TO B:PRINT#1,CHR$(
    PEEK(BS+I));:IF ST THEN800
                          :rem 171
790 NEXT:CLOSE1:CLOSE15:GOTO94
    0                     :rem 230
800 GOSUB1060:PRINT"{DOWN}
    {BLK}ERROR DURING SAVE:{4}
    ":GOSUB860:GOTO220 :rem 61
810 OPEN 1,8,8,IN$+",P,R":GOSU
    B860:IF A THEN220   :rem 57
820 GET#1,A$,B$:AD=ASC(A$+Z$)+
    256*ASC(B$+Z$):IF AD<>SA T
    HEN F=1:GOTO850       :rem 155
830 FOR I=0 TO B:GET#1,A$:POKE
     BS+I,ASC(A$+Z$):IF ST AND
    (I<>B)THEN F=2:AD=I:I=B
                          :rem 180
840 NEXT:IF ST<>64 THEN F=3
                           :rem 20
850 CLOSE1:CLOSE15:ON ABS(F>0)
    +1 GOTO960,970      :rem 12
860 INPUT#15,A,A$:IF A THEN CL
    OSE1:CLOSE15:GOSUB1060:PRI
    NT"{RVS}ERROR: "A$:rem 114
870 RETURN                :rem 127
880 POKE183,PEEK(FA+2):POKE187
    ,PEEK(FA+3):POKE188,PEEK(F
    A+4):IFOP=0THEN920:rem 178
890 SYS 63466:IF(PEEK(783)AND1
    )THEN GOSUB1060:PRINT"
    {DOWN}{RVS} FILE NOT FOUND
     ":GOTO690             :rem 34
900 AD=PEEK(829)+256*PEEK(830)
    :IF AD<>SA THEN F=1:GOTO97
    0                     :rem 201
910 A=PEEK(831)+256*PEEK(832)-
    1:F=F-2*(A<EA)-3*(A>EA):AD
    =A-AD:GOTO930          :rem 75
920 A=SA:B=EA+1:GOSUB1010:POKE
    780,3:SYS 63338     :rem 107
930 A=BS:B=BS+(EA-SA)+1:GOSUB1
    010:ON OP GOTO950:SYS 6359
    1                      :rem 38
940 GOSUB1080:PRINT"{BLU}** SA
    VE COMPLETED **":GOTO220
                          :rem 139
950 POKE147,0:SYS 63562:IF ST<
    >64 THEN970            :rem 39
960 GOSUB1080:PRINT"{BLU}** LO
    AD COMPLETED **":GOTO220
                          :rem 126
970 GOSUB1060:PRINT"{BLK}{RVS}
    ERROR DURING LOAD:{DOWN}
    {4}":ON F GOSUB980,990,100
    0:GOTO220             :rem 233
980 PRINT"INCORRECT STARTING A
    DDRESS (";:GOSUB360:PRINT"
    )":RETURN             :rem 145
990 PRINT"LOAD ENDED AT ";:AD=
    SA+AD:GOSUB360:PRINT D$:RE
    TURN                  :rem 159
1000 PRINT"TRUNCATED AT ENDING
      ADDRESS":RETURN :rem 166
1010 AH=INT(A/256):AL=A-(AH*25
     6):POKE193,AL:POKE194,AH
                           :rem 95
1020 AH=INT(B/256):AL=B-(AH*25
     6):POKE174,AL:POKE175,AH:
     RETURN               :rem 122
1030 IF AD<SA OR AD>EA THEN105
     0                    :rem 135
1040 IF(AD>511 AND AD<40960)OR
     (AD>49151 AND AD<53248)TH
     EN GOSUB1080:F=0:RETURN
                          :rem 104
1050 GOSUB1060:PRINT"{RVS} INV
     ALID ADDRESS {DOWN}{BLK}"
     :F=1:RETURN          :rem 224
1060 POKE SD+5,31:POKE SD+6,20
     8:POKE SD,240:POKE SD+1,4
     :POKE SD+4,33         :rem 19
1070 FOR S=1 TO 100:NEXT:GOTO1
     090                   :rem 90
1080 POKE SD+5,8:POKE SD+6,240
     :POKE SD,0:POKE SD+1,90:P
     OKE SD+4,17          :rem 182
1090 FOR S=1 TO 100:NEXT:POKE
     {SPACE}SD+4,0:POKE SD,0:P
     OKE SD+1,0:RETURN     :rem 8
```

©

Wishbringer

James V. Trunzo

Requirements: Commodore 64; Apple II-series computer with at least 48K RAM; Atari 400/800, XL, or XE with at least 48K RAM; IBM PC with at least 48K RAM; Expanded Model PCjr; Amiga; Atari 520ST; Macintosh; Kaypro CP/M; or a TRS-80 Model III. All versions require a disk drive. The Commodore 64 version was reviewed.

The latest entry from Infocom, the software industry's most prolific producer of text adventures, is a novel mystery/adventure entitled *Wishbringer.* It's billed as an introductory-level adventure, but veteran gamers should not be put off by the label. When Infocom calls a game "introductory," it simply means you might need only 20 or 30 hours to solve the adventure instead of 60 or 70 hours.

Actually, *Wishbringer* offers several very challenging puzzles, starting at the very beginning of the game when you have to map your way over the mountain leading to the Majick Shoppe. What makes *Wishbringer* slightly easier than a more advanced Infocom game is that some of the mapping is done for you, the scope of the storyline is not as broad, and the puzzles are slightly less devious. However, this should not be construed to mean that the game is child's play—far from it.

As the accompanying storybook says, you're in the role of an ordinary postal clerk in an "ordinary little town, and you've been performing your ordinary mail clerk's duties in an altogether ordinary way. But there's something quite extraordinary about today's mail." From that point your adventure begins, and nothing is the same any more.

A Piece Of The Rock

The adventure is twofold: First, you must seek out and obtain a magic stone known as the Wishbringer. To keep track of your location in the game's imaginary world, you should compile a map as you go along, even though a general map is included. If you find the Wishbringer, your second job is to use the powers of the stone (which are awesome in some ways, yet limited in others) to save your town—a town that no longer resembles what it was at the start of the adventure. Now it's filled with trolls, vultures, and other evil creatures.

Wishbringer conforms to the usual Infocom style. That is, it employs no graphics, relying on detailed descriptions and the player's imagination to provide the "pictures." The sophisticated parser, an Infocom trademark, lets you type in compound sentences rather than just primitive verb-noun commands. Other features let you save games in progress and send text to a printer. And as always with an Infocom package, *Wishbringer* is attractively designed. It includes a beautifully illustrated storybook, "The Legend of Wishbringer," and even a plastic Wishbringer stone that glows in the dark.

Starting with a simple premise—one that may seem almost childish at first—*Wishbringer* quickly becomes an enjoyable, playable adventure for all but the most hardened veterans of adventure games.

Wishbringer
Infocom
125 Cambridge Park Drive
Cambridge, MA 02140
$39.95

Remember!

Karen McCullough

Requirements: Commodore 64; Apple II-series computer with at least 64K RAM; IBM PC with at least 128K RAM and color/graphics adapter; or an Expanded Model IBM PCjr. All versions also require a disk drive. Joystick and printer optional.

Remember! bills itself as a "powerful, yet simple tool designed to help students from junior high through college master difficult subjects and improve memory skills." The claim is not exaggerated. This program presents an effective study system that teaches how to organize and memorize facts. It also helps you practice and test yourself on those techniques.

Two fundamental design principles give *Remember!* its power: You enter the facts you want to memorize only once, and the program then presents them to you in a variety of ways; and the program helps you build associations with the facts you're learning.

You begin by using the Create or Edit Lesson section to enter the facts you want to memorize into question and answer blocks. Once the lesson is entered, you can add hints to help you memorize the information. These hints can be in any of three formats: pictorial, musical, or written. Only one hint is allowed per question, and all hints for a given lesson must be in the same format. Editing functions allow you to make changes in the questions, answers, or hints at any time.

Entering hints is not quite as simple as entering the questions and answers. Although the program is generally flexible, drawing pictures or entering musical notation is not as intuitive as typing in questions. Both take some practice to master. One irritating aspect of entering pictorial or musical hints is that they are not automatically saved when you choose the Get Next Word option. This is the only time you must tell the program to save something, and it's easy to forget. (*Remember!* will remind you, however.)

Foreign Language Characters

Once the facts are entered, you have the option of reviewing them or testing yourself in various ways. The Familiarization option displays both questions and answers for review and study. When you feel thoroughly familiar with the material, you can choose the Practice option. In this mode, *Remember!* displays either the question or answer (your option), and you supply the missing part. If you can't remember the answer, pressing RETURN or Enter displays a hint (if you supplied one), and pressing the key again calls up the correct response. Finally, you can evaluate your progress with the Test option, which is similar to Practice mode.

Remember! has many attractive features that make it flexible and easy to use. You select all choices from menus, and an optional command menu allows quick, direct access to other program functions. Special characters are available so you can enter lessons in German, Spanish, Italian, or French. When using English, you can enter subscripts and superscripts for chemistry formulas. A Help function is always available. You can turn off the sound or change the screen display from dark lettering on a light background to vice versa. With a printer, you can print out copies of your lessons.

The program comes with a 64-page manual divided into four sections. The "User's Guide" section has extensive and reasonably clear directions for all program functions; "Learning How You Learn" covers theories of memory and how they relate to the way the program works; "Tips On Making Databases" gives practical suggestions on how to organize lessons in specific subject areas; and finally, there's a set of appendices and a bibliography. The package also includes a reference card for the special characters and an extra disk for storing lessons.

The few weaknesses of *Remember!* are minor. Disk error messages are not helpful, stating only that there is a problem with the disk; and as mentioned above, entering hints with pictures and music is awkward. But overall, versatility and attention to detail give *Remember!* the power to turn a home computer into an effective and powerful study aid.

Remember!
DesignWare Inc.
185 Berry Street
San Francisco, CA 94107
$79.95

Mudpies For Atari 520ST

Gregg Keizer, Assistant Book Editor

Requirements: Atari 520ST system with a color monitor. Joystick optional.

SPLAT! In your face, clown. He waddles away and another takes his place. There are enough clowns in this circus to fill ten of those tiny cars, and you need some breathing room. Welcome to *Mudpies*.

Mudpies is the first arcade-style game we've seen for the Atari 520ST. It turns you into Arnold, a fairly nasty kid who likes to disrupt the circus by throwing pies at clowns. Not that he's without provocation, for the clowns crowd him at every opportunity, sending him to the First Aid tent at the slightest touch. And the clowns throw things back at Arnold—those Indian clubs used by jugglers. Get beaned with one of those, and you guessed it—another trip to the First Aid tent. Three trips to First Aid and the game's over (although you can withstand additional trips for each 10,000 points scored).

Reminiscent of *Robotron*, *Mudpies* is a fast-action, grip-the-joystick (or mouse) kind of game. Six rooms in each level are littered with mudpies you can pick up and fling in the direction you're moving. The clowns, like robots really, swarm toward you. They're not very bright, and they can be avoided with a little fancy footwork. But there are enough of them, with more waiting in the wings, to make it interesting. Also, they bear more than a passing resemblance to Ronald McDonald.

Arnold has other things on his mind besides clowns, however. His energy level depends on the fast food he finds, and if he doesn't eat enough hamburgers, fries, and shakes, he'll slow down terribly. Eat too much, though, and the people watching you play will shout "You're getting fat!" Overeating slows down Arnold, too.

CUSTOM ROM $19^{95}

NOW YOU CAN HAVE YOUR COMPUTER POWER UP MESSAGE SAY ANYTHING YOU WANT. IN ANY COLOR YOU WANT.

- Color of Cursor
- Color Of Screen
- Color Of Border
- Choose Up To 31 Letters As Your **Power Up** Message

Also included is the 2 Key Load. Pressing the Shift/Run Stop will load "*", 8, 1.

You can choose from any of these colors: Black, White, Red, Light Red, Cyan, Purple, Green, Light Green, Blue, Light Blue, Yellow, Orange, Brown, Dark Grey, Medium Grey, Light Grey.

P.S. . . . It sure is nice having your own colors and message on the screen on power up!

MASTER LOCK

Here, at last, is a program that will protect your software programs from unauthorized duplicating. 1 to 1 million copies can be protected, fast and easy.

- Specially designed for the C-64 and 1541 Disk Drive.
- Completely encrypts and protects your programs.
- Fully compatible with almost all machine language and basic programs — can even support chained programs.
- Contains a special feature which protects your program from being 'broken.'
- Incorporates all the latest techniques in program encryption and protection.
- Each system has been specially prepared and is unique from all other systems — only **you** are able to make working duplicates of your own protected programs.
- Simple and easy to use — entirely menu drive with prompts.
- Will stop virtually all software copiers from duplicating your programs.
- Fast and reliable protection routine does not take away any useable space from your disk — all 664 blocks are available for use.
- Easy to follow step by step instructions are included on the disk.
- A **must** for all programmers who do not want their programs to be in public domain!

Even the "nibblers" can't copy them. Not even *Disk Maker, *Mister Nibbles, *Copy II, *Ultra Byte or *Fast Hackem. The time to protect you disk is only 5 (five) seconds and each Master Lock makes a different protection scheme. Only

$29^{95}

*Disk Maker™, Basix
*Mister Nibbles™, Full Circle
*Copy II™, Central Point Software
*Ultra Byte™, Ultra Byte
*Fast Hackem™, Basement Boys Software

GRAPHIC LABEL MAKER

Give your labels the professional touch. With Hi Rez Graphics make your own design or use one our 60 premade labels with easy to use on screen editor. You can insert up to three lines of text then choose the picture you want to put on the left hand side of the label. Then you can print out as many labels as you want. This has got to be the neatest label program out there and it's only **$24^{95}**

**Also available — the x-rated Graphic package for the 64 and your print shop. There's 60 Hi Rex pictures*

$24^{95}

TOOL BOX

This disk has over 100 routines, some of them are routines for protection, smooth scrolling, modem routines, and sound and color routines; bootmaker, paddle and joystick; read terminal, auto dial, auto answer. They can easily be incorporated into all of your programs. It is also fully documented. With this disk alone you could build your own program. This disk has a lot of tricks that are used in commercial software.

$19^{95}

1541 M.A.S.H.

Now you can service your own 1541 disk drive using 1541 M.A.S.H. Save big bucks on repair bills. Rate the performance of your drive. Test and adjust RPM's. Test and adjust head alignment. Step by step instructions that anyone can follow. Pays for itself the first time you use it to adjust a misbehaving drive. No knowledge of electronics is necessary. All you needs is a screwdriver and 20 minutes.

NOW ONLY $19^{95}

DISK TRACKER

Now you can log all of your disks into a neat filing system. Automatically record disk names and program titles. Allows you to quickly scan what you've got. Sorts titles; prints jacket covers; store up to 1600 disk directories on one disk. Search quickly through your entire disk collection for a misplaced program. Works with one drive or two. Fast, easy operation. This program is a must for everyone!

ONLY $19^{95}

SWIFTERM AND MODEM

This is the best package anywhere!

SWIFTERM:
is absolutely the easiest terminal program available anywhere.

- Works with the 1660, and Westridge Modem
- New printer and midwestern protocal
- Printer dump
- DOS commands access from menu
- Phone book
- Auto dial (with auto redial)
- 29K storage buffer
- Save to disk
- Standard ASCII up/down loading
- 300/1200 Baud

This is an excellent easy to use program for a very reasonable price.

THE MODEM:
Auto dial, auto answer, 300 Baud modem that is 100% compatible with Commadors 1650 modem, so all our software will run with it.

ALL THIS FOR ONLY $35^{96}

So why buy just another terminal program when you can get a modem, too!?!

300/1200 BAUD

Modem for the C-64 With SWIFTERM **$169^{00}**

THE XXXXXXXXXX

X-RATED GRAPHICS LIBRARY

Your Print Show May Never Be The Same!

X-Rated Graphics guaranteed to spice up your special letterheads, greeting cards, signs and banners! Everything from mild to Look Out Nelly! 60 Erotic additions to heat up your print shop graphics library, plus 5 ribald Screen Magic additions!

Print Shop is a trademark of Broderbund.

$24^{95}

GRAPHICS & GAME DISK

X-Rated adventure like you have never seen before. Complete with graphic simulations.

Plus: X-Rated Cartoons **$24^{95}**

HAVE WE GOT A MESSAGE . . . FOR YOU!

Message . . . is a unique graphics integration package which allows you to continuously display hiresolution and multicolor files. Up to 12 files* may be stored in memory and displayed from 1 to 59 seconds each.

Message . . . can use Print Shop 'Screen Magic' files, Flexidraw files, Doodle!, Koala Painter, and more! **Message** . . . even comes with a 'Screen Maker' program to create your own message files!

Use **Message** . . . for advertising displays, titles for home videos, art shows, etc. . . . **Message** . . . even plays background music to accompany displays and you can create your own music files with the 'Music Maker' program.

Also included are 'Graphic Aids' utilities for conversion of different file types, including conversion of Koala Painter files to Doodle! or Print Shop!

The price for all this? Just $19^{95} including shipping and handling

**Actual number of files dependent on file complexity.*

MAGNUM LOAD

Programed by Jim Drew

MAGNUM LOAD is a new replacement KERNAL (operating system) ROM chip for your Commodor 65 or 128 computer that will load and verify programs up to 6 times faster than before. The tape routines have been removed from the old chip and in their place have been put a high-speed loader, high-speed verify, and disk driver no-head-rattle routine. Unlike the older cartridge fast loaders, no ports are tied up at the rear of the computer, the screen is not blanked during loading and there is no wear-and-tear on the expansion port. For maximum convenience and performance, the chip is installed directly in the circuit board. Generally a socket has already been provided to make the operation easy, but occasionally some soldering may be required. Now you can give your 1541 disk drive "1571 speed."

Rather than give you more exagerated claims about how many times faster our ROM chip is compared to the slower cartridge versions, a comparison chart is supplied listing MEASURED loading times.

Program	Star-Dos	Reg. Load	Mach 5	Fast Load	MAGNUM LOAD
Pitstop II	?	144 sec.	43 sec.	41 sec.	31 sec.
Music Shop	?	105 sec.	105 sec.*	105*	21 sec.
Hitchhiker's Guide to the Galaxy	?	70 sec.	70 sec.*	N.G.**	68 sec.
On-field Football	?	149 sec.	66 sec.	63 sec.	56 sec.
EASY FINANCE I	?	58 sec.	13 sec.	13 sec.	11 sec.

$24.95 For 1541 or MSD Version

$39.95 for Combo Version

*Will not fast load - defaulted back to regular load

**Failed to load at all

D-CODER

- Translates any machine language program into easy-to-ready English descriptions with complete explanations of each command.
- Makes complete notations of all important memory locations accessed by the program (SID, VIC, MOS, KERNAL, etc.)
- Gives you three ways of accessing programs:
  1. **Will read and list programs from DISK**
  2. **Will read and list programs from MEMORY**
  3. **Direct user input (from magazines, etc.)**
- Can be used to locate and examine any machine language program's protection routines!
- Can be used to easily break apart machine language programs for study and examination!
- Printer option for complete hard copy listings!

You no longer need to be an EGGHEAD to read Machine Language.

$19.95

N-CODER

THE PERFECT COMPANION PROGRAM TO D-CODER!

Allows you to easily make changes in machine language programs . . . right on the disk!

- Rewrite ability allows code to be altered and then rewritten directly to the disk!
- Features sector-by-sector scrolling assembly language display of machine language programs!
- Notation of ASCII text equivalents for easy spotting of embedded text strings!
- Handy reference display of all assembly language commands and their ML numerical equivalents!
- Byte splitter for easy splitting of decimal addresses into low byte-high byte format!

$19.95

Top Secret Stuff I and Top Secret Stuff II

Programed by Jim Drew

Are both collections of 20 programs per diskette (that works out to about $1.00 per program!) that help you explore and enhance your Commodore 64 and/or 128 and 1541 disk drive. Now you can unlock many secrets formerly known only to top machine language programmers by using these sophisticated "tools." If you have ever been curious about the inner workings of your computer system, now is your chance to dig in and find answers with the help of these programs. These collections of programs have gotten rave reviews from actual users, and we are sure that you, too, will be pleased.

The programs include on each diskette are listed below

TOP SECRET STUFF I

The Dock (view/repair disk contents)
Sync Checker (diskette)
Imbedded Track Number Creator
Disk Manipulation System
3 Minute Copy (backup program)
Diskette Matcher (compare sectors)
Unscratch A File (recover file)
View BAM (block allocation map)
1541 Read/Write Test
½ Track Reader
Header Reader (display disk header)
Sync Maker
Device Number Change (disk drive)
Electronic Arts Backup
Drive Mon (disk drive m/1 monitor)
Diskette File Log (start-end address)
Write-Protect Sensor Test
Repair A Track (recover data)
Fast Format (10 seconds)
½ Track Formatter

$19.95

TOP SECRET STUFF II

RAM Test (test Computer RAM)
Copy $AOOO-$FFFF (under ROMS)
Display G.C.R. (All sector data)
Un-Write Protect (diskette)
Unnew Program
Wedge - $8000
Smooth Scroll (messages up screen)
Koala Dump (koala pad screen dump)
Disk Manipulation System
Disk Eraser (20 second clean wipe)
Split Screen (TWO screen colors)
Disk Protection System (stops copies)
Write Protect (diskette)
Boot Maker (autobook BASIC programs)
Wedge - $COOO
Diskmatcher II (high speed version)
No Drive Rattle (on reading errors)
3 Times Disk Drive Head Speed
Monitor Test (check video monitor)

$19.95

SOFTWARE PROTECTION HANDBOOK
Third Edition! Now Available!

If you're **tired** of being harassed by **protected software** and too many **copy** programs, then this is the book for you! This 224 page manual covers the gamut from legalities to protection methods to step-by-step back up procedures. Now you can learn both how to **protect and unprotect** software! The techniques covered include copying cartridges to tape or disk, tape protection, and disk protection. Disk protection covers error numbers 20, 21, 22, 23, 27 and 29 **plus** single track formatting, header modification, header swapping, half track reading and writing, reading and modified bit densities, formatting illegal track/sectors, sync writing and **more!** The Third edition **explains**, tells how to **detect** and how to **write** them with included software.

C-64 Book Only .... **$19.95 US**

Book & Disk of all Programs .... **$29.95 US**

This manual does not condone piracy

*Shipping: $2.00

'TRACK TRAP' The 1541 Disk Expander!
• Supplement to the Software Protection Handbook •

The most unusual and innovative protection analysis tool for the Commodore yet! — **Not For Beginners** — This system expands your 1541 drive giving capability otherwise only possible for **professional** disk duplication equipment. Now you can create or analyze exotic forms of disk protection. **'D.O.S. Kings' Take Note!'** — Entire tracks of data can be read and written without regard to 'standard' sync and format. You are not longer limited to sector by sector searches. Whole track readouts reveal hidden data even when all or most of the sectors have been erased. Uncovers and writes data under errors, pulse coded sync or data, hidden data and access codes, multiple track densities and more! This supplemental manual covers the complete implementation of the track trap system including necessary software and hardware documentation.

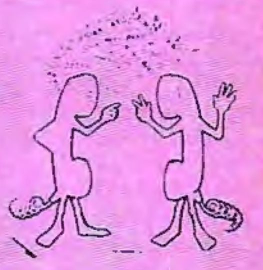
Track Trap disk expanded manual .... **$9.95 US**

WAR GAMES AUTODIALER

1. **Auto Dial** will automatically dial a set of numbers you choose.
2. **Review Numbers** will review numbers that were answered by a computer.
3. **Save Numbers** will save numbers where a computer answered.
4. **Hardcopy of Numbers** will print out list of numbers where a computer answered.
5. **LOAD Numbers** will load in numbers to continue where it left off.
6. **Continue** will pick up dialing where it was interrupted.

$29.95

BBS



- 300/1200 Baud
- Remote access for sysop
- 2 levels of security for up and down load
- 7 rooms (read and write) with 4 security levels
- Secret highest level
- Open chalk board
- Auto message cycling
- Printer option

Plus the only BBS with all three (3) proticals X modem, new punter and midwest term, so anyone can up/down load.

Don't be fooled by cheap imitations. This is the most comprehensive system available anywhere. Now at a new low price.

$39.95

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. Canada orders must be in U.S. Dollars. VISA — MASTER CARD — C.O.D.

Programs for C-64 — $2.00 S & H on all orders

Software Submissions Invited

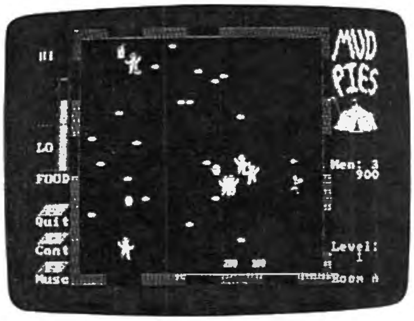
VISA — MasterCard

MegaSoft Limited

P.O. Box 1080, Battle Ground, Washington 98604

Phone 800-541-1541 • BBS 687-5205 After Hours Computer to Computer (BBS)

One or two people can play *Mudpies*, with two players alternating turns. The graphics are entertaining, as is the music (up to a point, when you might prefer to turn it off with a press of a function key). Looking like an IBM PC game more than anything else, down to the typeface of the display, *Mudpies* doesn't take full advantage of the ST. No menus, no icons, and you can't return to the GEM desktop without rebooting—but the game is still a tent full of fun.



Pesky clowns swarm toward Arnold as he defends himself by flinging mudpies.

Mouse Or Joystick?

The only way to score points is to splatter clowns. Advancing to the next room or level changes neither the scoring nor the speed or intelligence of the clowns. It makes for a repetitive game, and you can get good at it in a short time. One interesting feature is something called the Mudslinging Round, where the clowns don't chase you and don't throw things. You've got 30 seconds to mudpie as many of the purple-wigged clowns as you can, racking up bonus points left and right. It's great revenge.

Though you can control Arnold with the mouse, try a joystick instead. Mouse controllers may be great for manipulating Macintosh-style graphics, but for anyone who's arcade-trained, a joystick will probably feel more natural. In a fast game like *Mudpies*, you need every edge you can get.

If you rack up one of the highest scores, you can type your name on a vanity screen. However, the screen is not saved on disk. Unfortunate, for the game is addicting enough to make you want to beat whatever score was last posted. Function keys let you pause the game and turn off the music (though not the sound effects).

A few bugs cropped up while playing *Mudpies*. Several times the game paused on its own, the music suddenly turned itself on, and once the game stopped altogether and flipped back to the title screen. MichTech attributes all these bugs to the 520ST's operating system, saying that erratic control codes are being read from the joystick. The bugs are irritating, but not disastrous.

Keep an eye on this computer—its wide selection of colors, large memory, and speed make it a potentially great game machine. *Mudpies*, first out of the blocks, is a good beginning.

Mudpies
MichTron
567 S. Telegraph
Pontiac MI 48053
$29.95

BASIC XE For Atari XL & XE

Robert L. Riggs

Requirements: Atari XL or XE computer with at least 64K RAM. Disk drive recommended.

About two years ago, Optimized Systems Software brought out an extended BASIC cartridge for Atari computers. As a sequel to OSS's disk-based BASIC A+ introduced back in 1981, the BASIC XL cartridge was fantastic. It added 45 new commands and wrapped up Atari BASIC, BASIC A+, and Microsoft BASIC in one neat package. Furthermore, BASIC XL was made upwardly compatible with Atari BASIC, so it would run existing Atari BASIC programs. I discovered that games previously typed in from magazines—and abandoned because of their sluggish pace—ran at near-arcade speeds with BASIC XL. And it still offered Microsoft-style string handling, auto line numbering, block line deletions, and a host of other features.

When Atari introduced its 130XE this year, OSS upgraded BASIC XL for the new 128K machine. The result, BASIC XE, runs on all the XL computers but also adds commands to take advantage of the 130XE's expanded memory. The most important new command is XTEND. After you've typed or loaded a program into memory on the 130XE, you can use this command to move the program into the alternate 64K bank. At that point, your program and data space are separate—the former occupying the alternate 64K, and the latter occupying the main 48K (leaving about 35K free for data and strings). An optional third parameter for PEEK and POKE statements gives you access to any section of the 130XE's memory—the four extended banks of 16K or the main 48K RAM.

Of course, the XTEND command works only on the 130XE, not on the XL computers. Also, if you save a BASIC XE program which has been XTENDed, you can't load it back with either BASIC XL or Atari BASIC.

Like its predecessor, BASIC XE offers several additions to the Atari BASIC vocabulary, including ELSE, WHILE, ENDIF, ENDWHILE, PRINT USING, TAB, and TRACE/TRACEOFF. Atari's player/missile graphics are made easier to use via commands like MISSILE, BUMP, HITCLR, PMCOLOR, PMGRAPHICS, PMMOVE, PMWIDTH, and PMCOLOR.

Another extremely powerful instruction is SET. It lets you exercise control over a variety of system-level functions. You can quickly and easily disable or enable the BREAK key, change tab stop settings for the comma in PRINT statements, alter the prompt character for INPUT, automatically DIMension strings, and instruct the LIST formatter to indent structured statements. BASIC XE also has DOS commands, including DIR (directory), ERASE, PROTECT, UNPROTECT, and RENAME.

Memory Magic

There's much more. You get commands like DPEEK/DPOKE (for PEEKing and POKEing double-byte values), ERR (for reporting errors), FIND (a search command), HSTICK/VSTICK (for the joysticks), and SYS (for jumping directly to a memory address). And unlike Atari BASIC, BASIC XE lets you type your programs in lowercase or reverse characters. No more hitting the CAPS or reverse key after an annoying syntax error!

Normally you'd expect such a powerful BASIC to consume much more memory than standard 8K Atari BASIC. But the 16K BASIC XE cartridge cleverly bank-selects its ROM so that it displaces only 8K of RAM. Also, some of the commands (such as most of those for player/missile graphics) are stored on a disk that comes with the cartridge. Although you don't absolutely need a disk drive to use the BASIC XE cartridge, you won't be able to use these extra commands without one.

Among the extended commands on disk are a SORT statement that accepts numeric arrays as well as string arrays; a FAST command that tells BASIC XE to precompile the program currently in memory, so programs run several times faster than with Atari BASIC; RENUM, for renumbering BASIC programs; LVAR, to list variable names; RGET and RPUT, for reading or writing whole records with devices; and MOVE, a block memory transfer.

The manual thoroughly explains BASIC XE and is carefully indexed. The more you use BASIC XE, the more you realize it should have been built into the 130XE in the first place. If you intend to do much BASIC programming, especially on the 130XE, BASIC XE is a must.

BASIC XE
Optimized Systems Software
1221B Kentwood Avenue
San Jose, CA 95129
$79

Rescue Raiders For Apple

James V. Trunzo

Requirements: Apple II-series computer with at least 48K RAM, a disk drive, and a joystick.

Rescue Raiders is more than a little reminiscent of Brøderbund Software's popular *Choplifter*, yet easily stands on its own as an arcade game. It offers sufficient varieties of play and objectives to challenge even skilled arcade enthusiasts.

The theme is rather mundane, but the implementation contains a number of innovative elements. It seems that futuristic terrorists have time-warped sophisticated weapons into the middle of World War II and into the hands of the Germans. Fortunately, you, too, possess several of these "ultimate" weapons, and if you have the skill to use them, you can prevent history from being changed.

Controlling a command helicopter equipped with the latest in weaponry and radar technology, you can attack the enemy, observe the battlefield, transport troops, and attempt to destroy the enemy's helicopter—a clone of your own craft. The game demands solid arcade skills to manipulate your chopper and carry out these missions.

To add an element of strategy, you must win battles as economically as possible and learn how to employ your troops to their best advantage. *Rescue Raiders* definitely places more emphasis on arcade action than on strategy, however; except for a few prehistoric see-em blast-em games, nearly every arcade game has at least some element of strategy. A few factors do set *Rescue Raiders* apart, though. Economics enter the picture when you decide how to purchase men and equipment with funds earned by time spent in battle. Your score is partly determined by the amount of money you've spent and the number of lives lost in combat.

Other nice touches include excellent graphics and animation, plus capsule histories of European cities involved in World War II. (The histories appear after you win one of eight increasingly difficult battles in the game.) However, the game is almost totally devoid of sound effects. The omission is so noticeable that I first assumed the program disk was defective.

Overall, *Rescue Raiders* is a good, very challenging arcade game that could be even better with a frill here and there.

Rescue Raiders
Sir-Tech Software, Inc.
96 Main Street
Ogdenburg, NY 13669
$34.95

Field Of Fire For Atari & 64

James V. Trunzo

Requirements: Commodore 64 or an Atari 400/800, XL, or XE computer with at least 48K RAM. Both versions also require a disk drive and a joystick. The Atari version was reviewed.

Sarge, Freda, Billings, and Wild Bill are pinned down behind the seawall. The English Channel is at their backs and the buildings in front of them are full of Germans. Angry Germans. It's June 6, 1944—Omaha Beach—and the enemy machine gun and artillery fire is intense. Casualties mount. Finally Sarge, a veteran from the 1st Division's days in North Africa, remembers something he once heard. "What do you guys want, to live forever?" he shouts, and leads his rifle team over the wall.

The rest is history, or in the case of this World War II simulation, just part of the game. *Field of Fire*, written by Roger Damon, the creator of Brøderbund's *Operation Whirlwind*, is an enthralling game of computer combat. Eight separate scenarios, ranging from the 1942 campaign in Tunisia to the spring 1945 crossing of the Roeher River, and every major battle in between, puts you in command of Easy Company, part of the 1st Infantry Division, the Big Red One.

As the company commander, you give orders to the six-man teams armed with rifles, machine guns, bazookas, or mortars. Each team's abilities, such as weapon range and firepower, are based on true-life factors, and managing the various teams is a large part of a successful battle.

Separate game phases let you observe the battlefield, lay down fire, move, and assault nearby positions. All commands are selected by manipulating the joystick controller or pressing one of three keys on the keyboard. Three levels of difficulty, the eight scenarios, and an extended Campaign Game make *Field of Fire* a game you'll play more than once.

Stealth Makes Health

As in actual World War II battles, winning at *Field of Fire* requires attention to detail and a bit of luck. Maneuvering and firing—a phase in which some of your men move from one place of cover to another while others fire at suspected or known enemy positions—is a vital tactic to master. Charging blindly only brings immediate enemy response, pinning down your men. A few minutes of this could disintegrate your company. You have to make use of woods, ridges, hedgerows, streams, buildings, and roads. To be caught in the open is to risk annihilation.

Efficient use of your weapons teams is also important. Machine guns wreak havoc with enemy infantry, but against armor they'll merely force the tanks to "button up," just slightly restricting their movements. You must keep a few bazooka teams ready when the German panzers make an appearance. And your forward observers (when attached to your company) must be protected from all danger—placed at the tops of hills, their ability to call down artillery fire can turn the tide.

Field of Fire is quite good. You quickly get the flavor of combat and command. Some problems do crop up, however. Compared to the Germans in most scenarios, your men are just *too* good. Close assaults, in which your teams toss grenades and overrun a neighboring position, almost always result in a victory. Also, games sometimes seem to end abruptly. In one scenario, "Forever Road," you're supposed to move your company off the top of the map. But if you eliminate all the enemy units beforehand, the game ends with a less than acceptable victory level.

The bottom line is that *Field of Fire* is a sweaty palms kind of game. You won't smell the cordite or see the destruction, but your imagination fills that in quite sufficiently. Try it—but keep your eyes on that next treeline.

Field of Fire
Strategic Simulations, Inc.
883 Stierlin Road, Building A-200
Mountain View, CA 94043
$39.95

NEC PC-8401A Portable Computer

Gregg Keizer, Assistant Book Editor

Lap-sized portable computers are everywhere. You see them under the arms of business people, students, journalists, lawyers, and writers. It's almost impossible to board a jetliner without seeing one or hearing the clack of its keys. No wonder, for a lap portable lets you take your work virtually anywhere. And with the internal modems built into most lap portables, it's simple to communicate with other computers thousands of miles away.

The NEC PC-8401A portable computer is no exception to all of this. With four programs permanently stored in Read Only Memory (ROM), the NEC offers word processing, personal filing, spreadsheet analysis, and telecommunications—anywhere, anytime.

Like most lap portables, the NEC contains internal nickel-cadmium batteries which continuously trickle current to the Random Access Memory (RAM) chips, thus maintaining your data even when the power switch is turned off. The NiCad batteries, in turn, are continuously recharged by either four C batteries or an optional AC adapter. A low battery indicator warns when you're down to your last 30 to 60 minutes of power. The C batteries last three to eight hours, depending on their type. Even if the batteries fail and an external power supply isn't connected, the NEC retains saved files for up to five days.

The NEC has a full-stroke keyboard and a larger screen than most lap portables—80 columns by 16 lines. It has a total of 64K RAM and 96K ROM, plus a built-in operating system, CP/M (Control Program/Microcomputers). The package includes a phone cable for the modem and a cassette recorder cable for storing files on tape.

Onboard Software

Wordstar-to-Go, the word processing program, is a truncated version of the popular *Wordstar*. If you already know *Wordstar*, moving to the NEC's smaller version is a snap. Most of the commands are retained, and the ones that aren't, such as soft hyphenation and on-screen file directories, aren't really missed. Writing on the NEC is a pleasure. The keyboard has a solid feel and the screen is fast enough to keep pace with your typing.

Calc-to-Go is the NEC's spreadsheet program. You can create up to 64 columns and 256 rows, though the window on your screen is only 80 characters wide by 14 rows (a status line takes up the other two rows). The program has most standard spreadsheet features, such as entering data and formulas, arithmetic and logical functions, and editing or deleting data, rows, and columns.

Personal Filer lets you design cards and files which contain information such as addresses, phone numbers, and client notes. You can search and sort these cards, even use them to automatically dial phone numbers. Modifying, viewing, and entering new cards is fairly simple.

The fourth program included in the NEC is a telecommunications package called *Telecom*. Using the internal 300 bits-per-second modem (an external 1200 bps modem is optional), you can access information services, electronic bulletin boards, and almost any other computer connected to a modem. *Telecom* can upload and download files, dial numbers, and automatically log on to services. You can set up directory files and build log-on sequences. Both no protocol and Modem7 protocol are supported, and *Wordstar-to-Go* files can be converted to straight ASCII for uploading to other computers.

Documentation for these programs is extensive. Three manuals plus a general *User's Guide* come with the computer. Examples are easy to follow, for the most part, and cover almost all commands and features.

NEC offers a wide range of accessories, including an external 3½-inch disk drive (320K), a battery-powered 32K RAM cartridge, and cables for parallel printers, monochrome monitors, and RGB color monitors.

For the price, the NEC has much to offer. Word processing, spreadsheet management, filing, and telecommunications—all at your fingertips.

NEC PC-8401A
NEC Home Electronics (USA) Inc.
1401 Estes Avenue
Elk Grove Village, IL 60007
$999

MouseWrite For Apple IIe And IIc

Gregg Keizer, Assistant Book Editor

Requirements: Apple IIe with extended memory 80-column card and a disk drive or an Apple IIc. Mouse recommended.

Pull-down menus, overlapping windows, mouse-driven commands, and clipboards—seems like they're everywhere. The Macintosh, Amiga, Atari 520ST, and even the IBM PC with the GEM operating system take advantage of these tools, offering programs easy to learn and use. Word processing programs such as *MacWrite* and Microsoft's *Word* on the Macintosh, for instance, are built around this type of user interface. Now, with a program called *MouseWrite*, you can point and click your way through your prose on an Apple IIc or enhanced Apple IIe.

MouseWrite is a full-featured word processor, complete with all the standard text entry, editing, and formatting functions people have come to expect. What makes it different is not so much what it does, but how it does it. If you're able to point and click a button, you can delete text, change margins, do boldfacing or underlining, justify, and search and replace.

The program looks and works somewhat like a Macintosh application. The AppleMouse—though not required—is used to move the cursor, display menus, and select options. If you're not using a mouse, keyboard commands are available. A bar at the top of the screen contains eight menus, ranging from Windows and Page to Edit and File. Everything is within easy reach. A ruler showing margins and tabs can be displayed or hidden. Two windows can be open at the same time, letting you cut and paste sections of text from one version of a document to another. Printers can be selected and text formatted with a click.

Familiarity Breeds Content

If you've used a Macintosh word processor such as *MacWrite*, acclimation to *MouseWrite* is simple. Its operation is so comfortable that you can be up and writing within a few minutes of loading the program. Since the menus and commands are all just a click away, there's little need to pore over the manual. If you're unfamiliar with Macintosh-like programs, the documentation quickly gets you started, though many of *MouseWrite*'s features and commands will seem intuitive. Choosing the Find menu, for instance, leads you to three choices: Find Next, Replace Then Find, and Replace All. The text you want to find and replace is simply typed in. Even file and disk management, such as formatting disks, and opening, closing, and deleting files, is done with only a simple command or two.

Other *MouseWrite* features include automatic page numbering, headers, footers (both of which can display the current time and date), an optional onscreen clock (which unfortunately must be reset each time the program is booted), centering and justifying text, and three spacing options. Scroll bars let you move quickly through a long document, and windows can be resized by moving the mouse pointer.

MouseWrite makes writing what it should be—fun. With virtually no commands to memorize (at least when you use the mouse), you can concentrate on the words, not how the words get into the computer.

MouseWrite
Roger Wagner Publishing
P.O. Box 582
Santee, CA 92071
$125

Phantasie For Apple & 64

James V. Trunzo

Requirements: Apple II-series computer with at least 48K RAM and a disk drive; or a Commodore 64 with a disk drive. The Apple version was reviewed.

I have a strong predilection toward fantasy role-playing games, and when one comes along that seduces me into hour after hour of play, it's hard to keep from sounding overenthusiastic. But such a game has come along. *Phantasie*, from Strategic Simulations, may be the best fantasy role-playing game to come down the silicon pike since Sir-Tech conjured up *Wizardry*. As a matter of fact—at the risk of sounding blasphemous—in some ways *Phantasie* surpasses *Wizardry*.

Phantasie contains all the typical ingredients that go into a top-notch role-playing game: multiple characters derived from various races, standard professions (such as wizards, warriors, thieves, and so on), magic, monsters, and mayhem. However, the ways in which *Phantasie* combines these elements sets it apart from typical adventure games.

First, *Phantasie* employs a vertical split screen, the left side showing the area in which the party of adventurers is traveling and the right showing the party itself. During combat (a frequent occurrence in *Phantasie*), the enemy appears in front of the adventurers on the left, and the fight evolves into a semianimated affair. Spells hurl up and down the screen, striking each opponent. Swords, stingers, fangs, and claws are thrust in the direction of the enemy, with appropriate sound effects accompanying hits and misses. Of course, combat isn't always a must: You can greet an enemy, flee, accept a surrender, or attempt a bribe. Strategy is vital, especially early in the game when the adventurers are weak.

Phantasie keeps you in the dark, never showing you what's ahead until you explore the area. However, once you've explored the territory or dungeon (there are ten, all large and dangerous), the program "remembers" and displays the area when you reenter it later. There are 16 screens of territory in *Phantasie*'s land of Gelnor, so this eliminates the drudgery of painstakingly mapping a large area on paper, while retaining the game's mystery. In addition to saving lots of time, this feature emphasizes one of *Phantasie*'s major strengths: It is exceedingly playable, yet exceptionally challenging.

An Unfolding Story

Phantasie is aided tremendously by a strong story line that actually develops as you play. With help from 20 scrolls scattered throughout Gelnor, the party of adventurers is guided in its quest to defeat the Black Knights and their sorcerer master, Nikademus. These scrolls, hard-earned for the most part, contain hints that are necessary to emerge victorious. And don't be surprised if you notice a resemblance to J.R.R. Tolkien's *Lord of the Rings*.

The program allows a tremendous amount of interaction between characters and, even more importantly, between the player(s) and the game itself. It asks you to make choices often enough to involve you in the action without slowing up play, and the choices directly affect the course of the game. Too often in games of this sort, the choices are gratuitous, negating their value.

It's easy to go on about *Phantasie* because there are a tremendous number of facets to this program. Eighty different monsters, more than 50 spells, 100 pieces of equipment, plus potions, scrolls, rings, and other details make the game fascinating to play. Whether you're visiting a town or a dungeon, fighting a troll or a swarm of killer bees, or hiking through the mountains or the deserts, your persona will be challenged both physically and mentally throughout a game of *Phantasie*.

Phantasie
Strategic Simulations, Inc.
883 Stierlin Road, Building A-200
Mountain View, CA 94043
$39.95

©

ATARI · apple · COMMODORE

ATARI
800XL	CALL
130XE (NEW)	CALL
520ST (NEW)	CALL
1050 Drive	165
1010 Recorder	42
1020 Printer	55
1025 Printer	159
1027 Printer	179
850 Interface	109

ATARI SOFTWARE (NEW)
Codewriter	35.75
Filewriter	20.75
Reportwriter	20.75
Menuwriter	20.75
Home Integrator	19.75
Small Bus. Inventory	11.75
Salesman Expenses	11.75
Accs Rec/Pay	11.75
Retail Invoice	11.75
Final Legacy	15.75
Adventure Writer	18.75
Star Voyager	11.75

ATARI
XMM-801 Printer	177
XDM-121 Daisywheel	198
XM301 Modem	39
SF354 SSDD Drive	168
SF314 DSDD Drive	207

BRODERBUND (Atari)
The Print Shop	28.95
Graphics Library	18.95
Graphics Library II	19.50
Bank St. Writer	42.75
Whistler's Brother	18.95
Spelunker	18.95
Stealth	18.95
Serpent's Star	24.95
Mask of the Sun	24.95

MICROPROSE (Atari)
Kennedy Approach	21.75
Crusade in Europe	24.75
Decision in Desert	24.75
Solo Flight	20.75
Nato Commander	20.75
Spitfire Ace	18.75
F-15 Strike Eagle	20.75
Hellcat Ace	18.75

SUB LOGIC (Atari)
Flight Simulator II	*32.75*
Night Mission Pinball	20.75

SSI (Atari)
Colonial Conquest	24.75
Field of Fire	24.75
Kampfgruppe	36.75
Objective: Kursk	24.75
Market Garden	29.75
Six Gun Shootout	24.75
War in Russia	49.75
Computer Baseball	24.75
Computer Quarterback	24.75
Gemstone Warrior	22.75
50 Mission Crush	24.75
Questron	29.75

SCARBOROUGH (Atari)
NET WORTH (AT/C-64)	48.75
Mastertype	22.75
Improved Mastertype	23.75
Mastertype Filer	22.75

PERSONAL PERIPHERALS
Super Sketch C-64	29.95
Printer Utility C-64	14.99
Super Sketch Atari	25.95
Super Sketch Apple/Ibm	55.00

SYNAPSE (Atari)
SynFile	33.75
SynCalc (AT/C-64)	33.75
SynComm	26.75
SynTrend	26.75
SynChron	26.75
SynStock	26.75

ADVENTURE INT'L (Atari)
Diskey	32.75
Ultradissambler	32.75
Maxi Golf	18.75
Ralley Speedway	26.75
Whomper Stomper	18.75
Adventures 1-12 each	14.50
Adv. Hint Book	5.50

TRONIX
S.A.M. - Atari	38.50
S.A.M. - C-64	38.50

CONTINENTAL
Home Accountant	44.75
1985 Book of Atari Software	16.95

SPINNAKER
Delta Drawing Room	19.95
Cosmic Life Room	19.95
Up for Grabs Room	19.95

INNOVATIVE CONCEPTS
Flip-n-File 10	3.50
Flip-n-File 15	8.25

SCARBOROUGH (Apple)
NET WORTH	48.75
Improved Mastertype	22.75
Mastertype Filer	22.75

BRODERBUND (Apple)
The Print Shop	31.50
Graphics Library	18.50
Graphics Library II	18.95
Paper Refill	11.95
Bank St Writer 64K	41.50
Bank St Writer 128K	42.75
Bank St Filer	42.75
Bank St Mailer	42.75
Bank St Speller	42.75
Carmen Sandiego	22.75
Karateka	22.75
Captain Goodnight	22.75
Muppey Cruise	25.75

SUB LOGIC (Apple)
Flight Simulator II	*32.75*
Night Mission Pinball	20.75

MICROPROSE (Apple)
Crusade in Europe	25.75
Decision in Desert	25.75
F-15 Strike Eagle	21.75
Nato Commander	21.75

SSI (Apple)
Mech Brigade	36.75
Norway 1985	22.75
Market Garden	29.75
Six Gun Shootout	24.75
50 Mission Crush	24.75
Phantasie	24.75

Spinnaker
Alphabet	$18.75
Story Machine	$19.75
Kids on Keys	$18.75
Grandma	$19.75
Snooper Troop	$22.75

COMMODORE
C-128 NEW	CALL
1571 Drive	CALL
1572 Drive	CALL
1902 Monitor	CALL
1670 Modem	CALL
C-64 Computer	CALL
1541 Drive	189
MPS801 Printer	LOW
1702 Monitor	199
Simon's Basic	24.75
Assembler 64	34.75
Super Expander	22.75
Logo 64	49.75
Pilot 64	38.75
Easy Calc	34.75
Easy Script	38.75

CARDCO
Digitizer Camera	189.95
32K Printer Buffer	59.95
Numeric Keypad	34.95
CB/5 5-slot Board(64)	54.00
CB/2 2-slot Board(64)	25.00
S'More Basic Rom	49.95
Write Now-64	35.00
Mail Now-64	29.00
Spell Now-64	29.00
File Now-64	29.00
Paint Now-64	29.00
Calc Now-64	29.00
Tax Survival	29.00
Super Printer Utility	27.95
Write Now-Vic-20	29.95

EASTERN HOUSE
Rabbit C-64	19.95
Rabbit VIC-20	19.95
MAE C-64	27.95
Telstar 64	19.95
M.L. Monitor 64	18.95

QR & D
Copy Q	*27.95*
GPC Printer Interface	65.00

SPINNAKER (C-64ROM)
Cosmic Life ROM	19.75
Jukebox	19.75
Alphabet Zoo	17.75
All in Color Caves	19.75
Up for Grabs	19.75
Delta Drawing	19.75
Kids on Keys	16.75
Kindercomp	14.75
Facemaker	17.75
Fraction Fever	17.75

BATTERIES INCLUDED
Paper Clip	59.95
Spell Pak	34.95
Consultant	59.95
Paper Clip w/Spell Pak	75.95
Home Pak	34.95
Bus Card	129.95
80 Column Board	109.95

SSI (C-64)
Colonial Conquest	24.75
Wings of War	24.75
Computer Ambush	34.75
Field of Fire	24.75
Fighter Command	36.75
Kampfgruppe	36.75
Mech Brigade	36.75
Market Garden	29.75
Six Gun Shootout	24.75
Computer Baseball	24.75
Computer Quarterback	24.75
Imperium Galactum	24.75
Phantasie	24.75
Cartels & Cutthroats	24.75
50 Mission Crush	24.75
Questron	24.75

SYNAPSE-64
Essex	23.95
Brimstone	23.95
Lode Run Rescue	18.95
Syncalc	33.75
Syncalca Temp	12.95
Mindwheel	23.95

SCARBOROUGH (C-64)
Build A Book	24.75
Improved Mastertype	23.75
NET WORTH	48.75
Mastertype Filer	22.75
Boston 64 Diet	27.75

CONTINENTAL (C-64)
Home Accountant	44.75
1984 Tax Advantage	35.75
1985 C-64 Book of Software	16.95

PERSONAL PERIPHERALS
Super Sketch 64	32.75
Printer Utility	18.75

SUB LOGIC (C-64)
Flight Simulator II	*32.75*
Night Mission Pinball	20.75

MODEMS

US ROBOTICS
Password 1200M	229
Password 1200F	229
Password 300M	139
Password 300F	139
Autodial 212A	359
PCM5	319
PCM64	519
PCM256	769
S-100	379
Courier	469
Microlink	469
Telpac Ms-DOS	79

RACAL-VADIC
2400PC	549
2400PA	799
2400V	559
1200PC	329
300V	205
300PC	199
Genige Software	69

HAYES
Smartmodem 300	133
Smartmodem 1200	377
Smartmodem 1200B	347
Smartmodem 2400	598
Micromodem IIE	135
Westridge (C-64)	59
Mitey Mo Modem	69
Compuserve	19.85

ANCHOR
Volksmodem	55
Volksmodem 12	186
Mark 12	229

TELE LEARNING
Total Telecommunications (C-64)	29.95
AP-250 (300 Baud Apple)	69.95
IB-250 (300 Baud IBM)	69.95

NOVATION
IBM 300/1200 MS-DOS ext	319
IBM 300/1200 CPM-86 ext	319
IBM 300/1200/2400 ext	529
IBM 300/1200/2400 MS-DOS	579
IBM MS-DOS int	325
IBM CPM-86 int	325
Macmodem 300/1200	315
Macmodem 300/1200/2400	585
Apple Cat II	219
Upgrade Apple Cat II	225
Cat 300 Acoustic	139
J Cat RS232	89

MICROBITS
1064Modem	57

MONITORS

SAKATA
SG1000 12" Green	99
SA1000 12" Amber	109
SG1500 12" Green TTL	119
SA1500 12" Amber TTL	129
SC100 13" Color Comp.	209
SC200 13" RGB	389

NEC
JB-1260 Green	95
JB-1201 Green	135
JC 1215 Color	235
JC 1216 RGB	375
JC1460 Color	265
JB-1205 Amber	139

ZENITH
ZVM 122A Amber	75
ZVM 123G Green	75
ZVM 124 Amber IBM	129
ZVM 131 Color	275
ZVM 133 RGB	389
ZVM 135 Composite	449
ZVM 136 Hi Res Color	589
1220, 1230, 1240	CALL

TEKNIKA
MJ-10 Composite	179
MJ-22 RGB	255

TAXAN
115 12" Green Composite	CALL
116 12" Amber Composite	CALL
121 12" Green TTL	135
122 12" Amber TTL	145
220 14" Color Composite	259
410 12" RGB Hi Res IBM	329
420 12" RGB Super Hi IBM	409
440 12" RGB Ultra Hi Res	555
Tilt Stand	35

X-TRON
Comcolor I Composite Green	177

PANASONIC
DT1300D 13" RGB/Composite	247
DTM140 14" RGB/Composite	329
DTH103 10" RGB Hi Res	395
DTS101 10" Composite	175
DT1000G 10" RGB	166
TX12H3P 12" Color	419
TR120M1PA 12" Green	109
TR120MBPA 12" Amber	109
TR122M9P 12" Green IBM	148
TR122MYP 12" Amber IBM	148

AMDEK
300 Green	118
300 Amber	128
310 Amber IBM	155
Color 300 Audio	234
Color 500 Composite	369
Color 600	397
Color 700	495
Color 710	569

PRINCETON GRAPHICS
MAX-12 Amber	185
HX-12 RGB	465
SR-12 RGB	595

AMERICA'S MAIL ORDER HEADQUARTERS

LYCO COMPUTER

WORLD'S LEADER IN SALES & SERVICE

TO ORDER
CALL TOLL FREE
800-233-8760
In PA 1 717-327-1824
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

SpeedScript 3.0 REVISITED

Charles Brannon Program Editor

Since its publication in the March, April, May, and June 1985 issues of COMPUTE!, *response to the* SpeedScript 3.0 *word processor for the Commodore 64, VIC-20, Atari, and Apple computers has been tremendous. Hundreds of readers have written to comment on* SpeedScript, *ask questions, and report minor bugs. This article shows how to fix a few bugs confirmed in the Commodore and Apple versions, including the versions on the March, April, and June* COMPUTE! Disks.

Considering its size and the constraints under which it was developed and distributed, *SpeedScript 3.0* is remarkably bug-free. We made every effort to fully test *SpeedScript*; however, any sophisticated program is bound to have a few nooks and crannies where problems lurk. Even if you haven't encountered any of these bugs, you may want to make the following corrections to ensure that you have the most reliable word processor possible.

Corrections are much harder to make with an all-machine-language program like *SpeedScript* than with a BASIC program. You can't just insert or delete a line of machine language. You wouldn't want to type in *SpeedScript* all over again, either. Instead, corrections to machine language programs are usually in the form of a *patch*—a section of new code which replaces some existing code. Programming a patch properly is no small feat, especially if you need to add additional code. The patch must be merged with the original program to create a new, debugged program. Check the notes below for your computer to see how to enter the corrections.

If you have an Atari, you'll notice there aren't any corrections listed for your version of *SpeedScript*. The Atari version does have the same bug that afflicts the other versions: an odd character is printed whenever the underline toggle command is used. Fortunately, this odd character is a null (CHR$(0)) on the Atari, so nothing is printed, and no harm is done. Therefore, there's no need to make a correction.

Commodore 64 And VIC-20

These are known bugs in Commodore 64 and VIC-20 *SpeedScript 3.0* that these corrections fix:

The buffer should be preserved after an Erase All, but a change to one program module caused the buffer to be cleared with the rest of the text. The fix causes the Erase All routine to skip past the buffer clearing routine.

Sometimes the cursor does not appear when you first run *SpeedScript*, but this problem is also easily fixed with the changes below. Also fixed is a bug that prevents owners of dual disk drives (such as the MSD) from accessing drive 1 for loads and saves. (These bugs were first noted and corrected in the May 1985 "CAPUTE!" section of COMPUTE!.)

Recently confirmed and fixed is the underline bug, which prints an extraneous character whenever the **u** format command (CTRL-£, u) is used.

The RUN/STOP key behaves strangely when you use it to stop printing. Sometimes it works fine. Other times RUN/STOP does stop printing, but when you return to edit mode, it starts inserting five-space tabs. The printer handler is supposed to wait for you to release RUN/STOP before returning to edit mode. At worst, this bug causes your printer to continuously eject paper until you stop it.

Follow these steps to make corrections for all these bugs:

1. Load *SpeedScript*, but do not run it.

2. When you see the READY prompt, enter the following POKEs for your version of *SpeedScript*. Be extremely careful when typing these lines. If you enter any of these numbers incorrectly you may create new bugs that will be difficult to find and fix. Enter the POKEs without line numbers and press RETURN after typing each line:

Commodore 64:

```
POKE 2547,96:POKE 4316,200:POKE
  4946,234:POKE 4947,234:POKE 7716,50
POKE 5785,234:POKE 5786,234:POKE
  5787,234:POKE 7581,11:POKE 7590,76
POKE 7591,86:POKE 7592,29:POKE
  7593,201:POKE 7594,35:POKE
  7595,208:POKE 7596,23
```

VIC-20:

```
POKE 4625,1:POKE 5095,96:POKE
  7370,234:POKE 7371,234:POKE 10054,50
POKE 8145,234:POKE 8146,234:POKE
  8147,234:POKE 9937,11:POKE 9946,76
POKE 9947,138:POKE 9948,38:POKE
  9949,201:POKE 9950,35:POKE
  9951,208:POKE 9952,15
```

3. Save the modified *SpeedScript* by entering SAVE"*filename*" for tape or SAVE"0:*filename*",8 for disk. Be sure to use a different filename than the original *SpeedScript*.

Interface Confusion

Many Commodore *SpeedScript* users have reported problems that are not really the fault of the program at all. Instead, printer interfaces are to blame. The interfaces used to connect Commodore's serial peripheral bus to non-Commodore parallel printers are usually small computers in themselves. They have their own microprocessors, RAM, and ROM. Unfortunately, this intelligence sometimes makes the interfaces too smart for their own good.

For example, many readers complain of bizarre output when a printed line contains an odd number of quote marks ("). This is not the result of anything *SpeedScript* does; rather, your interface counts the occurrences of quotes in the line and turns on its own interpretation of Commodore's infamous quote mode when the count is odd. Other than the bugs described and corrected above, most of the other complaints about *SpeedScript* are actually the result of interface problems.

However, there is a way to get around this. Most printer interfaces have a setting called *transparent mode*, in which they pass all codes along to the printer unaltered. If your interface has such a setting, switch to that mode before printing. This may require changing your formatting commands—for example, when printing in transparent mode you may have to add the a (CTRL-£, a) command to change the output to true ASCII.

Apple II+, IIe, IIc

Apple *SpeedScript 3.0* has two bugs: the underline bug, which prints garbage characters whenever the underline toggle (CTRL-V, U) is used; and the header bug, which shifts the first header on a page to the right of its proper position. Also, Apple *SpeedScript* assumes a variation of the normal SHIFT key modification on the Apple II and II+ (this is the same variation used by *Apple Writer*). If you're having SHIFT key problems with *SpeedScript* on a II or II+, you'll either have to rewire the SHIFT key modification or make the program changes below. See your dealer for help on installing or changing a SHIFT key modification.

These corrections apply only to the DOS 3.3 version of *SpeedScript*. If you want to use this corrected *SpeedScript* with ProDOS, you must use the ProDOS Converter (COMPUTE!, July 1985).

The following program fragments should be typed in with "Apple MLX." Before you can load Apple MLX, you must reconfigure memory with the following POKEs. These POKEs prevent memory conflicts between MLX and *SpeedScript*:

```
POKE 104,32:POKE 8192,0:NEW
```

These are the same POKEs used to type in *SpeedScript* with Apple MLX, and must be used every time you wish to edit *SpeedScript* with Apple MLX.

After typing these POKEs and running Apple MLX, enter the following starting and ending addresses in response to the prompts:

```
STARTING ADDRESS? 0800
ENDING ADDRESS?  1E45
```

Next, press L to select (L)OAD FILE from the menu, and give the filename of the original *SpeedScript* file.

To make the first correction, press E to Enter Data, and enter 1C58 for the address. Type in these seven lines:

```
1C58: D0 10 38 AD D7 1E ED 53 45
1C60: 1E 38 ED D6 1E A8 A9 A0 68
1C68: D0 E0 C9 55 D0 0A AD EC C6
1C70: 1E 49 01 8D EC 1E 10 B1 B4
1C78: C9 63 D0 11 8C E5 1E AE 80
1C80: DD 1E AD DE 1E 20 24 ED 7A
1C88: AC E5 1E D0 9D AE E6 1E F4
```

When you've finished this block, press RETURN on the next line to get back to the menu. Press E to select Enter Data, then enter 1DE0 and type in this last line:

```
1DE0: D4 C9 CE C7 AE AE 8D 00 99
```

Press RETURN on the next line, then press S for (S)ave Data to save your modified copy of *SpeedScript*. Use a different filename than that of the original *SpeedScript*.

If you need to modify your version of *SpeedScript* for the Apple II/II+ SHIFT key problem described above, make the following corrections before saving to disk. (*Do not* make these corrections if you aren't having SHIFT key problems or if you have an Apple IIe/IIc.) The following ten lines of corrections must be made one line at a time. In other words, for each line, you must select option E (Enter Data) from the main menu, type the memory address preceding the colon, enter the numbers, and then press RETURN on the next line to return to the main menu. Then repeat the process for the next line of corrections. When you're done, save the corrected program to disk.

```
0A08: AD 61 C0 0D 44 1E 0D 63 4C

0CD8: 0D 44 1E 0D 63 C0 10 55 B0

0CE8: 0C AD 61 C0 0D 44 1E 0D 6D

0D98: 0D 44 1E 0D 63 C0 30 03 60

0E08: 0C AD 61 C0 0D 44 1E 0D 90

1068: 0D 44 1E 0D 63 C0 30 03 36

11E8: 0D 44 1E 0D 63 C0 10 03 78

1688: 1E 0D 63 C0 10 03 4C D5 7A

1AB8: C0 0D 44 1E 0D 63 C0 30 A2

1B60: 0D 44 1E 0D 63 C0 30 23 64
```

©

Apple Disk Booster

D W. Hoover

This unusual program increases the amount of storage space on Apple disks in DOS 3.3. It runs on any Apple II-series computer with a disk drive.

If you use a disk drive, you know that disk space is a precious commodity. One way to increase disk storage is to buy special hardware. But that's a costly proposition. "Apple Disk Booster" offers a simple, inexpensive alternative. It lets you format new disks with up to five extra tracks, creating more than 21,000 bytes of extra storage space per disk.

Type in Apple Disk Booster and save a copy before you run it. The program is written entirely in Applesoft BASIC. First the program prompts you to insert a blank disk in drive 1, and then it initializes the disk. Because different drives allow a different number of extra tracks, Apple Disk Booster will format only as many extra tracks as your drive can reliably use. The program automatically reads and verifies each extra track. If a track cannot be used, restart the initialization using the next lower track value. When it finishes the initialization, the program displays the number of tracks formatted on that disk.

Since Disk Booster is now the HELLO program on the disk, delete it by typing DELETE DISK BOOSTER,D1 and pressing RETURN. (This prevents you from accidentally running it again.) The disk is now ready for normal use.

As noted above, different drives may not be able to use the same number of extra tracks. If you want to use your modified disk on a different drive, it's a good idea to determine beforehand whether the drive can access the extra tracks. To do this, simply run Apple Disk Booster on the second drive and note the number of tracks displayed when the program ends. Once you know the number of tracks that both drives can access, substitute that number for 40 in line 60 of the program, and run it again as needed.

If you later need to transfer files to a normal disk, use the DOS FILEM utility on the Apple System Master disk.

Extra Tracks

Squeezing extra tracks onto an Apple disk is surprisingly easy to do. This program modifies values used by the DOS routines that initialize the disk and create its Volume Table Of Contents (VTOC). Apple disks are normally formatted with 35 tracks. The first POKE in line 130 forces DOS to format more tracks by substituting a larger number-of-tracks value.

The remaining POKEs in that line adjust the VTOC and bitmap accordingly. The bitmap is a portion of the VTOC that shows where free sectors are located on the disk. Each track has four bytes in the bitmap (two bytes are never used), and each bit represents a corresponding sector in the track. If a bit is off (set to 0), the sector is already allocated. If a bit is on (1), the sector is free. Here is the general format of the VTOC and bitmap.

Byte	Description
00	Not used
01	Track of first catalog sector
02	Sector of first catalog sector
03	DOS release number (3.3, etc.)
04–05	Not used
06	Volume number
07–26	Not used
27	Max number of track/sector pairs
28–2F	Not used
30	Last allocated track
31	Direction of allocation
32–33	Not used
34	Number of tracks per disk
35	Number of sectors per track
36–37	Number of bytes per sector
38–3B	Bit map of track 0
3C–3F	Bit map of track 1
40–43	Bit map of track 2
...	...
BC–BF	Bit map of track 33
C0–C3	Bit map of track 34
C4–FF	Bit maps of additional tracks (if desired)

To ensure that the VTOC and bitmap accommodate the extra tracks, the last two POKE statements in line 130 set new values for the number of tracks on the disk and the size of the bitmap. If 40 tracks are formatted, the bit map is 160 (40 × 4) bytes in size. Of course, it's important to be sure the disk drive can use the extra tracks reliably. Lines 440–510 of Disk Booster contain data for a machine language routine that checks the new tracks. It reads a random sector from each extra track and checks for read-back errors. If an error occurs, we assume the track cannot be accessed and reinitialize the disk without that track.

Apple Disk Booster

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
05 100 TRACKS = 40: REM  # OF TR
       ACKS VALUE
F5 110 BITMAPSZ = TRACKS * 4: RE
       M BIT MAP SIZE
53 120 FOR ML = 768 TO 829: REM
          LOC OF ML ROUTINE
58 130 READ BYTE: POKE ML,BYTE:C
       HKSUM = CHKSUM + BYTE
01 140 NEXT
```

240K Apple® Compatible Computer System

NEW SALE

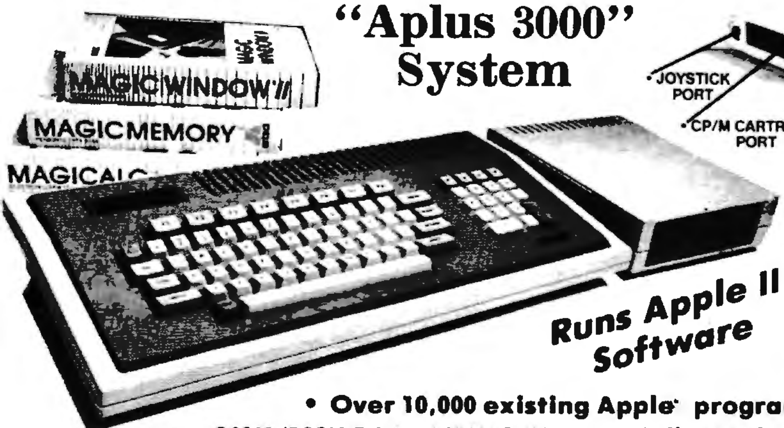
APlus 3000 computer system includes 192K RAM, 48K ROM (32K Microsoft Basic plus 16K ROM Emulator), 160K Laser 5¼" Disk Drive (Runs Apple II Software), Magic Window Wordprocessor, MagiCalc spreadsheet, Magic Memory Database. All for only $499.00

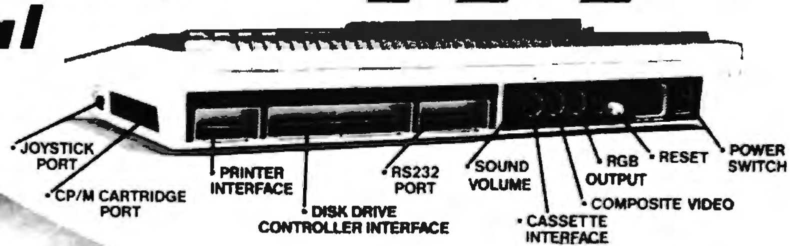
Complete System

$499.00

• 15 Day Free Trial

"Aplus 3000" System





Double Immediate Replacement Warranty

If any of the Aplus 3000 computer system equipment fails due to faulty workmanship or material within 180 days of purchase we will REPLACE it immediately with no service charge!!

- **Over 10,000 existing Apple® programs**
- **Centronics printer interface included**
- **240K (192K RAM, 48K ROM)**
- **ArtSci's Magic Window II, Magic Memory, and MagiCalc included**
- **160K Laser 5¼" Disk Drive (Runs Apple II software)**
- **RGB (80 columns in color) and composite included**

SPECIFICATIONS

A plus 3000 is a complete, self-contained computer based on the popular 6502A microprocessor and can tap into the tremendous software library of Apple II. Features include 192K Bytes RAM, 32KB Enhanced Microsoft BASIC, 80 column text, 560H X 192V color graphic display, 81 key sculptured keyboard and high efficiency switching power supply. Also included as standard are Centronics bus printer interface, Cassette interface, 4 channel sound generator, and 5¼" Apple Compatible Disk Drive.

• TEXT

— 40 columns X 24 rows or 80 columns X 24 rows software selectable.
— 5 X 7 characters in 7 X 8 matrix.
— Upper and lower case characters.
— One of Eight colors for characters/graphics and background. Red, Green, Blue, Cyan, Magenta, Yellow, Black and White.
— Character set with normal, inverse and flashing capabilities.

• GRAPHICS

— 280H X 192V 6 colors — Black, White, Violet, Green, Blue, Orange.
— 280H X 192V 8 colors bit image — Black, White, Red, Green, Blue, Cyan, Magenta, Yellow.
— 560H X 192V 6 colors — Black, White, Violet, Green, Blue, Orange. (High resolution color monitor required)

More Features than Apple® for less than Commodore®

Features	Aplus 3000	Apple IIe	Commodore C-128
RAM	192K	64K	128K
Runs Apple II Software	Yes	Yes	No
Function Keys	24	None	16
4 Voice, 6 Octave Sound	Yes	No	Yes
Composite Video	Yes	Yes	Yes
Disk Drive	Included	Extra Cost	Extra Cost
Numeric Keypad	Included	Extra Cost	Included
Video Cable	Included	Extra Cost	Extra Cost
RGB Color Card	Included	Extra Cost	Included
80 Column Card	Included	Extra Cost	Included
Centronics Printer Interface	Included	Extra Cost	Extra Cost
Drive Controller	Included	Extra Cost	Included
$150 Wordprocessor (Magic Window)	Included	Extra Cost	Extra Cost
$150 Spreadsheet (MagiCalc)	Included	Extra Cost	Extra Cost
$60 Database prg. (Magic Memory)	Included	Extra Cost	Extra Cost
Your Cost	**$499.00**	$1745.00	$1117.90

ACCESSORIES

	LIST	SALE
2nd Disk Drive	$299.95	$149.95
2 professional analog joysticks	$ 39.95	$ 24.95
Z-80 cart. allows CP/M use	$ 99.95	$ 59.95
RS232 adapter	$ 99.95	$ 59.95
R/F Modulator (TV hookup)	$ 29.95	$ 19.95
RGB cable (RGB Monitor hookup)	$ 24.95	$ 19.95
Centronics cable (for Centronics printer)	$ 34.95	$ 24.95
Technical reference manual	$ 29.95	$ 19.95
Comstar 10x 120-140 CPS dot matrix printer	$399.00	$179.00
80 columns Hi-Res Amber Monitor	$199.00	$ 89.95
80 column Hi-Res RGB Monitor	$399.00	$259.00

Super Apple Compatible Disk Drive Sale $149.95.
Quieter, Cooler, Better Disk Drives for your Apple II plus, IIe, IIc (specify when ordering). List $299.95. Sale $149.95.

15 Day Free Trial — If it doesn't meet your expectations within 15 days of receipt, just send it back to us UPS prepaid and we will refund your purchase price!!

Add $25.00 for shipping and handling!!
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. 1 day express mail! We accept Visa and MasterCard. We ship C.O.D. to continental U.S. addresses only. Add $10 more if C.O.D.

COMPUTER DIRECT

We Love Our Customers
22292 N. Pepper Rd., Barrington, Ill. 60010
312/382-5050 to order

```
61 150 IF CHKSUM < > 9911 THEN H
       OME : PRINT "ERROR IN DAT
       A STATEMENTS": GOTO 260
B8 160 POKE 216,0: ONERR GOTO 27
       0: REM RESET ONERR FLAG
A1 170 POKE 48894,TRACKS: POKE 4
       6063,TRACKS: POKE 44725,B
       ITMAPSZ
E8 180 HOME : PRINT "FORMATTING
       ";TRACKS;" TRACKS..."
F7 190 PRINT CHR$ (4);"INIT DISK
        BOOSTER,D1"
C9 200 CALL 768: REM  CALL VERIF
       Y TRACKS ML ROUTINE AT $0
       300
5B 210 PRINT "FORMATTING COMPLET
       E... ";TRACKS;" TRACKS WR
       ITTEN"
E5 220 PRINT
BA 230 PRINT "DON'T FORGET TO LA
       BEL"
88 240 PRINT "YOUR NEW DISK WITH
       "
9F 250 PRINT "THE NUMBER OF TRAC
       KS INITIALIZED!"
96 260 END
70 270 A = PEEK (222): REM  ERRO
       R CODE
76 280 ERRL = PEEK (218) + PEEK
       (219) * 256: REM ERROR LI
       NE
F5 290 IF A < > 8 THEN GOTO 350
84 300 HOME : PRINT "ERROR DETEC
       TED..."
78 310 PRINT "RESETTING NEW TRAC
       K VALUE"
EA 320 PRINT "PRESS <RETURN> TO
       CONTINUE"
0B 330 INPUT "";IN$: IF IN$ < >
       "" THEN GOTO 330
31 340 TRACKS = TRACKS - 1:BITMA
       PSZ = TRACKS * 4: GOTO 16
       0
E3 350 PRINT "ERROR NUMBER ";A;"
        DETECTED IN LINE ";ERRL
83 360 PRINT "CHECK DOS PROGRAMM
       ERS MANUAL"
65 370 PRINT "FOR ERROR TYPE"
95 380 PRINT "AND CORRECT IT ACC
       ORDINGLY"
25 390 GOTO 260
E2 400 REM  **ERROR CODES FOR AP
       PLE DOS 3.3**
87 410 REM
79 420 REM  ERROR #4 = WRITE PRO
       TECTED DISK (REMOVE PROTE
       CT TAB)
AE 430 REM  ERROR #11= SYNTAX ER
       ROR (CORRECT TYPOS)
AF 440 DATA 169,0,141,235,183,14
       1,240,183
AD 450 DATA 169,1,141,244,183,16
       9,16,141
0C 460 DATA 241,183,133,209,173,
       254,190
5E 470 DATA  141,236,183,56,233,
       35,144,22,133
94 480 DATA 210,169,183,160,232,
       32,181
1B 490 DATA 183,176,12,206,236,1
       83,198
3F 500 DATA 210,208,240,198,209,
       208,224
8A 510 DATA 96,169,8,141,92,170,
       76,213,166
                                ©
```

Atari Keypad

R. Alan Belke

Here's an alternative to buying an add-on numeric keypad—simply emulate one in software. This machine language utility redefines part of your existing keyboard as a numeric keypad which you can turn on and off at will. It works on all Atari 400/800, XL, and XE computers with a disk drive.

Because I type in lots of programs from magazine listings, I'm constantly looking for shortcuts. Some of the toughest programs to type are those which consist of hundreds of numbers—data for machine language routines or character sets. Although you can buy a plug-in numeric keypad, there is a more economical alternative.

The solution came to me when I first saw a TRS-80 Model 100 computer. Pressing the NUM key on that portable lap computer turns part of the keyboard (keys M, J-K-L, and U-I-O) into number keys (0, 1-2-3, and 4-5-6, respectively). Since keys 7-8-9 lie right above U-I-O, this lets you enter numbers 0–9 and press RETURN using only the fingers on your right hand. The makeshift keypad is a little slanted, but it's easy to adapt to. Also, notice that the D, A, and T keys are unaffected. You can type DATA statements all day without having to switch off the keypad to type the keyword DATA.

"Atari Keypad" duplicates the Model 100's arrangement on Atari computers. The program works by intercepting the keyboard interrupt routine and redirecting it to a new routine. You can toggle the keypad on and off by simultaneously pressing SHIFT-CTRL-N.

There was one major problem with carrying out this idea. Since Atari Keypad is most useful when entering BASIC programs—or when used in combination with other programs—it has to sit somewhere in memory without interfering with anything else. My original version of Atari Keypad locates itself in the upper half of page 6, a fairly safe section of memory which starts at location 1536 (hex $600). This protects it from the meanderings of Atari BASIC. But realizing that page 6 is used by a host of other programs and routines, including COMPUTE!'s "Automatic Proofreader," I've provided another version that hides in low memory. One version or the other should cover most situations.

Automatic Keypad

Program 1 creates an AUTORUN .SYS file on disk that loads Atari Keypad into page 6 when you boot the system. Program 2 creates an AUTORUN.SYS file that loads Atari Keypad into low memory when you boot. Since the exact location of MEMLO, the low-memory pointer, can vary, the version of Atari Keypad created by Program 2 automatically modifies itself for

any value of MEMLO. That explains all the IF-THEN statements which make the program look so strange.

To use Atari Keypad, try the version created by Program 1 first. If that doesn't work, try the other version. Use the second version with the Automatic Proofreader.

If you already have an AUTORUN.SYS file on your disk that you regularly use, you can append it to the Atari Keypad AUTORUN.SYS file so both will boot automatically. Follow these steps:

1. Boot up Atari DOS 2.0 or 2.5.

2. Rename your existing AUTORUN.SYS file. For example, call it OLDAUTO.

3. Exit to BASIC and run either Program 1 or Program 2 to create the keypad AUTORUN.SYS file on disk.

4. Enter DOS and select the COPY option. When the prompt appears, type OLDAUTO,AUTORUN.SYS/A. Don't forget the /A or you'll end up with your old AUTORUN.SYS file and have to start over again.

If your existing AUTORUN.SYS file happens to use the same memory as Atari Keypad, it would be overwritten when the keypad is booted. Another problem could crop up if your present AUTORUN.SYS file installs a routine at MEMLO and the routine isn't relocatable. If the keypad is installed at MEMLO first, the second routine would wind up at a different address than it was designed for. This would most likely cause the system to crash. Most of the time there's no trouble, however.

If you can touch-type on a keypad, you'll find Atari Keypad a great aid when entering DATA statements. But don't forget it can also be useful with other programs that call for numeric input.

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

Program 1: Atari Keypad For Page 6

```
PA 10 OPEN #1,8,0,"D:AUTORUN
      .SYS"
EK 20 FOR X=1 TO 170
LE 30 READ A:PUT #1,A
PL 40 NEXT X
CP 50 CLOSE #1
DN 60 END
AM 1000 DATA 255,255,128,6,2
        38,6,32,128,6,120
HB 1010 DATA 173,8,2,141,165
        ,6,173,9,2,141
OE 1020 DATA 166,6,169,156,1
        41,8,2,169,6,141
JK 1030 DATA 9,2,88,96,169,6
        ,72,169,167,72
KN 1040 DATA 8,8,76,164,6,17
        3,252,2,201,227
AH 1050 DATA 208,12,174,238,
        6,208,3,232,208,1
GJ 1060 DATA 202,142,238,6,1
        74,238,6,240,42,201
NJ 1070 DATA 37,208,2,169,50
        ,201,1,208,2,169
MH 1080 DATA 31,201,5,208,2,
        169,30,201,0,208
AE 1090 DATA 2,169,26,201,11
        ,208,2,169,24,201
NK 1100 DATA 13,208,2,169,29
        ,201,8,208,2,169
CO 1110 DATA 27,141,252,2,10
        4,64,0,0,6,42
DB 1120 DATA 6,165,12,141,12
        9,6,165,13,141,130
DK 1130 DATA 6,169,128,133,1
        2,169,6,133,13,120
HF 1140 DATA 173,8,2,141,165
        ,6,173,9,2,141
OI 1150 DATA 166,6,169,156,1
        41,8,2,169,6,141
LH 1160 DATA 9,2,88,96,226,2
        ,227,2,0,6
```

Program 2: Atari Keypad For Low Memory

```
KG 10 START=4+PEEK(743)+PEEK
      (744)*256
PB 20 OPEN #1,8,0,"D:AUTORUN
      .SYS"
DO 30 FOR I=1 TO 190
DG 40 READ X:IF I=3 THEN X=S
      TART-INT(START/256)*25
      6
AI 50 IF I=4 THEN X=INT(STAR
      T/256)
KK 60 IF I=5 THEN X=(START+1
      20)-INT((START+120)/25
      6)*256
BL 70 IF I=6 THEN X=INT((STA
      RT+120)/256)
IN 80 IF I=15 THEN X=(START+
      47)-INT((START+47)/256
      )*256
CG 90 IF I=16 THEN X=INT((ST
      ART+47)/256)
LF 100 IF I=21 THEN X=(START
       +48)-INT((START+48)/2
       56)*256
EN 110 IF I=22 THEN X=INT((S
       TART+48)/256)
LI 120 IF I=24 THEN X=(START
       +38)-INT((START+38)/2
       56)*256
FF 130 IF I=29 THEN X=INT((S
       TART+38)/256)
BM 140 IF I=35 THEN X=(START
       +128)-INT((START+128)
       /256)*256
IA 150 IF I=40 THEN X=INT((S
       TART+128)/256)
FJ 160 IF I=46 THEN X=INT((S
       TART+49)/256)
MI 170 IF I=49 THEN X=(START
       +49)-INT((START+49)/2
       56)*256
HN 180 IF I=135 THEN X=(STAR
       T+1)-INT((START+1)/25
       6)*256
FA 190 IF I=136 THEN X=INT((
       START+1)/256)
HE 200 IF I=140 THEN X=(STAR
       T+2)-INT((START+2)/25
       6)*256
EG 210 IF I=141 THEN X=INT((
       START+2)/256)
BN 220 IF I=143 THEN X=START
       -INT(START/256)*256
KA 230 IF I=147 THEN X=INT(S
       TART/256)
PA 240 IF I=155 THEN X=(STAR
       T+47)-INT((START+47)/
       256)*256
IJ 250 IF I=156 THEN X=INT((
       START+47)/256)
PB 260 IF I=161 THEN X=(STAR
       T+48)-INT((START+48)/
       256)*256
IJ 270 IF I=162 THEN X=INT((
       START+48)/256)
PE 280 IF I=164 THEN X=(STAR
       T+38)-INT((START+38)/
       256)*256
JB 290 IF I=169 THEN X=INT((
       START+38)/256)
EP 300 IF I=175 THEN X=(STAR
       T+128)-INT((START+128
       )/256)*256
LD 310 IF I=180 THEN X=INT((
       START+128)/256)
CI 320 PUT #1,X:NEXT I
GA 330 CLOSE #1
GO 340 END
ML 1000 DATA 255,255,0,29,12
        0,29,32,0,29,120
HF 1010 DATA 173,8,2,141,47,
        29,173,9,2,141
DM 1020 DATA 48,29,169,38,14
        1,8,2,169,29,141
OJ 1030 DATA 9,2,88,169,128,
        141,231,2,169,29
CD 1040 DATA 141,232,2,96,16
        9,29,72,169,49,72
LC 1050 DATA 8,8,76,46,29,17
        3,252,2,201,227
DD 1060 DATA 208,12,174,120,
        29,208,3,232,208,1
MA 1070 DATA 202,142,120,29,
        174,120,29,240,42,20
        1
NK 1080 DATA 37,208,2,169,50
        ,201,1,208,2,169
MI 1090 DATA 31,201,5,208,2,
        169,30,201,0,208
PM 1100 DATA 2,169,26,201,11
        ,208,2,169,24,201
NL 1110 DATA 13,208,2,169,29
        ,201,8,208,2,169
DA 1120 DATA 27,141,252,2,10
        4,64,0,0,6,52
JK 1130 DATA 6,165,12,141,1,
        29,165,13,141,2
OK 1140 DATA 29,169,0,133,12
        ,169,29,133,13,120
HK 1150 DATA 173,8,2,141,47,
        29,173,9,2,141
PB 1160 DATA 48,29,169,38,14
        1,8,2,169,29,141
OD 1170 DATA 9,2,88,169,128,
        141,231,2,169,29
DN 1180 DATA 141,232,2,96,22
        6,2,227,2,0,6
```

©

Million-Color Palette For IBM PC & PCjr

John Klein and Jeff Klein

It's amazing but true—with this stunning technique you can generate more than a million apparent color variations on a PCjr. You can even display 256 colors simultaneously. The effects are less dramatic on a PC, but it's still possible to generate many more than the standard 16 colors. The programs require an Enhanced Model PCjr or a PC with color/graphics card, plus a TV set or composite color monitor. The palette is more limited on an RGB monitor, but still impressive.

No longer is your PC or PCjr restricted to a palette of 16 colors and the inability to display them all in higher resolutions. Now you can choose to display 256 colors from a palette of over 1,000,000 colors in high resolution, and display an entire palette of 256 colors in medium resolution. And each color is distinct and solid.

The secret is a combination of a technique called *tile painting* and the trick of fooling a TV or composite monitor into displaying new solid colors. To understand how it works, let's examine the way graphics are stored, changed, and displayed on the IBM video screen.

A Byte Of Pixels

Graphics images are stored differently in the computer's memory for each different graphics mode or screen. In its simplest form, the color of each *pixel*—the smallest controllable dot on the screen—is stored in a section of memory. This video memory is arranged by its location or coordinates on the screen. The image you see on the screen, therefore, is a copy of the contents of video memory. (Actually, screens are divided into several layers when stored in memory, but that's not important for this discussion; we're concerned with how the colors of pixels are represented in memory, not how each pixel is arranged.)

To figure out how many pixels can be represented in a byte of memory, remember that a byte is made up of eight bits, and a bit is the smallest unit of memory (a bit is either a zero or a one). Simply divide the amount of memory required for a certain screen mode by the number of pixels on the screen. The memory requirements for each screen mode are shown in Table 1.

Remember that RGB stands for the three primary colors of light: red, green, and blue. All colors can be made by mixing these three primary colors. That's why RGB monitors, color TVs, and composite color monitors have three electron guns inside their picture tubes, instead of the single gun found in black and white TVs and monochrome monitors. There is a red gun, a green gun, and a blue gun, all of which are controlled by the computer to produce color. If none of the guns is lighting a pixel, the pixel appears black.

Colors are represented in memory by arranging bits to denote which electron guns should be turned on or off when lighting the corresponding pixel. For instance, if a certain pixel is supposed to be blue, the group of bits representing that pixel in memory shows the blue gun is on and the others off. (A bit set to 1 means on, and 0 means off.) All the possible combinations of the three electron guns account for eight colors. To get eight more colors, the intensity, also called *luminance*, is varied by mixing a little white with the first eight colors. That's why the IBM PC and PCjr have a total of 16 color variations: two shades each of eight colors.

Table 2 shows how each of the 16 colors is represented. Remember that each bit turns an electron gun either on or off. Notice how many bits it takes to represent all the

Table 1: Screen Mode Memory Requirements

Screen Mode	Resolution	Number of Colors	Memory per Screen	Pixels/Byte	Bits/Pixel
1	320 × 200	4	16K*	4	2
2	640 × 200	2	16K	8	1
3	160 × 200	16	16K	2	4
4	320 × 200	4	16K	4	2
5	320 × 200	16	32K	2	4
6	640 × 200	4	32K	4	2

*1K = 1024 bytes

possible combinations. It takes four bits, or half of a byte (sometimes called a *nybble*) to represent all 16 colors. So, all screen modes which use four bits to represent a pixel are 16-color modes. Only four-color combinations are possible with two bits, and only two combinations are possible with one bit. That's why some screen modes can display only four or two colors at a time.

The PCjr's PALETTE command can switch which colors are being displayed, but it can't add any more colors. You're still limited to the maximum number of colors for each screen mode.

Tile Painting

Once you're familiar with how pixels are represented in video memory, the technique of tile painting is easier to understand. Tile painting uses the PAINT command found in PCjr Cartridge BASIC and IBM BASICA to fill the bytes of screen memory with certain patterns of ones and zeros. This pattern is programmable, and it represents what is displayed on the TV or monitor. Instead of painting with the actual color, you paint with the bit pattern of the color. By using bit patterns, you can actually paint with more than one color around some specified border color.

PAINT (x,y), CHR$(bit pattern) + CHR$ (bit pattern) + ... ,boundary color

The bit pattern consists of eight bits, so its decimal equivalent can range from 0 to 255 (integers only). The bit pattern must represent the colors of the pixels per byte of the screen mode you're using. This means four colors can be painted at a time in SCREEN 4 and 6, while only two colors can be painted at a time in SCREEN 3 and 5. The color patterns are put in memory next to each other as vertical lines on the screen. The following example paints SCREEN 1 with vertical bands of blue and green lines:

```
10 SCREEN 1:CLS
20 PAINT (1,1),CHR$(102),3
```

The reason why the lines are blue and green can be seen when the number 102 is expressed in binary, revealing the bit pattern:

102 = 01100110

Table 3 shows how decimal 102 is derived from this binary number.

Table 2: Color Bits

Luminance	Red	Bits Green	Blue	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	Light Gray
1	0	0	0	Dark Gray
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Pink
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White

Table 3: Converting Binary to Decimal

Value for each digit	128	64	32	16	8	4	2	1
Binary	0	1	1	0	0	1	1	0

01 = 0001 = blue
10 = 0010 = green

```
128 * 0 =  0
 64 * 1 = 64
 32 * 1 = 32
 16 * 0 =  0
  8 * 0 =  0
  4 * 1 =  4
  2 * 1 =  2
  1 * 0 =  0
         ----
          102
```

SCREEN 1 stores four pixels per byte, so the pattern works out to these colors:

01	10	01	10
blue	green	blue	green

But here's where things get tricky. If the computer is plugged into a color TV or composite color monitor (not an RGB monitor), you won't see the blue and green vertical lines that are supposed to be there. Instead, you'll see a solid bar of color that's sort of blue. And the blue is not one of the normal 16 colors available. It is a new color—one of the 256 shades that can be created this way on SCREEN 1 of the PCjr, and one of the 16 shades that can be created on SCREEN 1 of the PC.

What's happening here is something called *artifacting*. This effect takes advantage of the limited resolution of TVs and composite color monitors. When two very small pixels are placed next to each other on these screens, there isn't enough resolution to display them properly. As a result, the pixels tend to blend together and create a false color—an artifact color. The color wouldn't be visible if the screen had more resolution, which is why you usually need a TV or composite color monitor to observe this effect. RGB monitors have enough resolution to display the pixels as they're supposed to appear.

Creating New Colors

If the binary pattern 10 01 10 01 is used in the above example instead of 01 10 01 10, the shade is slightly different—blue-green-blue-green does not appear the same as green-blue-green-blue on a color TV or or composite monitor. They mix differently to create an entirely new shade of blue-green.

The PC can mix a fewer number of colors than the PCjr for two reasons. The first is that the PC has only two graphics modes, SCREEN 1 and SCREEN 2. Tile painting produces only 16 colors in SCREEN 1 and five shades of gray in SCREEN 2. Still, these are more colors than what are normally available in these modes. The second reason is that the PC does not have a PALETTE command as the PCjr does. The PC does have a second color palette in SCREEN 1, but the mixed colors look the same as the first palette on a color TV or composite monitor.

On SCREEN 6, available only in PCjr Cartridge BASIC, there are four pixels per byte. Because the pixels are very small (640 × 400 per screen), vertical bands of four different colors can be mixed to form shades of any color. In medium resolution, 320 × 200, vertical bands of two different colors form new solid colors. Tile painting doesn't work in low resolution, 160 × 200, because the pixels are too large.

For a demonstration of how closely spaced vertical bands create new colors, enter and run Program 1 (for the PCjr only). Using the LINE command instead of PAINT, line 20 fills the first 40 columns of SCREEN 6 with purple bands on every line that is a multiple of four: 0, 4, 8, 12, and so on. Line 30 fills the next 40 columns with the same color on every vertical line that is a multiple of four plus one: 1, 5, 9, 13, and so on. Then the program fills the screen with lines of the other two colors available in SCREEN 6. The result, on a TV or composite monitor, is 12 different colors instead of the four you'd expect.

Adding up all the different combinations of four colors results in 256 shades, and all 256 can be displayed on the screen at the same time. When you take into account that the PALETTE command can change any of the four basic colors into any of the other 16 colors, there are 1,092,016 possible shades in high resolution.

Program 2 (for the PCjr only) proves it can be done. This program displays 256 shades on the screen by drawing the vertical lines using only the first four colors. After painting all the shades, it randomly changes the palettes. If the colors selected by the PALETTE command were never repeated, it would take about an hour and a half to cycle through all one million colors.

Colors In Other Modes

In SCREEN 5, there are 256 possible colors, as demonstrated by Program 3 (also for the PCjr only). In SCREEN 4 and SCREEN 1, which are the same resolution, only four basic colors are available, so tile painting lets us display up to 16 hues simultaneously. With the PALETTE command on the PCjr, you can select these 16 colors from 256 possibilities. Program 4 displays 16 shades, then uses the PALETTE command to get the rest. Vertical bands with four colors don't blend in this mode, so somehow bands of two must be painted. The secret is in line 40. Since there are four pixels per byte, the last half of the byte has to be reflected in the first half. This technique insures that only two colors are in each band of four. The first half is the same as the last half, so the first band of two will be the same as the last band of two. Program 4 will also work on the PC, but without the PALETTE command (line 80) you are limited to only 16 colors.

Tile painting doesn't work correctly in SCREEN 2, high resolution with two colors, because this screen is always in black and white. However, you can get five shades of gray, as shown by Program 5 (for PC and PCjr). Solid lines form the brightest white. Lines separated by one line of black give the next-brightest white. Lines separated by two or three lines of black yield the next two shades. The middle gray can't be displayed when using the PAINT command, because it's not possible to create a bit pattern that represents two blacks and then a white. Table 4 shows which bit patterns generate the various shades of gray.

Tile painting doesn't work at all in SCREEN 3 because the pixels are too large. To see a demo of tile painting in SCREEN 1 for the PC or PCjr, run Program 6. It fills the screen with circles, displaying up to 256 colors on the PCjr and 16 colors on the PC.

Program 7, for the PC and PCjr with an RGB monitor, demonstrates the usefulness of the many new colors in a fascinating experiment. It uses SCREEN 1 and tile painting, but in a different way than seen above. Closely spaced vertical lines don't blend together on an RGB monitor, so the previous technique won't work. So instead, Program 7 uses the second part of the PAINT command. The first CHR$(*bit pattern*) controls the horizontal line above the second CHR$(*bit pattern*). Now the PAINT command can control the horizontal as well as the vertical lines, forming a checkerboard.

Although the checkerboard blends the lines together to create new colors, the colors aren't as solid as those produced by vertical lines on a TV or composite monitor. Indeed, the effect won't look very pretty on a TV or composite monitor; it's passable on an RGB.

Program 8 (for the PCjr only) employs the same technique as Program 7, but uses SCREEN 5 on the PCjr to create all 240 possible colors on the RGB monitor at once. The PALETTE command won't create any new shades here, because all 16 colors and their possible combinations are displayed.

Program 9 (for the PCjr only) is the same as the last two, but uses SCREEN 6 on the PCjr. It does a much better job of blending, although the colors still aren't perfectly solid. Ten shades are displayed at once and the PALETTE command cycles through all 240 possible shades.

Painting Your Own Programs

To use the new colors in your own programs, simply choose one of the following example programs which uses the same screen mode. Table 5 summarizes the programs and the number of color variations possible in each.

If you're programming on a PCjr, remove the lines that deal with changing the palette. You can change palettes on the PCjr in direct mode until most of the shades you want are on the screen. We suggest not changing the palettes in the 16-color modes, because the unchanged palette creates the widest variety of colors with the least amount of extra work.

In four-color modes, the screen displays 16 shades. Pick the color you want, then refer to Table 6 for the corresponding decimal and hexadecimal translations of the bit patterns required.

If you're using a 16-color mode with a TV or composite monitor, the screen displays 256 shades and the bit patterns can be figured as follows: First choose the color. Then, starting at zero at the upper-left corner of the screen, count in hex across the screen to the column with the color you want. Remember to count in hex (0 through 9, then A through F). Then, still working in hex, count the number of rows down to the color you want. These two numbers form the bit pattern of the chosen color. Use them as shown below:

PAINT (x,y),CHR$(&H row column), boundary

Example: If row = A and column = 2, then

PAINT (x,y),CHR$(&HA2),boundary

If you're using an RGB monitor with a 16-color mode, choose which two of the 16 colors to make into the checkerboard. Then write each of their numbers in hex (0–F). Use these numbers as the bit pattern as shown below. Switching the first and second colors will create the checkerboard.

PAINT (x,y),CHR$&H 1st color 2nd color)+CHR$(&H 2nd color 1st color),boundary

Example: If 1st color = B (light cyan) and 2nd color = 2 (green), then

PAINT (x,y),CHR$(&HB2)+CHR$(&H2B)

IBM boasts of only the checkerboard technique for shading colors. I find the other method more fascinating. Now you can enhance your screens with a new palette of bright, solid colors, which formerly were thought to be impossible on an IBM.

Table 4: Gray Scales in SCREEN 2

	Binary	Decimal	Hex	Shade
color 1 =	1 1 1 1 1 1 1 1	= 256 =	&HFF =	White
	0 1 0 1 0 1 0 1	= 85 =	&H55 =	Dull White
	(Not accessible)		=	Middle Gray
	0 0 0 1 0 0 0 1	= 17 =	&H11 =	Dark Gray
color 0 =	0 0 0 0 0 0 0 0	= 0 =	&H00 =	Black

Table 5: Program Descriptions

Program	Screen Mode	Max Colors	Colors per Screen	PC or PCjr	Display Device
Program 1	SCREEN 6	1,092,016	256	PCjr	TV or CC*.
Program 2	SCREEN 6	1,092,016	256	PCjr	TV or CC
Program 3	SCREEN 5	256	256	PCjr	TV or CC
Program 4	SCREEN 1 or 4	256	16	PCjr	TV or CC
	SCREEN 1	16	16	PC	TV or CC
Program 5	SCREEN 2	5	5	PC/PCjr	TV or CC
Program 6	SCREEN 1	256	16	PCjr	TV or CC
		16	16	PC	TV or CC
Program 7	SCREEN 1 or 4	240	10	PCjr	RGB
	SCREEN 1	20	10	PC	RGB
Program 8	SCREEN 5	240	240	PCjr	RGB
Program 9	SCREEN 6	240	10	PCjr	RGB

*CC = Composite color monitor

Table 6: Translations of Bit Patterns in Four-Color Modes

	TV or Composite		RGB	
Shade Position	Decimal	Hex	Decimal	Hex
0	0	&H00	0 + 0	&H00 + &H00
1	17	&H11	17 + 70	&H11 + &H44
2	34	&H22	34 + 136	&H22 + &H88
3	51	&H33	51 + 204	&H33 + &HCC
4	70	&H44	70 + 17	&H44 + &H11
5	85	&H55	85 + 85	&H55 + &H55
6	102	&H66	102 + 153	&H66 + &H99
7	119	&H77	119 + 221	&H77 + &HDD
8	136	&H88	136 + 34	&H88 + &H22
9	153	&H99	153 + 102	&H99 + &H66
10	176	&HAA	176 + 176	&HAA + &HAA
11	187	&HBB	187 + 238	&HBB + &HEE
12	204	&HCC	204 + 51	&HCC + &H33
13	221	&HDD	221 + 119	&HDD + &H77
14	238	&HEE	238 + 187	&HEE + &HBB
15	255	&HFF	255 + 255	&HFF + &HFF

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!. Also, see Table 5 for a description of the programs.

Program 1: PCjr

```
IE 10 CLEAR,,,32768!:SCREEN 6:CL
      S:KEY OFF
DJ 20 FOR X=0 TO 40 STEP 4:LINE
      (X,0)-(X,200),3:NEXT
JH 30 FOR X=41 TO 80 STEP 4:LINE
       (X,0)-(X,200),3:NEXT
IA 40 FOR X=82 TO 120 STEP 4:LIN
      E (X,0)-(X,200),3:NEXT
GJ 50 FOR X=123 TO 160 STEP 4:LI
      NE (X,0)-(X,200),3::NEXT
FA 60 FOR X=160 TO 200 STEP 4:LI
      NE (X,0)-(X,200),1:NEXT
HN 70 FOR X=201 TO 240 STEP 4:LI
      NE (X,0)-(X,200),1:NEXT
FO 80 FOR X=242 TO 280 STEP 4:LI
      NE (X,0)-(X,200),1:NEXT
NM 90 FOR X=283 TO 320 STEP 4:LI
      NE (X,0)-(X,200),1:NEXT
OO 100 FOR X=320 TO 360 STEP 4:L
       INE (X,0)-(X,200),2:NEXT
MN 110 FOR X=361 TO 400 STEP 4:L
       INE (X,0)-(X,200),2:NEXT
OL 120 FOR X=402 TO 440 STEP 4:L
       INE (X,0)-(X,200),2:NEXT
GE 130 FOR X=443 TO 480 STEP 4:L
       INE (X,0)-(X,200),2:NEXT
```

Program 2: PCjr

```
IE 10 CLEAR,,,32768!:SCREEN 6:CL
      S:KEY OFF
LO 20 RANDOMIZE TIMER:Z=-1:A=INT
      (640/16)
HB 30 FOR Y=0 TO 15
OE 40 FOR X=0 TO 15:Z=Z+1
EC 50 LINE (X*A,Y*12.5)-(X*A+A,Y
      *12.5+12.5),3,B
FC 60 IF Z<>0 THEN PAINT (X*A+1,
      Y*12.5+1),CHR$(Z),3
MH 70 LINE (X*A,Y*12.5)-(X*A+A,Y
      *12.5+12.5),0,B
JI 80 NEXT X,Y
DD 90 PALETTE RND*3,RND*15:GOTO
      90
```

Program 3: PCjr

```
HF 10 CLEAR,,,32768!:SCREEN 5:CL
      S:KEY OFF
E9 20 RANDOMIZE TIMER:Z=-1:A=INT
      (320/16)
H9 30 FOR Y=0 TO 15
OE 40 FOR X=0 TO 15:Z=Z+1
ED 50 LINE (X*A,Y*12.5)-(X*A+A,Y
      *12.5+12.5),3,B
PC 60 IF Z<>0 THEN PAINT (X*A+1,
      Y*12.5+1),CHR$(Z),3
```

```
MH 70 LINE (X*A,Y*12.5)-(X*A+A,Y
      *12.5+12.5),0,B
JI 80 NEXT X,Y
KC 90 GOTO 90
```

Program 4: PC/PCjr

```
CA 10 SCREEN 1:CLS:KEY OFF:COLOR
       ,0
MM 20 RANDOMIZE VAL(RIGHT$(TIME$
      ,2)):Z=-1:A=INT(320/16):Y=
      0
OD 30 FOR X=0 TO 15:Z=Z+1
II 40 LINE (X*A,0)-(X*A+A,200),3
      ,B
KI 50 IF Z<>0 THEN PAINT (X*A+1,
      1),CHR$(Z+Z*16),3
CD 60 LINE (X*A,0)-(X*A+A,200),0
      ,B
QM 70 NEXT X
AL 80 PALETTE RND*3,RND*15:GOTO
      80:' Remove this line for
      PC
```

Program 5: PC/PCjr

```
CF 10 SCREEN 2,1:CLS:KEY OFF
MJ 20 FOR X=1 TO 100:LINE (X,1)-
      (X,200),1:NEXT X
DH 30 FOR X=101 TO 200 STEP 2:LI
      NE (X,1)-(X,200),1:NEXT X
HN 40 FOR X=201 TO 300 STEP 3:LI
      NE (X,1)-(X,200),1:NEXT X
KD 50 FOR X=301 TO 400 STEP 4:LI
      NE (X,1)-(X,200),1:NEXT X
IE 60 GOTO 60
```

Program 6: PC/PCjr

```
CA 10 SCREEN 1:CLS:KEY OFF:COLOR
       ,0
LO 20 RANDOMIZE VAL(RIGHT$(TIME$
      ,2))
GD 30 X=RND*320:Y=RND*200:R=RND*
      10+10:TILE=INT(RND*(15)+1)
BM 40 CIRCLE (X,Y),R,3:PAINT (X,
      Y),CHR$(TILE+TILE*16),3:CI
      RCLE (X,Y),R,0
AE 50 IF RND*10>8 THEN PALETTE R
      ND*3+1,RND*15:' Remove thi
      s line for PC
GA 60 GOTO 20
```

Program 7: PC/PCjr

```
CA 10 SCREEN 1:CLS:KEY OFF:COLOR
       ,0
CD 20 RANDOMIZE VAL(RIGHT$(TIME$
      ,2)):Z=-1:A=INT(320/16):Y=
      0:C=0
OD 30 FOR X=0 TO 15:Z=Z+1
NN 40 LINE (X*A,0)-(X*A+A,200),3
      ,B:Y=Z+Z*16:Q=Y*4:R=INT(Q/
      256):Q=Q-R*256+R
FF 50 IF Z<>0 THEN PAINT (X*A+1,
      1),CHR$(Y)+CHR$(Q),3
CD 60 LINE (X*A,0)-(X*A+A,200),0
      ,B
QM 70 NEXT X
GL 80 PALETTE RND*3,RND*15:GOTO
      80:' Remove this line for
      PC
EM 90 C=1-C:COLOR ,C:FOR Z=1 TO
      100:NEXT:GOTO 80:' Remove
      this line for PC
```

Program 8: PCjr

```
HY 10 CLEAR,,,32768!:SCREEN 5:CL
      S:KEY OFF
BB 20 RANDOMIZE TIMER:Z=-1:A=INT
      (320/16)
HB 30 FOR Y=0 TO 15
JE 40 FOR X=0 TO 15:Z=Z+1
JO 50 LINE (X*A,Y*12.5)-(X*A+A,Y
      *12.5+12.5),3,B:Q=(Z)*16:R
      =INT(Q/256):Q=Q-R*256+R
OC 60 IF Z<>0 THEN PAINT (X*A+1,
      Y*12.5+1),CHR$(Z)+CHR$(Q),
      3
MH 70 LINE (X*A,Y*12.5)-(X*A+A,Y
      *12.5+12.5),0,B
JI 80 NEXT X,Y
KC 90 GOTO 90
```

Program 9: PCjr

```
IE 10 CLEAR,,,32768!:SCREEN 6:CL
      S:KEY OFF
BB 20 RANDOMIZE TIMER:Z=-1:A=INT
      (640/16):Y=0
GD 30 FOR X=0 TO 15:Z=Z+1
NW 40 LINE (X*A,0)-(X*A+A,200),3
      ,B:Y=Z+Z*16:Q=Y*4:R=INT(Q/
      256):Q=Q-R*256+R
KF 50 IF Z<>0 THEN PAINT (X*A+1,
      1),CHR$(Y)+CHR$(Q),3
CD 60 LINE (X*A,0)-(X*A+A,200),0
      ,B
GM 70 NEXT X
BE 80 PALETTE RND*3,RND*15:GOTO
      80
```

©

Computed GOTOs And GOSUBs For Commodore 64

William M. Wiese

This short, relocatable utility permits computed GOTO and GOSUB statements in Commodore 64 BASIC.

You're probably familiar with GOTO and GOSUB statements, which pass control to another line in a BASIC program. In Commodore BASIC, these keywords can only be followed by a line number, as in GOTO 100. Some other versions of BASIC let you replace the line number with a variable, such as GOTO X, or even a complex expression, such as GOSUB X+100*ABS(Y). Since the line number is computed from the expression, the term *computed* GOTO or GOSUB is used to describe this feature.

Computing the destination from an expression offers two advantages. You can make your programs easier to understand by using meaningful variable names for subroutines instead of line numbers—for instance, replacing GOSUB 1000 with GOSUB DRAW. And computed GOTO and GOSUB statements offer a more flexible and efficient means of controlling program flow. For example, say that you write a program with six subroutines: The first starts at line 1000, the second is at 2000, and so on up to line 6000. The usual way to direct the computer to the correct subroutine would be with an ON-GOSUB statement:

```
ON A GOSUB 1000,2000,3000,4000,
  5000,6000
```

With computed GOSUBs, the same thing can be accomplished with the more compact statement GOSUB A. If A=1000, the computer performs the subroutine at line 1000. If A=2000, then GOSUB 2000 is performed, and so forth.

The program below adds both of these useful statements to Commodore 64 BASIC. Type in and save a copy before you run it. Enter line 130 exactly as shown (do *not* add an extra comma after the number 57812). The program automatically saves a machine language program named "CG0.ML" on disk. If you're using tape, change the ,8 to ,1 in line 130. Once the program has been created, load it with LOAD"CG0.ML",8,1 for disk or LOAD"CG0.ML",1,1 for tape.

Expressive Programming

Once the routine is loaded into memory, you can perform a computed GOSUB with the statement SYS 49152,*expression*. Replace *expression* with any variable or expression that evaluates to a valid line number (from 0–63999). Use SYS 49179,*expression* to perform a computed GOTO. For example, if the variable DRAW equals 1000, then SYS 49152,DRAW does the same thing as GOSUB 1000, and SYS 49179,DRAW does the same thing as GOTO 1000.

It's usually advantageous to substitute variables for 49152 and 49179 in such SYS statements. For instance, your program might contain the following lines:

```
10 CG=49152
90 SYS CG,DRAW
```

In Commodore BASIC, using variables in place of numbers speeds up a program. It takes the computer less time to find the value of the variable CG than it does to calculate the value of a constant such as 49152.

In some cases, you may want to use the memory locations starting at 49152 for a different machine language routine. If you use a disk drive, you can move the computed GOTO/GOSUB routine to the cassette buffer, which begins at location 828. Simply change lines 100, 140, 150, and 210 as shown here:

```
100 FOR I=828 TO 878   :rem 234
140 POKE193,60:POKE194,3
                        :rem 89
150 POKE174,110:POKE175,3
                       :rem 132
210 DATA 76,91,3,169,255,133
                       :rem 102
```

Before running the modified program, replace the name CG0.ML in line 130 with a new name (CGML/828 or whatever) that reflects the alteration. Then load the program as described above and use SYS 828,*expression* for computed GOSUB and SYS 855,*expression* for computed GOTO.

Occasionally, computed GOTOs and GOSUBs don't seem to work correctly. For example, suppose a program contains the statement SYS 49179, 5*COS(X). If X has the value 0, then this statement should do the same thing as GOTO 5 (to confirm this, type PRINT 5*COS(0) and press RETURN). Instead, the computer performs the equivalent of GOTO 4. Such effects are the result of slight rounding errors caused when the computer converts numbers from one format to another. The 64—like virtually every other computer—stores and manipulates numbers internally in a different format from the decimal numbers we ordinarily use. In this case, the computer evaluates 5*COS(0) as 4.99999999, then throws away the fraction, ending up with the integer (whole) value of 4. To prevent such rounding errors, add a small number (.00001 is a good value) to the expression. For instance, SYS 49179, 5*COS(0)+ .00001 correctly performs GOTO 5.

How It Works

Computed GOTOs and GOSUBs are surprisingly easy to add to Commodore BASIC. When the computer performs an ordinary GOSUB, it "remembers" its current place in the program by storing an address and the current BASIC line number in a special memory area called the stack. An additional byte is stored on the stack to show that a GOSUB caused the stack entry. This makes it possible for the computer to find its way back to the right spot when the subroutine ends with RETURN.

From this point onward, GOSUB and GOTO share the same code and work exactly the same. The computer looks at the ASCII line number stored in the BASIC program text (if it finds anything other than ASCII numerals, it stops with an UNDEF'D STATEMENT error). Then it converts the line number to integer form and stores it in locations 20–21. Finally, the computer searches the program text for the matching line number and (if the line exists) continues forward.

To make computed GOTOs and GOSUBs possible, this utility duplicates the way a GOSUB statement stores return information on the stack. But it adds something new to the common routine that retrieves the line number from the program text. Instead of getting the line number in the old manner, we call BASIC's main evaluation routine at memory address 44446. This routine, usually labeled FRMEVL, can evaluate any BASIC expression (unlike the normal routine, which accepts only numerals). After calling a second routine at 47095 to convert the number into a two-byte address, the utility stores the line number in locations 20–21. Since this is exactly where the GOTO routine expects to find the line number, we then jump into the computer's normal routine at address 43171.

Computed GOTOs And GOSUBs

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
100 FOR I=49152 TO 49202
                      :rem 167
110 READ A:POKE I,A    :rem 14
120 NEXT              :rem 210
130 SYS 57812"CGO.ML",8
                      :rem 226
140 POKE 193,0 :POKE 194,192
                      :rem 140
150 POKE 174,51:POKE 175,192
                      :rem 193
160 SYS 62954         :rem 160
170 DATA 169,0,133,2,169,3
                      :rem 250
180 DATA 32,251,163,165,123,72
                      :rem 193
190 DATA 165,122,72,165,58,72
                      :rem 156
200 DATA 165,57,72,169,141,72
                      :rem 152
210 DATA 76,31,192,169,255,133
                      :rem 201
220 DATA 2,32,253,174,32,158
                       :rem 90
230 DATA 173,32,247,183,32,163
                      :rem 195
240 DATA 168,165,2,240,1,96
                       :rem 47
250 DATA 76,174,167   :rem 176
```

©

Refurbish Your 64

Richard Roffers And Jeffrey Hock

Enhance your Commodore 64 by modifying its built-in operating system. This unusual program eliminates several annoying bugs and adds convenient new features as well.

While the Commodore 64 is a remarkable computer, its operating system, the *Kernal*, has a few notorious shortcomings. Some 64s lock up if you type a line more than 80 characters long at the bottom of the screen, then delete a character. POKEs to screen memory are invisible on some models, and none of them handle the ASC value of a null string ("") correctly. "Refurbish Your 64" corrects these problems and makes several other improvements as well. Of course, the changes are only temporary. Restarting the computer returns it to normal.

Type in and save the accompanying program, and be sure to remove any cartridges from the expansion port. When you run the program, the computer behaves as if you just turned the power on—but with a difference. The startup message reveals that the Kernal has been modified. As you may know, the 64 has programmable RAM (Random Access Memory) "underneath" the ROM (Read Only Memory) addresses where BASIC and the Kernal are stored. This program works by copying BASIC and the Kernal from ROM into the underlying RAM, modifying them, and then turning off the ROM to make the computer use the RAM-based Kernal and BASIC.

Don't worry if that seems unclear. You can use this program without knowing how all the details work. For now, notice that the number of bytes free is shown as 51,216, far more than the usual number (38,911). Since the 64 now has RAM instead of ROM at locations 40960–49151, it thinks its BASIC program space stretches all the way from location 2048 to 53264. But that's just an illusion. We can't use the RAM from 40960–49151 without destroying the modified BASIC we just put there. Before you do anything else, reset the top-of-memory pointer to its normal value by typing the following line and pressing RETURN:

```
POKE 55,0:POKE 56,160:POKE 643,0:POKE
  644,160:NEW
```

This line must always be entered immediately after you run the program (or perform a cold start with SYS 64738). Once that is done, your modified 64 is ready to go. Let's look at each modification in turn and note how you can customize this program to suit your own tastes.

Screen Colors

Everyone seems to have different preferences for default screen colors. If you don't like the usual colors, they're easy to change. Lines 1460, 1500, and 1550 define the default background, border, and character colors, respectively. Change the values in those lines to whatever color numbers you like, then rerun the program. The chosen colors will reappear whenever you press RUN/STOP–RESTORE or cause a cold start with SYS 64738.

In this and other parts of the program, you'll notice that each group of DATA statements represents one change, with the first DATA statement in each group specifying the starting address and the number of bytes to be changed in ROM. The remaining DATA statements in each group contain the actual bytes that are POKEd into RAM to make the change. The REM statements in each section explain which values you may change.

Default Device

Although most 64 owners use a disk drive, the default device for LOAD, SAVE, and VERIFY is the Datassette. Lines 1600–1700 change the default device number to 8 so that a command like SAVE "FILE" (without the ,8) saves to the disk drive rather than cassette. You can still use tape by adding device number 1 to your commands (for instance, SAVE"FILE",1). However, for nonrelocating disk loads you must still add ,8,1 to the command (as in LOAD"FILE",8,1). Replace the 8 in line 1650 with a 1 if you don't have a disk drive.

Auto Load/Run

When you press SHIFT-RUN/STOP, normally the 64 loads and runs the first program on tape. Since the disk drive is now the default device, the SHIFT-RUN/STOP routine has been modified to perform the equivalent of LOAD"*",8 followed by RUN. This was necessary because disk loads (unlike tape) always require a filename. The command LOAD"*",8 normally loads the first program file on the disk. However, in some cases the wildcard symbol * is equal to the last filename used rather than the first file on disk.

Screen POKEs

Depending on the age of your 64, POKEs to screen memory (like POKE 1024,42) may produce white characters, invisible characters (the same color as the background), or characters the same color as the cursor. This program makes all screen POKEs appear in the cursor color as on the newest 64s.

Moving CLR/HOME

This is a change you may or may not find desirable, so we've made it optional. Some people often hit the CLR/HOME key by accident when trying to press the INST/DEL key. Instead of inserting a character in a line that you're editing, the screen clears and your work is lost. To eliminate this problem, remove the REMs from lines 2060–2280. This modification exchanges the positions of the CLR/HOME key and the £ key, moving CLR/HOME to a less vulnerable position. If you make this change and use this program frequently, you may want to exchange the keycaps for those keys as well. The keycaps are easily removed by prying them straight up.

INPUT Prompt

As you probably know, INPUT permits a prompt message (for example, INPUT"YOUR CHOICE" ;A$ prints YOUR CHOICE?). If the prompt message is longer than one screen line, INPUT either tacks the entire prompt message onto the front of your response (when accepting string input) or causes a REDO FROM START error (when accepting numeric input). Lines 2310–2340 eliminate this bug.

LIST Freezing

The 64 normally lets you slow screen scrolling (caused by PRINTing or LISTing to the screen) by pressing the CTRL key. In many cases, it's more convenient to freeze such displays rather than merely slow them down. When the modified Kernal is installed, SHIFT (or SHIFT-LOCK) will freeze screen scrolling. Pressing CTRL while a screen is frozen causes it to scroll at the normal rate as long as both SHIFT and CTRL are pressed.

Keyboard Buffer Option

Since this modification may not be useful to everyone, we've made it optional. The computer's keyboard buffer stores keystrokes temporarily. If you type faster than the computer can digest the keystrokes, the keyboard buffer remembers them until the system is ready. The buffer is normally ten characters long; when you type more than ten characters "ahead" of the system, the extra characters are lost. There are times when a longer buffer would be useful—for example, to prevent a fast typist from overflowing the buffer or to let the computer execute long direct-mode commands as if they were typed directly on the keyboard.

If you remove the REMs from lines 2530–2720, the keyboard buffer is moved from its normal location (631–640) to an 80-byte area in the cassette buffer (starting at 828). Note that many programs expect to find the keyboard buffer in its normal place and may misbehave or crash as a result of this relocation. For this reason, be careful to test the program after incorporating this change.

Power-Up Message

This message is displayed when you first run the program, and thereafter (as long as the computer remains on) when you cause a cold start with SYS 64738. After performing a cold start, you must always reset the top-of-memory pointer as explained above. Lines 3190–3220 contain the data for the new startup message. The numbers are ASCII character codes (listed in your user's guide). To replace this message with one of your own, replace these codes with ASCII codes for the characters that you want. Do not try to add any extra characters (there's no room for them in the modified Kernal). Note that *the last ASCII code must be a 0.* If you omit the final 0, the computer may crash when it tries to print the message.

Screen Lockup

Some early models of the 64 suffer from the infamous bottom-of-screen lockup bug, caused when you type in a line more than 80 characters long at the bottom of the screen, then delete a character. This bug has been eliminated.

New Erase Key

Commodore computers provide excellent full-screen editing capabilities. However, some people prefer an "erase" key that acts like a mini black hole. When you press it, the character under the cursor disappears, and everything to the right of that character moves left one space. This is the equivalent of pressing CURSOR RIGHT followed by DELETE. We chose to make the seldom-used SHIFT-£ combination into an erase key. To erase a character, just press SHIFT-£. The new erase key repeats when you hold it down, just like the cursor keys and INST/DEL (SHIFT-9 now repeats as well, an unavoidable side effect). If you need the graphics character that SHIFT-£ normally prints, use PRINT CHR$(169).

Null String Fix

The 64's normal ASC function can't handle a null string (two quotation marks with nothing between them). A statement like PRINT ASC("") causes an ILLEGAL QUANTITY error. This is one of the easiest ROM bugs to fix, requiring only a one-byte change.

For most ordinary programming, a RAM-based Kernal and BASIC work just fine. However, since other programs may use the same RAM area (to store a high-resolution screen, make other modifications to BASIC, or whatever), you must be alert for conflicts. If another program POKEs into the RAM where the modified BASIC and Kernal are stored, the computer may crash. As mentioned earlier, turning the computer off and on restores the original, ROM-based versions of BASIC and the Kernal. The only way to make these changes permanent is to store the modified BASIC and Kernal in EPROM (Eraseable Programmable ROM) chips and substitute them for the existing ROM chips—a job requiring specialized equipment and expertise.

Refurbish Your 64

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
1000 REM COMMODORE 64 KERNAL M
     ODIFIER.              :rem 238
1030 REM{4 SPACES}THIS SECTION
       OF CODE IS A       :rem 130
1040 REM{4 SPACES}SMALL MACHIN
     E LANGUAGE           :rem 91
1050 REM{4 SPACES}PROGRAM WHIC
     H DOWNLOADS          :rem 224
1060 REM{4 SPACES}THE KERNAL I
     NTO THE              :rem 100
1070 REM{4 SPACES}UNDERLYING R
     AM AND THEN          :rem 143
1080 REM{4 SPACES}BANKS OUT TH
     E KERNAL ROM.        :rem 206
1150 PRINT"{CLR}{9 DOWN}
     {15 SPACES}{11 @}"
                          :rem 208
1200 PRINT"{15 SPACES}{RVS}PLE
     ASE WAIT"            :rem 149
1210 PRINT"{2 DOWN}{4 SPACES}D
     ON'T FORGET TO RESET THE
     {SPACE}TOP OF"       :rem 105
1220 PRINT"{DOWN}{7 SPACES}MEM
     ORY POINTERS (SEE TEXT)."
                          :rem 149
1230 FOR I=49152 TO 49212:READ
      B:POKE I,B:NEXT I:rem 87
1240 DATA 169,0,133,251,169,16
     0,133,252,76,24,192,234,1
     69,0,133,251         :rem 226
1250 DATA 169,224,133,252,76,2
     4,192,234,162,32,160,0,17
     7,251,145,251         :rem 20
1260 DATA 200,208,249,230,252,
     202,208,242,96,234,120,16
     5,1,37,253           :rem 118
1270 DATA 133,1,88,96,234,120,
     165,1,5,253,133,1,88,96
                          :rem 54
1300 REM{4 SPACES}THIS DOWNLOA
     DS BASIC.            :rem 27
1320 POKE 253,1:SYS 49204:SYS
     {SPACE}49152         :rem 102
1350 REM{4 SPACES}THIS DOWNLOA
     DS THE KERNAL.       :rem 92
1370 POKE 253,2:SYS 49204:SYS
     {SPACE}49164         :rem 111
1400 REM{4 SPACES}THE DATA BEL
     OW MODIFIES          :rem 109
1410 REM{4 SPACES}THE DEFAULT
     {SPACE}BACKGROUND,
                          :rem 156
1420 REM{4 SPACES}BORDER, AND
     {SPACE}CHARACTER :rem 245
1430 REM{4 SPACES}COLORS (IN T
     HAT ORDER).          :rem 65
1450 DATA 60633,2         :rem 68
1460 DATA 6:REM{3 SPACES}THE B
     ACKGROUND COLOR :rem 121
1470 REM{10 SPACES}CODE OF YOU
     R CHOICE             :rem 90
1480 REM{10 SPACES}(VALUES 0 -
      15).                :rem 195
1500 DATA 6:REM{3 SPACES}THE B
     ORDER COLOR          :rem 82
1510 REM{10 SPACES}CODE OF YOU
     R CHOICE             :rem 85
1520 REM{10 SPACES}(VALUES 0 -
      15).                :rem 190
1540 DATA 58677,1         :rem 82
1550 DATA 14:REM{2 SPACES}THE
     {SPACE}CHARACTER COLOR
                          :rem 85
1560 REM{10 SPACES}CODE OF YOU
     R CHOICE             :rem 90
1570 REM{10 SPACES}(VALUES 0 -
      15).                :rem 195
1600 REM{4 SPACES}THE DEFAULT
     {SPACE}DEVICE NUMBER
                          :rem 10
1610 REM{4 SPACES}FOR 'LOAD',
     {SPACE}'SAVE', AND
                          :rem 169
1620 REM{4 SPACES}'VERIFY'.
                          :rem 254
1640 DATA 57818,1         :rem 79
1650 DATA 8:REM{3 SPACES}THE D
     ISK DRIVE WILL       :rem 250
1660 REM{10 SPACES}BE THE NEW
     {SPACE}DEFAULT       :rem 8
1670 REM{10 SPACES}DEVICE.
     {2 SPACES}IF YOU DO
                          :rem 175
1680 REM{10 SPACES}NOT HAVE A
     {SPACE}DISK DRIVE,
                          :rem 218
1690 REM{10 SPACES}REPLACE THE
      8 IN LINE           :rem 136
1700 REM{10 SPACES}1650 WITH A
      1.                  :rem 84
1730 REM{4 SPACES}THE SCREEN P
     OKE FIX.             :rem 148
1740 REM{4 SPACES}THE SCREEN C
     OLOR MEMORY          :rem 169
1750 REM{4 SPACES}WILL NOW BE
     {SPACE}FILLED WITH:rem 80
1760 REM{4 SPACES}THE CURRENT
     {SPACE}CHARACTER     :rem 67
1770 REM{4 SPACES}COLOR.
                          :rem 96
1790 DATA 58586,3         :rem 90
1800 DATA 173,134,2       :rem 160
1830 REM{4 SPACES}THE LOAD/RUN
      MODIFICATION.       :rem 121
1840 REM{4 SPACES}IF YOU DO NO
     T HAVE A DISK        :rem 81
1850 REM{4 SPACES}DRIVE, DELET
     E LINES              :rem 134
1860 REM{4 SPACES}1810 THROUGH
      1890.               :rem 158
1880 DATA 60647,9         :rem 87
1890 DATA 76,207,34,58,42,13,8
     2,213,13             :rem 84
1920 REM{4 SPACES}THIS SECTION
      OF CODE WILL        :rem 229
1930 REM{4 SPACES}EXCHANGE THE
     £ KEY WITH           :rem 86
1940 REM{4 SPACES}THE CLR/HOME
      KEY.{2 SPACES}IF THE
                          :rem 83
1950 REM{4 SPACES}REMS IN LINE
     S 2060 THROUGH       :rem 229
1960 REM{4 SPACES}2280 ARE REM
     OVED THEN:           :rem 211
1970 REM                  :rem 181
1980 REM{4 SPACES}1){2 SPACES}
     THE TWO KEY CAPS MUST
                          :rem 68
1990 REM{8 SPACES}BE PHYSICALL
     Y EXCHANGED          :rem 199
2000 REM{8 SPACES}AND :rem 121
2010 REM{4 SPACES}2){2 SPACES}
     THIS PROGRAM SHOULD
                          :rem 33
2020 REM{8 SPACES}ALWAYS BE RU
     N                    :rem 245
2030 REM{8 SPACES}IMMEDIATELY
     {SPACE}AFTER THE  :rem 48
2040 REM{8 SPACES}COMPUTER IS
     {SPACE}TURNED ON. :rem 82
2060 REM DATA 60337,1     :rem 38
2070 REM DATA 19          :rem 49
2090 REM DATA 60340,1     :rem 35
2100 REM DATA 92          :rem 44
2120 REM DATA 60402,1     :rem 28
2130 REM DATA 147         :rem 96
2150 REM DATA 60405,1     :rem 34
2160 REM DATA 169         :rem 103
2180 REM DATA 60467,1     :rem 45
2190 REM DATA 147         :rem 102
2210 REM DATA 60470,1     :rem 33
2220 REM DATA 168         :rem 99
2240 REM DATA 60584,1     :rem 42
2250 REM DATA 255         :rem 99
2270 REM DATA 60587,1     :rem 48
2280 REM DATA 28          :rem 52
2310 REM{4 SPACES}INPUT PROMPT
      MESSAGE FIX.        :rem 54
2330 DATA 58918,2         :rem 79
2340 DATA 234,234         :rem 65
2370 REM{4 SPACES}THE FOLLOWIN
     G CODE (WHICH        :rem 248
2380 REM{4 SPACES}WILL NOT BE
     {SPACE}EXECUTED   :rem 184
2390 REM{4 SPACES}BECAUSE OF T
     HE REM               :rem 4
2400 REM{4 SPACES}STATEMENTS)
     {SPACE}WILL RELOCATE
                          :rem 98
2410 REM{4 SPACES}THE KEYBOARD
      BUFFER TO           :rem 58
2420 REM{4 SPACES}THE CASSETTE
      BUFFER AND          :rem 118
2430 REM{4 SPACES}WILL EXPAND
     {SPACE}THE KEYBOARD
                          :rem 215
2440 REM{4 SPACES}BUFFER TO 80
      CHARACTERS.         :rem 129
2450 REM{4 SPACES}IF YOU WISH
     {SPACE}TO HAVE THE:rem 30
2460 REM{4 SPACES}KEYBOARD BUF
     FER MODIFIED,        :rem 40
2470 REM{4 SPACES}REMOVE THE R
     EMS FROM LINES       :rem 70
2480 REM{4 SPACES}2530 THROUGH
      2720, AND           :rem 103
2490 REM{4 SPACES}ALSO, THE RE
     MS PRECEEDING        :rem 252
2500 REM{4 SPACES}THE DATA STA
     TEMENTS IN           :rem 69
2510 REM{4 SPACES}LINES 2930 A
     ND 2940.             :rem 197
2530 REM DATA 58669,1     :rem 55
2540 REM DATA 80:REM{2 SPACES}
     NEW BUFFER LENGTH:rem 181
2560 REM DATA 58871,2     :rem 54
2570 REM DATA 59,3        :rem 153
2590 REM DATA 58569,2     :rem 61
2600 REM DATA 60,3        :rem 139
2620 REM DATA 58575,2     :rem 52
2630 REM DATA 60,3        :rem 142
2650 REM DATA 58805,2     :rem 51
2660 REM DATA 60,3        :rem 145
2680 REM DATA 58813,2     :rem 53
2690 REM DATA 60,3        :rem 148
2710 REM DATA 58810,2     :rem 44
2720 REM DATA 61,3        :rem 143
2750 REM{4 SPACES}SCREEN LOCK-
     UP FIX.              :rem 130
2770 DATA 58769,9         :rem 98
2780 DATA 228,201,240,3,76,237
     ,230,96,234          :rem 233
2800 DATA 58748,21        :rem 131
2810 DATA 32,240,233,169,39,23
     2,180,217            :rem 130
2820 DATA 48,6,24,105,40,232,1
     6,246,133            :rem 121
2830 DATA 213,76,36,234
                          :rem 112
```

```
2860 REM{4 SPACES}THE ERASE KE
     Y.                 :rem 28
2880 DATA 60220,3       :rem 69
2890 DATA 32,194,228   :rem 228
2910 DATA 58562,16     :rem 131
2920 DATA 201,169,208,8,169,29
     ,157,119,2,232,169,20,157
     ,119,2,96         :rem 103
2930 REM DATA 58562,16:rem 105
2940 REM DATA 201,169,208,8,16
     9,29,157,60,3,232,169,20,
     157,60,3,96       :rem 229
2970 REM{4 SPACES}THE FOLLOWIN
     G PATCH CAUSES    :rem 124
2980 REM{4 SPACES}THE SHIFTED
     {SPACE}£ KEY TO :rem 135
2990 REM{4 SPACES}AUTO REPEAT.
                       :rem 224
3010 DATA 60157,6       :rem 67
3020 DATA 32,183,228,234,234,2
     34                 :rem 35
3040 DATA 58551,11     :rem 119
3050 DATA 201,41,240,6,201,20,
     240,2,201,32,96   :rem 130
3080 REM{4 SPACES}THE CHANGE O
     F THE COLD        :rem 206
3090 REM{4 SPACES}START AND WA
     RM START          :rem 214
3100 REM{4 SPACES}ROUTINE.
                       :rem 252
3120 DATA 64982,1       :rem 74
3130 DATA 229          :rem 126
3160 REM{4 SPACES}THE NEW STAR
     TUP MESSAGE.      :rem 223
3180 DATA 58483,56     :rem 137
3190 DATA 147,13,32,32,32,32,4
     2,32,82,69,86,73,83,69,68
     ,32                :rem 56
3200 DATA 82,65,77,45,82,69,83
     ,73,68,69,78,84,32,75,69,
     82                 :rem 47
3210 DATA 78,65,76,32,42,13,13
     ,32,67,79,77,77,79,68,79,
     82                 :rem 36
3220 DATA 69,32,54,52,32,32,0,
     0                 :rem 227
3250 REM{4 SPACES}THIS CHANGE
     {SPACE}ALLOWS THE :rem 63
3260 REM{4 SPACES}SHIFT KEY TO
      INHIBIT          :rem 192
3270 REM{4 SPACES}SCROLLING.
                       :rem 139
3290 DATA 59723,11     :rem 128
3300 DATA 173,141,2,201,1,240,
     240,160           :rem 254
3310 DATA 0,132,198    :rem 161
3330 DATA 59710,4       :rem 73
3340 DATA 141,2,201,1 :rem 244
3354 REM{4 SPACES}FIX ASCII NU
     LL STRING          :rem 21
3356 DATA 46991,1       :rem 85
3357 DATA 5             :rem 33
3370 REM{4 SPACES}THE END OF D
     ATA MARKER        :rem 218
3390 DATA 99999          :rem 6
3420 REM{4 SPACES}THE FOLLOWIN
     G CODE READS      :rem 201
3430 REM{4 SPACES}THE DATA STA
     TEMENTS AND       :rem 132
3440 REM{4 SPACES}POKES THE DA
     TA INTO THE        :rem 71
3450 REM{4 SPACES}KERNAL RAM.
                       :rem 123
3470 READ A0:IF A0=99999 THEN
     {SPACE}POKE 253,253:SYS 4
     9194:SYS 64738:REM COLDST
     ART.               :rem 50
3480 READ N:FOR I=A0 TO A0+N-1
     :READ A%:POKE I,A%:NEXT I
     :GOTO 3470         :rem 63
```

©

Apple ProDOS Disk Menu

K Michael Parker

Here's a fast method of loading and running programs at the touch of a key. The program requires an Apple IIc or IIe with the ProDOS operating system.

How many times have you found yourself wishing for an easier way to load and run programs? The process of calling up a disk catalog, looking for the desired pathname, then typing (or mistyping) it can be a frustrating experience—especially if the pathname is something cryptic like FNINPT.BAO.2. Perhaps a better alternative is to select the program from a menu taken from the disk directory.

That's exactly what you can do with "ProDOS Disk Menu." To use it, create a startup disk by saving both ProDOS and BASIC.SYSTEM on a disk. Then save Disk Menu with the filename STARTUP.

When you boot this disk, a menu containing the first 16 programs in the directory appears on the 40-column screen. If more than 16 programs are on the disk, press P to view the next page. Pressing P on the last page returns you to the first page. (Disk Menu accepts both uppercase and lowercase commands.)

If you don't find the program you want, press C. A screen prompt asks you to switch disks, then Disk Menu reruns itself.

To select a program, press the up/down arrow keys to position the cursor over the desired filename, then press RETURN. A screen prompt offers three choices: (R)UN, (L)OAD, OR (U)NDO. If you made a mistake and selected the wrong program, press U to return to the menu.

Loading Multiple Programs

There are three ways to exit Disk Menu: run any program, load a BASIC program, or press Q to quit. Notice that loading a machine language program *does not* exit Disk Menu. Therefore, if you have a BASIC program that utilizes several ML subroutines, you could load the ML routines into memory one after the other, then exit DISK MENU by running the BASIC program.

The programming techniques used in Disk Menu are quite simple. The program retrieves the volume name from the disk and opens the volume directory. It reads the directory into an array, skipping all nonprogram files (except the type mentioned below). Then, depending on the current page, the program reads the filenames into the page array for display and selection.

A few parts of the program may need some explanation. For example, line 325 skips past the first few records on the volume directory, which do not contain information essential to the menu.

Disk Menu does not list any file types other than BASIC and binary files (.BAS and .BIN). Although data is sometimes stored in binary files, it is usually considered good practice to store data and programs on separate disks.

When writing Disk Menu, I was tempted to make it include files contained in subdirectories, but refrained because in my experience such files are usually chained to other programs, and would therefore only clutter up the menu. However, if you use subdirectories differently, the necessary alterations should be fairly simple.

Apple ProDOS Disk Menu

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
51 10 D$ = CHR$ (4)
53 12 L = 1
E2 20 A$(20) = "":A$(21) = "PRES
      S <RETURN> TO ACCEPT CHOIC
      E":A$(22) = "'Q'-QUIT":A$(
      23) = "'P'-PAGINATE  'C'-G
      ET NEXT CATALOG"
44 100 HOME
54 110 INVERSE : PRINT "  D I S
       K   M E N U ": NORMAL
34 200 REM GET VOLUME LABEL
81 210 PRINT D$;"PREFIX/"
3A 220 PRINT D$;"PREFIX"
15 230 INPUT VL$
F7 240 PRINT D$
24 250 VTAB 2: PRINT VL$: REM DI
       SPLAY VOLUME NAME
ED 300 REM GET DIRECTORY
C9 310 PRINT D$;"OPEN";VL$;",TDI
       R"
D7 320 PRINT D$;"READ";VL$
D1 325 INPUT Z$: INPUT Z$: INPUT
        Z$
CC 330 INPUT L$(L)
DF 335 CH$ = MID$ (L$(L),18,3)
4E 340 IF L$(L) = "" THEN 360
FA 345 IF CH$ < > "BAS" AND CH$
       < > "BIN" THEN 330
6A 350 L = L + 1: GOTO 330
8F 360 PRINT D$;"CLOSE"
72 370 MAX = L - 1
75 400 PAGE = 0
F4 410 GOSUB 2000: REM LOAD PAGE
        INTO ARRAY
52 420 GOSUB 3000: REM PRINT ARR
       AY
FF 430 GOSUB 4000: REM ACCEPT IN
       PUT
F1 440 GOTO 5000: REM RUN/LOAD
B6 2000 REM  INITIALIZE ARRAY
68 2010 FOR I = 4 TO 19:A$(I) =
        "": HTAB 5: VTAB I: PRIN
        T SPC( 16): NEXT
27 2020 REM   LOAD PAGE INTO ARR
        AY
64 2030 N = PAGE * 16:PAGE = PAG
        E + 1
20 2040 IF (MAX - N) > = 16 THEN
         LIM = 16: IF (MAX - N)
        = 16 THEN PAGE = 0
4D 2045 IF (MAX - N) < 16 THEN L
        IM = MAX - N:PAGE = 0
A5 2047 A = 4
A1 2050 FOR I = (N + 1) TO (N +
        LIM)
DB 2060 A$(A) = L$(I)
4B 2070 A = A + 1: NEXT
F2 2080 RETURN
4D 3000 REM PRINT ARRAY
B0 3020 FOR I = 4 TO 19
76 3030 HTAB 5: VTAB I
09 3040 PRINT MID$ (A$(I),2,16)
B7 3050 NEXT
C5 3060 FOR I = 21 TO 23: VTAB I
        : PRINT A$(I): NEXT
43 3062 CR = 4: INVERSE : VTAB C
        R: HTAB 5: PRINT MID$ (A
        $(CR),2,16): NORMAL
A2 3065 VTAB CR: HTAB 4
EF 3070 RETURN
A3 4000 REM ACCEPT INPUT
53 4010 GET C$
3C 4020 IF C$ < > CHR$ (10) AND
        C$ < > CHR$ (11) AND C$
        < > CHR$ (13) AND C$ < >
         "Q" AND C$ < > "C" AND
        C$ < > "q" AND C$ < > "c
        " AND C$ < > "P" AND C$
        < > "p" THEN 4010
88 4030 IF C$ < > CHR$ (10) AND
        C$ < > CHR$ (11) THEN 45
        00
02 4040 REM MOVE CHOICE
5D 4050 VTAB CR: HTAB 5: NORMAL
        : PRINT MID$ (A$(CR),2,1
        6)
EF 4060 ON ASC (C$) - 9 GOSUB 41
        00,4200
8F 4070 INVERSE : VTAB CR: HTAB
        5: PRINT MID$ (A$(CR),2,
        16): NORMAL
A7 4075 VTAB CR: HTAB 4
74 4080 GOTO 4010
D6 4100 REM DOWN
DA 4110 IF CR = 19 THEN CR = 3
D3 4120 IF CR < > 19 THEN CR = C
        R + 1
E2 4130 RETURN
77 4200 REM UP
6D 4210 IF CR = 4 THEN CR = 20
CD 4220 IF CR < > 4 THEN CR = CR
         - 1
E4 4230 RETURN
82 4500 IF C$ = "Q" OR C$ = "q"
         THEN HOME : END
2E 4510 IF C$ = "P" OR C$ = "p"
         THEN POP : GOTO 410
8D 4520 IF C$ = CHR$ (13) THEN R
        ETURN : REM  LOAD/RUN SU
        BROTINE
02 4530 REM C$ MUST BE C OR c SO
         CONTINUE
CC 4540 HOME : INPUT "INSERT NEW
         DISK THEN PRESS <RETURN
        >";ANS$
7F 4550 POP : GOTO 10
21 5000 REM RUN/LOAD
A7 5002 FILE$ = MID$ (A$(CR),2,1
        6):TYPE$ = MID$ (A$(CR),
        18,3)
EA 5010 HTAB 5: VTAB 3
82 5020 PRINT "(R)UN,(L)OAD, OR
        (U)NDO": HTAB 28: VTAB 3
5D 5030 GET E$
55 5040 IF E$ < > "R" AND E$ < >
         "L" AND E$ < > "1" AND
        E$ < > "r" AND E$ < > "U
        " AND E$ < > "u" THEN 50
        30
DA 5045 HTAB 5: VTAB 3: PRINT "
                              "
50 5047 IF E$ = "U" OR E$ = "u"
        THEN PAGE = 0: GOTO 410
7F 5050 IF E$ = "L" OR E$ = "1"
        THEN 5100
9E 5060 HOME : PRINT D$;"-";FILE
        $
17 5070 NEW
E0 5100 IF TYPE$ = "BAS" THEN HO
        ME : PRINT D$;"LOAD";FIL
        E$
44 5110 IF TYPE$ = "BIN" THEN PR
        INT D$;"BLOAD";FILE$
D9 5120 GOTO 420
```

©

Atari Fine Scrolling

Karl E. Wiegers

Unlock the secrets of fine-scrolling screen displays with this step-by-step tutorial, complete with example programs. Recommended for intermediate BASIC and machine language programmers. The techniques work on all Atari 400/800, XL, and XE computers.

An especially powerful graphics feature of Atari computers is their ability to scroll all or part of a screen display. Both text and graphics screens can be scrolled horizontally, vertically, or diagonally by various increments. Scrolling is seen in such diverse applications as racing games, in which the moving roadway lends apparent motion to the stationary cars, and in strategic war games, in which players can manipulate the screen as a "window" over a much larger map.

There are two general types of scrolling: *coarse scrolling* and *fine scrolling*. Coarse scrolling moves the screen in increments of eight pixels (the size of one character); fine scrolling moves the screen in increments of one pixel and is much more realistic. Earlier articles have addressed the rudiments of coarse scrolling (see "Fun with Scrolling" by David Plotkin in *COMPUTE!'s Second Book of Atari*). As we'll see in a moment, fine scrolling is actually a combination of coarse and fine scrolling. Since these techniques require machine language to work properly, we'll present a vertical blank interrupt routine you can add to your BASIC programs to obtain the smooth, continuous scrolling effect seen in many Atari games.

Coarse Scrolling

Let's review some details about how Atari computers display information on the video screen. A special microprocessor chip called ANTIC governs the display process. ANTIC gets its instructions from a short program in memory called the *display list*. The display list tells ANTIC what kind of graphics mode to use for each display line, how many lines to show, where in RAM to find the data to be displayed, and other information. The starting RAM location of the display list can be found using this formula:

```
DL=PEEK(560)+256*PEEK(561)
```

Ordinarily, the block of RAM containing screen display data is defined when a GRAPHICS statement is executed in BASIC. The first byte of screen memory—which is displayed in the upper-left corner of the screen—is identified by the fifth and sixth bytes in the display list in the usual low-byte, high-byte format:

```
MEMST=PEEK(DL+4)+256*PEEK
  (DL+5)
```

An elegant feature of the Atari operating system is that the section of RAM to be displayed on the screen can be altered simply by changing the values in DL+4 and DL+5, the pointers to screen RAM. For example, consider a graphics mode 2 display, with 20 characters or bytes of RAM per line. If we add 20 to the screen RAM pointers, the twenty-first byte of the original block of screen RAM would appear in the upper-left corner of the screen. This causes every part of the display to jump up by one mode line: a vertical coarse scroll. Conversely, subtracting 20 from the screen RAM pointers scrolls the display downward by one mode line in graphics mode 2.

Program 1 is a simple vertical coarse-scrolling routine written in BASIC for graphics mode 0. (Type this listing with the line numbers shown; we'll be adding to it later.) In line 150, the starting byte of screen RAM is incremented by 40 to generate each step of the scroll. Then the starting location is factored into its corresponding high- and low-byte values (lines 160—170), which are inserted into the display list (lines 180—190). Coarse scrolling can only change the position of display information in relatively large jumps, equal to the height of a character in whatever graphics mode is being used. It yields a jerky, rough appearance when scrolling a screenful of data.

These same principles apply to the concept of horizontal scrolling. However, horizontal scrolling is a bit more complex because it involves fooling the computer into thinking that each mode line is wider than the usual screen display. To make things easier, we'll stick to vertical scrolling.

Mixed Scrolling

The secret to fine scrolling, as mentioned above, is to mix coarse and fine scrolling. Atari computers were designed to allow vertical fine scrolling in increments of one video scan line (there are 192 in a normal full-screen display). In graphics mode 0, which has eight scan lines per mode line, the fine scrolling capability thus permits seven increments of vertical movement between mode lines. To scroll a display by more than just one mode line, your program must execute seven fine scrolls, then one coarse scroll. The final coarse scroll, in effect, appears onscreen as the eighth fine scroll.

All of this requires two basic steps. First, the program must in-form ANTIC which mode lines in the display are enabled for fine scrolling. Second, the program must store into an appropriate hardware register an integer representing the number of scan lines to scroll.

The first step, enabling the desired mode lines for scrolling, takes us back to the display list. We've already seen how to find the display list in RAM and how to alter the bytes pointing to the start of screen memory. Most of the other instructions in the display list identify the kind of graphics mode line to display. (For a more detailed discussion of display lists, see Craig Chamberlain's article "How to Design Custom Graphics Modes" in *COMPUTE!'s First Book of Atari Graphics.*) To enable a mode line for vertical fine scrolling, you must set bit 5 of its display list instruction. This is equivalent to adding 32 to the contents of the byte, and it must be done for each mode line you want to scroll. If you like, you can define several blocks of scrollable lines. Mode lines which don't have bit 5 set can be coarse-scrolled, but not fine-scrolled.

The second step, telling ANTIC how many scan lines to scroll, requires a simple POKE into a register called VSCROL at location 54277 (hex $D405). VSCROL affects all lines which have been enabled for vertical fine scrolling. For instance, the statement POKE 54277,4 shifts the display in each enabled mode line upward by four scan lines. Notice that you can POKE only positive integers into VSCROL (or into any other byte, for that matter). In effect, this means you can scroll the display upward but not downward. To simulate downward scrolling, you must start with the display scrolled fully upward (store a 7 in VSCROL for graphics modes 0 or 1, or 15 for mode 2, and so on), then POKE a smaller number into VSCROL to move the contents of each mode line downward by one or more scan lines.

Here, then, is the procedure for a complete mixed-scrolling routine:

1. Fine scroll a number of scan lines which is one less than the pixel height of the graphics mode.

2. Reset VSCROL to the starting value (0 if scrolling up, the maximum value if scrolling down).

3. Coarse scroll by one mode line.

4. Repeat the procedure.

A Fine Example

To add vertical fine scrolling to our previous example of coarse scrolling, merge the lines in Program 2 with those in Program 1. After running the program, you must press SYSTEM RESET to restore the original display list.

Notice that the bottom of the screen moves up slightly after running this program. Because of the way that ANTIC works, a block of mode lines enabled for fine scrolling results in a loss of one mode line of display area. This shortens the screen display.

To make this program scroll downward rather than upward, change the following lines:

```
80 POSITION 2,5
110 FOR S=7 TO 0 STEP -1
150 MEMST=MEMST-40
200 POKE 54277,7
```

As the display scrolls downward, you'll see the display list itself come into view, since it's normally found immediately before the start of screen memory. The display list appears mostly as a string of uppercase Bs. That's because the internal character code for an uppercase B is 34, the same as the display list instruction for a graphics 0 line enabled for vertical fine scrolling. You'll also see the fifth (and occasionally sixth) character in the display list change with each coarse scroll. These are the pointers to screen memory we discussed earlier.

To see a scrolling demo in graphics mode 1 instead of graphics 0, press SYSTEM RESET, type NEW, reload Program 1, and once again add the lines in Program 2. Then substitute these lines:

```
10 GRAPHICS 1+16
60 POKE DL+6+X,38
90 PRINT #6;"MODE ONE DEMO"
150 MEMST=MEMST+20
220 GOTO 220
```

Now to convert it for graphics mode 2, press RESET and make these changes:

```
10 GRAPHICS 2+16
50 FOR X=0 TO 9
60 POKE DL+6+X,39
80 POSITION 2,11
90 PRINT #6;"MODE TWO DEMO"
100 FOR D=1 TO 9
110 FOR S=0 TO 15
```

Scrolling Behind The Scenes

As you run these demos, you'll notice that they still suffer from some unsightly flickers and jumps, even though they're clearly a big improvement over simple coarse scrolling. The problem is that BASIC can't POKE the display list and scroll registers fast enough to synchronize with the TV or monitor's electron beam which is displaying the video image. To achieve smooth, flicker-free scrolling, your program must change all the registers during the split-second when the beam is displaying nothing on the screen. This *vertical blank interval* happens 60 times a second when the beam returns from the bottom to the top of the screen to sweep another video "frame." Since BASIC isn't nearly fast enough for this job, a machine language routine is required.

Program 3 is a BASIC loader which incorporates such a routine. (Program 4 is the source code for machine language programmers; don't type it in unless you have an assembler.) Be sure to save a copy of Program 3 before running it for the first time. When you type RUN, it stores the machine language routine in memory page 6 (starting at location 1536, hex $600), then sets up a *vertical blank interrupt (VBI),* a mechanism which calls the routine during each vertical blank interval. The program also modifies the display list as described above and initializes a few memory locations (203–206) for the VBI routine.

After the screen clears, you'll see it fill with a mass of apparently random letters, numbers, and graphics symbols. That's because the program has scrolled the display past the end of usable RAM and into the BASIC cartridge itself. The scrolling continues until you press SYSTEM RESET.

An apparent limitation of a VBI scrolling routine is that it can't scroll the display faster than 60 times a second, because it's called only 60 times a second. If you want to scroll faster, you can scroll more than one scan line at a time—although it won't appear as smooth.

There's also a way to scroll more slowly. This routine uses a counter at location 203 to control the scroll rate. It checks to see how many vertical blank intervals have passed since the last fine scroll, then compares the result against a preset limit to see if it's time for another fine scroll. To make the routine wait for more than one vertical blank interval between fine scrolls, change the 1 in line 60 of Program 3 to a higher number.

The comments in Program 4 tell machine language programmers how to modify this VBI routine to work in other graphics modes.

Program 5 is a BASIC loader for a downward-scrolling VBI routine. It's not a stand-alone program—it must be combined with certain lines in Program 3 as described in the REM statements. (Program 6 is the source code for Program 5 so machine language programmers can study the technique. Again, don't type in Program 6 unless you have an assembler.)

So far we've seen simple demos of the Atari's scrolling capabilities. Now let's use them for something fun.

Empire State Building

Scrolling is most often used in programs that have a larger display than can be shown on a single screen. By scrolling across parts of the display data, you can use the screen as a window onto other sections of RAM. Consider, for example, that a graphics mode 2 screen has 12 lines of 20 bytes each, or only 240 bytes of information. That leaves enough memory in the computer to create a display containing thousands of bytes of data—maybe a dozen or more screens. This is the technique seen in such classic Atari games as *Caverns of Mars* and *Eastern Front 1941*.

Let's try a simple example. Program 7 shows the Empire State Building as it might appear to a parachutist leaping out of a helicopter over Manhattan. The building is composed of redefined graphics mode 2 characters. It took 1200 bytes of RAM to store the building and background, which are conveniently located in a character string called ESB$. The beauty of this approach to allocating memory is that your program can easily find the first byte of ESB$ with BASIC's string ADR function. Then it can use this address as the upper-left corner of the screen by modifying the screen display pointers in the mode 2 display list. The 1200 bytes of ESB$ amount to five screens of graphics mode 2 data.

The VBI routine used in the Empire State Building example is slightly different from that in Program 4. First, it had to be modified for graphics mode 2. Second, it has a counter which is incremented after each coarse scroll. When the counter reaches a preset value (corresponding to street level in this case), the scrolling stops. You can change the 48 in line 150 of Program 7 to stop the scrolling at some other point. Press SYSTEM RESET each time before running this program to keep the redefined characters from getting messed up.

Just The Beginning

These examples illustrate the power of the graphics scrolling ability of Atari computers, but they're just a start. We don't have room in this article to cover extensions of these techniques, such as horizontal fine scrolling; diagonal scrolling; joystick-controlled scrolling; and altered perspective scrolling, in which cleverly designed character sets are combined with scrolling routines to create effective three-dimensional effects. With the ideas presented here, you can probe some of these techniques on your own.

For instructions on entering these listings, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

Program 1: Coarse Scrolling Demo

```
OC 1Ø  GRAPHICS Ø
PH 2Ø  DL=PEEK(56Ø)+256*PEEK(
       561):REM Start of disp
       lay list
PN 3Ø  MEMST=PEEK(DL+4)+256*P
       EEK(DL+5):REM Start of
        screen memory
KB 8Ø  POSITION 2,15
HD 9Ø  PRINT "THIS IS A DEMO
       OF COARSE SCROLLING"
NP 1ØØ FOR D=1 TO 15:REM Loo
       p to scroll through 1
       5 lines
LB 13Ø FOR DLAY=1 TO 1ØØ:NEX
       T DLAY:REM Delay loop
GO 15Ø MEMST=MEMST+4Ø
HI 16Ø HI=INT(MEMST/256):REM
        New high byte for sc
       reen memory in displa
       y list
MJ 17Ø LO=MEMST-HI*256:REM N
       ew low byte
HO 18Ø POKE DL+4,LO
HG 19Ø POKE DL+5,HI
BG 21Ø NEXT D
```

Program 2: Fine Scrolling Demo

(Merge these lines with Program 1.)

```
II 4Ø  REM Enable vertical fi
       ne scroll on all mode
       lines except first and
        last
BH 5Ø  FOR X=Ø TO 21
JM 6Ø  POKE DL+6+X,34
PD 7Ø  NEXT X
NI 9Ø  PRINT "THIS IS A DEMO
       OF FINE SCROLLING"
JL 11Ø FOR S=Ø TO 7:REM Fine
        scroll 7 times
EK 12Ø POKE 54277,S
LL 13Ø FOR DLAY=1 TO 2Ø:NEXT
        DLAY
CH 14Ø NEXT S
DH 2ØØ POKE 54277,Ø:REM Rese
       t vertical fine scrol
       l register
```

Program 3: VBI Routine BASIC Loader

```
OC 1Ø  GRAPHICS Ø
IP 2Ø  REM Read data for VBI
       routine
OM 3Ø  REM Increase the 1 in
       line 6Ø to slow the sc
       roll rate
PM 4Ø  FOR X=1 TO 63:READ A:P
       OKE 1535+X,A:NEXT X
LC 5Ø  DATA 1Ø4,16Ø,1Ø,162,6,
       169,7,76,92,228
GI 6Ø  DATA 23Ø,2Ø3,169,1,197
       ,2Ø3,2Ø8,42,23Ø,2Ø4
OL 7Ø  DATA 165,2Ø4,141,5,212
       ,169,Ø,133,2Ø3,169
HE 8Ø  DATA 7,197,2Ø4,176,25,
       169,Ø,141,5,212,133
PM 9Ø  DATA 2Ø4,169,4Ø,16Ø,4,
       24,113,2Ø5,145,2Ø5,144
ND 1ØØ DATA 7,2ØØ,169,Ø,113,
       2Ø5,145,2Ø5,76,98,228
NB 11Ø REM Modify display li
       st for vertical fine
       scrolling on all line
       s
HF 12Ø DL=PEEK(56Ø)+256*PEEK
       (561)
EO 13Ø POKE DL+3,98
II 14Ø FOR X=Ø TO 21:POKE DL
       +6+X,34:NEXT X
ML 15Ø POSITION 2,2Ø
OP 16Ø PRINT "FINE SCROLL WI
       TH VERTICAL"
JO 17Ø PRINT "BLANK INTERRUP
       T"
HS 18Ø REM Initialize variab
       les for VBI routine
BF 19Ø POKE 2Ø3,Ø:POKE 2Ø4,Ø
JF 2ØØ POKE 2Ø5,PEEK(56Ø)
JI 21Ø POKE 2Ø6,PEEK(561)
NJ 22Ø REM Turn on VBI routi
       ne; press SYSTEM RESE
       T to make it stop
CN 23Ø A=USR(1536)
```

Program 4: VBI Routine Source Code

(An assembler is required to enter this listing.)

```
10 ;VBI routine
20 ;for combined fine and coarse scrolling
30 ;in graphics mode 0.
40 ;
50 ;Change the 1 in line 180 to scroll slower.
60 ;Change the 7 in line 260 to the number of scan lines
70 ;per mode line minus 1 for other graphics modes.
80 ;Change the 40 in line 320 to the number of
90 ;bytes per mode line for other graphics modes
0100 ;
0110 *=$0600 ;Load into page 6.
0120 PLA ;Remove argument count.
0130 LDY #10 ;These 4 statements set up
0140 LDX #6 ;a deferred vertical blank
0150 LDA #7 ;interrupt - use LDA #6 for
0160 JMP $E45C ;an immediate VBI.
0170 INC $CB ;$CB is counter for number of
0180 LDA #1 ;VB cycles before next scroll.
0190 CMP $CB ;If not up to desired interval
0200 BNE EXIT ;then exit VBI
0210 INC $CC ;$CC is counter for number
0220 LDA $CC ;of fine scrolls.
0230 STA $D405 ;Store in vertical fine scroll register.
0240 LDA #0 ;reset VB counter
0250 STA $CB
0260 LDA #7 ;Have we done 7 fine
0270 CMP $CC ;scrolls yet?
0280 BCS EXIT ;No, exit VBI.
0290 LDA #0 ;Yes,
0300 STA $D405 ;reset vertical scroll register,
0310 STA $CC ;reset scroll counter,
0320 LDA #40 ;and coarse scroll by
0330 LDY #4 ;adding 40 to low byte
0340 CLC ;of screen memory pointer.
0350 ADC ($CD),Y
0360 STA ($CD),Y
0370 BCC EXIT ;If carry not set then exit VBI,
0380 INY ;else increment high byte
0390 LDA #0 ;of screen memory pointer.
0400 ADC ($CD),Y
0410 STA ($CD),Y
0420 EXIT JMP $E462
```

Program 5: Downward VBI BASIC Loader

(Combine with Program 3.)

```
DI 10 REM This loads a VBI routine
GC 20 REM for downward fine scrolling
AI 30 REM into page 6 (1536, $600).
CH 35 REM Call it with A=USR(1536).
HF 40 REM Initialize locations 203-205 as in Program 3 before running,
HF 50 REM but execute POKE 204,7 instead.
PB 60 REM Don't forget to modify display list for
DP 70 REM fine scrolling before using this routine
LC 80 FOR X=1 TO 65:READ A:POKE 1535+X,A:NEXT X
LF 90 DATA 104,160,10,162,6,169,7,76,92,228,230,203,169,1,197,203,208
EE 100 DATA 44,165,204,201,255,240,8,141,5,212,198,204,76,58,6,169,7,133
AC 110 DATA 204,141,5,212,160,4,56,177,205,233,40,145,205,176,8,200,177
CP 120 DATA 205,56,233,1,145,205,169,0,133,203,76,98,228
```

Program 6: Downward VBI Source Code

(An assembler is required to enter this listing.)

```
10 ;VBI routine
20 ;for downward fine scrolling.
30 ;
40 *=$0600 ;Load into page six
50 PLA ;Remove argument count
60 LDY #10 ;Set up deferred VBI
70 LDX #6
80 LDA #7
90 JMP $E45C
0100 INC $CB ;Increment VB counter
0110 LDA #1 ;Change the 1 to scroll slower
0120 CMP $CB ;Time to fine scroll yet?
0130 BNE EXIT ;No, exit VBI
0140 LDA $CC ;Yes, see if time for coarse scroll
0150 CMP #255
0160 BEQ COARSE ;Yes, go to coarse scroll routine
0170 STA $D405 ;No, then fine scroll
0180 DEC $CC ;Decrement fine scroll counter
0190 JMP INCREMENT ;Exit via VB counter
0200 COARSE LDA #7 ;Coarse scroll routine
0210 STA $CC ;Reset fine scroll counter
0220 STA $D405 ;Reset fine scroll register
0230 LDY #4 ;Get low byte of screen memory pointer
0240 SEC
0250 LDA ($CD),Y
0260 SBC #40 ;Subtract 40 from it
0270 STA ($CD),Y
0280 BCS INCREMENT ;Exit via VB counter unless
0290 INY ;you need to get the high byte
0300 LDA ($CD),Y
0310 SEC
0320 SBC #1 ;and subtract 1 from it
0330 STA ($CD),Y
0340 INCREMENT LDA #0 ;Reset VB counter
0350 STA $CB
0360 EXIT JMP $E462 ;Exit from VBI
```

Program 7: Empire State Building Demo

```
JJ 30 REM Reserve memory for redefined characters in GR. 2
LD 40 POKE 106,PEEK(106)-2:CHBASE=256*PEEK(106)
JG 50 GRAPHICS 18:POKE 559,0:REM Turn off video display
MA 60 SETCOLOR 0,12,8:REM Green for top of building
AN 70 SETCOLOR 1,8,6:REM Blue for sky
LD 80 SETCOLOR 3,1,8:REM Yellow for windows
FF 90 REM ESB$ is screen display RAM, A$ is character for wall with windows
MG 100 DIM A$(16),ESB$(1200)
PJ 110 A$="■":A$(16)=A$:A$(2)=A$
NA 120 ESB$="A":ESB$(1200)=ESB$:ESB$(2)=ESB$
FP 130 REM VBI routine
NM 140 FOR X=1 TO 71:READ A:POKE 1535+X,A:NEXT X
MI 150 DATA 104,160,10,162,6,169,7,76,92,228,165,207,201,48,240,52,230,203,169,2
NG 160 DATA 197,203,208,44,230,204,165,204,141,5,212,169,0,133,203,169
CM 170 DATA 15,197,204,176,27,169,0,141,5,212,133,204,230,207,169,20,160,4,24
KN 180 DATA 113,205,145,205,144,7,200,169,0,113,205,145,205,76,98,228
```

```
EH 190 REM Turn on new chara
       cter set
AB 200 POKE 756,CHBASE/256
LJ 210 REM Read redefined ch
       aracters
OC 220 FOR X=1 TO 5:READ OFF
       SET:REM A,B,C,D
HM 230 FOR J=0 TO 7:READ A:P
       OKE CHBASE+8*OFFSET+J
       ,A:NEXT J:NEXT X
CB 240 DATA 1,255,255,255,25
       5,255,255,255,255
AL 250 DATA 2,252,252,252,25
       2,252,252,252,252
IN 260 DATA 3,63,63,63,63,63
       ,63,63,63
LP 270 DATA 4,0,68,68,0,0,68
       ,68,0
FN 280 DATA 35,255,255,255,2
       55,255,255,255,255
OG 290 REM Create Empire Sta
       te Building with rede
       fined characters
HO 300 FOR X=230 TO 270 STEP
        20
PC 310 ESB$(X,X+1)="BC":NEXT
        X
ID 320 FOR X=290 TO 330 STEP
        20
LF 330 ESB$(X,X+1)="##":NEXT
        X
JD 340 FOR X=349 TO 429 STEP
        20
EJ 350 ESB$(X,X+3)=A$(13):NE
       XT X
JL 360 FOR X=448 TO 588 STEP
        20
EL 370 ESB$(X,X+5)=A$(11):NE
       XT X
JL 380 FOR X=606 TO 986 STEP
        20
CG 390 ESB$(X,X+9)=A$(7):NEX
       T X
OC 400 FOR X=1003 TO 1183 ST
       EP 20
ME 410 ESB$(X,X+15)=A$:NEXT
       X
OP 420 REM These are suppose
       d to be doors
AK 430 ESB$(1166,1166)="E":E
       SB$(1170,1170)="E":ES
       B$(1174,1174)="E"
NK 440 REM Set up GR. 2+16 d
       isplay list for verti
       cal fine scrolling
HL 450 DL=PEEK(560)+256*PEEK
       (561)
KH 460 POKE DL+3,103
PG 470 FOR X=DL+6 TO DL+15:P
       OKE X,39:NEXT X
LN 480 REM Start display RAM
        at beginning of ESB$
JJ 490 HI=INT(ADR(ESB$)/256)
BI 500 LO=ADR(ESB$)-256*HI
JB 510 POKE DL+4,LO:POKE DL+
       5,HI
GO 520 REM Initialize variab
       les for VBI
HB 530 POKE 203,0:POKE 204,0
       :POKE 207,0
JN 540 POKE 205,PEEK(560)
JP 550 POKE 206,PEEK(561)
KD 560 REM Turn video displa
       y back on
AB 570 POKE 559,34
JD 580 REM Start VBI going;
       it will stop automati
       cally
DG 590 A=USR(1536)
GI 600 REM This is needed to
        keep GR. 2 display o
       n the screen
GH 610 GOTO 610
```

©

Commodore Program Chaining

Orlando Lee Stevenson

Take advantage of Commodore's automatic chaining feature to link two or more BASIC programs together. The method illustrated here applies to all Commodore computers.

Program chaining is a method of linking separate programs together, making them run, in effect, as one large program. Why would you need to chain? Some BASIC programs simply grow too large to fit into memory: Chaining lets you break them into two or more program modules that work together as one. This method also lets you interconnect an entire group of programs, moving from one to another whenever you like.

LOAD In Program Mode

Let's say you have two programs you want to chain together. The solution can be as simple as placing LOAD "PROGRAM NAME",8 (disk) or LOAD "PROGRAM NAME" (tape) in place of an END statement. In Commodore BASIC, a LOAD command executed as part of a program automatically loads and runs the specified program. If the programs are completely unrelated, nothing more needs to be done.

However, if the programs are related, you'll probably want to pass variable values from one to the other as well—a procedure that requires some care. On all Commodore computers except the 128, variables and arrays are stored in memory immediately following the end of BASIC program text. Since different programs are of different lengths, the actual location of variables depends on the length of the program. The computer uses two-byte address pointers to keep track of where everything is stored, and updates the pointers as needed while the program runs. When you perform LOAD in program mode, the computer does not reset the pointers for variables, arrays, and strings. Thus, after it loads a second program, the computer still knows how to find and use all of the first program's variables.

The success of this procedure depends on the relative length of the chained programs. If the first program is *longer* than the second, all is well: When the second program loads in, its shorter program text doesn't extend as far as the area where variables are stored. (Remember, the first program's variables are still located in the same place). However, if the first program is *shorter* than the second, you have trouble. When the second program loads, its longer text overwrites the variables. Though the pointers still point to the right area, the variable data which used to be there has been replaced with program lines. Once that happens, the variables are lost.

This is not a problem with BASIC 7.0 on the Commodore 128, because it keeps variables in a separate 64K bank of memory. Thus, 128 programs can be chained freely without worrying about overwriting variables, and all the following discussion about preserving variables does not apply. However, you should still read the section entitled "Chain with Care." And don't forget that variables *will* be overwritten if you're running the 128 in 64 mode, just as they would be on a 64.

Changing The Signposts

The easiest solution is to make sure the first program in a chain is longer than all the rest. However, in many cases the first program in a chain is quite short. It may be a menu program—one that simply lets you choose among several programs to load and run.

Fortunately, there's an answer. By resetting the first program's pointers, you can make it store variables in an area that won't be disrupted by following programs in the chain. Here are the steps to follow (you can use any BASIC programs to practice this technique):

1. First, find the length of every program in the chain. Load the program and type the appropriate line below in direct mode (without a line number), then press RETURN.
   For the VIC, 64, 128 in 64 mode, Plus/4, and 16:
   ```
   PRINT PEEK(45)+PEEK(46)*256
   ```
   For PET/CBM (Upgrade and 4.0 BASIC):
   ```
   PRINT PEEK(42)+PEEK(43)*256
   ```
   This number is the location where the program's text ends and its variable storage begins. Write down the end-of-text number and note which program it belongs to, then repeat for every program in the chain.
2. Scan the list of numbers to find the longest program: It's the one with the highest end-of-text number. Now reload that program and find the contents of the two addresses you PEEKed above. For the VIC, 64, Plus/4, or 16, type **PRINT PEEK (45),PEEK(46)** in direct mode; substitute the proper addresses if you are using a PET/CBM. Two numbers are printed. These are the actual pointer values for the longest program. Write them down, labeling the first number LO and the second HI. You now know the lowest safe storage address for variables in this chain.
3. Reload the first program in the chain. Do not run it or enter any direct mode statements that would create variables. Enter the following lines, *replacing LO and HI with the numbers you recorded in step* 2. For instance, if LO is 20 and HI is 9, you would type the first line as POKE 45,20:POKE 46, 9+1. Don't forget to press RETURN after each line.
   For the VIC, 64, Plus/4, and 16:
   ```
   POKE 45,LO:POKE 46,HI+1
   POKE 47,LO:POKE 48,HI+1
   POKE 49,LO:POKE 50,HI+1
   ```
   For PET/CBM (Upgrade and 4.0 BASIC):
   ```
   POKE 42,LO:POKE 43,HI+1
   POKE 44,LO:POKE 45,HI+1
   POKE 46,LO:POKE 47,HI+1
   ```
4. Finally, resave this program. *Do not delete the original version* (see explanation below). Step 3 sets the first program's end-of-text and variable pointers to an address 256 bytes above the end of the longest program (the extra bytes provide a margin for error). Though it artificially increases the length of the first program, this method lets you run the entire package without losing variables.

Chain With Care

This method of program chaining has limitations. User-defined functions—created with DEF FN()—cannot be passed at all, since their definitions are stored in program text, not as variables. Such functions must be redefined in every program that uses them.

Strings may cause problems as well. In the VIC, 64, and PET/CBM versions of BASIC *dynamic* strings (which result from a string operation such as A$="HELLO"+B$) are stored outside the program text, they can be passed like other variables. The same is not true of *static* strings. Like a function definition, a static string exists only in a program line (10 A$="HELLO"). If you need to pass a static string, simply add a null string to it (for instance, replace 10 A$="HELLO" with 10 A$= "HELLO"+""). The string operation (+) turns it into a dynamic string, storing it outside the program. This is not a problem in the 128, Plus/4, and 16 versions of BASIC, where all strings are effectively dynamic.

Be careful when editing chained programs. If you lengthen a program, it may become the longest one in the chain and overwrite variables when it loads. Do *not* edit and resave a program after breaking out with RUN/STOP (since the pointers are set at artificially high locations, the program's length is abnormal). Instead, reload the program to set the pointers correctly, then make the changes and save it again. Whenever you edit any of the programs in the chain, you should also repeat steps 1–4, using the *original version* of the first program. It's critical that you know the true length of this program, not the inflated length it was given in steps 3 and 4.

There are other ways to pass variables while chaining, but they're inevitably cumbersome. One approach is to store variable data in a separate memory area while one program loads another. For instance, say that A=10. Just before the first program loads the second, it POKEs the value of A into a safe memory location (say, 49152 for the Commodore 64). The first thing the second program does is retrieve A's value with a statement like A=PEEK(49152). Since a single memory location can hold only a number from 0–255, it requires multiple POKEs and PEEKs to pass larger numeric values. Passing arrays, strings, or more complex numbers (negative values, for instance) takes even more work and ingenuity. ©

Commodore Dynamic Keyboard
Part 3

Jim Butterfield, Associate Editor

Parts 1 and 2 of this series showed how the dynamic keyboard technique—which allows the computer to seemingly type on its own keyboard—lets you do things that would otherwise be difficult or impossible from within a program. Now we'll look at the trickiest application of this technique—writing a program that changes itself as it runs.

Let's quickly review how the dynamic keyboard technique works. First, the program prints the desired command at a specific screen location. Then a RETURN character is placed in the keyboard buffer. Finally, the program stops with the cursor flashing over the screen command. The RETURN in the keyboard buffer causes the operating system to read the command on the screen and carry it out, just as if you pressed the RETURN key. Using the same principle, we can put several commands on the screen and make the program execute them all.

The following table shows the location of the keyboard buffer counter and the start of the keyboard buffer on most Commodore computers:

	Counter	Buffer
VIC-20, Commodore 64	198	631
Commodore 16, Plus/4	239	1319
PET/CBM (Upgrade and 4.0 BASIC)	158	623
Original ROM PETs	525	527
B128 (Model 700)	209	929

For a single-line command, POKE a value of 1 into the counter and a value of 13 (RETURN) into the buffer. To execute more than one screen line of commands, use a higher count and more RETURN characters. On the B128 computer, it's wise to execute a BANK 15 command before the POKEs.

Self-Editing Programs

The usual way to change a program is to type in a new line and press RETURN. The line is either added to the program or it replaces an existing line with the same line number. A program can do this, too, using the dynamic keyboard technique. But there's a hitch. Whenever you enter a program line, the computer performs a CLR command, which closes all open files and clears the contents of all variables and arrays. This can be annoying, since it's hard for a program to continue running after its variables are gone. But with some careful programming, you can still make things work.

The solution is to identify your key variables, make the program change itself, then reinstate the variables with the dynamic keyboard technique. In effect, the variables are temporarily stored on the screen and put back in the program by the equivalent of a direct command. Tricky? Crude? Whatever your opinion of this method, the point is that it works. There are other ways to do the job, but you usually want to get it done in the most direct way possible.

You might be wondering why you'd ever need to design a program that modifies itself, anyway. Here's an example. Suppose you have something in a special part of memory—a machine language program, a screen picture, or a data table. Whatever it is, you want to take the information and build it into a series of DATA statements so it can be reconstituted by a BASIC program when needed. Perhaps you'd like to publish a small machine language program in a newsletter or magazine, and want readers to be able to type it in as DATA statements rather than the more complex hexadecimal code. How to do it?

First, let's write some data into memory so that you'll have something to convert to DATA statements. Here's a quick program to put a series of prime numbers into memory locations 828 to 881:

```
100 POKE 828,2               :rem 192
110 POKE 829,3               :rem 195
120 N=3                      :rem 81
130 FOR A=830 TO 881         :rem 216
140 N=N+2                    :rem 203
150 FOR M=3 TO SQR(N)+.1 STEP2
                             :rem 106
160 T=N/M                    :rem 242
170 IF T=INT(T) GOTO 140
                             :rem 22
180 NEXT M                   :rem 37
190 PRINT N;                 :rem 176
200 POKE A,N                 :rem 124
210 NEXT A                   :rem 19
```

That's not the most efficient prime number generator, but it does put the numbers into memory. The last number should be 251. Now, suppose you want these values in DATA statements so that a different program will be able to POKE them back at the start of the run.

Frenzied Activity

Type NEW to make space for the new program. The following program is written for the Commodore 64. If you're using another computer, refer to the table above to find the right POKE values for lines 75 and 80.

```
10 L=100:A=830              :rem 206
15 PRINT CHR$(147)          :rem 225
20 PRINT                    :rem 239
25 PRINT                    :rem 244
30 PRINT L;"DATA";           :rem 16
35 D$=STR$(PEEK(A))          :rem 50
40 IF N>0 THEN D$=","+MID$(D$,
   2)                        :rem 51
45 PRINT D$;                :rem 153
50 A=A+1:N=N+1:IF N<10 AND A<8
   82 GOTO 35                :rem 54
55 PRINT                    :rem 247
60 PRINT "L=";L+10;":A=";A;
                            :rem 193
65 PRINT ":GOTO 15"          :rem 21
70 PRINT CHR$(19)           :rem 176
75 POKE 198,1:IF A<882 THEN PO
   KE 198,2                 :rem 224
80 POKE 631,13:POKE 632,13
                             :rem 85
85 END                       :rem 68
```

Be sure to type the semicolon at the ends of lines 30, 45, and 60, and note that some of the strings printed in lines 60 and 65 start with a colon character. When you RUN the program, you'll see a frenzy of activity on the screen for a few moments. Then the action stops with the cursor over a line which says L=160:A=882:GOTO 15. Don't execute this line. Instead, move the cursor down, type LIST, and press RETURN. You'll find that the program contains six new lines of DATA statements.

Start the new DATA lines at line number 100 (variable L). Since the data maker program ends at a lower line number, there's no danger of replacing existing lines with new ones. Never increase the line number directly. Instead, print a higher value onto the screen. When the dynamic keyboard reinstates the variable, it's ten higher than before. There's no need to set variable N back to zero. The CLR caused by changing the program effectively clears all variables to zero.

After the DATA lines have been created—you've generated only a few—you might want to get rid of the program that made them. You could do this manually by clearing the screen and giving the direct command:

```
FOR J=10 TO 85 STEP 5:PRINT J:
  NEXT J
```

This prints the line numbers on a blank screen. You could then move the cursor back and strike RETURN 16 times, eliminating the lines. It would take a little ingenuity, but you could even cause the program to wipe itself out using the dynamic keyboard. (Hint: Crunch the program into less than ten lines—then stuff the keyboard buffer with the same number of RETURN characters.)

Convert ASCII To BASIC

Occasionally, you might have a sequential file on disk or tape that contains a program. You'd like to run this program, but LOAD can't handle a sequential file. Dynamic keyboard lets you bring the file into memory and convert it to a regular program file. How could this need arise? There are several possibilities. First, the program might have been transmitted over a communications link. It's possible to download a program in a form that's ready to run, but it's common to transmit a program in ASCII form—as ASCII characters only, rather than the usual mixture of ASCII and BASIC tokens. Now, however, you must change the listing back into a working program.

Here's another way it might happen. You want to transfer a program between two slightly incompatible computers. Perhaps you have a PET program that you'd like to use on a Plus/4, or vice versa. You may be surprised to find that the SCRATCH command used in a PET program doesn't load correctly on a Plus/4. Knowing the technical reason for this (the computers use different tokens for some commands) doesn't help solve the problem. Since ASCII listings contain no tokens, you'll find that they transport more easily than ordinary programs.

Another possibility is that you want to merge two programs into one. The dynamic keyboard offers one way to do this. Note that we're breaking down the distinction between data and programs—as personified by sequential and program files, respectively. This opens the door to such things as program-analyzing programs and program-writing programs.

Change A Program To A File

Let's start by writing a simple program. Anything will do, but let's use the following:

```
100 FOR J=1 TO 10
110 PRINT J,SQR(J)
120 NEXT J
```

Remember to type NEW before entering this program. Now store it as a sequential file:

```
OPEN 1,8,6,"0:PROGFILE,S,W":CMD
  1:LIST
```

After you press RETURN, the disk drive operates briefly, then the cursor returns. Now type:

```
PRINT#1:CLOSE 1
```

It is *very important* to close the file in exactly this manner.

The program is now stored as an ASCII listing in a sequential file named PROGFILE. If displayed on the screen, it would look almost like the original program. But it consists of nothing but ASCII characters. Thus, the first line contains the characters 1-0-0 (the line number), then a space, then the letters F-O-R, and so on. This is quite different from the tokenized form in which programs are usually stored.

This file has a few oddities caused by the way in which it was stored. Unlike most data files, it begins with two RETURN characters. And it ends with the word READY (after all, when you LIST a program to the screen, it always ends with the word READY). None of this is critical, but if you plan to do advanced work with such a file, keep these things in mind.

Keep The File Open

However, there's another problem to consider. Every time you enter a new line with the dynamic keyboard, CLR closes the file you're using. You've learned how to recreate variables, but how do you reinstate the file? You can use the following fact: The file isn't really closed, it's just "disconnected."

CLR signals that no files are open simply by putting the value zero in the computer's number-of-open-files counter. If this is the only file you're handling, you can reconnect it by POKEing a 1 into the counter. The counter is found at one of the following locations, depending on your machine:

	File Counter
VIC-20, Commodore 64	152
Commodore 16, Plus/4	151
PET/CBM (Upgrade and 4.0 BASIC)	174
Original ROM PETs	610
B128 (Model 700)	864

Let's write a program to bring in this file. You know in advance that the line numbers start at 100, so you can safely put the loader program at lower numbers. If that isn't true for other situations, you might try renumbering the program with very high line numbers (above 60000, for instance).

Again, the following example runs on the VIC-20 and Commodore 64. Change the POKEs in lines 50, 65, and 70 to suit your machine. Be sure to include the semicolon at the end of line 40.

```
10 OPEN1,8,8,"PROGFILE":rem 84
15 PRINT CHR$(147)      :rem 225
20 PRINT                :rem 239
25 PRINT                :rem 244
30 GET#1,X$:X=ASC(X$) :rem 178
35 IFX>47 AND X<58 THEN F=1
                        :rem 175
40 IF F=1 THEN PRINT X$;
                        :rem 26
45 IF X<>13 GOTO 30     :rem 202
50 IF ST=0 THEN PRINT "POKE 15
   2,1:GOTO 15"         :rem 5
55 IF ST<>0 THEN PRINT "CLOSE
   {SPACE}1"            :rem 241
60 PRINT CHR$(19)       :rem 175
65 POKE 198,2           :rem 154
70 POKE 631,13:POKE 632,13
                        :rem 84
75 END                  :rem 67
```

When you run this program, it loads and merges the sequential ASCII listing. Typing 14 lines in order to add three more may seem inefficient. But the principle works on programs of any size.

It's been a long voyage. If you've stayed with it, you can probably see how the dynamic keyboard technique expands what you can do with the computer. Though it requires care, it also creates new possibilities. "Dynamic keyboard" is not just a buzzword, although you may add it proudly to your vocabulary. It's a new resource. ©

Advanced Commodore 128 Video

Jim Butterfield Associate Editor

Here's how to relocate screen memory and set up a custom character set on the Commodore 128—two valuable techniques worth mastering on any computer. When you run the example program, be ready for a surprise. For intermediate and advanced BASIC programmers.

You can do a lot of graphics on the Commodore 128 with an elementary knowledge of the new BASIC: circles, squares, lines, and points appear by means of simple BASIC commands. But advanced programmers may still need to get into the mechanics of video. Here's a simple exercise for 128-mode 40-column screens that will give a little insight into the "works."

The question often arises: How can I implement a new character set? Some people want to design their own personalized codes or graphics symbols for the screen; others are interested in foreign languages. In 40 columns, the 8564 video chip is practically identical to the 6567 of the Commodore 64. With a few new rules, we can put the chip's features to work in the same way.

Because the Commodore 128 makes it easy, I'll be including some hexadecimal addresses in the following listing. If you'd rather use decimal numbers, the computer will do quick conversions for you, and you can make the substitutions in the program.

Changing Addresses

Let's build the program step by step and note points of interest.

```
100 POKE 58,DEC("C0")
110 CLR
```

I'm planning to put the screen and its new character set into memory bank 1, at addresses $C000 to $CBFF—character set at $C000, screen at $C800. (By the way, if you'd rather use the decimal value 192 instead of DEC("C0"), be my guest. I prefer C0 because it's easier to visualize it as part of the full address $C000. Be sure to type a zero and not the letter O or you'll get an error.) Bank 1 is where BASIC puts its variables; we wouldn't want these to get mixed up with our screen. So we cut down the top-of-variable-memory pointer to $C000. There's really no danger of a memory conflict with this small program, but we might as well do it right.

The CLR command makes sure the other variable pointers don't get mixed up by this change.

```
120 TRAP 500
```

This command may be unfamiliar to many Commodore programmers. It sets up an *error trap* so that if anything goes wrong in the following code, the computer hops to line 500, which will restore the screen. This saves us from the horrible prospect of watching the program stop with a syntax error while the screen is still scrambled and unreadable. The TRAP command gives us another bonus: If the computer freezes—or is just too slow—we can press STOP, and the program zips to line 500 and wraps things up.

```
130 BANK 15
```

We're about to fiddle with the insides of computer chips (registers), so this command calls for memory bank 15 to make the chips accessible. This assures that the next few POKEs will be directed to the right place.

```
140 POKE DEC("DD00"),148
```

Except for the decimal number conversion ($DD00=56576), this POKE is identical to the way it's done on the Commodore 64. Briefly, it means: Display video out of the memory slice in the range $C000 to $FFFF. We haven't specified the bank yet, but we'll get around to it in a moment.

```
150 POKE DEC("0A2C"),32
```

We're still in bank 15, but this address isn't a chip. The address $0A2C (decimal 2604) is below $4000 (16384). When we're using bank 15, all such low addresses go to RAM, bank 0. This POKE sets the position of the character set and the screen within the video slice we've selected. The calculation goes like this: We want the screen to be at $C800, which is 2K above the start of the video slice at $C000, so multiply the 2 by 16 and add a similar value for the character set. In this case, the character set is right at the start of the slice; so we add 0 to get a value of 32.

On the Commodore 64, we'd do exactly the same calculation, but we'd put the result in address $D018 (53272). In fact, that's the same address at which our value will end up in the Commodore 128, but we must let the computer's interrupt routine deliver it there for us. So instead of POKEing the value directly into $D018, we store it at $0A2C (2604). As part of the computer's interrupt procedure, it will copy the contents of this location into $D018.

```
160 POKE DEC("D506"),68
```

This tells the computer to take video from bank 1. If we wanted video from bank 0, we'd POKE a value of 4—or just leave this line out, since that's the value that will be there in any case.

```
170 POKE 217,4
```

This POKE tells the computer to take its video from RAM, not ROM. We don't need to give this one for the addresses we have chosen, since there is no conflict. This very low address has a special banking rule: All addresses below hex $400 (1024) go to RAM bank 0, regardless of the bank which has been specified.

Relocating The Screen

Now our video is set up and ready to go. We'd better put something on the screen so we can see it working. It seems sensible to copy our old screen to the new place; then we'll copy the character set. We'll make a slight change so you can see how to create a new set of characters.

First, our screen must move from bank 0, address $400, to bank 1, address $C800. We must move the whole thousand characters.

```
200 FOR J=0 TO 999
210 BANK 0:X=PEEK(1024+J)
220 BANK 1:POKE DEC("C800")+J,X
230 NEXT J
```

This moves screen memory, but since the character set is not in place, the result would look rather muddy. We can read the character set by selecting bank 14; it is found in this bank at addresses $D000 to $D7FF. There are 256 characters times 8 bytes per character, which means 2,048 bytes to move. Just as we moved the screen in the lines above, we must move the character bytes one at a time, flipping between banks 14 and 1.

We'll also change the characters slightly as we move them. This allows us to see that indeed we've taken control of the character set.

```
300 FOR J=DEC("C000") TO
  DEC("C7FF") STEP 8
310 FOR K=0 TO 7
320 BANK 14
330 X=PEEK(J+4096+7-K)
340 BANK 1
350 POKE J+K,X
360 NEXT K
370 NEXT J
```

This puts the character set in place. When you run the program (after typing in the additional lines below), you should see your original computer screen—slightly changed. We could insert a delay loop to prolong the effect, but the screen takes long enough to change that you'll have plenty of time to see what happens.

Cleaning Up

We're finished—almost. We must be neat and put everything back the way it was. This also gives you a chance to see the original values that were in the various registers and addresses.

```
500 BANK 15
510 POKE DEC("DD00"),151
520 POKE DEC("0A2C"),20
530 POKE DEC("D506"),4
540 POKE 217,0
```

These lines restore the original screen. A little study should enable you to guess at what each POKE does—or undoes.

Finally, we need two last lines to complete the job. But there's an important note: *Do not* enter these lines until you've tested the program and found it good. If your program has a problem, you'll want to be able to look at the variables (by using commands such as PRINT J) to find out what went wrong. These final lines make it impossible for you to do so.

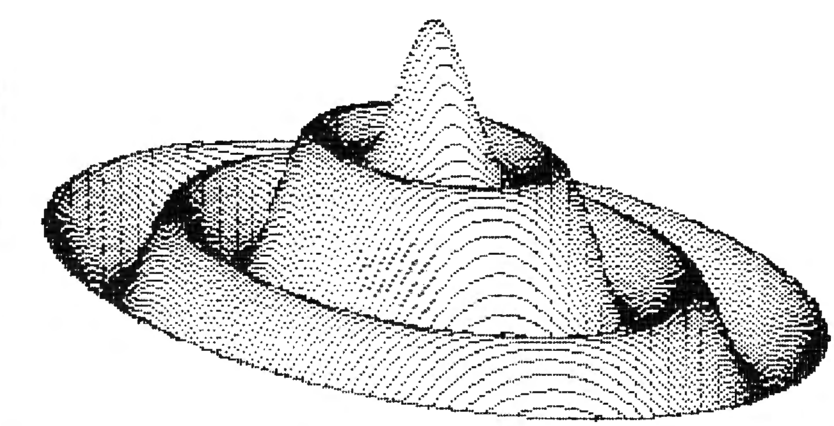
```
550 POKE 58,DEC("FF")
560 CLR
```

We've given back to the computer its variable storage memory. And the job is complete. ©

Apple Hi-Res Screen Dump

Mark Russinovich

You can easily dump high-resolution graphics pictures onto a dot matrix printer with this efficient machine language utility. It's also an ideal way to add a screen dump option to your own BASIC programs. It requires an Apple IIe or II+ computer with at least 48K RAM and an Epson or Epson-compatible printer, as well as an Epson or Epson-compatible parallel interface card that connects to slot 1. For both DOS 3.3 and ProDOS.



Have you ever wished you could print out an image that appears on the hi-res screen? This can be useful for inserting graphs or charts directly into text, or just saving interesting pictures and mathematical plots. With the program below, "DUMP," you can do all these things with minimal effort.

Using DUMP

To get started, type in Program 1 using the "Apple MLX" machine language entry program found elsewhere in this issue. Be sure you understand the instructions for using MLX before you begin entering the data from Program 1. The required starting and ending addresses for DUMP are:

Starting address: 9000
Ending address: 91DF

After you finish typing in the data, use MLX to save it to disk with the name DUMP. To install DUMP in memory for later use, just type BRUN DUMP. It loads itself into memory, protects itself from Applesoft by resetting HIMEM, and changes the ampersand vector to allow access from Applesoft.

Program 2 makes it easy to catalog disks, load hi-res pictures, view the pictures, and dump them on the printer. Just select the function you want from the menu. When you choose to print the hi-res screen, the program asks you to specify the size of the printout. There are nine sizes, ranging from a small block to a full page. (Owners of Epson MX-series printers should note that sizes 2, 3, 6, and 7 will *not* work with their printers. These sizes use codes available only on Epson's newer FX and RX models.) Next you'll be asked for a tab value. This lets you position the picture exactly where you want it. Specify the tab value in pica characters, making sure the value does not exceed the width of the page minus the width of the picture. Otherwise,

Table of DUMP Sizes

Size	Width Pica	Width Elite	Width Inch	Height Char	Height Inch
1	24	28	2.31	14	2.31
2	35	42	3.50	14	2.31
3	35	42	3.50	32	5.31
4	47	56	4.62	14	2.31
5	47	56	4.62	32	5.31
6	58	69	5.75	32	5.31
7	58	69	5.75	48	8.00
8	70	84	7.00	32	5.31
9	70	84	7.00	48	8.00

the picture might be cut off at the edge of the page, or wrap around to the middle. If you enter a tab value of zero, DUMP automatically centers the picture on the page.

To embed a picture within the text of a document, you should leave room for the pictures in your document by changing the margins. For your convenience, the accompanying table shows the widths and heights of all nine print sizes. After printing out your document, rewind the paper to position the print head about one line above the space you left for the picture. Then run Program 2 and request the size and tab value you planned for. This procedure might take a little practice before you can place a picture exactly where you want it.

Note that DUMP sets the printer to all of its default values after running. If you were using a special mode or typeface, you'll have to restore that mode after running DUMP.

DUMP With Other Programs

DUMP is especially handy when used with graphing and drawing programs, and for this reason you may want to add it to programs of your own. To do this, add a line at the beginning of the program similar to this:

10 PRINT CHR$(4);"BRUN DUMP"

Later in the program, add a screen dump option to your menu. Prompt the user for size and tab values, then enter this command:

&P,S,T

where P specifies hi-res page (1 or 2), S is the size (1–9, but remember the MX-series limitation mentioned above), and T is the tab value. Program 2 is an example of how this is done. Numbers, variables, or expressions can be used in the command. For instance, to print out hi-res page 1, with a size of 3 and a tab of 15, this form could be used:

10 A=15
20 & 1,A/5,A

After DUMP has finished printing the picture, it returns control to Applesoft and the program continues running.

The ampersand command can also be entered in immediate mode.

Program 1: MLX Data For DUMP

```
9000: A9 4C 8D F5 03 A9 18 8D 96
9008: F6 03 A9 90 8D F7 03 A9 9F
9010: FF 85 73 A9 8F 85 74 60 77
9018: 20 5B 91 20 F8 E6 E0 01 7A
9020: F0 07 E0 02 F0 07 20 C9 65
9028: DE A9 20 D0 02 A9 40 85 F0
9030: E6 20 BE DE 20 F8 E6 E0 26
9038: 0A B0 1A CA 8E D5 91 20 B9
9040: BE DE 20 F6 E6 E0 00 D0 97
9048: 06 AC D5 91 BE 6B 91 8E C0
9050: DB 91 4C 58 90 20 C9 DE 4A
9058: A0 00 84 F9 84 FA 84 EF 03
9060: 84 FB 84 FD 84 FE 84 FF 5C
9068: AC D5 91 A9 1B 20 62 91 D1
9070: A9 33 20 62 91 B9 74 91 4B
9078: 20 62 91 B9 7D 91 8D D6 34
9080: 91 B9 86 91 8D D7 91 B9 6B
9088: 8F 91 8D D8 91 AA CA 8E 70
9090: D9 91 B9 98 91 8D DA 91 CD
9098: 98 0A A8 B9 A8 91 85 06 D5
90A0: B9 A9 91 85 07 A0 00 B1 FF
90A8: 06 99 DC 91 C8 C0 03 D0 08
90B0: F6 A9 0A 20 62 91 AC DB 89
90B8: 91 F0 08 A9 20 20 62 91 52
90C0: 88 D0 F8 A0 00 A9 1B 20 80
90C8: 62 91 B9 DC 91 20 62 91 E7
90D0: C8 C0 03 D0 F5 A5 FB 85 B7
90D8: FC A5 FC C9 C0 B0 28 A6 DD
90E0: F9 A4 FA 20 11 F4 A4 EF 1F
90E8: B1 26 A4 FF 39 A1 91 F0 65
90F0: 16 AD DA 91 A4 FD F0 0B 07
90F8: AE D6 91 4A CA D0 FC 88 1A
9100: 4C F6 90 05 FE 85 FE E6 5C
9108: FC E6 FD A5 FD CD D8 91 E7
9110: D0 C7 A5 FE AC D7 91 20 3A
9118: 62 91 88 D0 FA A9 00 85 F2
9120: FE 85 FD E6 FF A5 FF C9 B2
9128: 07 D0 06 E6 EF A9 00 85 DD
9130: FF E6 F9 D0 02 E6 FA A5 A0
9138: F9 C9 18 D0 98 A5 FA F0 1D
9140: 94 A9 00 85 EF 85 F9 85 7F
9148: FA A5 FB 6D D9 91 85 FB C4
9150: C9 C0 B0 03 4C B1 90 20 39
9158: 5B 91 60 A9 1B 20 62 91 E3
9160: A9 40 AE C1 C1 30 FB 8D AE
9168: 90 C0 60 1D 16 16 11 11 1E
9170: 0A 0A 05 05 15 15 12 15 42
9178: 12 12 12 12 12 01 01 02 25
9180: 01 02 02 03 02 03 01 03 36
9188: 03 01 01 05 05 03 03 07 1F
9190: 07 03 07 03 03 02 03 02 31
9198: 40 40 60 40 60 60 70 60 C1
91A0: 70 01 02 04 08 10 20 40 BD
91A8: BA 91 BD 91 C0 91 C3 91 C3
91B0: C6 91 C9 91 CC 91 CF 91 CB
91B8: D2 91 4C 18 01 5A 48 03 B9
91C0: 5A 48 03 4B 18 01 4B 18 AB
91C8: 01 5A 78 05 5A 78 05 4C 6D
91D0: 48 03 4C 48 03 A2 D6 C9 01
91D8: A5 8A A0 C5 C8 FF A0 31 9A
```

Program 2: DUMP Example

```
81 10 ONERR GOTO 40
52 20 D$ = CHR$ (4)
D5 30 PRINT D$"BRUN DUMP"
55 40 TEXT : HOME
13 50 HTAB 9: PRINT "************
      ***********": HTAB 9: PRINT
       "*                      *":
      HTAB 9: PRINT "* APPLE HI-
      RES DUMP *": HTAB 9: PRINT
       "*                      *":
      HTAB 9: PRINT "************
      ***********"
1F 60 PRINT : PRINT : PRINT : HT
      AB 12: PRINT "ENTER CHOICE
      :": HTAB 12: PRINT "------
      --------": PRINT : HTAB 12:
       PRINT "1) CATALOG": PRINT
       : HTAB 12: PRINT "2) LOAD
       SCREEN"
BA 70 PRINT : HTAB 12: PRINT "3)
       VIEW SCREEN": PRINT : HTA
      B 12: PRINT "4) PRINT SCRE
      EN": PRINT : HTAB 12: PRIN
      T "5) QUIT"
D7 80 VTAB 22: HTAB 12: GET A$:
      IF A$ = "1" THEN 140
2A 90 IF A$ = "2" THEN 160
F6 100 IF A$ = "3" THEN 170
74 110 IF A$ = "4" THEN 200
48 120 IF A$ = "5" THEN END
D8 130 PRINT CHR$ (7): GOTO 80
68 140 PRINT : HOME : PRINT D$"C
       ATALOG"
E3 150 PRINT : PRINT "PRESS ANY
       KEY:";: GET A$: GOTO 40
DD 160 PRINT : VTAB 22: INPUT "E
       NTER FILE NAME: ";FL$: PR
       INT D$"BLOAD"FL$",A$2000"
       : HOME : GOTO 40
81 170 POKE - 16302,0: POKE - 16
       297,0: POKE - 16304,0: PO
       KE - 16368,0
87 180 X = PEEK ( - 16384): IF X
        < 128 THEN 180
28 190 POKE - 16368,0: TEXT : HO
       ME : GOTO 40
11 200 PRINT : VTAB 22: INPUT "E
       NTER SIZE OF DUMP (1-9):
       ";S: IF S < 1 OR S > 9 TH
       EN 200
DD 210 VTAB 24: PRINT "(0=AUTO C
       ENTER)";: VTAB 23: HTAB 1
       : INPUT "ENTER TAB SETTIN
       G: ";T: IF T < 0 OR T > 5
       0 THEN 210
7A 220 POKE - 16302,0: POKE - 16
       297,0: POKE - 16304,0: &
       1,S,T: TEXT : HOME : GOTO
        40
```

©

Disassembler For Atari

William Casner

This versatile utility disassembles any machine language program in memory or on disk. It can also display a memory dump and check disks for bad sectors. The program works on any 400/800, XL, or XE with at least 16K RAM for tape or 24K for disk.

Here is a BASIC utility for disassembling machine language (ML) programs and examining the contents of your Atari's memory. Type in "Disassembler" and save it to disk or tape before running it for the first time. Since this program is largely self-prompting, you should be able to use it with little or no instruction. To choose one of its three main options, press the OPTION, SELECT, or START keys as prompted. In each case, you may choose to send output to a printer rather than to the screen.

Using The Disassembler

The first option, disassembly, translates ML object code into its 6502 mnemonics. After you choose this option, the computer asks whether you wish to disassemble a particular memory area, a particular sector on the disk, or a binary file stored on disk. This allows you the freedom to disassemble virtually any ML program, even autoboot programs that normally take control of the computer as soon as you load them into memory.

The size of the disk file you can disassemble depends on the memory capacity of your computer: With 48K or 64K, you can disassemble files as large as 21K (more than 21,000 bytes). When disassembling memory, you must provide hexadecimal starting and ending addresses of the area you wish to disassemble.

The second option is a listing, or memory dump. Again, you can look at a particular memory area, a particular disk sector, or a binary file. In this case, however, the disassembler displays each byte in ASCII form rather than as a 6502 mnemonic. This function is useful for examining parts of a program that contain data rather than ML instructions.

Finally, you can scan a disk for bad sectors. After you select this option, the program checks every sector on the disk, listing the type and sector location of any errors that are found.

As you may know, CTRL-1 can be used to pause any scrolling screen display. Press Q at any input point (except the menus themselves) to return to the main menu. If you wish to abort a disassembly or memory dump, press the START key: The computer asks you to press any key to continue, then returns you to the main menu.

Take special care while typing the DATA statements in lines 1230–1310. Don't omit any commas or spaces, but don't add any extra ones, either. Mistakes could lead to incorrectly decoded mnemonics. If the program stops with an ERROR 3, 6, or 8 message in line 1220, it probably means you have a typing error somewhere in the DATA lines.

Disassembler For Atari

For instructions on entering this listing, please refer to "COMPUTE!'s Guide to Typing In Programs" published bimonthly in COMPUTE!.

```
DN 10  DIM R$(1032),SC$(128),
       ML$(36),ML2$(34),G$(31
       ),A$(20),T$(15),F$(4),
       U$(1)
EO 20  DIM P(4):P(1)=4096:P(2
       )=256:P(3)=16:P(4)=1
PO 30  DIM TY$(11):TY$="VUBC@
       ZJYXNR":SPACE=FRE(0)-6
       00:DIM S$(SPACE)
MN 40  FOR J=1 TO 36:READ B:M
       L$(J)=CHR$(B):NEXT J
BP 50  DATA 104,104,141,11.3,
       104,141,10,3,104,141,5
       ,3,104,141,4,3,169,1,1
       41,1,3
MJ 60  DATA 169,82,141,2,3,32
       ,83,228,173,3,3,133,20
       3,96
AA 70  FOR J=1 TO 34:READ B:M
       L2$(J)=CHR$(B):NEXT J
MM 80  DATA 104,104,141,105,3
       ,104,141,104,3,104,141
       ,101,3,104,141,100,3,1
       69,7
DM 90  DATA 141,98,3,162,32,3
       2,86,228,173,99,3,141,
       3,2,96
OA 100  GOSUB 1180:GOTO 200
HG 110  REM
BB 120  S=0:FOR X=1 TO 4
DI 130  A=ASC(A$(X,X))-48:IF
        A>9 THEN A=A-7
```

```
PC 140 S=S+P(X)*A:NEXT X:RET
       URN
HK 150 REM
CN 160 F=0:F$="":FOR X=1 TO
       4
KI 170 F=INT(A/P(X)):A=A-(F*
       P(X)):IF F<10 THEN F=
       F-7
AB 180 F$(X)=CHR$(F+55):NEXT
        X:RETURN
HO 190 REM
FB 200 ? "{CLEAR}":POSITION
       2,8:? "Press OPTION f
       or Disassembler":? "
       {6 SPACES}SELECT for
       Code Lister"
KP 210 ? "{6 SPACES}START f
       or Sector Scan":? :?
       ">";
DE 220 ON PEEK(53279)-2 GOTO
        230,220,240,980:GOTO
        220
FG 230 DIS=1:T$="Disassemble
       r":G$="{CLEAR}ADDR ME
       MORY{3 SPACES}OPCODE
        OPERAND":GOSUB 1120:
       GOTO 250
MA 240 DIS=0:T$="Lister":G$=
       "{CLEAR}{9 SPACES}BYT
       E #{4 SPACES}BYTE ASC
       II"
IA 250 ? "{CLEAR}":? :? "
       {13 SPACES}";T$
JH 260 POSITION 2,8:? "Press
        OPTION Memory":? "
       {6 SPACES}SELECT Sect
       or":? "{6 SPACES}STAR
       T File"
MO 270 FOR X=1 TO 50:NEXT X:
       ? :? ">";
ED 280 ON PEEK(53279)-2 GOTO
        290,280,360,500:GOTO
        280
AG 290 ADD=1:SA=SS:? :? "Sta
       rting address(4 digit
        hex)";
CM 300 INPUT A$:GOSUB 1090:I
       F LEN(A$)<>4 THEN ? "
       {3 UP}":GOTO 290
MK 310 GOSUB 120:Y=S
OC 320 ? :? "Ending address(
       4 digit hex)";
CJ 330 INPUT A$:GOSUB 1090:I
       F LEN(A$)<>4 THEN ? "
       {3 UP}":GOTO 320
BC 340 GOSUB 120:SE=S:S=Y:IF
        DIS=0 THEN 860
GK 350 GOTO 630
NI 360 ADD=0:MAXS=INT(SPACE/
       128):S=ADR(S$):GOSUB
       1120
EJ 370 ? :? "Starting sector
        (1-719)";:INPUT A$:I
       F LEN(A$)=0 THEN SS=1
       :GOTO 390
LA 380 GOSUB 1090:SS=VAL(A$)
       :IF SS<1 OR SS>719 TH
       EN ? "{3 UP}":GOTO 37
       0
JF 390 ? :? "Ending Sector (
       1-719)";:INPUT A$:IF
       LEN(A$)=0 THEN ES=SS:
       GOTO 420
PG 400 GOSUB 1090:ES=VAL(A$)
       :IF ES<SS OR ES>719 T
       HEN ? "{3 UP}":GOTO 3
       90
MG 410 IF ES-SS>MAXS THEN ?
       "Only room for ";MAXS
       ;" sectors":? "{6 UP}
       ":GOTO 370
CO 420 SECTOR=SS-1:? :? "Pre
       ss RETURN to begin";:
       INPUT S$
MN 430 SECTOR=SECTOR+1:A=USR
       (ADR(ML$),SECTOR,ADR(
       SC$)):IF PEEK(203)=1
       THEN 460
CA 440 ? "Sector ";SECTOR;"
       bad: ";PEEK(203):IF S
       ECTOR=ES THEN 1060
GJ 450 GOTO 430
AC 460 S$(LEN(S$)+1)=SC$:IF
       SECTOR=ES THEN SE=S+L
       EN(S$):GOTO 480
GL 470 GOTO 430
EF 480 IF DIS=0 THEN 860
GP 490 GOTO 630
ND 500 ADD=1:S$(1)=" ":S$(SP
       ACE)="|":S$(2)=S$
MB 510 SS=0:SE=0:? :? "Enter
        D#:filename.ext";:IN
       PUT A$:IF LEN(A$)=0 T
       HEN ? "{3 UP}":GOTO 5
       10
LE 520 GOSUB 1090:IF A$(2,2)
       <>":" AND A$(3,3)<>":
       " THEN ? "BAD FILENA
       ME FORMAT":? "{4 UP}
       ":GOTO 510
LK 530 CLOSE #2:TRAP 550:OPE
       N #2,4,0,A$:GET #2,A:
       IF A=255 THEN GET #2,
       A:IF A=255 THEN 570
PA 540 CLOSE #2:TRAP 40000:?
        :? "Not a BINARY LOA
       D file":GOTO 1060
DC 550 CLOSE #2:TRAP 40000:I
       F PEEK(195)=170 THEN
       ? "FILE NOT FOUND":?
       "{4 UP}":GOTO 510
BK 560 ? "UNRECOVERABLE DISK
        ERROR":GOTO 1060
KN 570 GET #2,A:GET #2,B:SS=
       B*256+A
YA 580 GET #2,A:GET #2,B:SE=
       B*256+A
DJ 590 NOBYTES=SE-SS+1:IF NO
       BYTES>SPACE THEN ? :?
        "Not Enough RAM ":GO
       TO 1060
JE 600 A=USR(ADR(ML2$),NOBYT
       ES,ADR(S$)):IF PEEK(2
       03)=255 THEN ? :? "ER
       ROR #";PEEK(203):GOTO
        1060
NK 610 CLOSE #2:S=ADR(S$):SA
       =S:SE=S+NOBYTES
ES 620 IF DIS=0 THEN 860
EK 630 IF S>SE THEN 1060
NL 640 ? G$:IF PTR=1 THEN ?
       #3;G$
BD 650 IF PEEK(53279)=6 THEN
        1060
II 660 G$="{31 SPACES}"
HF 670 A=S-SA+SS:GOSUB 160:G
       $(1,4)=F$
HL 680 Z=PEEK(S):A=Z:GOSUB 1
       60:G$(6,7)=F$(3,4)
HG 690 IF R$(Z*4+1,Z*4+1)="
       " THEN G$(17,19)="???
       ":S=S+1:GOTO 630
DG 700 G$(17,19)=R$(Z*4+1,Z*
       4+3):U$=R$(Z*4+4):IF
       U$=" " OR U$="A" THEN
        G$(24,24)=U$:S=S+1:G
       OTO 630
DC 710 G$(24,24)="$":A=0:FOR
        J=1 TO 11:IF U$=TY$(
       J,J) THEN A=J:J=11
FM 720 NEXT J:ON A GOTO 730,
       740,750,760,770,780,7
       90,800,810,820,830:ST
       OP
GN 730 G$(27,28)=",Y":GOTO 7
       80
GN 740 G$(27,28)=",X":GOTO 7
       80
CN 750 G$(23,23)="(":G$(27,2
       9)=",X)":GOTO 780
CP 760 G$(23,23)="(":G$(27,2
       9)="),Y":GOTO 780
PE 770 G$(23,23)="#"
NM 780 A=PEEK(S+1):GOSUB 160
       :G$(9,10)=F$(3,4):G$(
       25,26)=F$(3,4):S=S+2:
       GOTO 630
KK 790 G$(23,23)="(":G$(29,2
       9)=")":GOTO 820
GB 800 G$(29,30)=",Y":GOTO 8
       20
FE 810 G$(29,30)=",X"
HO 820 A=PEEK(S+2)*256+PEEK(
       S+1):GOSUB 160:G$(9,1
       0)=F$(3,4):G$(12,13)=
       F$(1,2):G$(25,28)=F$:
       S=S+3:GOTO 630
CI 830 Z=PEEK(S+1):A=Z:GOSUB
        160:G$(9,10)=F$(3,4)
       :IF Z<128 THEN G$(22,
       22)=">":A=S-SA+SS+Z+2
       :GOTO 850
PD 840 G$(22,22)="<":A=S-SA+
       SS+Z-254
NE 850 GOSUB 160:G$(25,28)=F
       $:S=S+2:GOTO 630
MB 860 ? G$:?
EN 870 IF S>SE THEN 1060
BI 880 IF PEEK(53279)=6 THEN
        1060
FI 890 IF ADD=0 THEN ? ,S-AD
       R(S$),:GOTO 910
HG 900 A=S-SA+SS:GOSUB 160:?
        "{8 SPACES}";F$;" ";
       S;"  ";
NO 910 Z=PEEK(S):A=Z:GOSUB 1
       60:? F$(3,4);"
       {5 SPACES}";
LG 920 IF Z=125 THEN ? "
       {ESC}{CLEAR}":GOTO 97
       0
NM 930 IF Z=157 THEN ? "
       {ESC}{INS LINE}":GOTO
        970
NN 940 IF Z=158 THEN ? "
       {ESC}{DEL LINE}":GOTO
        970
KL 950 IF Z=29 THEN ? "{ESC}
       {DEL LINE}":GOTO 970
IK 960 ? CHR$(Z)
PB 970 S=S+1:GOTO 870
FG 980 ? "Insert diskette to
        scan":? :? "Press RE
       TURN when ready";:INP
       UT A$:GOSUB 1090
PC 990 N=0:SECTOR=0:? "
       {CLEAR}":POSITION 14,
       2:? "SECTOR SCAN":? :
       ? "{12 SPACES}Bad Sect
       ors"
FY 1000 ? ,"Sector","Error"
DN 1010 SECTOR=SECTOR+1:A=US
        R(ADR(ML$),SECTOR,AD
        R(SC$))
BI 1020 IF PEEK(203)<>1 THEN
         ? ,SECTOR,PEEK(203)
        :N=N+1
JN 1030 IF SECTOR=720 THEN ?
         :? ,N;" Bad Sectors
        ":GOTO 1060
AI 1040 IF PEEK(53279)=6 THE
        N 200
MB 1050 GOTO 1010
NP 1060 ? :? "Press any key
        to continue":? ">";
DO 1070 IF PEEK(764)=255 THE
        N 1070
GG 1080 POKE 764,255:GOTO 20
        0
```

```
EN 1090 IF LEN(A$)=0 THEN 1110
IK 1100 IF A$(1,1)="Q" THEN POP :GOTO 200
KD 1110 RETURN
KB 1120 PTR=0:? "Do you wish to print(Y/N)";
NM 1130 IF PEEK(764)=43 THEN PTR=1:GOTO 1150
DJ 1140 IF PEEK(764)=255 THEN 1130
KC 1150 POKE 764,255:IF PTR=1 THEN CLOSE #3:TRAP 1170:OPEN #3,8,0,"P:":TRAP 40000
JK 1160 ? :? :RETURN
PG 1170 CLOSE #3:TRAP 40000:? :? "Unable to start printer":GOTO 1060
AI 1180 ? "{CLEAR}6502 DISASSEMBLER AND CODE LISTER":? :?
PF 1190 ? "{7 SPACES}PLEASE WAIT...."
OG 1200 R$(1)=" ":R$(1032)="|":R$(2)=R$
MM 1210 SC$(1)=" ":SC$(128)="|":SC$(2)=SC$
PI 1220 FOR X=0 TO 255:READ F$:R$((X*4)+1,(X*4)+4)=F$:NEXT X:RETURN
OG 1230 DATA BRK ,ORAB,,,,ORAZ,ASLZ,,PHP ,ORA@,ASLA,,,ORAN,ASLN,,BPLR,ORAC,,,,ORAU,ASLU,,CLC ,ORAY,,,,ORAX
KH 1240 DATA ASLX,,JSRN,ANDB,,,BITZ,ANDZ,ROLZ,,PLP ,AND@,ROLA,,BITN,ANDN,ROLN,,BMIR,ANDC,,,,ANDU,ROLU,,SEC ,ANDY,,,
AP 1250 DATA ANDX,ROLX,,RTI,EORB,,,,EORZ,LSRZ,,PHA ,EOR@,LSRA,,JMPN,EORN,LSRN,,BVCR,EORC,,,,EORU,LSRU,,CLI,EORY,,,
IA 1260 DATA EORX,LSRX,,RTS,ADCB,,,,ADCZ,RORZ,,PLA ,ADC@,RORA,,JMPJ,ADCN,RORN,,BVSR,ADCC,,,,ADCU,RORU,,SEI,ADCY,,,
HB 1270 DATA ADCX,,,,STAB,,,STYZ,STAZ,STXZ,,DEY,,TXA ,,STYN,STAN,STXN,,BCCR,STAC,,,STYU,STAU,STXV,,TYA ,STAY,TXS ,,
AJ 1280 DATA STAX,,,LDY@,LDAB,LDX@,,LDYZ,LDAZ,LDXZ,,TAY ,LDA@,TAX ,,LDYN,LDAN,LDXN,,BCSR,LDAC,,,LDYU,LDAU,LDXV,
EN 1290 DATA CLV ,LDAY,TSX ,,LDYX,LDAX,LDXY,,CPY@,CMPB,,,CPYZ,CMPZ,DECZ,,INY ,CMP@,DEX ,,CPYN,CMPN,DECN,,BNER
FC 1300 DATA CMPC,,,,CMPU,DECU,,CLD ,CMPY,,,,CMPX,DECX,,CPX@,SBCD,,,CPXZ,SBCZ,INCZ,,INX,SBC@,NOP ,,CPXN,SBCN,INCN,
ID 1310 DATA BEQR,SBCC,,,,SBCU,INCU,,SED ,SBCY,,,,SBCX,INCX,
```
©

CAPUTE!

Modifications or Corrections To Previous Articles

Atari Witching Hour

Goblins apparently invaded our lister program while this Halloween game from the October issue (p. 54) was printing. The mysterious {=} character in lines 1310 and 1320 should instead be the vertical line character, SHIFT-=.

Skyscape

The Commodore 64, Atari, and TI versions of this astronomy plotting program from the November issue (p. 62) do not work properly for latitudes between the equator and 24 degrees south. Trying to plot a skyscape for a location in this area—Peru or northern Australia, for example—results in an ILLEGAL QUANTITY ERROR message or a misplaced sun. In the Commodore 64 version (Program 1), the culprit is the second ABS in line 2510. The line should read as follows:

```
2510 IF ABS(LL)<24 THEN LB=40*
  INT(LL/7+.5)
```

The correction is the same for the Atari version (Program 2), except that the line number is 2540. For the TI version (Program 5), make the change to line 2440.

All About IBM Batch Files

The {CTRL-P} character which appears in Programs 2, 3, and 4 of this overview of batch files is not correct. Wherever this character appears, you should instead type whatever key or key combination produces an ESCape character, CHR$(27). If you use the EDLIN text editor from the IBM PC-DOS system disk, the proper replacement is {CTRL-V}[. That is, hold down the CTRL key and type V, then release CTRL and V and type [. Note that the left bracket ([) is in addition to any brackets that are already in the listing. For example, with EDLIN the first line of Program 3 would be typed as follows:

```
{CTRL-V}[[2J {CTRL-V}[[32m
```

Other text editors or word processors may require another combination. Check the manual for the editor you are using to see what you need to type to produce ASCII character 27.

There is also a correction for the last paragraph in the article (p. 88). The statement shown as:

```
IF .– – %1. GOTO .NOPARAM
```

should read:

```
IF .== %1. GOTO :NOPARAM
```

64 Color Plotter

There are no errors in this graphics utility program from the "64 Multicolor Graphics Made Easy" article in the October issue (p. 90). However, there was one point that the article failed to make completely clear: Programs with "Color Plotter" commands work *only if they are typed in while Color Plotter is active.* If you type in a program containing Color Plotter commands—for example, Program 2 from the article—in regular BASIC, then activate Color Plotter, the program appears correct when you list it, but will not run. Instead, all Color Plotter commands will cause syntax errors. You can convert the faulty program statements to true Color Plotter statements by activating Color Plotter, listing the problem line on the screen, moving the cursor to that line, and pressing RETURN. Always be sure that Color Plotter is active before typing in any programs using its special commands. And remember that you have to reactivate Color Plotter each time you press RUN/STOP–RESTORE. ©

Classified

SOFTWARE

TWO NEW DISKS FOR 64-128
1-Electronic Encyclopedia (first of series)
menu select from OHMS thru FILTERS-
all needed formulas + explanations.
2-A new personal sci-fi game - great
sounds - very interactive - uses your name.
$29/disk. Both for $55 + $3 p/h per order.
TECH ED LTD., RFD #2, Exeter, NH 03833.

COMMODORE: TRY BEFORE YOU BUY. Top 25 best-selling games + classics, new releases. Visa, MasterCard. Free brochure. Rent-A-Disk, 908 9th Ave., Huntington, WV 25701. (304) 522-1665

ATARI ST USERS - Life organizer & entertainment jackpot. Big software package. Write: MC, 94 Macalester Bay, Winnipeg, Manitoba R3T 2X5 Canada

TIM, The Investment Manager. COMEX, gold and silver management. FUN-ANL stock analysis program. C64. All three $19.95 or write for free details to: Author's Club Software, 6027 S. High, Suite 410, Oklahoma City, OK 73149

TI-99/4A QUALITY SOFTWARE for Business, Home and Entertainment ** BONUS Software Offer! ** Send for FREE Catalog to MICRO-BIZ HAWAII, Box 1108 Pearl City, HI 96782

Free Educational Software Catalog
158 pp, color pictures, 1000+ programs for Com., Apple, IBM, Atari. Send name & address to: Interstate Software, P.O. Box 8952, Boise, ID 83707, (208) 342-3347

ORGANIZE YOUR GENEALOGY with BRANCHES on your C64 & 1541 drive. Send $14.95 to Inventure Soft., 3353 S. Main, Suite 101, Salt Lake City, Utah 84115

TI-99/4A Software/Hardware bargains. Hard-to-find items. Huge selection. Fast service. Free catalog.
D.E.C., Box 690, Hicksville, NY 11801

GOMUKO — Compiled basic program 15 by 15 board, Unmove, load/save game user changable logic: $14.95. Star traders for 2 to 5 players. Show off your 64 when guests come over: $14.95. Send check or m.o. to: Author's Club Software, 6027 S. High, Suite 410, Oklahoma City, OK 73149

LOTTO PICKER. Go for Million Dollar Jackpots! Picks all USA Lotto games +! PRO FOOTBALL ANALYST. Beat the points consistently and easily! They pay for themselves! IBM/C64/TI99 $35. Order 1-800-341-1950 Ext. 77. Mail Orders: RIDGE, 170 Broadway, Suite 210-C NYC, NY 10038.

FREE PROGRAMS! FREE PROGRAMS! C64/C16/+4/V20/TI99-4A/TIMEX 1000/2068/IBM pc/TRS80III/4/PC3/CoCo/MC10. Send stamps! EZRAEZRA, Box 5222 TDE, San Diego, CA 92105

FREE SOFTWARE CATALOG!
Call Toll-Free 1-800-554-1162, Tevex, Inc. Save ⅓ off retail prices. We carry SSI, Elect. Arts, Infocom, and many more!

NEW BARMAID IS THE LIFE OF THE PARTY!
Barmaid software shows you how to mix over 100 drinks. Add more, too. Search by full or part name of drink. Shows the ingredients & prep. Pass sobriety test before exiting. Great gift for the IBM-PC family who have everything. $34.95 ck or m.o., Visa/MC from: DPR Software, Inc., 588 Rte. 70 West, Bricktown, NJ 08723, (201) 920-8890

BIBLE QUIZ GAMES and other Bible software for the C64. Fun and learn for all ages. BIBLE-MATCH-WITS I (easy), II (hard). STAIRWAY TO HEAVEN 1, 2, 3. Colorful, animation, graphics, and sound. Each $29.95. Order or send SASE for brochure to COMPEDS, P.O. Box 147, Narrows, VA 24124

80 Column Word Processor for 48K Atari.
No additional hardware required. Any printer. Easy to use!! disk, complete manual, warranty, $69.95. FLAPS LANDING, 393 Carmen Rd., Amherst, NY 14226

COMPUTER SOFTWARE! 35%–40% OFF!
Send phone number, specify software.
We will call with quote. Next day shipping!
C & D Assoc., Box 851, Mt. Prospect, IL 60056

ATARI-ENHANCEMENTS TO BASIC-800/XL/XE A disk based program that fixes the 800 and 800XL lockup bug and also adds many valuable commands to BASIC. Reviewed in May '85 *Analog* and June '85 *COMPUTE!* $24.95 VISA/MC/CK/MO/COD 412-627-3596 FIRST BYTE, Box 32, Rices Landing, PA 15357

WARGAME to prevent war! - STATE OF WAR situation simulator predicts conflict/peace events for all nations, alliances, guerrilla groups. Free info: Kilborn, Box 4692-Stn. E, Ottawa, Canada K1S 5H8

Free Spirit Software for the C64:
POSTMASTER - Simple, efficient, mailing list program. Disk: $19.95
BASICally SIMPLE - A quick, easy method to learn Basic programming. Disk: $20
TECHNIQUE - Learn to program graphics, animation, sound, music easily. Disk: $29.95
ITALY - Travel and educational game. Disk: $15
Order from: Free Spirit Software, Inc.
5836 S. Mozart, Chicago, IL 60629

HARDWARE

Trade in your used Commodore or Atari on a brand new C-128 or Atari ST. This offer may not be available through retail outlets. Brochure $2.00 and SASE. NEW WEST TECHNOLOGY, 4B Monroe Pkw., Box 200, Ste. 134, Lake Oswego, OR 97034

HARDWARE & SOFTWARE 30% BELOW RETAIL. Apple, Atari, C64, IBM-PC, TI-99. Over 1000 titles. Hard to find items. Send $1.00 for catalog. Specify computer. Multi-Video, P.O. Box 246, East Amherst, NY 14051

DISK SERVICE MANUAL II:
Comprehensive maintenance + repair manual on standard-bus 5.25", 8" drives, microfloppies and Apple/Commodore drives. No special software or equipment required! Over 100 labeled photos or illustrations. Manual plus free $1 catalog: $22. CONSUMERTRONICS, P.O. Drawer 537-X, Alamogordo, NM, 88310

PROWRITER, C.ITOH 8510 P, APPLE DMP, and IMAGEWRITER OWNERS: Add an extra 2K of buffer capacity. 100% compatible. Only $19.95! Send to: PRINTER-BUFFER, P.O. Box 1097, St. Louis, MO 63026

MISCELLANEOUS

64 AUTHOR'S CLUB - We get you published. Send for free details or send $25.00 (a 50% savings) to: Author's Club, 6027 S. High, Suite 410, Oklahoma City, OK 73149

HELP IS ON THE WAY!
Just call 1-800-334-0868 to get your free copy of the latest COMPUTE! Books Catalog! If you need help in getting information on all of the latest COMPUTE! book titles available plus all COMPUTE! backlist titles, call us today!

RIBBONS for ANY PRINTER at LOW PRICES!!
DELTA MICRONICS
BOX 10933, ERIE, PA 16514
(814) 455-5667

*** MR. SOFTWARE CO. ALL POPULAR TITLES ***
* Printers, Monitors, Drives, VISA, MC *
heavy discounts - Send $1.00 for catalog
11-9 Exton Complex, Somers Point, NJ 08244

FREE! USE YOUR MODEM! Call our innovative electronic shopping center, FANTASY PLAZA. VISA and MASTERCARD accepted. 300 BAUD. You've never seen anything like it! Use your Modem NOW! (818) 840-8066

EARN MONEY, PART OR FULL TIME, AT HOME with your computer. 50 page manual with forms. Money back guarantee, $9.95, JV Tech, P.O. Box 563, Ludington, MI 49431

STOCKING STUFFER
1986 Calendar. Complete history of computers, especially micros. Spiral bound, $6 ch or m.o. Same day mailing. Heat Stroke Software, Box 62171, Tucson, AZ 85734-6171

COMPUTE! Classified is a low-cost way to tell over 350,000 microcomputer owners about your product or service.

Rates: $25 per line, minimum of four lines. Any or all of the first line set in capital letters at no charge. Add $15 per line for boldface words, or $50 for the entire ad set in boldface (any number of lines.)

Terms: Prepayment is required. Check, money order, American Express, Visa, or MasterCard is accepted. Make checks payable to COMPUTE! Publications.

Form: Ads are subject to publisher's approval and must be either typed or legibly printed. One line equals 40 letters and spaces between words. Please underline words to be set in boldface.

General Information: Advertisers using post office box numbers in their ads must supply permanent address and telephone numbers. Orders will not be acknowledged. Ad will appear in next available issue after receipt.

Closing: 10th of the third month preceding cover date (e.g., June issue closes March 10th). Send order and remittance to: Harry Blair, Classified Manager, COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. To place an ad by phone, call Harry Blair at (919) 275-9809.

Notice: COMPUTE! Publications cannot be responsible for offers or claims of advertisers, but will attempt to screen out misleading or questionable copy.

The Beginners Page

Tom R. Halfhill, Editor

No Strings Attached

For the past few months, we've been discussing various kinds of *numeric variables*—those that store numbers. But BASIC has a second general type of variable that's worth knowing about, too—*string variables.*

Instead of storing numbers, string variables store characters. Characters can be letters of the alphabet, the numerals 0–9, punctuation marks, the foreign letters or graphics symbols found on some keyboards, spaces, and even special codes which have meaning only to computers.

In program listings, string variables resemble regular variables, but are denoted with a trailing dollar sign, as in A$ (pronounced "A-string"). Usually, all the rules that apply to numeric variable names on your computer's BASIC also apply to string variable names. For instance, the Commodore 64 allows variable names of any length, but the computer recognizes only the first two characters for purposes of telling them apart; ditto on the Apple; the IBM also allows names of any length, but recognizes the first 40 characters; the TI allows names up to 15 characters long, recognizing all 15; and the Atari allows names of any length and recognizes all characters.

String variables are easy to set up and use. You're probably already familiar with *literal strings,* such as the word HELLO found in the following program line:

```
10 PRINT "HELLO"
```

A literal string is analogous to a numeric constant—it doesn't change. PRINT "HELLO" always prints the word HELLO. But storing a string of characters in a string variable has the same advantages as storing a number in a numeric variable—your program can manipulate the variable (and therefore the characters it stores) at will. Here's a quick example:

```
10 A$="HELLO"
20 PRINT A$
```

(Atari users should add this line: 5 DIM A$(10). We'll explain why later.) When you run this program, it prints HELLO just like the previous program. But now add these lines:

```
30 A$="HI MOM!"
40 PRINT A$
```

Even though the PRINT statement in line 40 is identical to the one in line 20, it prints a different message: HI MOM! instead of HELLO. The reason, as you may have surmised, is that we assigned a new string of characters to the string variable A$ in line 30. In effect, we changed the "value" of A$ from HELLO to HI MOM!.

This is just a taste of how string variables can be modified by a running program. We'll cover many more possibilities over the next few columns. The important thing at this point is to grasp the advantage of string variables: They allow your programs to manipulate characters, words, and sentences instead of just numbers.

A DIM Memory

Take another look at the statements in lines 10 and 30 above. These are the string variable versions of assignment statements, just as the statement A=10 assigns the value 10 to the numeric variable A. (In case you're wondering, the rarely seen keyword LET—as in LET A=10—can be used in string variable assignments, too, but is optional in almost all BASICs these days. It's customary to omit it.)

When you assign a string of characters to a string variable, BASIC stores the string in computer memory and uses the variable as a reference marker—sort of like the thumb tabs on the pages of a large dictionary. A program statement such as PRINT A$ tells BASIC to look up the string of characters in memory, retrieve it, and print it on the screen.

In TI-99/4A BASIC and most Microsoft-style BASICs (including those supplied with Commodore, Apple, and IBM computers), there's a limit on the length of the string that can be assigned to a string variable—255 characters. If you try to assign a longer string, you'll either get an error message or the string will be *truncated* (cut off) at the 255-character limit.

In Atari BASIC, a string can be of any length up to the limit of available program memory. On a 48K or 64K Atari with the Disk Operating System (DOS) and BASIC in memory, there's room for a string of more than 30,000 characters—although that wouldn't leave much memory for a very long program. Because the length of Atari strings is so flexible, Atari BASIC requires you to declare the maximum length of a string variable before using it in the program. Otherwise, the computer wouldn't know how much space to reserve for the string (Microsoft BASIC always knows that strings won't be longer than 255 characters).

The Atari BASIC statement for declaring a string's length is DIM(*x*), where *x* equals the maximum number of characters (DIM stands for DIMension). An example of the DIM statement is in line 5 above. It reserves memory for a string up to ten characters long, room enough for HI MOM! with a few characters to spare. The DIM statement *must* precede the first use of the string variable in the program, or you'll get an error. If you try to assign a string longer than the DIMed length, the string is truncated at the limit *without* an error message.

Next month we'll start delving a little deeper into how to use string variables in various ways in your programs. ©

David D. Thornburg, Associate Editor

Another Kind Of Home Computing

At first glance, the Emergency Housing Consortium of Santa Clara County may seem to be an unlikely place to find personal computers. This agency, founded four years ago by Barry Del Buono, helps meet the emergency and long-term housing needs for residents of Santa Clara and San Mateo counties—two of the most populous counties in California's Silicon Valley.

To an outsider, the apparent affluence of this area masks its pockets of poverty—poverty that strikes quite hard, given the high cost of local housing. With rental units costing as much as $2,000 per month, many families who are down on their luck end up living in their cars or on the streets.

This is where Del Buono's agency steps in. In four years, the Emergency Housing Consortium has grown from one person to a staff of 35 people who oversee four shelters housing 600 people per night. In addition, the Consortium helps people find permanent housing and jobs.

As the agency began to grow, Del Buono contacted the Community Affairs program at Apple Computer, Inc. to apply for a corporate grant of computer equipment. He was convinced that computer technology could help his clients gain an edge on locating permanent housing. He envisioned an interagency network that would include a constantly updated list of low-cost area housing. Such a network was needed because by the time most of his people found out about a low-cost rental opportunity, it was already taken.

Apple granted four complete computer systems to the Consortium to share with three other housing agencies. The equipment included an Apple IIe computer with the extended 80-column card (expanding the memory to 128K RAM), a monitor, two disk drives, a ten-megabyte hard disk, a 1200 bps modem, and an Imagewriter printer. Apple also provided numerous pieces of its own software, as well as some products from other manufacturers (such as HabaMerge).

From Fast Food To Figures

The Apple Community Affairs grants are awarded primarily to nonprofit groups interested in using microcomputer networks to communicate and share information with other groups that have similar social objectives. Apple emphasizes the importance of cooperation between groups and the ways in which computers can help people cooperate across organizational boundaries.

When Apple provided the Consortium with the equipment and support it needed, the database envisioned by Del Buono became a reality. The legwork was done by volunteers and by the homeless clients themselves. "Pretty soon we were coming up with incredible stuff," says Del Buono. "We had the information available on a daily basis, and it was being updated all the time. Walk-ins could now come to our center and, in a short time, could walk out again with a list of appropriately priced rentals."

The computers became useful in other ways, too. Because the machines also store information about the Consortium's clients, it's easy to compile detailed statistics on them. This type of information is important to an agency that obtains funding from public sources.

Perhaps more importantly, the computers have provided opportunities for the clients themselves to learn how to use today's technology. One woman who had last worked for a fast-food restaurant is now the agency's statistician. She is so good at her job that she recently led a workshop at Apple. Other formerly homeless people working for the Consortium are also acquiring job skills that are transferable to the private sector. They are seeing how access to technology has a direct impact on improving their lives. This helps them recognize the importance of developing appropriate job skills in the information age.

Fringe Benefits

Meanwhile, thanks to the combined efforts of the clients and volunteers, the Consortium's constantly updated housing list is so valuable that it's now being sold to other agencies on a subscription basis. Even corporations are calling the Consortium to get rental information for their new employees!

Del Buono is convinced the Consortium couldn't be what it is today without the help of computers. His agency is decentralized, operating four shelters in two counties, and is linked to other agencies as well.

Above all, Del Buono has shown that computer technology can benefit the very poor—to create a concrete product that improves their quality of life. "When you don't have a lot of money, you need a competitive edge," he says. "That's what we get with the computer."

For more information on the Apple Community Affairs program, contact Fred Silverman at Apple Computer, 20525 Mariani Avenue, Cupertino, CA 95014. Tax deductible donations can be made to the Emergency Housing Consortium of Santa Clara County, P.O. Box 2346, San Jose, CA 95109.

The World Inside the Computer

Fred D'Ignazio, Associate Editor

Pieces Of Our Past—The Computer Puzzle

Last month I told the story of my "Phantom Programmer," Hunter Baker, a high school student I recruited to organize my attic and computer room. The story ended with me nervously charging into the computer room waving a machete in the middle of the night after mistaking Hunter and his friend, Amy Powell, for burglars. Actually, they were working on a school project for National History Day: a history trivia game for the IBM computer.

Hunter and Amy entered their program in the regional History Day competition and won first place in the Senior Media Presentation category. And no wonder! They had spent dozens of hours collecting hundreds of history questions and typing them into the computer, where they were stored as six random data files representing six question categories: Presidents, Places, Historical Figures, U.S. Constitution, Wars and Battles, and Trivial Trivia. And while Hunter was writing a program that managed all the questions, Amy was using Mouse Systems' *PC Paint* to create seven beautiful picture screens—a title screen and a screen for each category.

Confident after their victory in the regional competition, Hunter and Amy took their history trivia game—called "Pieces of Our Past"—to the state competition at Lynchburg College, in Lynchburg, Virginia.

But the two young people received a rude shock. The state judges said their program was not a media project at all, and gave them low grades in almost every category. One judge wrote that the project "shows no work." Another judge gave Hunter and Amy 0 out of a possible 15 points for "Quality of the Medium." Several judges gave the program low grades for historical accuracy, yet every one of Hunter and Amy's questions and answers came from reliable sources such as textbooks and encyclopedias.

Worst of all, the judges refused to interact with the program. During the judging they sat in their chairs, far away from the computer screen and keyboard, and declined to come any closer—even when Hunter and Amy invited them. Later, one judge wrote on the judging sheet: "Not effective media presentation. I couldn't see the screen."

Hunter and Amy returned from the History Day competition disappointed and bewildered. They had put an enormous amount of work into their project. They had come up with an innovative approach to learning history facts, and they had demonstrated a mastery of their medium. Hunter's program made use of random data files, elaborate graphics (created, pixel by pixel, by Amy), and music. By storing the pictures in the IBM's video memory and the music in another memory buffer, Hunter's program was able to display a picture, play music, and build the question arrays all at the same time.

But their program lost. Why? Do history teachers fear the computer? Don't they recognize the computer as a valid educational medium, like slides, filmstrips, videotapes, or 8mm movies?

A New Media Is The Message

I think history teachers are no more afraid of computers than anyone else, but like almost everyone else, few of them see the computer as "media." And since the computer is a new form of media, with its own special needs and limitations, no one was quite prepared for Hunter and Amy's project, which was so different that it bewildered the judges, confounded the rules, and didn't fit into any of the project categories.

I imagine the judges had no idea how much work and original thinking went into "Pieces of Our Past." All this work was stored, invisibly and electronically, inside the computer as hundreds of lines of code, computer records, and screen maps.

And the judges were not prepared to *interact* with a media project. In the past, they had sat back, passively, and been informed, educated, or entertained. Now they were being asked to sit down in front of an unfamiliar keyboard, read the display screen, and *answer questions without any preparation.* What a fright! They might have pressed the wrong key and looked foolish. Or worse, they might have answered one of the history questions incorrectly in front of their colleagues (all fellow history teachers, instructors, and professors).

Everyone—Hunter, Amy, the judges—was burned by this experience. In the future, I doubt if Hunter and Amy will be quite as innovative or work quite as hard or independently on a project like this. And I know the judges feel bad, too. They saw merit in Hunter and Amy's project, but they didn't understand it, and they didn't know how to compare it with the other projects, or rate it according to the rules of the competition.

This was just a small incident, but I fear similar ones are occurring all over the U.S. when young people try to incorporate computers into projects that baffle and confuse their elders. Bright, self-motivated young people can come up with all sorts of ingenious uses for computers that many of us older folks have never dreamed of. But I'm worried that we may not be ready for them when they do.

What do you think? Have you had any similar experiences? Please write me care of COMPUTE!. ©

Arlan R. Levitan

In Pursuit Of Lower Phone Bills

For months I'd been bugging a friend about his reluctance to add a modem to his home computer. Then, while visiting a computer store one day, I was busy inspecting a surge protector designed to protect the surge protector I already own when something caught my eye. It was a modem that would work with John's Commodore 128, complete with software for only $39.95. I walked over to him and waved the modem package slowly back and forth before his eyes for maximum hypnotic effect.

"That sure is a good price for a modem," John admitted. "But wouldn't I end up paying at least that much every month in phone bills and information service charges?"

He had me there. I recalled my own introduction to telecomputing and the trauma induced by various bills totaling over a hundred dollars for an uncontrolled spree of telecomputing.

Node-To-Node Networking

Sound like a familiar complaint? Now there's a solution. How would you like unlimited access to hundreds of computer bulletin boards all over the country for a flat fee of $25 a month?

If you live in the metropolitan areas of Atlanta, Boston, Chicago, Dallas, Denver, Detroit, Houston, Los Angeles, New York, Philadelphia, San Francisco, or Washington D.C., such a service is available. It's called PC Pursuit, and it's marketed by GTE Telenet, one of the giants of the packet-switching business.

What's packet switching? It's a system used daily by hundreds of businesses that have centralized computer systems linked to branch offices in different cities. Rather than leasing expensive data lines to link each branch office to the central computer, they call a local node or connection point that hooks into a special long-distance network. The call is routed through the packet-switching network to another node that is local to the firm's central computer.

The vast bulk of data traffic on packet-switching networks occurs during the business day. Although the networks are also used by people accessing commercial information services during off-hours, there's still a lot of extra capacity. PC Pursuit is an attempt by GTE Telenet to make productive use of those idle resources. Here's how it works:

Registered users call a special access number via their modem and computer. When the connection is established, the PC Pursuit system asks for their phone number, the city they wish to call, and the phone number they're trying to reach. Next, the system disconnects, temporarily freeing the phone line. Within 20 seconds, the system calls back. The user re-establishes the modem link, and then the system rings the number of the BBS. If a computer answers, PC Pursuit reports that the connection is complete. It's as if the user had directly called the remote computer himself. While the process may sound somewhat complicated, it actually requires only three pieces of information from the caller, and takes only about a minute.

A Few Limitations

For the most part, PC Pursuit works well. I spent my first evening calling BBSs in Los Angeles, Houston, and Dallas that I had been limiting my use of to keep my long distance bill from resembling the national debt. The quality of the connections is quite good, and the few noisy lines I've encountered can probably be blamed on a poor local connection at either end of the telecomputing link.

The cost savings can be significant, especially if you're a heavy user. I figured that the cost of making all of my PC Pursuit calls for the first month alone would have been well over $200. Prospective users must consider whether the one-time $25 registration and monthly $25 usage fees will actually save money.

PC Pursuit does have its limitations. You can use the service only from a single, registered phone number, typically the home number your computer is connected to. The service is offered only during Telenet's off-hours, from 6 p.m. to 7 a.m. Monday through Friday, and on weekends from 6 p.m. Friday to 7 a.m. Monday. Each PC Pursuit connection can last only 60 minutes at a time (you can, however, make multiple calls to the same number).

Since it takes longer to make a call and is a somewhat more complicated process, it is more difficult to use the redialing routines within terminal programs to bust the busy signals of the most popular bulletin boards. And although PC Pursuit tells you if the requested number is busy, the actual call cannot be monitored via a speaker on direct-connect modems. That makes it impossible to hear recorded messages informing you that a line has been disconnected or its number changed.

Also, packet switching reduces the speed of the telecomputing link. I clocked my average PC Pursuit connection at just a little less than 1000 bps (bits per second), even though I was using a 1200 bps modem. Things slow down even more when transferring files with protocols such as XMODEM—I clocked the speed at 720–950 bps.

If you're interested in more information, call GTE's bulletin board at 1-800-835-3001. Or, if you prefer to talk to a human, call 1-800-368-4215.

Programming the TI

C. Regena

Christmas Graphics

Try this special Christmas program. (It can only be typed in on a TI-99/4A console.)

We Three Kings

```
100 REM  WE THREE KINGS
110 CALL CLEAR
120 T=375
130 CALL SOUND(2*T,494,2,39
    2,6,165,8)
140 CALL CHAR(152,"00010103
    03FF7F1F")
150 CALL CHAR(153,"07070F1F
    1C30304")
160 CALL SCREEN(2)
170 CALL CHAR(154,"8080C0C0
    E0FFFEF8")
180 CALL CHAR(155,"E0E0F078
    180C0C02")
190 CALL CHAR(33,"000000051
    5DF7F78")
200 CALL SOUND(T,440,2,370,
    6,165,9)
210 CALL CHAR(34,"020666606
    06773F1")
220 CALL CHAR(35,"000000004
    0C0E0F")
230 CALL SOUND(2*T,392,2,33
    0,6,165,8)
240 CALL CHAR(36,"E0C0D2929
    3333373")
250 CALL CHAR(37,"010701000
    1010101")
260 CALL CHAR(38,"50F0C0103
    080B0B")
270 CALL CHAR(39,"010001030
    70704")
280 CALL CHAR(40,"F00080889
    8C86")
290 PRINT TAB(10);"!"
300 CALL SOUND(T,330,2,196,
    6,165,9)
310 CALL CHAR(41,"7B7B7BFBF
    BFBFBFB")
320 CALL CHAR(42,"808080E0F
    CF8F8F")
330 CALL SOUND(T,370,2,311,
    6,123,9)
340 CALL CHAR(43,"030307060
    4010307")
350 CALL CHAR(44,"90000020F
    0F0F0F")
360 CALL SOUND(T,392,2,311,
    7,123,9)
370 CALL CHAR(45,"030707070
    70F0F0F")
380 CALL CHAR(46,"80C0C0C0E
    0E0E0F")
390 CALL SOUND(T,370,2,311,
    6,123,8)
400 CALL CHAR(47,"000008081
    8183839")
410 CALL CHAR(48,"F9F9F8F8F
    9FDFDFC")
420 CALL SOUND(2*T,330,2,19
    6,6,165,8)
430 PRINT TAB(10);CHR$(34)
440 PRINT TAB(8);"# $  %&"
450 CALL CHAR(49,"F6E6E0400
    89CFEFE")
460 CALL CHAR(50,"070707000
    E0F01")
470 CALL CHAR(51,"F0F0F8FC7
    E7C3911")
480 CALL CHAR(52,"000000000
    01C3F8F")
490 PRINT TAB(7);"'( )* +,"
500 CALL CHAR(53,"000000000
    030F8FC")
510 CALL CHAR(54,"000001030
    707071B")
520 CALL CHAR(55,"1F9FCFC7E
    7F3F9FC")
530 CALL SOUND(2*T,494,2,39
    2,6,165,8)
540 CALL CHAR(56,"F0F0E6F0E
    0C3CF1F")
550 CALL CHAR(57,"39190949E
    1F1F9F9")
560 CALL CHAR(58,"FDFDFDFDF
    DFDFDFD")
570 CALL CHAR(59,"7C0CE0FCF
    FFFFFFF")
580 CALL CHAR(60,"000103030
    383C7C7")
590 CALL SOUND(T,440,2,370,
    6,165,9)
600 CALL CHAR(61,"E7F3F8FCF
    EFFFFFF")
610 CALL CHAR(62,"C0E6FE7E3
    F1F8FC7")
620 CALL SOUND(2*T,392,2,33
    0,6,165,8)
630 CALL CHAR(63,"070F00383
    F000F3F")
640 CALL CHAR(64,"FFFFFF3F8
    000FEFE")
650 CALL CHAR(65,"80C0C0D01
    02C4C1C")
660 PRINT TAB(7);"-./01 234
    5"
670 CALL CHAR(66,"030307030
    91C1F3F")
680 CALL CHAR(67,"FCFEFFFFF
    F1FC0CE")
690 CALL SOUND(T,330,2,196,
    6,165,9)
700 CALL CHAR(68,"FCFEFFFFF
    F1FC0CE")
710 CALL CHAR(69,"000001010
    1091939")
720 CALL SOUND(T,370,2,311,
    6,123,8)
730 CALL CHAR(70,"7BF9F9F8F
    8FCFEFF")
740 CALL CHAR(71,"F8F0E0C30
    70F1F3F")
750 CALL SOUND(T,392,2,311,
    5,123,8)
760 CALL CHAR(72,"9F9F9F9F9
    F9F9F9F")
770 CALL CHAR(73,"F9F9F9F1F
    1F1F3F3")
780 CALL SOUND(T,370,2,311,
    6,123,9)
790 CALL CHAR(74,"FFFFFFFFF
    FFF")
800 CALL CHAR(75,"E7E7E7EFE
    F870723")
810 CALL SOUND(2*T,330,2,19
    6,6,165,8)
820 CALL CHAR(76,"FFFFFFFFF
    FFFFFFF")
830 CALL CHAR(77,"E1F0F8FCF
    CFEFEFE")
840 CALL CHAR(78,"3F3F3F3F3
    F3F3F3F")
850 PRINT TAB(6);"6789:;<=>
    ?@ABCD"
860 CALL CHAR(79,"FCF8F0E0C
    0C1C1C3")
870 CALL CHAR(80,"180080808
    ")
880 CALL CHAR(81,"3F7F00007
    FFFFFFF")
890 CALL CHAR(82,"FF930040C
    0818101")
900 PRINT TAB(5);"EFGHI:JKL
    MNOPQRS"
910 CALL SOUND(2*T,392,4,33
    0,8,165,10)
920 CALL CHAR(83,"383878108
    08")
930 CALL CHAR(84,"79F9F9F9F
    9710101")
940 CALL CHAR(85,"FFFFFFFEF
    EFCF8F")
950 PRINT TAB(5);"TUVHW:LXL
    YVZ[\]"
960 CALL CHAR(86,"3F7F7F7FF
    FFFFFFF")
970 CALL CHAR(87,"F3F3F3F3F
    3F3F3F3")
980 CALL SOUND(T,392,5,330,
    9,165,11)
990 CALL CHAR(88,"636171717
    078787C")
1000 CALL CHAR(89,"F7F7F7FC
     F8F8F1E1")
1010 CALL SOUND(2*T,440,4,3
     70,8,147,10)
1020 CALL CHAR(90,"C2828688
     8080802")
1030 CALL CHAR(91,"01010103
     03070F1F")
1040 CALL CHAR(92,"FEFCFCF8
     F8F8F8F")
1050 PRINT TAB(4);"^_`LHWaL
     bcdVeHf"
1060 CALL CHAR(93,"03030306
     040C18E")
1070 CALL CHAR(94,"07070707
     07070703")
1080 CALL SOUND(T,440,5,370
     ,9,147,11)
1090 CALL CHAR(95,"F9F9F9F9
     F9F9F9F9")
1100 CALL CHAR(96,"F8F0F0F0
     E0E4E4C4")
1110 CALL SOUND(2*T,494,3,3
     92,7,196,9)
1120 CALL CHAR(97,"FCFCFCFC
     FCFCFCFC")
1130 CALL CHAR(98,"60000003
     0F3F3F3F")
1140 CALL CHAR(99,"3F3F1F00
     80E1FBFA")
1150 PRINT TAB(4);"g_hLHijL
     Nklmno"
1160 CALL CHAR(100,"C3870F3
     F7F7F7F75")
```

```
1170 CALL CHAR(101,"337373F
     3F3F3F3F3")
1180 CALL SOUND(T,494,4,392
     ,8,196,10)
1190 CALL CHAR(102,"F0F0E0E
     0C0808")
1200 CALL CHAR(103,"2323636
     3C3C38383")
1210 CALL SOUND(T,587,2,392
     ,6,247,8)
1220 CALL CHAR(104,"C4C68E0
     E0E0E0E9E")
1230 CALL CHAR(105,"E7E7E7E
     7E0E0E0E5")
1240 CALL SOUND(T,523,2,370
     ,6,220,8)
1250 CALL CHAR(106,"FCFCFCF
     C000000F4")
1260 CALL CHAR(107,"F2F2F2F
     2F2E60404")
1270 CALL SOUND(T,494,2,392
     ,6,196,8)
1280 CALL CHAR(108,"7F7FFCF
     CF0C1031F")
1290 CALL CHAR(109,"8F0F1F1
     F1F9DBC38")
1300 CALL SOUND(T,440,3,370
     ,7,220,9)
1310 CALL CHAR(110,"F3F3F3F
     3F3F0F0F")
1320 CALL CHAR(111,"9E9C988
     08")
1330 CALL SOUND(T,494,3,392
     ,7,220,9)
1340 CALL CHAR(112,"070F1F3
     F7F7F7E7C")
1350 CALL CHAR(113,"F8F1E3C
     78E18")
1360 CALL SOUND(T,440,3,370
     ,7,220,9)
1370 CALL CHAR(114,"9C99830
     70F1F3F7F")
1380 CALL CHAR(115,"1F1F1F0
     00000707")
1390 CALL SOUND(2*T,392,3,3
     30,7,247,9)
1400 CALL CHAR(116,"E5E5E50
     000387C78")
1410 CALL CHAR(117,"F4F4F40
     0000C1C7C")
1420 CALL CHAR(118,"FFFFFF0
     00000E0E")
1430 PRINT TAB(4);"pqrUstuv
     wxyz{"
1440 CALL CHAR(119,"3C3C3C3
     C3E3E3E3E")
1450 CALL CHAR(120,"C07F3F0
     3")
1460 CALL SOUND(T,370,3,311
     ,7,123,9)
1470 CALL CHAR(121,"FFFEFEF
     000010101")
1480 CALL CHAR(122,"3830707
     0F0E0E0E")
1490 CALL SOUND(3*T,330,3,1
     96,7,165,9)
1500 CALL CHAR(156,"FF")
1510 CALL CHAR(157,"0102040
     81020408")
1520 CALL CHAR(158,"8080808
     08080808")
1530 PRINT TAB(4);"| } ~"
1540 PRINT TAB(4);"|"
1550 CALL HCHAR(2,27,152)
1560 CALL HCHAR(3,27,153)
1570 CALL HCHAR(2,28,154)
1580 CALL HCHAR(3,28,155)
1590 CALL CHAR(123,"F87C7C3
     C3E1F1F0E")
1600 CALL CHAR(124,"3030303
     03030303")
1610 CALL CHAR(125,"FFFCFCF
     8F0F0E0E")
1620 CALL SOUND(2*T,370,2,2
     94,6,220,8)
1630 CALL CHAR(126,"707070F
     0F0F0F0F8")
1640 CALL CHAR(127,"F8F1F1E
     1E1E0E0E")
1650 CALL CHAR(128,"FCF8F0F
     0F0F0F0F")
1660 CALL HCHAR(22,11,127)
1670 CALL HCHAR(22,12,128)
1680 CALL CHAR(129,"C0C0C0C
     0C0C0C0C")
1690 CALL CHAR(130,"1F1F0F0
     F07070703")
1700 CALL SOUND(T,440,2,262
     ,8)
1710 CALL CHAR(131,"0103030
     303030303")
1720 CALL CHAR(132,"C0C0C08
     0808")
1730 CALL SOUND(2*T,392,0,2
     94,6,196,8)
1740 CALL COLOR(16,16,1)
1750 CALL CHAR(133,"0E0C1C1
     C383060E")
1760 CALL CHAR(134,"0000010
     103030307")
1770 CALL CHAR(135,"E0C0C08
     08")
1780 CALL HCHAR(22,13,129)
1790 CALL CHAR(136,"78783C1
     C0C030606")
1800 CALL CHAR(137,"6060404
     0404")
1810 CALL SOUND(T,392,1,294
     ,7,196,9)
1820 CALL HCHAR(2,23,156,3)
1830 CALL VCHAR(4,28,158,5)
1840 CALL HCHAR(4,26,157)
1850 CALL HCHAR(5,25,157)
1860 CALL SOUND(2*T,392,0,2
     94,6,196,8)
1870 CALL CHAR(138,"0303030
     10101")
1880 CALL CHAR(139,"8080808
     08080C0C")
1890 CALL CHAR(140,"0606030
     C0C1C1C3C")
1900 CALL HCHAR(22,14,130)
1910 CALL HCHAR(22,16,131)
1920 CALL CHAR(141,"0103070
     F0F0E0E0C")
1930 CALL CHAR(142,"C0C08")
1940 CALL SOUND(T,294,1,247
     ,6,196,8)
1950 CALL CHAR(143,"30383C3
     C0E")
1960 CALL CHAR(144,"0603030
     F0F070707")
1970 CALL SOUND(2*T,392,1,2
     47,7,165,9)
1980 CALL CHAR(145,"0000000
     00080C0E")
1990 CALL CHAR(146,"000080C
     0E0E")
2000 CALL CHAR(147,"E0F0703
     8")
2010 CALL HCHAR(22,17,132)
2020 CALL HCHAR(22,18,133)
2030 CALL CHAR(148,"C0C0C0C
     0F0783C1C")
2040 CALL CHAR(149,"3C1C0E0
     707")
2050 CALL SOUND(T,330,1,262
     ,6,131,8)
2060 CALL CHAR(150,"1C1C180
     8")
2070 CALL HCHAR(23,7,134)
2080 CALL HCHAR(23,8,135)
2090 CALL HCHAR(23,10,136)
2100 CALL HCHAR(23,11,137)
2110 CALL SOUND(2*T,392,1,2
     94,6,165,8)
2120 CALL HCHAR(23,12,124)
2130 CALL HCHAR(23,13,129)
2140 CALL HCHAR(23,14,138)
2150 CALL HCHAR(23,15,139)
2160 CALL HCHAR(23,16,140)
2170 CALL HCHAR(23,17,141)
2180 CALL HCHAR(23,18,142)
2190 CALL HCHAR(24,6,143)
2200 CALL HCHAR(24,7,144)
2210 CALL HCHAR(24,8,145)
2220 CALL HCHAR(24,10,138)
2230 CALL HCHAR(24,11,146)
2240 CALL HCHAR(24,12,143)
2250 CALL HCHAR(24,13,147)
2260 CALL HCHAR(24,15,148)
2270 CALL HCHAR(24,16,149)
2280 CALL HCHAR(24,17,150)
2290 CALL SOUND(T/2,9999,30
     )
2300 CALL SOUND(2*T,392,2,2
     94,7,165,9)
2310 CALL SOUND(T,392,4,294
     ,9,165,11)
2320 CALL SOUND(2*T,392,2,2
     94,7,165,9)
2330 CALL SOUND(T,294,2,247
     ,6,196,8)
2340 CALL SOUND(2*T,392,2,2
     47,6,165,8)
2350 CALL SOUND(T,330,2,262
     ,6,131,8)
2360 CALL SOUND(2*T,392,2,2
     94,6,196,8)
2370 CALL SOUND(T,9999,30)
2380 CALL SOUND(2*T,392,3,2
     47,7,165,9)
2390 FOR C=1 TO 15
2400 CALL COLOR(C,16,1)
2410 NEXT C
2420 CALL SOUND(T,392,4,247
     ,8,165,10)
2430 CALL SOUND(2*T,440,2,3
     70,6,147,8)
2440 CALL SOUND(T,494,2,370
     ,7,147,9)
2450 CALL SOUND(2*T,523,1,3
     92,5,131,8)
2460 CALL SOUND(T,494,1,392
     ,5,196,7)
2470 CALL SOUND(2*T,440,1,3
     92,6,147,8)
2480 CALL SOUND(T,494,2,370
     ,6,147,9)
2490 CALL SOUND(2*T,392,2,2
     47,6,196,8)
2500 CALL SOUND(T,392,3,294
     ,7,196,9)
2510 CALL SOUND(2*T,392,2,2
     47,6,196,8)
2520 CALL SOUND(T,294,2,196
     ,7,123,8)
2530 CALL SOUND(2*T,392,2,3
     30,6,131,8)
2540 CALL SOUND(T,330,2,262
     ,6,131,9)
2550 CALL SOUND(3*T,392,2,2
     94,6,247,9)
2560 CALL COLOR(16,12,1)
2570 CALL COLOR(16,16,1)
2580 CALL KEY(0,K,S)
2590 IF S<1 THEN 2560
2600 CALL CLEAR
2610 PRINT "HAVE A HAPPY HO
     LIDAY SEASON!":::::
2620 END
```

©

INSIGHT: Atari

Bill Wilkinson

The Hidden Power Of Atari BASIC

This month we're going to look at good old Atari BASIC. For once, though, I'm not going to talk about its problems. Instead, I'm going to tell you about a few of its many virtues. If you've been reading my column since it first appeared in the September 1981 issue of COMPUTE!, then some of this may seem repetitive; but it's time to introduce newcomers to some of this material.

Unfortunately, I am beginning to see more and more poorly written Atari BASIC programs. Generally, what happens is that someone not too well-versed in Atari BASIC attempts to translate a program from another computer's BASIC and botches the job. The last straw, for me, was a recently released book which is full of CAI (Computer Assisted Instruction) programs. All the programs do much the same thing, and all the programs are...well, just a lot of work for so little value.

Now, I'm all for using a computer for drill and practice, even though most of the educational programs which do this are dull and unimaginative (and often overpriced). But even the plainest of CAI programs can at least free up a teacher or parent for 20 or 30 minutes while a student is checking his or her knowledge. And if all you want your CAI program to do is ask questions and wait for a response, then all such programs can be essentially the same. So that's what I'm going to give you this month: a "formula" program for drill and practice.

I also mentioned that we would look at some of the virtues of Atari BASIC, so let's do that first. Among microcomputer BASICs, Atari BASIC is nearly unique in its flexibility in the use of GOTO, GOSUB, and RESTORE. Specifically, each of these statements accept any numeric expression as the line number they reference. Combined with Atari BASIC's variable-name flexibility, this means you can code such oddities as:

```
GOSUB CALCULATEGROSSPAY
```

and

```
RESTORE 20000+10*CURRENTROOM
```

Most Atari BASIC books refer to these capabilities briefly, if at all. But there is some real hidden power here, as we are about to find out. Rather than belabor the point, let's take a look at the accompanying listing and analyze it a step at a time.

Using Variables As Labels

Line 1010 is fairly obvious, so let's start with lines 1060 to 1080. The variables being set here are actually going to be used as labels, the targets of GOTO and GOSUB statements. The only thing you have to be careful of with this method is renumbering—some renumbering utilities warn you when they encounter a variable being used as a label, and some don't.

Now, after setting the DATA pointer in line 1090, we get a line of DATA, assigning the first byte to the variable TYPE$. The action we take next depends on what type of line we got. We use an exclamation point to indicate a screen clear is needed, a colon for an extra blank line, and a period to flag an ordinary text line. In any of these cases, we print the rest of the line and get another one. If the type is an asterisk, the program halts. If the type is a question mark, then it's time for the student to answer.

At this time, let's look at the DATA in lines 10000–10003. The first line begins with an exclamation point, so the screen is cleared and it is printed. Then the colon asks for a blank line before the next line is displayed. Finally, the question mark tells the program to ask for a response. But what's the rest of that funny stuff: 1=,Y,0,10010?

Back at lines 1200–1260, you can see that the digit (a 1 in line 10002) tells the number of possible answers to the question, and the next character indicates the type of answer which is acceptable (the equal sign here asks for an exact match). The program then prompts the user for an answer (the #16 suppresses the INPUT prompt) and prepares to test its validity. The loop in 1310–1360 checks each valid answer against the user's response.

If an exact answer is needed, even the length of the answer counts. (Example: In line 10002, we have allowed only a single exact answer, the letter Y.) Another flag indicates whether the valid answer can be found somewhere in the user's response line. Line 10012, for example, passes any answer containing the word GRANT (such as MIGRANT WORKERS), so some care is needed in using this type. Finally, if none of the valid answers matches the user's response, the program falls through to lines 1400–1420.

So far, all this has been very straightforward, and it would work on almost any BASIC. Now comes the tricky stuff. Look at line 1320, where we READ numbers into the variables GOSUBLINE and DATALINE. What we're doing is establishing an action to take and a new set of DATA to access if the user's response matches a valid answer. Similarly, in line 1420 we read values to be used if no valid answer is given. Finally, the "magic" of this program is revealed in lines 1510 and 1520.

If we READ a number other than zero for GOSUBLINE, the program actually GOSUBs to that number. And, in any case, we change the DATA pointer to the new DATALINE. If you can't predict what happens if you answer DUCK to the second question (because of the DATA in lines 10012–10014), please type this program and try it out.

Now, the real beauty of this program is that it works with almost any kind of question and answer session. It allows for multiple choice questions (use a format like ?3=,A,0,100,B,0,100,C,0,200), true/false, and so on. It provides for special help if needed (via the GOSUBLINEs). And, last but by no means least, it is expandable. You could add many different statement types, question types, or whatever quite easily. And it's all made possible thanks to Atari BASIC.

Multiple Choice Quiz

```
DL 1000 REM === INITIALIZATION ===
MF 1010 DIM LINE$(120), ANS$(20), TYPE$(1)
ID 1060 INEXACT=2000:EXACT=2100
NI 1070 MAINLOOP=1100:QUESTION=1200
MC 1080 MATCHED=1500
CC 1090 RESTORE 10000:REM where we start
PL 1100 REM === THE MASTER LOOP ===
FE 1110 READ LINE$:TYPE$=LINE$
FJ 1120 IF TYPE$="?" THEN GOTO QUESTION
DN 1130 IF TYPE$="!" THEN PRINT CHR$(125);
DC 1140 IF TYPE$=":" THEN PRINT
GN 1150 IF TYPE$="*" THEN END
CE 1160 PRINT LINE$(2)
GB 1170 GOTO MAINLOOP
ON 1200 REM === PROCESS A QUESTION ===
JI 1210 QCNT=VAL(LINE$(2,2))
DI 1220 TYPE$=LINE$(3)
PL 1230 POSITION 2,20
KG 1240 PRINT CHR$(156);CHR$(156);
DC 1250 PRINT "Your Answer ?";
FL 1260 INPUT #16,LINE$
OG 1300 REM === PROCESS THE ANSWER ===
NF 1310 FOR ANS=1 TO QCNT
CK 1320 READ ANS$,GOSUBLINE,DATALINE
BL 1330 IF TYPE$="#" THEN GOSUB INEXACT
JP 1340 IF TYPE$="=" THEN GOSUB EXACT
CD 1350 IF MATCH THEN GOTO MATCHED
OL 1360 NEXT ANS
EI 1400 REM === ANSWER DOESN'T MATCH ===
JA 1410 REM (read error conditions and fall thru)
PJ 1420 READ GOSUBLINE,DATALINE
ND 1500 REM === ANSWER MATCHED ===
FF 1510 IF GOSUBLINE THEN GOSUB GOSUBLINE
CO 1520 RESTORE DATALINE
GB 1530 GOTO MAINLOOP
LD 2000 REM === INEXACT MATCH ROUTINE ===
GK 2010 MATCH=0:ALEN=LEN(ANS$)
GB 2020 SIZE=LEN(LINE$)-ALEN+1
AL 2030 IF SIZE<1 THEN RETURN
BH 2040 FOR CHAR=1 TO SIZE
GL 2050 IF LINE$(CHAR,CHAR+ALEN-1)=ANS$ THEN MATCH=1:RETURN
CF 2060 NEXT CHAR
KJ 2070 RETURN
BN 2100 REM === EXACT MATCH ROUTINE ===
ED 2110 MATCH=(ANS$=LINE$)
KF 2120 RETURN
KK 10000 DATA !Ready to try out this program?
KP 10001 DATA :  (answer Y or N)
LJ 10002 DATA ?1=,Y,0,10010
FL 10003 DATA 0,10000
PJ 10010 DATA !A tribute to Groucho Marx:
CK 10011 DATA :Who is buried in Grant's tomb?
MH 10012 DATA ?2#,GRANT,0,10040
HF 10013 DATA DUCK,10020,10030
CJ 10014 DATA 10050,10060
OH 10020 REM special sound routine
KI 10021 FOR FREQ=120 TO 20 STEP -10
CG 10022 FOR VOLUME=15 TO 0 STEP -0.5
JK 10023 SOUND 0,FREQ,10,VOLUME
LF 10024 NEXT VOLUME:NEXT FREQ
NI 10025 RETURN
LB 10030 DATA !You said the secret word!
LH 10031 DATA :You win $100.
DY 10032 DATA *
KL 10040 DATA !Great!  You get the consolation
PE 10041 DATA .prize of $50.
DL 10042 DATA *
LE 10050 REM raspberry
FK 10051 FOR VOLUME=15 TO 0 STEP -0.25
PK 10052 SOUND 0,4,80,VOLUME:NEXT VOLUME
NJ 10053 RETURN
KM 10060 DATA !Sorry.  You lost.
DM 10061 DATA *
```

©

Donald B. Trivette

Diary Of A Home Application

Few of us appreciate how much time and effort goes into successful software. This month we're going to take an inside look at one of today's best-selling home programs for IBM computers—*Andrew Tobias' Managing Your Money*.

September 1982. Micro Education Corporation of America (MECA) looks for ways to enter the lucrative personal computer market. A market survey and analysis shows that home users are interested in financial software. MECA decides to develop a tutorial-type financial program for the Atari and to employ a "big name" to promote the product. Louis Rukeyser of *Wall Street Week* is contacted; he graciously declines. Another proposal goes to syndicated columnist Sylvia Porter; no reply is received. MECA looks for another name.

December 1982. Andrew Tobias, author of the best-selling book *The Only Investment Guide You'll Ever Need*, agrees to provide his name and guidance—for a percentage. (He too almost missed out when his agent, unaware of the potential profits, neglected to return MECA's calls. Eventually MECA contacted Tobias directly.)

Tobias and Jerry Rubin, the master programmer/designer and president of MECA, meet to discuss possibilities. They decide to develop the product for the IBM PC, which seems to be gaining momentum in the market. The program begins to evolve from the initial concept of a tutorial with cartoon characters and balloons of text to a more serious program that can be used to plan and record financial transactions.

Steve Wagar, a recent Yale computer science graduate, begins writing a special computer language called SEESAW (System Elegantly Enmeshing Screens And Worksheets) in which *MYM* will be programmed. Spencer Martin, taking a year off between high school and college to swim in the Olympic trials, joins the programming team. So does Jim Russell, a student at the Massachusetts Institute of Technology.

Summer 1983. Rubin gleefully demonstrates a screen to Tobias, who, knowing nothing about computers and programming, fails to appreciate its significance. "My God," he thinks, "six months and all we've got is one or two screens where I can type my name and a few numbers."

Fall 1983. The project is far enough along so that Tobias, using a *WordStar* interface to SEESAW, can write the help screens and compose the program's text—a job that eventually takes six months.

January 1, 1984. The goal for the release of *MYM* slips by while initial product testing begins in Westport, Connecticut. A group of 20 people—some experienced computer users, some not—are given copies to take home. Later they are invited to headquarters for a debriefing while Rubin, Tobias, and others watch tensely through a one-way mirror, suppressing the urge to pound on the wall and yell, "No, you idiot, not that key, the other key!"

MYM takes shape as an integrated financial program with nine sections or chapters. Chapter 1 is for new users who know nothing about computers; Chapter 2 is a reminder pad; Chapter 3 is a budget and checkbook; Chapter 4 is an income tax estimator; Chapter 5 is for insurance planning; Chapter 6 is a calculator; Chapter 7 is a portfolio manager; Chapter 8 is a net worth summary combining data from the other chapters; and Chapter 9 is a comprehensive index.

March 19, 1984. Tobias appears on the *Today* show to introduce *MYM* and the first 300 copies are shipped to dealers. A bug is uncovered and MECA replaces all 300 copies at its own expense. Another bug is uncovered and 500 copies are replaced. A WATS line with 12 customer support people is set up to answer questions and help users. MECA continues to improve *MYM* and to provide free updates to registered owners.

Summer 1984–Spring 1985. Tobias travels more than 60,000 miles promoting *MYM* in software stores, at trade associations, and on radio and TV talk shows. The program gets good reviews and IBM markets a cartridge version for the PCjr.

Summer 1985. Starting with a wish list compiled from customer suggestions, MECA begins work on version 2.0—a major update. Rubin, Martin, Russell, and four new programmers add 75 enhancements. Rubin and Russell spend weeks on an option to make the fiscal year different from the calendar year; Tobias writes an expanded 15,000-word manual. MECA engages a software-testing company which generates more than 120 Trouble Reports that ultimately have one of three resolutions: Already Fixed; To Be Fixed; and Not a Bug After All. The testing costs more than $50,000.

October 1985. *Andrew Tobias' Managing Your Money* version 2.0 ships to dealers. Current owners who signed up for the newsletter and warranty plan ($40/year) receive the update free. Other registered owners can purchase the update for $50.

(MYM 2.0 lists for $199.95 and requires an IBM PC or compatible with two disk drives and at least 192K of RAM; IBM markets version 1.0 in cartridge form for the Enhanced Model PCjr at the same price. MECA, 285 Riverside Avenue, Westport, CT 06880.)

What the world really needs is a 69 cent Double Sided, Double Density Diskette with a LIFETIME WARRANTY!

And DISK WORLD! has it.

Introducing Super Star Diskettes: the high quality diskette with the lowest price and the best LIFETIME WARRANTY!

In the course of selling more than a million diskettes every month, we've learned something: higher prices don't necessarily mean higher quality.

In fact, we've found that a good diskette manufacturer simply manufactures a good diskette...no matter what they charge for it. (By way of example, consider that none of the brands that we carry has a return rate of greater than 1/1,000th of 1 percent!)

In other words, when people buy a more expensive diskette, they aren't necessarily buying higher quality.

The extra money might be going toward flashier advertising, snazzier packaging or simply higher profits.

But the extra money in a higher price isn't buying better quality.

All of the good manufacturers put out a good diskette. Period.

How to cut diskette prices ...without cutting quality.

Now this discovery posed a dilemma: how to cut the price of diskettes without lowering the quality.

There are about 85 companies claiming to be "diskette" manufacturers.

Trouble is, most of them aren't manufacturers.

Rather they are fabricators or marketers, taking other company's components, possibly doing one or more steps of the processing themselves and pasting their labels on the finished product.

The new Eastman Kodak diskettes, for example, are one of these. So are IBM 5¼" diskettes. Same for DYSAN, Polaroid and many, many other familiar diskette brand names. Each of these diskettes is manufactured in whole or in part by another company!

So, we decided to act just like the big guys. That's how we would cut diskette prices...without lowering the quality.

We would go out and find smaller companies to manufacture a diskette to our specifications...specifications which are higher than most...and simply create our own "name brand" diskette.

Name brand diskettes that offered high quality at low prices.

Super Star diskettes are sold in multiples of 50 only. Diskettes are shipped with white Tyvec sleeves, reinforced hubs, user ID labels and write-protect tabs.

Boy, did we get lucky. Our Super Star Diskettes are the same ones you've been using for years...without knowing it.

In our search for the low priced, high quality diskette of our dreams, we found something even more interesting.

We found that there are several manufacturers who don't give a hoot about the consumer market for their diskettes. They don't spend millions of dollars in advertising trying to get you, the computer user, to use their diskettes.

Instead, they concentrate their efforts on turning out the highest quality diskettes they can.. because they sell them to the software publishers, computer manufacturers and other folks who (in turn) put their name on them...and sell them for much higher prices to you!

After all, when a software publisher or computer manufacturer or diskette marketer puts their name on a diskette, they want it to work time after time, everytime. (Especially software publishers who have the nasty habit of copy-protecting their originals!)

Super Star Diskettes. You already know how good they are. Now you can buy them...cheap.

Well, that's the story.

Super Star diskettes don't roll off the boat from Pago-Pago or emerge from a basement plant just east of Nowhere.

Super Star diskettes have been around for years... and you've used them for years as copy-protected software originals, unprotected originals. Sometimes, depending on which computer you own, the system master may have been on a Super Star diskette. And maybe more than once, you've bought a box or two or more of Super Star diskettes without knowing it. They just had some "big" company's name on them.

Super Star Diskettes are good. So good that a lot of major software publishers, computer manufacturers and other diskette marketers buy them in the tens or hundreds of thousands.

We buy them in the millions.

And than we sell them to you.

Cheap.

When every little bit counts, it's Super Star Diskettes.

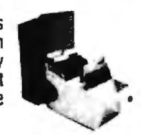
You've used them a hundred times. .under different names.

Now, you can buy the real McCoy, the same diskette that major software publishers, computer manufacturers and diskette marketers buy...and call their own.

We simply charge less.

Super Special!

Order 50 Super Star Diskettes and we'll be happy to sell you an Amaray Media-Mate 50 for only $8.75, shipping included...a lot less than the suggested retail price of $15.95.



Regular DISK WORLD! price: $9.69 ea. + $2.00 Shpng.

The Super Star LIFETIME WARRANTY!

Super Star Diskettes are unconditionally warranted against defects in original material and workmanship so long as owned by the original purchaser. Returns are simple: just send the defective diskettes with proof of purchase, postage-paid by you with a short explanation of the problem, and we'll send you the replacements. (Incidentally, coffee stained diskettes and diskettes with staples driven through them don't qualify as "defective".)

DISKETTE STORAGE CASES

DISK CADDIES
The original flip-up holder for 10 5¼" diskettes. Beige or Grey only.
$1.65 ea. + .20 Shpng.

DISKETTE 70 STORAGE
Dust-free storage for 70 5¼" diskettes. Six dividers included. An excellent value.
$9.95 ea. +$3.00 Shpng.

HOW TO ORDER:

ORDERS ONLY:
1-800-621-6827
(In Illinois: 1-312-256-7140)
INQUIRIES:
1-312-256-7140

FOR FASTEST SERVICE, USE NO-COST MCI MAIL: Our address is DISKORDER. It's a FREE MCI MAIL letter. No charge to you. (Situation permitting, we'll ship these orders in 24 hours or less.)

SHIPPING: 5¼" & 3½" DISKETTES—Add $3.00 per each 100 or fewer diskettes. **OTHER ITEMS:** Add shipping charges as shown in addition to other shipping charges. **PAYMENT:** VISA, MASTERCARD and Prepaid orders accepted. **COD ORDERS:** Add additional $5.00 special handling charge. **APO, FPO, AK, HI & PR ORDERS:** Include shipping charges as shown and additional 5% of total order amount to cover PAL and insurance. We ship only to United States addresses, except for those listed above. **TAXES:** Illinois residents, add 7% sales tax.

MINIMUM ORDER: $35.00

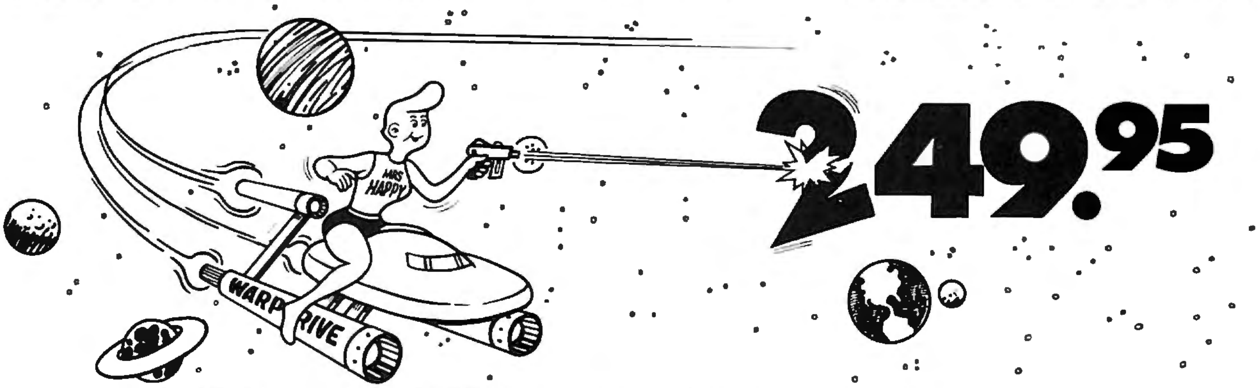
HOURS:
Human: 8AM-6PM Central Time, Monday through Friday
Answering Machine: 6PM-8AM, All Times
MCI MAIL: 24 hours a day.

WE WILL MEET OR BEAT ANY NATIONALLY ADVERTISED PRICE ON THE SAME PRODUCTS AND QUANTITIES SUBJECT TO THE SAME TERMS AND CONDITIONS.

DISK WORLD!, INC.

629 Green Bay Road
Wilmette, Illinois 60091

ATARI DISK DRIVE OWNERS . . .
HAPPY BLASTS RETAIL PRICE—ORDER TOLL FREE!



THE FAMOUS HAPPY ENHANCEMENT NOW ONLY $149.95
for 1050 order number HC1C, for 810 order number HC8C

Makes your disk drive read and write faster, and allows you to execute the HAPPY WARP SPEED SOFTWARE. Available only for ATARI 1050 and 810 disk drives. 1050 version allows true double density plus the original single and enhanced density. PRICE INCLUDES WARP SPEED SOFTWARE BELOW, installation required.

HAPPY WARP SPEED SOFTWARE REV 7 (not sold separately)

Includes the famous HAPPY BACKUP and COMPACTOR which are the most powerful disk backup utilities available for your ATARI computer, plus MULTI DRIVE which allows high speed simultaneous reading and writing with up to 4 HAPPY ENHANCED drives, plus SECTOR COPIER which is the fastest disk copier that supports the 130XE RAMDISK, plus the WARP SPEED DOS which improves ATARI DOS 2.0s to allow fastest speed, plus HAPPY'S DIAGNOSTIC which allows comprehensive disk drive testing.

HAPPY 1050 CONTROLLER $64.95 order number HC2C

For use with HAPPY ENHANCED 1050 disk drives only. Allows easy access to HAPPY 1050 slow and fast speeds and ultimate control of disk drive write protect, including writing to disk back side and protecting valuable data disks. Printed circuit board has switches and write protect indicator LED, installation required.

GET YOUR FAVORITE HIGH SPEED DOUBLE DENSITY DOS

Both of these disk operating systems support the fastest speed with both HAPPY 810* and 1050, and with HAPPY 1050 you get true double density. WARP SPEED DOS XL is HAPPY's own version of OSS DOS XL, and includes under cartridge, under ROM and AXLON RAM disk version, and is order number HC4C at $29.95. TOP DOS version 1.5 from ECLIPSE SOFTWARE has more menu driven features, operates in all three densities, supports the 130XE RAMDISK, and is order number HC6C at $39.95. *Note: 810 requires upgrade below.

810 VERSION 7 UPGRADE $49.95 order number HU3C-XXXX

Allows older 810 HAPPIES to use newer software. Includes custom plug in IC and rev 7 WARP SPEED SOFTWARE. Same price for all HAPPY 810s registered or not. When ordering replace XXXX in part number with the serial number of your HAPPY COMPUTERS manufactured 810 board, or with a 2732 or 2532 which corresponds to the EPROM part number in your HAPPY 810 socket A102 of your side board modified HAPPY (not made by HAPPY COMPUTERS), installation required. Upgrade not needed for new 810 HAPPYS and serial number over 8000.

SUPER PACKAGE SPECIALS

Get a HAPPY 1050 ENHANCEMENT and CONTROLLER and WARP SPEED DOS XL for just $199.95 order number HS5C, or get the same with TOP DOS 1.5 instead of DOS XL for just $214.95 order number HS7C. If you already have the 1050 ENHANCEMENT you can get the HAPPY 1050 CONTROLLER and WARP SPEED DOS XL for $74.95 order number HXL9C, or get the HAPPY 1050 CONTROLLER and TOP DOS 1.5 for just $84.95 order number HTD9C. For other specials and dealer pricing call (408) 779-3830.

All prices include UPS shipping in USA, add $10.00 for shipment outside USA. California residents add sales tax. No extra charge for credit cards or COD, VISA or MASTERCARD accepted. Our toll free number is an order taking service, not our line. To ORDER ONLY call (800) 538-8157 outside California, or (800) 672-3470 inside California, ask for extension 817 and have your credit card, part number and quantities ready. Toll free hours 6 am to 12 pm Mon.–Fri., 8 am to 8 pm Sat. & Sun., Pacific Time. For answers to questions call HAPPY COMPUTERS at our number below. Office hours 9–5 Mon.–Fri. Pacific Time.

HAPPY COMPUTERS, INC. * P.O. Box 1268 * Morgan Hill, CA 95037 * (408) 779-3830

To receive additional information from advertisers in this issue, use the handy reader service cards in the back of the magazine.

MUST LIQUIDATE At Far Below Dealer Cost!
TOTAL Personal Computer System

Factory Reconditioned with Factory Warranty!

Carries easily as a suitcase! Plugs into 115V outlet

GREAT GIFT IDEA!

Sorry, we're not permitted to PRINT the famous brand-name. BUT, we CAN "tell all" if you call us TOLL FREE: 1-800-328-0609!

THE COMPUTER

Snap-on computer keyboard! 64K RAM, 20K ROM. Full-size typewriter keyboard. Upper and lower case letters, numerals, symbols, reverse characters. 2 cursor control keys, 4 function keys, programmable to 8. Music synthesizer with 3 independent voices, each with 9 octave range. Input/output ports accommodate . . . user, serial, ROM cartridge, joysticks, external monitor, phone modem.

Built-in disk drive! Intelligent high speed unit with 5¼" floppy disk recorder. 170K formatted data storage; 35 tracks. 16K ROM. Uses single sided, single density disk. Serial interface. Second serial port to chain second drive or printer.

Built-in color monitor! Displays 40 columns x 25 lines of text on 5" screen. High resolution. 320 x 200 pixels. 16 background, character colors.

Built-in ROM cartridge port! Insert ROM program cartridge. Multitude of subjects available in stores across the nation!

Original List Price $995.00

Liquidation Priced At Only **$388**

Item H-764-63631-00 Ship, handling: $20.00

THE PRINTER

Print method: Bi-directional impact dot matrix.
Character matrix: 6 x 7 dot matrix.
Characters: Upper and lower case letters, numerals and symbols. All PET graphic characters.
Graphics: 7 vertical dots — maximum 480 columns. Dot addressable.
Character codes: CBM ASCII code.
Print speed: 60 characters per second.
Maximum columns: 80 columns.
Character spacing: 10 characters per inch.
Line feed spacing: 6 lines per inch in character mode or 8 lines per inch selectable. 9 lines per inch in graphics mode.
Line feed speed: 5 lines per second in character mode. 7.5 lines per second in graphics mode.
Paper feed: Friction feed.
Paper width: 4.5" to 8.5" width.
Multiple copies: Original plus maximum of two copies.
Dimensions: 13"W x 8"D x 3¼"H. Wt.: 6½ lbs. **Power:** 120V AC, 60 Hz.

Original List Price: $200.00

Liquidation Priced At **$118**

Item H-764-63681-00 Ship, handling: $7.00

THE SOFTWARE

"Easy Script" One of the most powerful word processors at any price! Cut re-typing, create documents from standard paragraphs, do personalized letters, see and change a document before it is printed. Instruction manual has extensive training section that simplifies use . . . even for someone who has never used a computer or word processor before!

"The Manager" A sophisticated database manager for business or home use. Business uses: accounts payable/receivable, inventory, appointments, task manager. Home uses: mailing lists, home inventory, recipes, collection organizer, investment tracking, checkbook balancing. School uses: research article index, gradebook.

Original List Price: $73.98

Liquidation Price **$24**

Item H-764-64011-03 Ship, handling: $3.00

BUY INDIVIDUAL UNITS OR GET THIS ULTRA-FAMOUS SYSTEM AT ONE LOW PACKAGE PRICE!

TOTAL Personal Computer System available at FAR BELOW dealer cost!

Original List Price $1,293.00

TOTAL PACKAGE PRICE **$488**

Item H-764-64011-02 Ship, handling: $24.00

Compatible with above Computer System (Not included in package price.)

JOYSTICKS (Set of 2)
Mfr. List: $59.90 pr.
Liquidation Price . . . . **$18** pr.
Item H-764-63622-01 S/H: $6.00 pr.

64K MODEM (Factory Reconditioned w/Warranty)
Mfr. List: $124.95
Liquidation Price **$19**
Item H-764-63646-01 S/H: $4.00

VISA MasterCard

Credit card members can order by phone, 24 hours a day, 7 days a week.

Toll-Free: 1-800-328-0609

TeleCheck — Your check is welcome! No delays when you pay by check!

C.O.M.B. CO.®
Authorized Liquidator
14605 28TH AVENUE NORTH
MINNEAPOLIS, MINNESOTA 55441-3397

SEND ME THE ITEMS I HAVE LISTED BELOW
Price subject to change after 60 days. Sales outside continental U.S. are subject to special conditions. Please call or write to inquire.

Item No.	#	Item	Price	S/H
		TOTAL		

C.O.M.B. CO.® Item H-764
14605 28th Ave. N./Minneapolis, MN 55441-3397

Send the items indicated at left. (Minnesota residents add 6% sales tax. Please allow 3-4 weeks delivery. Sorry, no C.O.D.)
☐ My check or money order is enclosed. (No delays in processing orders paid by check, thanks to TeleCheck.)
Charge: ☐ MasterCard ☐ VISA
Acct. No. ______ Exp. ______
PLEASE PRINT CLEARLY
Name ______
Address ______
City ______
State ______ ZIP ______
Phone ______
Sign here ______

The Sider™ 10 MB hard disk

Only $595 Thru December 31, 1985

from First Class Peripherals

Decisions, decisions. First you had to choose between Apple and IBM. Now you have to decide which hard disk subsystem to purchase—and they all seem about the same. *But are they?*

First Class Peripherals can make your hard disk decision a lot easier. Because whether you use an Apple II+ or IIe...or IBM PC* or XT...we offer a Sider 10 MB hard disk subsystem just right for all your storage needs.

The most reliable, affordable 10 MB hard disk on the market

The Sider features state-of-the-art Winchester disk technology. Direct booting without floppies. Self-contained power supply. And compatibility with the most popular Apple or IBM software.

In addition, the Sider is *plug and play.* Everything you need for quick, easy installation is included: cable, host adapter, software and manual.

Built to last by Xebec

The Sider has won rave reviews for its performance and reliability. That's because it's manufactured exclusively for First Class Peripherals by Xebec, the industry's leading manufacturer of computer disk drives and controllers. And it's sold *direct to you,* so there are no dealers or distributors to hike up the cost.

Full guarantee and free tech hotline

You can choose your Apple or IBM Sider with confidence. Simply order and use your Sider for 15 days. If you're not 100% satisfied, return it for a full refund. Keep it, and you'll enjoy a full one-year limited warranty...plus access to our toll-free hotline, should you ever have a technical or service question.

It's easy to order your Sider

The Sider is priced at just $595 for the Apple model...$595 for the IBM. *That's hundreds of dollars less than what you'd expect to pay for the comparable "big name" models.* To order, use the coupon below...or for faster service, order by phone using Visa, MasterCard or American Express. Call toll-free:

1 800 538-1307
Extension 702

*Must contain hard disk ROM.

FIRST PERIPHERALS CLASS

3579 Highway 50 East, Carson City, NV 89701

☐ Yes, please send me the Sider, including 10 megabyte hard disk drive, host adapter card, cable, complete installation software and documentation for my: ☐ **Apple II+ or IIe** ☐ **IBM PC or XT**

I prefer to pay as follows:

☐ I've enclosed my check or money order for $595* plus $15 shipping and handling, payable to First Class Peripherals.

☐ Please bill the following credit card account for $595* plus $15 shipping and handling:

☐ VISA ☐ MasterCard ☐ American Express

Card # *Exp. Date*

Signature

*Residents of CA, NV and PA, please add appropriate sales tax.

Name

Address

City

State *Zip*

Telephone (area code)

Mail to: First Class Peripherals
3579 Highway 50 East, Carson City, NV 89701

702

Advertisers Index

This holiday season put your fellow PC users on line with

COMPUTE!

The magazine for Home, Education and Recreational Computing.

SPECIAL HOLIDAY GIFT SAVINGS

For each gift you give, save 25% off the regular $24 subscription rate. You pay only $18 per gift subscription.

A SEASONAL CARD WILL ANNOUNCE EACH GIFT

YOUR NAME ____________________ ☐ PAYMENT ENCLOSED ☐ PLEASE BILL ME

ADDRESS ____________________ APT # ______

CITY/STATE/ZIP ____________________

1.	2.
TO ____________	TO ____________
ADDRESS ________ APT # ____	ADDRESS ________ APT # ____
CITY/STATE/ZIP ____________	CITY/STATE/ZIP ____________
GIFT CARD TO READ ____________	GIFT CARD TO READ ____________

N

PRICES GOOD IN USA ONLY. FOR CANADA AND FOREIGN PLEASE ADD $6 PER SUBSCRIPTION. M1087

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7478 DES MOINES, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!
PO BOX 10954
DES MOINES, IOWA 50347

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7478 DES MOINES, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!

PO BOX 10954
DES MOINES, IOWA 50347

COMPUTE!'s
FREE Reader Information Service

Use these cards to request FREE information about the products advertised in this issue. Clearly print or type your full name and address. Only one card should be used per person. Circle the numbers that correspond to the key number appearing in the advertisers index.

Send in the card and the advertisers will receive your inquiry. Although every effort is made to insure that only advertisers wishing to provide product information have reader service numbers, COMPUTE! cannot be responsible if advertisers do not provide literature to readers.

Please use these cards *only* for subscribing or for requesting product information. Editorial and customer service inquiries should be addressed to: COMPUTE!, P.O. Box 5406, Greensboro, NC 27403. Check the expiration date on the card to insure proper handling.

Use these cards and this address only for COMPUTE!'s Reader Information Service. Do not send with payment in any form.

COMPUTE!

101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117
118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134
135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168
169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185
186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202
203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236
237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253

Circle 101 for a one year new U.S. subscription to COMPUTE!: you will be billed for $24.

Please let us know. Do you own: plan to buy:

own		plan to buy
☐ 270	Apple ________	☐ 271
☐ 272	Atari ________	☐ 273
☐ 274	Commodore ________	☐ 275
☐ 276	IBM ________	☐ 277
☐ 278	TI-99/4A ________	☐ 279
☐ 280	Other ________ (specify model)	☐ 281

Please print or type name and address.
Limit one card per person.

Name ____________________

Address ____________________

City ____________________

State/Province ____________ Zip ________

Country ____________________

Please include zip code. Expiration date 1/31/86 C01285

Place
Stamp
Here

COMPUTE! Reader Service
P.O. Box 2141
Radnor, PA 19089