PROOF OF PURCHASE

Software Giveaway with Purchase of this Magazine
See back cover

HOME COMPUTER® magazine

FOCUSING EXCLUSIVELY ON • APPLE • ATARI • COMMODORE • IBM • TEXAS INSTRUMENTS

Vol. 5 No. 5

$3.50 in USA
$4.50 in Canada

Now Expanded To Include ATARI COVERAGE!

Memoranda Processing

—Computer-Assisted Correspondence Comes Home

Featuring A Ready-To-Use Electronic Typewriter for:

- Memoranda Of All Kinds
- Quick & Easy Correspondence
- Short Lists, Notes, & Schedules
- Custom Text-Format Templates

Contains 41 Type-In Programs!

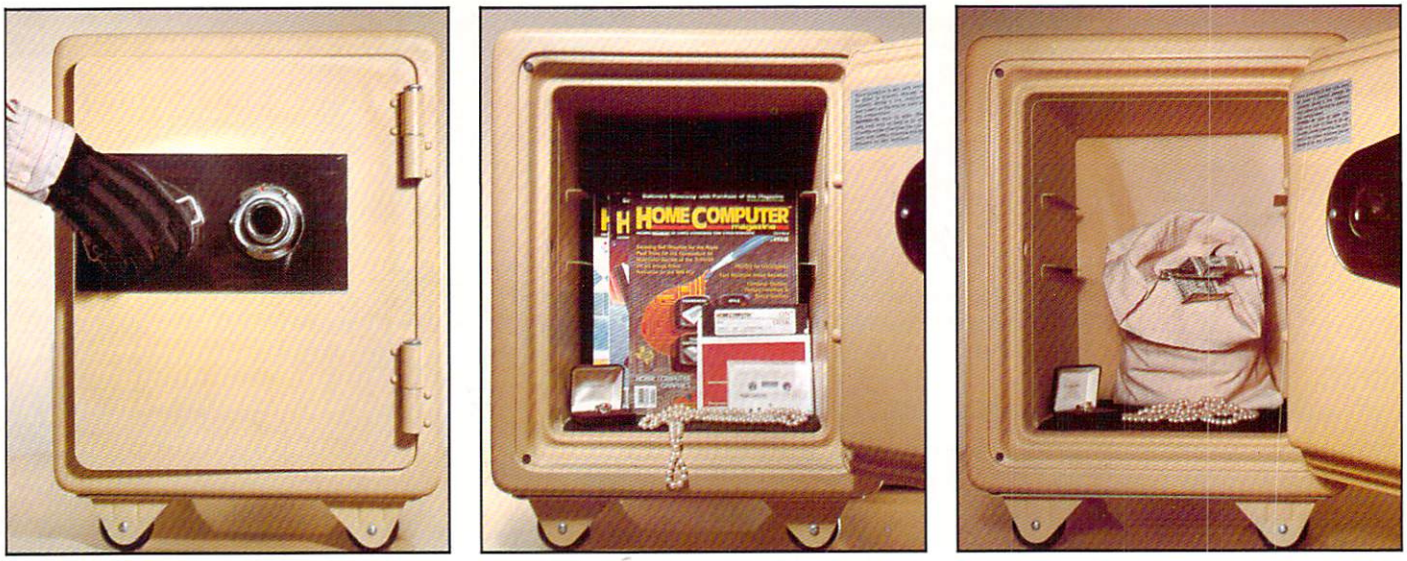
Special Apple, Atari, Commodore, IBM, & TI Software:

- Fantastic Voyage Through the Microprocessor
- Cardio-Vascular Bio-Simulation
- Cybernetic Card Capers
- King Arthur's Final Round

—Reviews Galore:

- ★ Home Print-Studio Tools
- ★ Computer Rock Videos
- ★ Music/Sound-Dazzler Enhancements
- ★ Inter-Galactic Hitchhiking

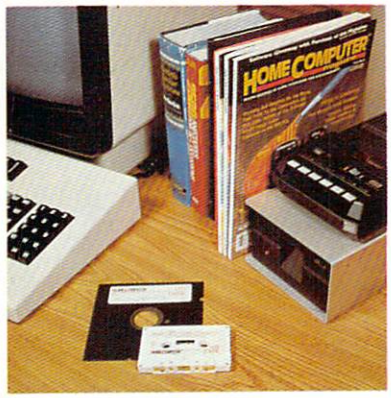
05



MISSING ANY VALUABLES?

If you're missing any back issues of HOME COMPUTER™ magazine you're missing more than you'll ever know . . .

Having each issue of *Home Computer Magazine* readily at hand provides you with direct access to a valuable reference library of home computer knowledge —unequaled anywhere!



A valuable reference library of each *Home Computer Magazine* issue is the One Essential Peripheral™ for your home computer.

Back issues of HCM's program service— ON DISK™ or ON TAPE™ are also available.

ON DISK™ and ON TAPE™ are the convenient, accurate and affordable ways to save hundreds of typing hours.

Collect all the programs from each magazine issue on a ready-to-RUN quality floppy disk or cassette tape available in separate versions for Apple, Commodore, IBM, and Texas Instruments home computers.

"Safeguard" Your Home Computer Knowledge— Order Valuable Back Issues Today!

To Order, Use Bind-In Card at Center of Magazine.

HCM BACK ISSUES

. . . and discover what's in store for you— an extraordinary resource value!

VERSIONS SUPPORTED:

Machine	Media
APPLE II Family (A)	ON DISK™
Atari (At) (coverage commenced with issue 5.5)	ON DISK™/ON TAPE™
COMMODORE 64 (C)	ON DISK™/ON TAPE™
IBM PC/PCjr (I)	ON DISK™
TI-99/4A (T)	ON DISK™/ON TAPE™

*** = No ON TAPE™ available, even if normally supported**
TX = Extended BASIC programs only
PCjr = Available for PCjr only

Apple owners: Please note that ON DISK™ Media for *HCM* 4.1-4.3 is in DOS 3.3 format only, and all Apple programs beginning with *HCM* 4.4 are in ProDOS format. All programs will RUN on a 64K Apple II+ (with Applesoft BASIC in ROM), an Apple *IIe*, or an Apple *IIc*.

Apple & IBM "clone" owners: Some *HCM* programs may not RUN (without modification) on your machines, because of differences in hardware and/or BASIC interpreters.

Issue 4.1:

Premier Issue * Uncle Larry's Fiddle Tunes * Electronic Sheet Music * Music in Mini Memory * PCjr: A Look Inside the Peanut's Shell * 66 Keys to Graphics Success: A Primer for the Commodore 64 * Have No Fear: Assembly Language Won't Byte, Part 3 * Porsches and other Pipedreams: Computer Assisted Savings * 3DIIe: Apple Graphics in Three Dimensions, Part 1 * Biting Into Your Apple * Don't Be A SlowPOKE * Down Memory Lane: Don't let programmable characters gobble up your memory * Easy As Pie: Apple programming for intricate works of art * Microcomputer Accuracy * What is LOGO? * Lyrical LOGO * LOGO Shoots for the Moon: A lesson in structured problem-solving * Product Reviews * Flak Attack * Slots * Meltdown * Challenging the Tower of Hanoi * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Flak Attack (A,C,I,T)
Air-to-ground battle game
Applesoft 3D (A)
Apple graphics in three dimensions
Tower of Hanoi (A,C,I,T)
An ancient brain teaser
Saving (A,C,I)
Computer-assisted savings plan
LOGO Poet (A,C*, I)
Recursion frees the poet in your console
LOGO Apollo (C*, T)
A lesson in structured problem-solving
Slots (T)
An intriguing Las Vegas simulation
Uncle Larry's Fiddle Tunes (C, I, T)
Play ten beloved fiddle tunes
Music Magic (TX)
"Joy to the World" in harmonious BASIC
Music Assembler (T)
Assembly language simplifies composition
Autosprite (C)
Routines to keep your graphics lively
Meltdown (TX)
Debug the reactor and save the world

Issue 4.2:

Graphics * Sea of States * San Francisco Tourist * Building Your Character: A Graphics Editor for the VIC-20 * Quick Pixel Tricks: A Graphics Editor for the C-64 * Follow the Bouncing Ball: On the rebound with graphics fundamentals * 3DIIe: Apple Graphics in Three Dimensions, Part 2 * Double Your Color, Double Your Fun: Sprites try on a layered look * Musical Mystery Words * Matrix Muncher * Elementary Addition and Subtraction for the VIC-20 * IBM Animation: Controlling the pallet on the PCjr * Jr. Sounds Off: Access Jr's Special Sound Enhancements * The Electronic Home Secretary * Files in LOGO * LOGO Spans the Generation Gap: A review of Commodore LOGO * FROGO: LOGO Invades the Arcade * Product Reviews * Tablut * Cannibals * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Cannibals (A, C, I, T)
Livingston Stew, I presume?
FROGO (T)
A logical LOGO learning lesson
The Home Secretary (A, C, I, T)
Address & inventory recordkeeping
LOGOFILES (A, C*, I, T)
Access your DATA files in LOGO
Sea of States (C, TX)
State Capitals and dive for booty
Tablut (C, I, TX)
14th-century strategy revisited
Matrix Muncher (C)
Solve unknowns simultaneously
Graphic Editor (C)
Pixel tricks create easeful graphics
Mystery Words (A, I)
Reading between the treble clefs
PCjr Animation (PCjr)
Exploring Junior's graphic modes
Applesoft 3-D IIe (A)
Edit your 3-D graphic shapes

Issue 4.3:

Productivity * Snap-Calc: A Homespun ready-to-use spreadsheet * Bars and Plots: Create colorful graphic charts of your records * Elementary Addition and Subtraction for the 99/4A and C-64: A powerful children's learning tool * Spider Graphics: Spin a colorful web on screen * Convertible for Comfort: Automatically convert your machine-language programs to DATA statements * Programming: The Name of the Game: Designing your own game—a complete tutorial * Colorfun on your VIC-20 * Product Reviews * Binary Forest: Branching out with LOGO * LOGO Flakes: Creative explorations with snowflake designs * Robochase * Cyber-Cipher * Wild Kingdom * Speeder * Boolean Brain * Missile Math * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Snap-Calc (A, C, I, TX)
Home sweet spreadsheet
Robochase (A, C, I)
Run from the rampaging robots
Spider Graphics (A, I)
Spin a myriad of rainbow filaments
Boolean Brain (A, I)
A graphic Adventure inside computers
Wild Kingdom (A, C, I, TX)
Flee ferocious felines
Missile Math (A, C, I, T)
Launching grade-school arithmetic
Binary Forest (A, C*, I)
Branching-out with leafy LOGO trees
Bars & Plots (T)
Color your chart factfully
Cyber-Cipher (T)
Decode correct color combinations
Elem. Addition & Subtraction (C, T)
BASIC preschool arithmetic skill-builder
LOGO Flakes (T)
Snowflakes in June? This must be LOGO
Convertible for Comfort (C)
Machine Language DATA auto-conversion

Issue 4.4:

Computer Sports * IIc: The Core of a New Machine * On the Home Court: Computer Sports Simulation * Razzle Dazzle: Quick Graphics Magic for the 99/4A * Simon Sez: Plug in 114 new BASIC commands to the Commodore 64 * Tax Deduction Filer: A complete tax recordkeeping program convinces you that makes tracking of deductions a breeze * Kaleido Computer: Creating a myriad of mosaic designs on your home computer * Multiplan Medium, Part 8 * Have No Fear: Assembly Language Won't Byte, Part 4 * The RS-232 Interface: Understanding Your Link to the Periphery * One for the Money, Two for the Slow—Adding a Second Drive to the PCjr * Missionary Impossible: A Logic Puzzle in LOGO pits you against hungry Cannibals * Product Reviews * Boolean Brain * Stadium Jumping * Market Madness * Elementary Addition and Subtraction: An arithmetic tutor (for Apple and IBM PC and PCjr systems) * HCM TECH NOTES: Apple, C-64, IBM and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Boolean Brain (C, TX)
A graphic Adventure inside computers
Tax Deduction Filer (A, C, I, TX)
SAVE-ing with your tax deductions
Market Madness (A, C, I, TX)
Exciting Stock market simulation
Stadium Jumping (A, C, I, T)
Horsing around an Olympic Stadium
LOGO Spreadsheet (A, C*, I, T)
And you thought LOGO was kidstuff
Missionary Impossible (A, C*, I, T)
Watch out for Cannibals with LOGO
Elem. Addition & Subtraction (A, I)
BASIC preschool arithmetic skill-builder

And There's More On The Next Page

Issue 4.5:

Building Up Your Software Library * Quiz Construction Set: Create a Quiz or Take a Quiz—a must for students and teachers * Personal Loan Calculator: Find out where your interest lies * Jumping Ahead With Game Programming: A complete game programming tutorial includes a program example * Sketch-64: Joystick graphics with just a flick of the wrist * Simon Sez: New string-related commands explained * Razzle Dazzle: Character manipulation on the 99/4A * Division Tutor: Teaching BASIC math learning skills * Putting The Puzzle All Together: Apple IIc Programming Considerations * Bird Brain * Slither * LOGO Clones: TI Graphics In a Turtle-Shell * Build A LOGO Adventure, Part 1 * Product Reviews * HCM One Liners * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Bird Brain (A, C, I, T)
Keep your fishing feathers dry
Division Tutor (A, C, I, TX)
Expand elementary math skills
Personal Loan Calc (A, C, I, T)
Find out where your interest lies
Sketch-64 (C)
Use a joystick to draw graphics
Quiz Construction Set:
Quiz-Make/Quiz-Take (A, C, I, T)
Complete tutorial with file examples
Peg Jump (A, C, I, T)
Learn BASIC game programming
Slither (A, C, I, T)
A maze of snake-like proportions
LOGO Clones (T)
TI-Graphics in a Turtle-Shell
LOGO Adventure (A, I, C*)
Pt. 1: Creating interactive fiction

Issue 5.1:

Thought Processing: A New Frontier in Home Computing * The Organizer: Store and organize your thoughts * Orbital Defender * Quiz-Print/Quiz-Print Tutorial: This educational enhancement is a tool for use with your Quiz Construction Set (see *HCM* 4.5) * Electronic Backgammon: A modern version of an ancient game of skill * Razzle Dazzle: Screen patterns with graphics characters on the 99/4A * Kors-Elf: An Arcade Typing-Tutor Game * Personal Loan Calculator: Find out where your interest lies * Apple Seedlings: A ProDOS Date-Setting Utility * IBMpressions: Create a beautiful pie chart * Build A LOGO Adventure, Part 2 * LOGO Sailing: A Premier Yachting Event * Simon Sez: Composing music is simple * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News * Group Grapevine, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Orbital Defender (A, C, I, T)
Split-second battle decisions
Electronic Backgammon (A, C, I, TX)
Pit your pips against the computer
Kors-Elf (A, C, I, TX)
An arcade typing adventure
The Organizer (A, C*, I, TX*)
A versatile Thought Processor
Quiz-Print (A, C, I, T)
Format printouts of your quizzes
Apple Seedling (A)
BASIC utility dates ProDOS files
LOGO Adventure (A, C*, I)
Pt. 2: Creating interactive fiction
Merging Files (C)
Experienced hackers only!
Personal Loan Calc (T)
Find out where your interest lies
Razzle Dazzle (T)
Wormwood your character graphics
LOGO Sailing (T)
Turtles race for the America's Cup
IBMpressions (I)
Create a beautiful pie chart

Issue 5.2:

Number Crunching: The Building Blocks of All Computing * It Figures: An equation calculator that'll crunch your numbers accurately * Evacu-Pod: See if you can rescue all the miners in this challenging space game * Switch 'n' Spell: Electronic anagram brain teasers to puzzle over (for children, and adults) * Laserithmetic: Strut your math skill with this space fantasy edu-game * Organizer Reports: An enchancement to print-out your organized thoughts (see The Organizer *HCM* 5.1) * Razzle Dazzle: Tinker with musical sounds, or Play it Maestro! * What is CP/M?: Learn the Basics of Control Programming for Microcomputers * Apple Seedlings: Sorting out your ProDOS Catalog * Commodore Hornblower: Discover what's inside the Commodore 64's SID chip * IBMpressions: Create 3-D surface drawings in BASIC * Field & Screen: A tutorial for using a Data Base System—correctly * Product Reviews * HCM One Liners * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A * Product News, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Evacu-Pod (A, C, I, TX)
Miner rescue in space
It Figures! (A, C, I, TX)
A mighty equation calculator
Laserithmetic (A, C, I, T)
Blast aliens with your math skills
Organizer Reports (A, C*, I, TX*)
Print your organized outlines
Switch 'n' Spell (A, C, I, T)
A spelling aid that's fun to boot
Apple Seedlings (A)
Sort your ProDOS catalogs
Commodore Hornblower (C)
Inside the SID chip
IBMpressions (I)
3-D surface drawing in BASIC

Issue 5.3

Computerized Budgeting: Featuring a ready-to-use budget processor (Budgetron) * Honing your Geometry skills (Geometrix) * LOGO Adventuring (Build A LOGO Adventure, Pt. 3) * Survive a nuclear plant disaster (Over-Reaction) * Guard the seaways with nuclear submarines (Torpedo Alley) * Turtles race with Zeno's theory (Achilles and the Turtle) * Apple Seedlings: Character graphics on the hi-res screen * Commodore Hornblower: Select waveforms and envelopes from SID * Razzle Dazzle: Multi-layered animation with TI sprites * IBMpressions: Blending sign waves into complex patterns * MAC-ROs: Expanding BASIC on Macintosh * Speeding Up a BASIC Program * Product Reviews * HCM One Liners * Group Grapevine * Product News, * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Budgetron (A, C, I, T)
Budget your income and expenses
Geometrix (A, C, I, T)
Sharpen your geometry skills
Over-Reaction (A, C, I, T)
You're at a nuclear plant's controls
Torpedo Alley (A, C, I, T)
Keep the enemy's ships at bay
Achilles & the Turtle (T)
A LOGO demonstration of Zeno's Theory
LOGO Adventure, Pt. 3 (A, C*, I)
Apple Seedlings (A)
Character graphics in hi-res
Commodore Hornblower (C)
Waveforms & envelopes from SID
Apple Tech Note (A)
Key-in checking routine
IBM Tech Note (I)
Selective keyboard input
Commodore Tech Note (C)
Merging programs from disk
TI Tech Note (T)
A full-screen editor

Issue 5.4

Time Management: Computer-Assisted Efficiency Comes Home * Run-Day-View: Let your computer streamline your day * Trig-Trix: Use the triangle as a measuring tool * Archeodroid: Participate in a future archeological dig * Mine Over Matter: Hone your business skills in this simulation of uranium mining operations * MAC-ROs: Create custom graphic shapes on Macintosh * IBMpressions: Create your own computer windows * Razzle Dazzle: Explore sound-on-sound recording with this 3-track recorder program * Apple Seedlings: Use a pie chart as a visual aid * Commodore Hornblower: Change filters on the SID chip * Algorithm-A-Tricks: Create invisible ripples * Speeding up a BASIC Program: Part 2 * Build a Logo Adventure: Part 4 * Product Reviews * HCM One Liners * Home Computer Industry Journal * Product News * HCM TECH NOTES: Apple, C-64, IBM, and 99/4A, and much, much more!

CONTENTS: ON TAPE™ & ON DISK™

Run-Day-View (A, C, I, T)
Organize your daily schedule
Trig-Trix (A, C, I, T)
The triangle makes measurement easy
Archeodroid (A, C, I, T)
Join a future archeological dig
Mine Over Matter (A, C, I, T)
Manage a uranium mine
LOGO Adventure, Pt. 4 (A, C, I)
Apple Seedlings (A)
Creating pie charts
Commodore Hornblower (C)
Changing filters on the SID chip
Apple Tech Note (A)
Exiting error routines
Commodore Tech Note (C)
Merging programs from tape
IBM Tech Note (I)
Using special character graphics

NOTE:
Programs for the IBM PC/PCjr will run on the Tandy 1000 with modifications specified on page 130.

This space reserved for Issue 5.5

Atari users please note that coverage in HCM didn't commence until Issue 5.5—the issue you are now viewing.

Also commencing with this issue, programs for the IBM PC/PCjr will run as is on the Tandy 1000.

◄◄◄◄ HCM BACK ISSUES

The Plain & Simple Truth About HOME COMPUTER™ magazine

Chock Full of Valuable Software & How-To Articles Without Filler

Every issue is a software "horn of plenty" with dozens of type-in-and-RUN programs printed in an easy-to-read listings format. Our programs are also available on inexpensive disks or cassettes for those who prefer the convenience of ready-to-RUN software. Step-by-step tutorials round out each issue, providing the solid facts you need without fluff or filler. Thus, each issue functions as an excellent reference work, as well as a valuable software source.

No Outside Advertising

Freed from the pressures of servicing *advertisers*, we concentrate on serving our *readers*. Each issue provides uninterrupted editorial flow and graphic layouts for better comprehension—plus unbiased product reviews which focus on true strengths and weaknesses, wherever the chips may fall . . . And we don't have to worry about losing advertisers because of publishing software in the magazine that is "too good." Consequently, we can provide the best free software available anywhere.

Focused on the 5 Hot Home Brands

We are 5 system-specific magazines under one wrapper—not a sprawling, "general interest" publication which attempts to cover too wide a field, only to spread itself too thin. The other side of the coin to this focused approach is the knowledge you gain from being exposed to the many tips, ideas, and techniques we provide for 4 of the 5 systems you may not even have. You'll learn more about your Apple, Atari, Commodore, IBM, or Texas Instruments home computer from this one magazine than from a host of more limited sources.

A Balanced Mix For a Perfect Recipe

In each issue we strive for a perfect balance of productivity, entertainment, education, utilities, and computer literacy—serving the needs of novice and pro alike. Every issue is a full-course meal, with a smorgasboard of tasty dishes for all palates. Whereas other computer magazines may dish out lumps of "editorial indigestion," we serve up a satisfying blend—one digestible byte at a time.

—Welcome to Our World of Home Computing

HOME COMPUTER magazine

Home Computer Magazine (ISSN 0747-055X) is published ten times per year by Emerald Valley Publishing Co., P.O. Box 70288, Eugene, OR 97401. The editorial office is located at 1500 Valley River Drive, Suite 250, Eugene, OR 97401 (Tel. 503-485-8796). Subscription rates in U.S. and its possessions are $25 for one year, $45 for two years, and $63 for three years. In Canada add $7 per year. Other foreign countries $43 for one year surface mail. Inquire for air delivery. Single copy price in U.S. and its possessions is $3.50, and $4.50 in Canada. Foreign subscription payment should be in United States funds drawn on a U.S. bank. Second-class postage paid at Eugene, OR 97401, and Columbia, MO 65201.

POSTMASTER: Send all address changes to **Home Computer Magazine**, P. O. Box 70288, Eugene, OR 97401. Subscribers should send all correspondence about subscriptions to above address.

Address all editorial correspondence to the Editor at **Home Computer Magazine**, 1500 Valley River Drive, Suite 250, Eugene, OR 97401. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage and self-addressed envelope. Not responsible for lost manuscripts, photos, or program media. Opinions expressed by the authors are not necessarily those of **Home Computer Magazine**. All mail directed to the Editor or to the "Letters to the Editor" column will be treated as unconditionally assigned for publication, copyright purposes, and use in any other publication or brochure, and are subject to **Home Computer Magazine's** unrestricted right to edit and comment. **Home Computer Magazine** assumes no liability for errors in articles, programs, or advertisements. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by **Home Computer Magazine** or the publisher unless explicitly stated.

Publisher/Editor-in-Chief	Gary M. Kaplan
Executive Editor	David G. Brader
Managing Editor	Walter Hego
Associate Editor	Wayne Koberstein
Sr. Technical Editors	
William K. Balthrop, Roger Wood	
Technical Editors	
D. Donaldson, Tom Green, Edward McManus, G.R. Michaels, Patricia Swift, Randy Thompson, Andy Widders-Ellis, Scott Williams	
User Group Editor	Judy Campbell
Assistant Editors	
Dana M. Campbell, Laile L. DiSilvestro Michael Hosmar, Steven P. Nelson	
Program Translators	
Stephen A. Cordon, Jeff Harper, Galen Huntington, James Jentile, Ross Johnson, Robert Paschelke, Ken Thomas, Nancy Vendelin, Mark White	
Production Manager	Norman Winney, Jr.
Creative Director	Gei-Lei Gom
Photography	
Nelson Stevens, K.D. Wainsworth	
Customer Service	**Tel. (503) 341-1029**
Dealer Sales & Distribution	**Tel. (503) 341-1036**
Main Switchboard	**Tel. (503) 485-8796**

Antique typewriter provided by Earl Kelso (Vida, OR).

Outside HCM

What kind of contraption is this? If you've seen our last few covers, you know by now that we like to transform old gizmos into new ideas—illustrating how your home computer can go way beyond the abilities of past, single-purpose machines. We've depicted an antique coffee grinder as a "number cruncher," a classic alarm clock as a computerized scheduler, and now, an ancient typewriter (circa 1905) as still another transfigured electronic tool—the "memoranda processor." What next? If our imaginations are not limited, neither are the uses of our amazing home computers.

Inside HCM

Memory . . .in a machine? Remember the last time you sat in front of your computer. Was it like the time before? Or did your trusty machine show you an entirely different face? In fact, each time you boot up another piece of software, your computer seems to become almost a different machine.

In humans, such inconsistent behavior can either be sign of senility, or of acting ability. But your machine is not senile. Like the man of a thousand faces, the computer is a multi-faceted performer. Its versatility lies in its incredibly *flexible* memory. Software fills this memory with a *script*, telling the machine what to do, how to *act*.

In this issue of *HCM*, we give your computer many roles to play. Our software runs the gamut from memorable simulations to memory helpers. Take *Electronic Typewriter*, a simple tool for processing memos, letters, lists, and short schedules. This program has all the virtues of a memory typewriter—it combines quick and easy typing with line-by-line editing.

A computer's memory resides like a ghost in the machine. Hardware provides a storage place for memory; software provides memory-instructions which the computer must read and turn into action. Giving the computer a different set of instructions is a bit like changing its "face"—giving it a different appearance, and another purpose. *NanoProcessor* is a software simulation of the malleable machine on its most fundamental level. Learn how to enter simple instructions into this computer model *bit-by-bit*, creating programs (even music routines) in a language of ones and zeros.

From the heart of the machine, we move on to *Vital Signs*, a simulation of our own heart and respiratory system. If you can keep this heart model beating—as you respond *consciously* to the effects of exercise and changing air quality—you may learn how to prolong the beat of your own heart.

King Arthur has a place for the stout of heart on his final battlefield: *The Plains Of Salisbury*. One hint: In this strategic exercise—played out on a terrain much more complicated than a chessboard—the key to planning future moves is the ability to remember past ones.

And let us remind you: Each machine brand we cover gets its own quick and helpful program within a mini-tutorial column. In this issue, *Apple Seedlings* blends sine waves into vibrational applesauce; *Commodore Hornblower* sets sail with SID's special effects; *IBMpressions* casts 3-D shadow graphics; *Razzle Dazzle* calculates super-accurately on the 99/4A; and our new column, *Atari Atrium*, illuminates 4-channel sound-on-sound recording.

Can your computer become a source for print graphics? Discover the ins and outs of this new software genre in our comprehensive *Home Print Studio* review. Then examine several music and sound products, including *Music Construction Set*, *Music Video Kit*, *Music Synthesizer*, and *Mockingboard*, a sound board for the Apple II family. And , if you're in need of galactic guidance—and some cosmic humor, check out our review of the *Hitchhiker's Guide To the Galaxy*.

Fill your computer's memory with our software, and you can immediately see the variety of roles it can play in your life. Read our reviews, and find out what your machine *might* do with what's available in the commercial market.

As the machine remembers, it thinks about what to do next. *What* the machine remembers is what it *does*. So, go ahead—make use of *Home Computer Magazine* and give your computer some very useful memories. You're in for a "memorable" treat.

Until next time, have fun reading, learning, and RUNing HCM

On Screen

By Gary M. Kaplan
Publisher & Editor-in-Chief

"Many people have been put off . . . by the term 'electronic music' which unfortunately invokes auditory images of spacey-sounding squeaks and squeals right out of some '50s sci-fi movie."

This issue represents another milestone in the evolution of *Home Computer Magazine*. As we start mapping out the editorial content for our sixth publication year which commences this January, we do it as a larger magazine—expanded in size and content with coverage of a fifth major home-computer user base: the diverse family of Atari 800-compatible machines.

I would thus like to take this opportunity to welcome all our new Atari friends to the *HCM* fold. Many of our long-time readers may be wondering why we are adding this coverage now. For some time, we have been receiving a tremendous number of requests from the Atari community—coming in the form of letters, phone calls, and visits to our booth at Consumer Electronics Shows. The message to *HCM* is now crystal-clear: *Atari users want in*! It didn't take a proverbial "whack on the side of the head" for us to realize that a dual service and expansion opportunity now existed: We could provide new service to the large, active Atari user base; and we could foster additional cross-pollination of fresh ideas within a larger base of *HCM* readers. Before this expansion, however, we first wanted to put in place a series of format enhancements—including our Programmer's Windows, Glossary, Industry Journal, Counterpoints, sectional edge-tab IDs, and so on. You've seen all this implemented over the last several months.

In this issue we are presenting another major enhancement—our new debugging aid for readers who prefer to type in our program listings, rather than order them ready-to-run ON TAPE or ON DISK. We have optimized our Bug-Out Codes (BOCs) and checking utilities to make the job of catching typing errors as simple and painless as possible. And we've accomplished this without sacrificing any readability or clarity in *HCM*'s widely acclaimed listings format.

Long-term readers have witnessed a steady improvement in our published programs over the past year—demonstrating our commitment to excellence. Although we now craft *each* of our magazine programs with extra-special care and attention to detail, there is, on occasion a program that stands a little taller than the rest. A case in point is this issue's *NanoProcessor*. It artfully employs what a computer does best—dynamic simulation—to "demystify" the machine's own inner workings. In my opinion, this simulation is destined for the Software Hall of Fame. Though many have tried hard to *teach* the rudiments of machine language, most have only succeeded in *scaring off* the uninitiated. But, with *NanoProcessor*, even the meek can inherit this down-to-earth knowledge—*and have loads of fun doing it*! Next issue, we will follow with *NanoAssembler*, a tool for "playing" your way into the formerly esoteric realm of assembly language.

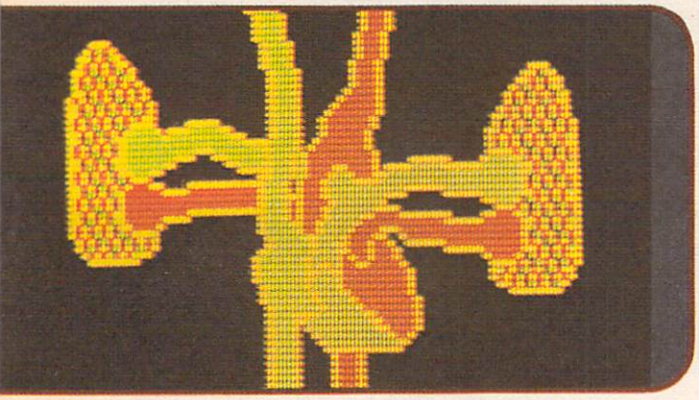
On another score, the hills around *HCM*'s Emerald Valley are alive with the sound of music, as we premiere "Soundbytes." This column bridges *HCM* to its new sister publication, *Music & Electronics*. I confess that we have somewhat of an ulterior motive for providing this bridge: We'd like to bring you all across! So, check out our two-page announcement following "Soundbytes" for details about the new magazine and how you can become a charter subscriber.

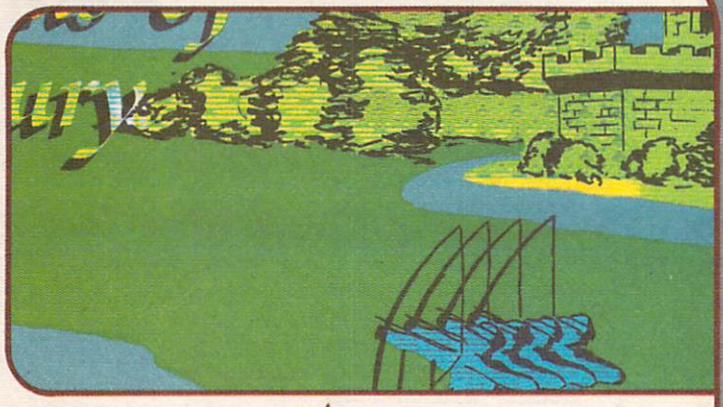
Preparing for the launch of *Music & Electronics* has, in fact, been a real "ear-opener" for me—providing a measure of excitement that I can only liken to the beginning of the home computer revolution. As I've discovered, the link-up with music synthesizers and other gadgets in the production of electronic music is truly a harmonious use for home computers. Many people have been put off, however, by the term "electronic music," which unfortunately invokes auditory images of spacey-sounding squeaks and squeals right out of some '50s sci-fi movie.

Let me assure all of you that both the sound quality and stylistic possiblilties of what is more aptly called "electronically produced music" must simply be heard to be believed. And what's even more amazing is that you don't have to be a *musician* to participate in all the creative fun. Even those of us who don't know the difference between treble and tribbles can produce musical recordings that are out of this world . . . Those of you who first want to "hear it with your own eyes" (or "see it with your own ears"), will get a small taste of all this sensory excitement in an upcoming issue of *HCM*—replete with some bound-in musical surprises.

In our next issue of *HCM*, we will have a new challenge for our readers as we launch the "Problems In Productivity" series. This first challenge requires the use of *HCM*'s *Snap-Calc* spreadsheet tool to analyze typical financial alternatives for a college education, and come up with a cost-effective solution. *Snap-Calc* was originally published in Volume 4, Number 3; so if you don't yet have the program, get ready by ordering the magazine and software ON DISK or ON TAPE now. And while you're at it, take the opportunity to fill in any other gaps in your *HCM* back-issue reference collection.

HOME COMPUTER® magazine



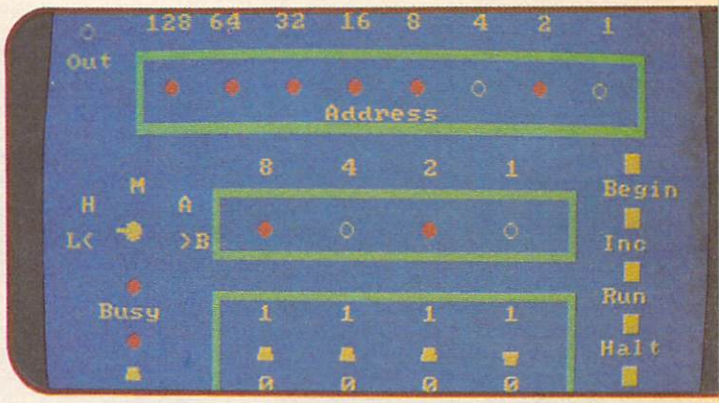


FEATURES

PRODUCT REVIEWS

CONTENTS

VOLUME 5 NUMBER 5



DEPARTMENTS

MACHINES SUPPORTED

Machine	Requirements	Media
APPLE II family	At least 64K RAM & Applesoft BASIC	ON DISK
FRANKLIN ACE	At least 64K RAM	ON DISK
ATARI 800, 800XL, 130XE	-None-	ON DISK/ON TAPE
COMMODORE 64	-None-	ON DISK/ON TAPE
COMMODORE 128	Must be in 64 mode	ON DISK/ON TAPE
IBM PC	BASICA	ON DISK
IBM PCjr	Cartridge BASIC	ON DISK
TANDY 1000	GW-BASIC Version 2.02	ON DISK
TI-99/4A	TI BASIC or Extended BASIC	ON DISK/ON TAPE

SPECIAL NOTES

Apple Owners: Apple ON DISK media for *HCM*, Vol.4, No.1-Vol.4, No.3 is in DOS 3.3 format only. Beginning with *HCM,* Vol.4, No.4, all Apple programs are in ProDOS format. All programs **RUN** on Apple II+, Apple *IIe*, or Apple *IIc* computers.

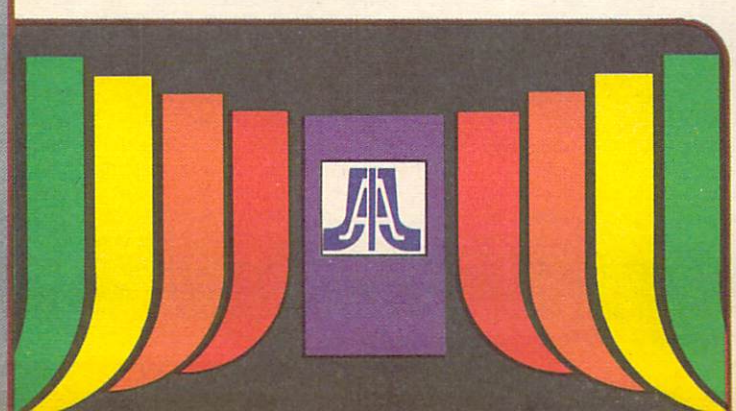
Franklin Owners: Beginning with *HCM*, Vol.4, No.4, all Apple ON DISK media is in ProDOS format only. Booting ProDOS on a Franklin requires the following steps:

1. Boot ProDOS. When the system hangs up, press [RESET].
2. Type 265B:EA EA and press [RETURN].
3. Type 2000G (insert no spaces between the last zero and the G) and press [RETURN].

See *HCM*, Vol. 5, No.4, page 13 for more information.

Tandy 1000 Owners: Starting with *HCM*, Vol.5, No.5, all of our IBM PC programs run on the Tandy 1000 without modifications. Programs prior to Vol.5, No.5 may need minor changes as explained on page 130 of *HCM*, Vol.5, No.4.

Errata: In Program Listing Contents on page 75, the Atari Tech Note is incorrectly shown as page 57. The correct location is page 52.



Letters to the Editor

Just In Atari Time

Dear Sir:

I would like to compliment you on your magazine. I find it to be one of the most useful and informative magazines I have read for home computers.

However, since subscribing to your magazine, I have sold my TI-99/4A and upgraded to an Atari 800 computer.

Frankly, I don't understand why you don't provide coverage for the Atari line of home computers.

Do you have plans to include them in your magazine now that Atari has introduced their new XE and ST lines?

Barry Gray
Sacramento, CA 95828

A timely question, Barry. You are one of many Atari users who have written in the past several months to ask for Atari coverage. Well . . . Surprise! In this very issue, as you can see, we are premiering full coverage of the Atari 800 family and compatibles. Each issue will include Atari versions of all our major programs, as well as an Atari Tech Note and "Atari Atrium." Starting with this issue, Atari programs are also available ON DISK and ON TAPE. Letters to the Editor from other Atari users will be published in this column commencing with our next issue. So hurry and get them in to us! This seems like the right time to make the following announcement:

ATTENTION:
ATARI USERS GROUPS

HCM periodically publishes Group Grapevine, a bulletin board of users group activities across the nation, and in other countries as well. Beginning in the next issue (Vol. 5, No. 6), Group Grapevine will cover Atari Users Groups in addition to our normal coverage of Apple II, Commodore 64, IBM PC & PCjr, and TI-99/4A users groups. We hereby issue an open invitation to Atari Users Groups everywhere to join our "grapevine" of computer enthusiasts. If you are an active member or officer of your local Atari Group, please write or call our office immediately to insure coverage of your organization. And if you have a message to put out to other groups, if you are starting a new group, or if you have an interesting item to share, send a note or picture—or better yet, a group newsletter—to the Users Group Editor, Home Computer Magazine, 1500 Valley River Drive, Suite 250, Eugene, OR 97401, Tel. (503) 485-8796.

Great Julian Date Mixup Fixed

Dear Sir:

In regard to Max A. Shelhorse's letter in HCM Vol. 5, No. 3, page 11:

As an avid amateur astronomer for almost 15 years, I've had occasion to write a program which calculates the positions of the planets in their orbits for any date and time. This, of course, includes a routine to determine the Julian date.

Upon examining Mr. Shelhorse's Julian date algorithm, I recognized it as being correct and could not see how it would determine Jan. 1 to Mar. 1 as 65 days. Closer scrutiny of your listing, however, revealed that the error seems to stem from a possible quirk in the TI-99/4A's interpreter.

Note that in line 130 the second parenthesis is not closed in the calculation of DP. While this would result in a syntax error on my Spectravideo SV-328 computer (which uses the same BASICA & GWBASIC as the IBM), it apparently goes undetected on the TI-99/4A for which the routine was written. I was able to duplicate your erroneous value of 65 by writing the calculation as DP – INT(30.6001)*(MF). It appears that somehow the TIs interpret the typographical error in this way, rather than refusing it.

Simply closing the last parenthesis should resolve the "Great Julian Date Mix-Up." Following is a slightly modified, Microsoft BASIC, version which creates fewer "local" variables and makes the routine somewhat more universal by also allowing for BASICs which do not have the ELSE clause:

```
500 'Julian Date subroutine (enter with
    YR=year, M=month #, D=day)
510 F=YR :MF=M :IF M<3 THEN
    F=YR-1 :MF=M+12
520 IF YR<=1582 AND M<=10 AND
    D<=15 THEN JD=0 :GOTO 540
530 JD=INT(F/100)
    :JD=2-JD-INT(JD/4)
540 JD=JD+INT(365.25*F)+INT(30.60
    01*(MF+1))+D+1720994.5
550 RETURN
```

Gregory S. Vigneault
Toronto, Ontario, Canada

Gregory, you discovered a typo in our listing, not a bug in the TI BASIC interpreter—line 130 does require a closing parenthesis just as you have noted. With this corrected, the more important bug from the original letter of Max Shelhorse is still apparent when this program is run. Line 130 of the original listing starts like this:

```
130 A=INT(F/100):: B=2-A+INT(A/4) . . . .
```

By comparing it with line 530 in your listing above, we discovered the real problem—the plus (+) should have been a minus (–). If you make these two fixes, Max Shelhorse's listing and yours give identical (and correct) Julian Dates. Thanks for helping us unravel this celestial mystery.

Commodoring In Argentina

Dear Sir:

I'm the happy owner of a Commodore 64 and I'm very interested in telecommunicating. I have several questions I'll be very happy if you could answer.

Could I telecommunicate if I have the Datasette or do I need the disk drive?

What things do I need to telecommunicate; is it enough only having the Vic-modem?

I'm also interested in making a RESET. How can I make it; can't a key be programmed for that?

Any assistance you can provide me will be greatly appreciated.

Ignacio Adrogue
Buenos Aires, Argentina

You need only two things to make telecommunicating easy: one is a VicModem, or any other C-64 compatible modem unit; the other is software that can send, receive, and handle data via modem. Although you can write your own software, you may want to obtain some type of commercially-available "terminal-emulation" software package, such as VIP Terminal from Softlaw, or SKIWriter II from Prentice Hall (reviewed in HCM Vol. 5, No. 4). As for the RESET feature, see the Commodore Tech Note, "Installing a Reset Switch" in Vol. 4, No. 4.

Wants Apple Frequency Blender

Dear Sir:

I own an Apple *IIc* computer and find your magazine very helpful. In your last issue (Vol. 5, No. 3), there was an article for the IBM entitled "Frequency Blender" which interested me. I have been trying to find a program which extracts cycle frequency and amplitude from data using fast Fourier analysis and also would like an Apple program which could blend the cycles into a composite curve like the IBM program. Could your magazine and staff include a program in Home Computer Magazine or suggest a source to find a program which would do this? Thank you.

Richard L. Laughlin
Tulsa, OK 74105

The answer to the second part of your request is right beneath your fingertips, Richard. In this issue, "Frequency Blender" for the Apple II family appears in our regular mini-tutorial, "Apple Seedlings." We have some doubt, however, that the 6502 processor in the Apple IIc is fast enough to do the kind of Fast Fourier Transformations (FFT's) you desire.

2nd Disk Drive and PCjr Warranty

Dear Sir:

Having sought information about adding a second disk drive to my PCjr for almost four months now, I have found that the only thing that exceeds the amount of information available from sales reps, shop techs, and other "experts" is the amount of misinformation one acquires during the search. The most valuable piece of information I acquired was relevant to your fine, fine magazine. I latched onto Volume 5.2 and 5.3—straight scoop without the poop! Obviously I require Vol. 4, No. 4 which contains the original article. Check enclosed.

One of the criteria established (in consultation with the wife) was that the addition of a second drive could not result in our version of the national debt. That ruled out third-party manufacturers. Hello! Do it yourself! Your instructions and kit seem to offer the only hope. Thanks, HCM.

As the result of this letter to you, I called Randall Baxter, who lives in my area, about the procedures you published for adding the second drive. A point he made was that any modification to the disk drive controller card would probably invalidate the balance of the year's warranty on my machine. Ah, serious consideration time. It would be most advisable for me to wait until my warranty expires. Mind you, I'm not superstitious, but I don't take chances either.

In light of my decision to wait until the magic November 1985 date when my warranty expires, do you suppose that the kit and instructions you have so thoughtfully provided at an extraordinarily reasonable cost would be available until then? I could invest in it now, but the way the PC market is now, I might have already moved into an IBM PC by that time. (That statement resulted in some mention of a dead body by my wife—it wasn't clear whom she meant!)

Not intended to cause palpitations, but anything in the works about a do-it-yourself project for adding a hard disk (or did I miss that too)?

Anyway, thanks for having a fine magazine with great articles and even better programs.

F. E. DiGirolomo, Jr.
Duncanville, TX 75116

Yes, Mr. DiGirolomo, installing the drive kit will indeed void your warranty, as we stated in the original article (Vol. 4, No. 4). However, we will continue to sell the kit as long as there is any demand for it—so if you want to wait, don't worry. We'll have it in November when you need it.

More SIDs for "Music of Sound"

Dear Sir:

Your reviews get better and better, and this two-part series on "The Music of Sound" for the C-64 is a prime example. Have you heard of anyone making a cartridge or interface to add more SID chips to the Commodore so more voices/tones could be created? The Commodore 64 Programmer's Reference Guide on the 6581 SID (sound interface device) chip suggests on page 460 that: "SID can process external audio signals, allowing multiple SID chips to be daisy-chained or mixed in complex polyphonic systems." Imagine, if you will, what such a successful addition would do to music—even speech synthesis! Can this not be done?

Souping-up the C-64 this way seems pretty cost-effective from this consumer's point-of-view. Can I make a general appeal to those IC wizards out there? Any word of this being attempted by anyone?

By the way, in hooking up your C-64 to a stereo amplifier any buzz produced from poor grounding can be stopped by reversing the outlet, or cables' plugs. Try 'em all until the buzzing quits or as in my case, make sure the entire setup is plugged into a grounded circuit.

Elizabeth Schelper
Ft. Myers, FL 33901

Thanks, Elizabeth, for the compliment on our "Music of Sound" reviews. All the effort put into these articles seems to have been well worth it—judging by the favorable response from our readers. If the link between music and computing is one of your chief interests, you'll love our soon-to-be-released sister publication, Music & Electronics. We share your intrigue with the idea of daisy-chaining multiple SIDs together to—as the manual states—"create complex polyphonic systems." According to Commodore, there is no definite source for the SID chip apart from the C-64 itself. However, we do know of one supplier: The Jameco catalog (1355 Shoreway Rd, Belmont, CA 94002, Tel. 415-592-8097) offers the SID chip for $32.95. Perhaps some of our readers know of other sources. You might also contact local computer repair shops to see if they have a "junkpile" of otherwise unrepairable C-64s with intact SID chips that you can salvage. If any HCM readers attempt this project, we'd be interested in hearing about the results.

Expanding His TI-99/4A

Dear Sir:

Since I have become a TI-99/4A owner just seven months ago, I have become very involved with learning and enjoying all aspects of the computer.

I then discovered HCM on a newsstand and have purchased many back issues and software from your firm, enhancing my knowledge and enjoyment, but the software is on cassette tape.

I now want to add a disk drive, but do not have an expansion box. Please tell me how to add one in the most practical and inexpensive manner.

I now subscribe to HCM and look forward to each new issue. Keep up the good work!

Joseph A. Nicosia
Auburn, CA 95603

We know of three disk drive expansion units currently marketed for the 99/4A: the Texas Instrument Peripheral Expansion Box (TI PEB), the Myarc Peripheral Expansion System, and the Corcomp 9900 Micro Expansion System. The TI PEB sells for about $300 and includes one single-sided/single-density disk drive, a disk controller card capable of handling up to three disk drives, and a 32K Memory Expansion Card. Myarc's system sells for about $600 and includes a double-sided/double-density disk drive, a double-density disk controller that can control up to 4 drives, and the 32K memory expansion. Corcomp's system sells for around $325 and comes with an RS232 port, a disk drive controller, and a disk-based disk manager—the disk drive is extra. All three systems are available through several major TI-related product catalogs (Triton, Unisource, TexComp, Tenex, etc.), although there may be other regional or local sources. Thanks for the good words, Joseph, and good luck!

A Better Way To Write Thank You?

Dear Sir:

I want to thank you for such a great magazine. I have a PCjr and find your magazine to give the best coverage for the PCjr. I also like your new coverage of the back issues of HCM Vol. 5, No. 3—I am ordering two more back issues to complete my total collection of HCM. I also want to thank you for your Tech Notes—especially Vol. 4, No. 3 on Format A:/S. This saves a lot of time and hassle and provides many hours of enjoyment with my PCjr. Would you run a program on some sort of word processing or something similar so that I can sit down and write a letter such as this without having to worry about margins or running out of screen room? At the present, I use the Function + Prtsc key to print a letter, but when my letter is of greater length than my screen allows, I lose part of my letter.

I am looking forward to each and every issue of HCM. Keep up the great work.

James R. Delaney
Tedxico, IL 62889

With this issue, your wish is our command, James! The Electronic Typewriter (starting on page 19) will just fit the bill. It is ideal for letters, memos, or any other short correspondence.

What Am I Doing Wrong?

Dear Sir:

The purpose of this letter is twofold. First, my congratulations to you on a fine magazine written without the normal advertising. Prior to last month, I always purchased the magazine at a bookstore. I now have subscribed and will look forward to receiving it through the mail on a timely basis.

The second reason [for writing] is I am not a very experienced computer operator, and I am having trouble updating your programs when corrections are to be made. When I follow the sequence of commands as directed by the flyer included with the disk and try to save the merged version, I get a PATH NOT FOUND message. If I type the correction, I am able to merge the program with the correction. But being a hunt-and-peck typist you can see the problem this presents.

Can you tell me what I'm doing wrong? Thank you.

Dorsey Williams
St. Louis, MO 63116

You do not say what brand of computer you own, Dorsey, but from the kind of error message you are receiving, we assume you have an Apple II operating under ProDOS. ProDOS (Apple's Professional Disk Operating System) is an improvement over all previous versions of DOS for the Apple—it Loads and Saves files more quickly and lacks several of the bugs that existed in the earlier operating systems. It does, however, contain a few new aspects which might cause trouble for the inexperienced, Dorsey. One thing that you're sure to encounter is Prefixing. Every disk formatted for ProDOS has a Volume Name which makes up the first part of the Prefix for any file you wish to access. All Apple HCM disks have the name ON.DISK followed by the volume and issue numbers that correspond to the issue of that software. For example, this issue's disk has the name /ON.DISK.5.5 (the slash is added by ProDOS as a delimiter). As ProDOS "boots-up" from our disks, the Prefix is set to the ON DISK volume name. If you wish to access a different disk, you must do one of two things: (1) null the Prefix, or (2) set the Prefix to a different disk. The easiest way to null the Prefix is to simply type PREFIX */then press [RETURN]. To reset the Prefix, you must place the new disk in the drive and execute a* PREFIX *command. If you know the Volume Name, you can type* PREFIX /name, *where* name *is the Volume Name of the disk. An alternative way to reset the Prefix is to place the disk in drive one and type* PREFIX,D1. *The system will then check that drive and reset the Prefix. If you are swapping a number of disks in and out (say, you are Cataloging a number of disks just to see what is on them), nulling the Prefix is most convenient. If, however, you want to make sure that a file is written to or read from a particular disk, Prefixing that disk is an added safety measure. For more information about Prefixing see the "Home Computer Tech-Note for Apple" in HCM Vol. 4, No. 5.*

The Elusive C-64 One-Liners

Dear Sir:

I think your magazine is the greatest! I work on a PCjr at school and have a Compaq and a TI-99/4A at home, so I get quite a lot of use out of each one of your magazines. My girlfriend, however, knows only four words for her Commodore 64 (LIST, RUN, LOAD, and PRINT). So, when she asked me to show her something that her C-64 could do, I got real excited! I figured I would type in one or two of your HCM one-liners and really impress her. First, I typed in your Graphics Spectacular (issue 4.5) and nothing happened. The computer simply responded with <READY>. Not too discouraged, I NEWed the first program (SYS64759) and typed in another (It's Alive!, issue 5.3). It flashed the bottom two rows of the screen in white for about half a second. Needless to say, she was real impressed. I checked what I typed in with what you printed twice for each program. What's the matter here? Is it simply my ineptitude on the Commodore computer?

Jason Harper
Fairfield, OH 45014

Debugging typed-in software through the mail is just about impossible, Jason, but your difficulties probably stemmed from a lack of familiarity with the C-64. Because a line on the Commodore can contain no more than 80 characters, One-Liners make use of a number of abbreviations (such as F [SHIFT] O for FOR and N [SHIFT] E for NEXT). After you've keyed in such abbreviations, however, the Commodore expands them to full-size commands again. This means that if you make an error when entering the program, you will have to change all the expanded commands back to abbreviations when you edit the line so that it still fits within the 80 character maximum. We hope this gives you some idea of the nature of your errors.

99/4A Disk Manager Available

Dear Sir:

Having read your Vol. 5, No. 3 issue of HCM, I decided to try and locate a TI Disk Manager II module. After only making two telephone calls, I found that the following business has these modules "sitting on their shelves." The cost is $19.95 plus $3.00 for S&H. Contact Altex Electronics, 10731 Gulfdale, San Antonio, TX 78216. You can even order via the telephone (1-800-531-5369).

Hope this will be of help to any other readers who are currently looking for the Disk Manager II. Evidently, Edward Stack didn't call or contact the right people.

Keep up the good work and thanks for not forgetting about all of us that have the TI-99/4A.

Jerry Petrel
Auburn, KS 66402

Thank you, Jerry. We have verified your information by phone. It's good that we can rely on people like you to help keep track of TI peripherals—considering that specific items are often available only from regional or local sources.

New PCjr Owner Gets More Help

Dear Sir:

After haunting the magazine section of our local bookstore for the past five months looking for PCjr articles, suddenly your magazine appeared. Feeling like the proverbial Edsel owner, it was with humble gratitude that I carried HCM home. I was hoping for a few crumbs such as I had been finding in other computer magazines; but such a feast! Thank you. The same day that I purchased Vol. 5, No. 1, one check was written for the subscription and another for all of the back issues I had missed. When the back issues arrived it was interesting to see the evolution of your format. The present format is extremely readable and the type-in programs the most legible I have seen. Also, the Debugs on Display has gone from relatively indecipherable to very clear.

With the compliments comes a request for help or information as the case may be. I have typed in several short graphics programs (Ripples and HCM One Liner, Vol. 5, No. 2 and IBM Animation, Vol. 4, No. 2) with no results. I know it takes time for the calculations to occur, but how much time is reasonable? I have left the computer alone for up to an hour with nothing to show for it except a black screen and an irregularly blinking cursor in the upper left corner. I received Ripples ON DISK and tried that too with the same results. I'm willing to wait to see results, but not for hours. Is there something I am missing in trying to run these programs? In case it makes any difference, I have the enhanced PCjr with the Tecmar Captain board for 256K and use a memory configuration program worked out by IBM that frees up the memory in better fashion than the Tecmar software.

Thanks again for a very useful magazine. I really enjoy the productivity programs and the Tech Notes.

Lynn Cox
Kerrville, TX 78028

We are always happy, Lynn, when readers applaud our efforts to constantly improve the quality of this magazine. As for the problem you are experiencing with running our software—it is probably due to the IBM software that reconfigures your memory for use with the Tecmar board. You can use such software when running applications (such as Lotus 1-2-3) which can employ more memory than the PCjr normally offers. IBM Cartridge BASIC can only access 64K of memory, however, so any extra memory will not be used. In addition, the reconfiguring software places memory into an arrangement that differs from what the BASIC interpreter expects to find. This can have disasterous results, especially with graphics routines. When you wish to work in BASIC, just boot up with a standard DOS 2.1 master disk, and you should have no trouble running our software. Of course, the Tecmar memory will not be active, but then, neither will it be necessary. In short, save the reconfiguration software for applications that can use it.

ProDOS For Franklin

Dear Sir:

I received your May edition (Vol. 5, No. 3) of HCM and started to read the "Letters to the Editor" column. I came to an article requesting information on how to boot ProDOS on a Franklin. I thought I might try to help a little considering that I also own a Franklin ACE 1200.

I had the same problem myself when I purchased an AppleMouse which included software written in ProDOS. I recently learned of how to permanently fix a disk so ProDOS will boot automatically on a Franklin.

You need to use a program such as Copy][+, Nibbles Away, Bag of Tricks, or Byte Zap (found on Apple Mechanics) with a "Sector Editor" option to change the data on a ProDOS disk. For ProDOS version 1.0.1, you need to alter two locations on Track 01, Sector 09 of the disk: locations 5B, and 5C hexadecimal. These two locations will contain a D0 and a 03 respectively. Change both of these to EA and your Disk will now boot on a Franklin without a hitch.

I hope I could be of some help to any other computer enthusiasts who own a Franklin.

I am 15 years old and I enjoy reading your magazine!

Henry James Curry
Hope Mills, NC 28348

Thanks, Henry, for the Franklin Disk modification. We checked out this ProDOS/Franklin fix and found that it works great and does not affect the disk's ability to run on Apple computers. Although any modification like this entails a certain amount of danger (you could "blow up" your disk if you made a mistake), Franklin owners may find it convenient to modify ProDOS startup disks this way.

Converts Father To TI-99/4A

Dear Sir:

Please enter this gift subscription as soon as possible. My father spent most of his life as an electronics engineer and later was chief test director for a large aerospace company which is part of the giant Textron conglomerate. He purchased a TI-99/4A for me about two years ago, and I was immediately hooked. Last year, over his protests, I bought him a TI and an Extended BASIC cartridge, and now he is equally addicted. Up until that point he had never used a computer and was afraid that he was too old to learn and enjoy the world of computing. Now, after less than a year, his acute mathematical and engineering abilities have made him a veritable whiz!

He inspected several issues of your fine magazine when he visited last time, and he is now writing an article for submission to you, dealing with the graphic representation of simple and complex mathematical equations, a task which he feels the TI-99/4A is over-qualified to perform. He also feels, after looking at the publications available, that your magazine is the only choice he would make as far as submitting an article.

One closing note. I have been computing about two years and I also feel that your magazine is the finest available and that your program listings are far and away the finest published anywhere.

Thanks for great entertainment and instruction. I'll be reading you faithfully.

Robert P. Marsh
Greensboro, NC 27407

We thank you for the encouraging words, Robert, and look forward to hearing from your father!

Response to Quadram Review

Dear Sir:

The content of HCM has been able to shake up my grey matter with bits and pieces of information. These pieces come from all parts of the magazine. It appears that IBM has inadvertently placed in the hands of six or seven hundred thousand people a rather remarkable machine. That is, of course, if we can figure out just what it is that we have in our hands.

Contrary to your review, I claim that the Quadram equipment that I have attached to my IBM PCjr will operate other programs while outputting to the printer from QSPOOL buffer.

Software used:

1. Quadram QuadMaster jr Version 1.03 of 2/01/85.
2. IBM Writing Assistant Version 1.01
3. IBM DOS 2.10

In Quadram CONFIG.SYS, it is absolutely essential that JRVIDEO.SYS be placed first for configuration. The configuration that works for me is as follows:

```
DEVICE = JRVIDEO.SYS
DEVICE = QUADCLOK.SYS
DEVICE = QSPOOL1.SYS 48 48
DEVICE = RAMDRIVE.SYS 295
```

When booting up with the above, I was able to make all features of Writing Assistant work perfectly, including the QSPOOL print buffer. I have to believe that there must be others experimenting as I do. BBS is OK. Media does what it can. As I have heard, "what the mind can conceive, machines can do." HCM just may be on the right track as an information dispenser for the Orphans of the computer society.

I am told that Racore Corporation, 10 Victor Square, Scotts Valley, CA 95066, (408) 438-7255 is the designer and producer of the hardware and software that I have described in this letter.

Lloyd E. Howard
Chelsea, MI 48118

Thank you, Lloyd, for the information. You say you use version 1.03 of the Quadram software. When we reviewed the product, we had version 1.02, and this apparently made all the difference between your experience and ours. We received the updated version shortly after your letter arrived, and we found that many of the functions that we reported as faulty, such as the QSPOOL, did indeed work great with the new software. In fact, the batch files that we reported having trouble with worked just fine once we used the new version of the software. Thanks for putting us on the right track. Your note about Racore also proves correct. Racore was the original manufacturer of the Quadram, and now also markets the enhancement unit described in the review. In addition, the company recently released a 10 MB harddisk (they quoted a $1299 price when we called them), and will soon release a DMA controller for the PCjr ($149). Racore claims that its software is entirely different from Quadram's. We found it to be quite similar, however, except that no QSPOOL is available. Racore also informed us that the printer buffer option will now come as part of the DMA board.

Two Commodore Questions

Dear Sir:

I have two questions about the Commodore 64. First, I would like to know if there is a company somewhere that sells programs that will change the C-64 to run Assembler programs. Second, I would like to know if there are any bulletin boards for Commodore in my area. Thanks for your time.

Tim Gitchel
Irving, TX 75060

Tim, any C-64 can run assembler programs. There are many assembler programs available for the C-64 that allow you to write routines in assembly language and then convert that code into machine language. Here are three such assembler packages: Assembler Development System from Commodore, Merlin-64 from Roger Wagner Publishing Co., and the MAE Assembler from Eastern House. It is very likely that most of the programs you are using are already in machine language. Such programs can be identified by the LOAD"name",8,1 format used to load them from disk. As for your second question, we suggest you call Bill Marshall of the Irving Commodore Users Group (214-256-1402). He will probably be happy to hook you up to the local BBS scene.

TI To Okidata—Come In, Please

Dear Sir:

In Home Computer Magazine Vol. 4, No. 4, a letter from Mr. Nisius states that one can hook the parallel output of the TI-99/4A to an Okidata with a cable where one wires:

TI-99/4A		OKIDATA	
Term.	Description	Term.	Description
1	Handshake Out	1	Data Strobe
2-9	Data	2-9	Data
270K-ohm resistor in series with:			
10	Handshake In	11	Busy
11	Logic Ground	10-30	Data Return

Does this mean 10-30 on one side, the Okidata side, is all run to 11 on the TI side? Or does it mean the only other change is to cross 10 and 11 and add a resistor? I'd like to make my own cable. Please advise.

Stanley Page
Vancouver, BC, Canada

According to Tom Nisius, his solution is sufficient for his printer, Stanley. He placed a 270K-ohm resister in series with pin 10 of the TI and pin 11 of his Okidata, and connected pin 11 of the TI directly to pins 10-30 of the Okidata. It is important that you realize that this procedure is not for the novice and that it might not work on your printer. We did a little checking with two companies that market cables necessary for the interface (Tenex, P. O. Box 6578, South Bend, IN 46660; and Innovative Electronics and Computing, 4150 Fox Street, A-5, Denver, CO 80216). They verified that the 270K-Ohm resistor solution will probably not work on many models of Okidata printers. Technicians at both these companies stated that, apart from the pin differences you've recanted above, the timing of the Handshake In and Busy signals is not compatible between the TI and Okidata printers. Both companies' cables contain capacitors and even have active circuitry to put the two units in sync with each other.

Junior Graphics Dump

Dear Sir:

First, I would like to congratulate you on your fine magazine and wish you continued success. HCM is the only magazine that is really a voice of the users—especially PCjr users. While all the "PC" magazines take their monthly shot at the PCjr, you tell everyone the many things it does very well. Some magazines can't seem to understand that they are comparing a $799.00 machine with a $2900.00 machine.

I recently had a problem that perhaps other IBM users can benefit from. I have a Star Gemini 10x printer that normally works fine. When I run GRAPHICS.COM on the DOS disk, however, and then do graphics screen dumps to the printer, the line spacing is incorrect (leaving 10/144 of an inch between lines). Using Debug, I located the line feed spacing and corrected it. Now the program works fine. The procedure is listed below:

```
A>debug graphics.com
-a0168
0905:0168 mov ax,000e
0905:016B <return>
-ngrprint.com
-w
Writing 0315 bytes
-q
A>
```

Now simply use GRPRINT.COM instead of GRAPHICS.COM> The 0e in 016a controls the line feed size.

Keep up the good magazine! PCjr owners need you!

Todd Vernon
Warrensburg, MO 64093

Many thanks, Tom, both for your encouragement, and for your helpful update. We discussed the problem in the "Home Computer Tech Note for IBM" in Vol. 5, No. 1 and presented a solution in BASIC. Using DEBUG.COM to actually alter the GRAPHICS.COM routine is a very interesting solution. We did find that in completing the process with our Gemini 10, the number 0008 worked better than 000E, so anyone who tries this fix might want to experiment a bit.

Trouble In ProDOS

Dear Sir:

My computer is a 64K Apple IIe and I have only one disk drive. Your notes with "ON DISK" say that I need not buy anything to use these disks, however I am having far too much trouble with ProDOS. I hope that you can help me.

I learned for the first time of Prefix,D1 from your Tech Note in Vol. 4, No. 5. After formatting a disk I have typed Prefix,D1 then saved programs to my disk but I can't get it to boot up. I get the message "UNABLE TO LOAD PRODOS."

Several weeks ago I called your Customer Relations and it was suggested to me that I try loading and saving Startup—that did not work either. I tried to run it and got "HAS TO BE BOOTED." Well, I got back the "UNABLE" message again.

I listed and read your "input.namefile" and saw in line 360 "link to disk access routine." It would seem that access routine is what I need. Could you get this routine to me in the simplest of terms (as I am a novice) before I pull my hair out!

If the "fixes" for the programs can't be made without booting up each time, I will never get them right. I love the type of programs that are offered—just what I need. The magazine is excellent, but personally I could use some simple "type this and that" sort of thing. I will be subscribing to the magazine soon as it is hard to find around my home. I will be waiting for your help anxiously.

Mrs. E. O. Coldiron
Bridgeview, IL 60455

No, Mrs. Coldiron, you don't need to re-boot every time you want to make a change. You normally need to boot up only when starting a session. To start a session, boot up with the ON DISK original disk, select Exit To Applesoft BASIC, and then place the disk with the files you wish to work with in the disk drive. Then type PREFIX,D1and you're ready to LOAD, RUN, or modify any programs on that disk. Each time you change disks, you should change the Prefix. For more tips on Prefixing, see our answer to the letter from Dorsey Williams ("What Am I Doing Wrong") in this section.

If, however, you re-boot the system either by turning the power off and on or by pressing [CONTROL] [OPEN-APPLE] [RESET] then you will need to put a disk in the drive that contains two very important files: PRODOS and BASIC.SYSTEM. If these files are not on the disk when you re-boot, then you will indeed get the UNABLE TO LOAD PRODOS message. Simply formatting a ProDOS disk does not add these files to the disk, and therefore the disk is not a "boot disk." All of our Apple ON DISK products since Vol 4, No. 4 contain these two startup files and are thus boot disks. If you wish to make additional startup disks, you will need a ProDOS filer utililty available from Apple.

MissingLink Inventor Comments

Dear Sir:

Thank you for your request for response to a review by Pat Swift of the Missing Link in Vol. 5, No. 3 of HCM. I cannot thank you enough for alerting TI owners of its existence as it has been an uphill battle, and I think Pat did a good review of it. I sent Pat some thoughts separately and do agree that the package was disorganized and have added additional sheets to help people find their way around (enclosed).

I think Pat may have emphasized one or two things too much and may have missed one or two things that deserve emphasis. If you left planet Earth three years ago with prices on the TI home computer, you would have realized that it took $500 to do word processing on a printer. If someone brought you back and told you that for $70 you could also do word processing on a printer, you might have trouble believing it. I've been living under that scourge for three years. MissingLink has worked flawlessly for me for three years and I have tested it to at least an error rate of 1 in a billion. Pat seemed to emphasize the problem with VPLinK too much (her OKI possesses different control codes from my Epson based printer) and I think some people would think (1) it doesn't convey information accurately, or (2) it's too complicated to use. The title "The Zero Bug" also was used; if I were skimming, I would think it meant that the device doesn't work reliably.

I also think many PES owners would like an inexpensive backup to their RS232 cards. A guy who owns the PES and RS232 almost surely has to own the 32K, hence for only $30 he can have a backup for his system if it ever fails. People who buy it like that feature.

George A. Bowman
Midwest Engineering Consultants
Vernon Hills, IL

Thanks, George, for the feedback on our review. We're glad to hear about the continuing demand for the Missing Link. Your added documentation does answer many of Pat Swift's concerns with the previously incomplete and disorganized documentation. We're glad that our reviewer's input has helped you to improve your package.

Some Copyright Considerations

Dear Sir:

I wish to compliment you on your fine magazine. After buying three issues from the supermarket stand, I have recently called in my subscription. I am most enthusiastic about the programs you provide at such modest cost! It has long been my opinion that software is almost always overpriced in comparison to say a fine technical manual. Your magazine offers some relief from this situation.

It is exactly this enthusiasm and respect that causes me to write at this time. What is the exact legal obligation connected with the copyright laws? I ask this question because several of my computer friends have asked me for copies of the programs. In addition, my "friends" on the lines of our local user's group have also been interested. I can see two concerns for this situation. The first being that the programs may be ripped off and in some manner commercially sold. That, of course, is always a possibility but your registration would be at the beginning of each program so that the thieves would have to make a conscious effort to do so. Secondly, I realize that you are in the business of selling magazines. I might point out that with the acknowledgement of your magazine as the source, these few programs would introduce and sell more subscriptions. In short, I am asking if it would be possible to share this program amongst these interested parties? What are the acceptable parameters to be used?

I do appreciate your work and especially the programming for the PCjr . . . an unfortunate orphan before its time. Keep up the good work and let me know about the copyright thing.

Adelia Ramey
South Bend, IN 46637

As much as we appreciate your glowing compliments, Adelia, we cannot permit you to give our software away. As stated on page 6 of this issue, "EVP [Emerald Valley Publishing—that's us!] grants to such purchaser only [that's you only!], the limited license to enter these program's into the purchaser's computer, and to place such programs on a diskette or cassette for the purchaser's personal use." In short, our licensing agreement doesn't allow you to copy these programs for a friend.

As you well know, HCM does not receive any revenues from outside advertising—as does virtually every other computer magazine—and is therefore largely supported by the revenue generated from its sale of media and back issues. Because this revenue is so crucial to our existence, we must take a firm stand on this copyright issue, and we request that our readers do the same to safeguard our (and your) material.

This, of course, does not prevent you from showing the software to your friends and letting them know where they can buy HCM and ON TAPE or ON DISK. We believe that the price of back issues and their ON DISK contents is so very reasonable that your friends should take advantage of the legitimate media availability. We also ask that our readers please report to us any observed copyright violations so that we can take appropriate action.

How About The TI Pro?

Dear Sir:

I will not bore you or insult your intelligence by arguing the merits of the Texas Instruments Professional compared with the IBM PC.

Neither can I blame you for your choice in publishing programs targeted to this (IBM) market. I am certain there are infinitely more IBM users out there than we fewer, but more discriminating TI owners.

As an architect, I use a TI Pro with AutoCad at the office and also have another unit at my home. My 15-year-old son is able to rewrite most IBM programs to run on the TI. I can't!

My question is this: Since most algorithms are so similar for both machines, why can't you list both in your programs? There are several hundred thousand TI Pro owners who would welcome any sort of entertainment programs. There is only one periodical devoted exclusively to the TIP and it is so business oriented that it is no fun at all.

Please give us a break. I would subscribe in a minute if there were programs I can use and enjoy.

John Vaden
Ft. Worth, TX 76107

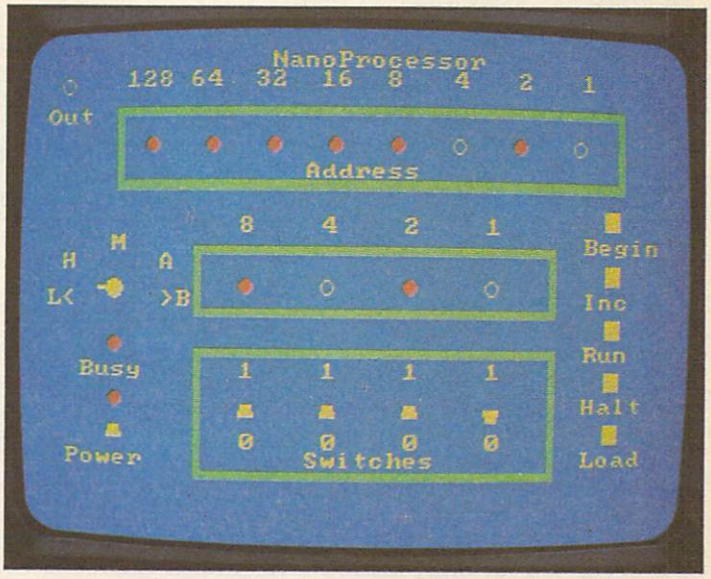
John, we agree that the TI-Pro is a fine machine—we use several in our offices. But, unfortunately, the machine does not run IBM BASICA, which is the language our IBM programs are written in. This makes it virtually impossible to convert our software to the TI -Pro without major changes to the IBM code. On the other hand, the Tandy 1000's BASIC is nearly identical to IBM's, and thus we have extended coverage to this machine. We would have to provide a complete new program listing for each TI-Pro version; at this time, we simply do not have the space in the magazine to provide that level of coverage. Not to be facetious, but perhaps your son could perform this conversion service for you when you become a subscriber.

HCM

The NanoProcessor

by Roger Wood
and
Wayne Koberstein
HCM Staff

With this simple simulation of the machine's inner workings, you can discover how easy (and fun!) it is to communicate with computers in their own language.



Since the premiere of the movie *Tron*—in which the hero has to fight his way out of a computer's microcircuits—many people have held a fascination for the inner workings of this "thinking machine." Are you one of them? Perhaps your interest has always been there, but you have not yet "taken the plunge" into machine-level programming. Or perhaps you know a great deal about this subject already, but would appreciate a very clear and simple demonstration of how computers "think." If so, you're ready for *NanoProcessor*—a program that emulates the computer at its most fundamental level.

At the heart of a computer, there is nothing but an immense set of on and off switches. But how can such a simple foundation foster such a complex information-handling system? In short, how are all these switches organized? A "real" computer, such as the one you have at home, is such a large system that it would be difficult to see the forest for the trees. But, with *NanoProcessor*, you have a chance to operate and see a much-simplified model of how a computer performs its tasks.

Brain Central

All computers—including the *NanoProcessor*—have a central "brain." It's called the CPU (Central Processing Unit). This brain recognizes and responds to different sets of numbers as instructions. These instructions direct the CPU to carry out certain operations—much as our brains store, handle, and act on information encoded in switch-like neurons. In a computer, information travels along parallel paths of wires and printed circuits called "buses."

As humans, we may think in English, Spanish, or any other language—some subtle, some exact. Computers also "think" in languages—such as BASIC and LOGO. CPU's like our own brains, must translate these high-level languages into encoded information. In computers, this information takes the form of machine language—a set of codes and numerical values expressed as binary numbers. Binary means "two," and implies two choices: on or off; or, in purely numerical terms, 1 or 0.

People tend to think in terms of a ten-based number system because they have ten fingers—but a switch has only two "fingers." (For a detailed look at converting between these number systems see the sidebar "Numbers To Bits And Back.") When you RUN *NanoProcessor* you will notice the row of switches at the bottom of the screen—your only means of shuttling information through this simulated computer (See Photos 1, 2, 3). Each switch only has two positions—up for *on* (1), or down for *off* (0). A switch is therefore the perfect means for conveying binary information.

Banking on Memory

Every computer has a memory area, called "Random Access Memory" (RAM), and a Central Processing Unit (CPU). Memory is the computer's capacity to store information, and is measured in terms of "bytes." A byte generally consists of 8 *bits* of information—where a bit is one binary (on or off) condition.

A CPU performs all the arithmetic that manipulates the numerically-encoded data—the ones and zeros—stored in a computer's RAM. This memory is made up of discrete "locations" in the machine, each of which has an "address." It helps to think of each memory location as a mailbox that not only has an address attached to it, but also a place to put the mail. This mail is the data stored at that location. Each "mailbox" has a limited amount of space that depends on the machine design. Because each of *NanoProcessor*'s memory locations can only store 4 bits, (one nibble), we say it is "nibble-addressable." By simply requesting a particular address, the CPU can immediately find what is contained at that address. This direct addressability of memory by the CPU is what gives a computer the power of *random access*.

The CPU and RAM are connected by three buses: the address bus (8 parallel wires), the data bus (4 parallel wires), and the control bus (See Figure 2). The first provides *access* to each memory location; the second simply moves data to and from each location; and the third carries control signals which control the flow of data between the CPU and memory. Furthermore, the CPU is organized into a system of discrete "registers" that serve as temporary stations for storing and shuffling data.

Look at the *NanoProcessor* front panel. On the middle-left side of the screen is a "rotary switch" with various letters positioned around it. The letters on the right-hand side of this switch—A and B—stand for the A and B registers in the CPU. It is between these two registers that the actual "arithmetic" and logic operations take place. The A register is also called the Accumulator because this is where the answers to many of the commands end up—or *accumulate*.

NUMBERS TO BITS AND BACK

One of the most important aspects of machine language programming (but sometimes most confusing for the novice) is converting digital numbers to binary and vice versa. To make this as easy as possible, we have employed two aids: 1) Whenever we list a *binary* number, we precede it with a percent (%) sign; and 2) *NanoProcessor* displays the decimal equivalent of each bit above the address and data windows of the front panel (see diagram below). We refer to these decimal equivalents as the "weight" of the bits.

To quickly convert a binary number to a decimal number, simply add up the weights of the "1" (on) bits. For example, to convert %1111 1010, refer to the following diagram:

	128	64	32	16	8	4	2	1
	*	*	*	*	*	*	*	*
%	1	1	1	1	1	0	1	0

Then add 128+64+32+16+8+2 and you can easily arrive at the correct decimal equivalent: 250. (Also, see Figure 1 for converting the numbers 0—15 to binary.)

Figure 1

Decimal	Binary
0	%0000
1	%0001
2	%0010
3	%0011
4	%0100
5	%0101
6	%0110
7	%0111
8	%1000
9	%1001
10	%1010
11	%1011
12	%1100
13	%1101
14	%1110
15	%1111

Turning On

First, press **P** to turn on the Power to your *NanoProcessor*. Make sure the rotary switch is pointing to the letter M, for Memory. You move this switch left (counter-clockwise)with the < (less than) key, and right (clockwise) with the > (greater than) key.

At the top of the screen, you should see an address box containing a long row of "lights" with numbers across the top. This is the "location counter" shown inside the CPU of Figure 2. It displays the 8-bit address of the location currently being interrogated by the CPU. Notice the vertical row of buttons at the right side of the screen. These buttons represent *NanoProcessor's* functions. Press the **B** (for **B**egin) key on your keyboard. This effectively turns off all the lights in the address box, indicating that you have returned to the first address in memory: the 0 (zero) location. Now press the **I** key, for **I**ncrement. This moves you to the next address: location 1. If you repeatedly press **I**, you will continue to step through successive locations.

Notice that, as you step through each location, the row of 8 lights in the address box changes. These lights display the *address* of the "mailbox." To view the *contents* of this mailbox, look at the row of 4 lights directly above the toggle switches. This shows the value stored at the current location. If you were to move the rotary switch pointer to A, you would see the contents of the A register. To examine the B register, point the switch to the letter B. Now, move the pointer to the letters H or L at left. These access the "high nibble" (the first or left-most 4 bits) and the "low nibble" (the last or right-most 4 bits) in the 8-bit address.

Entering Data

The next step is to "fill" these locations so that the processor has something to process. With the rotary switch in the M position, try toggling the switches in the switch box. Nothing happens? Don't worry; turn some of these switches "up" and then press **L**, for **L**oad. Now you have something. Any switch that is *on* has a corresponding light glowing just above it.

You have just entered your first "data" into the *NanoProcessor*. Now move the rotary switch to the H position and try the same exercise. This time, when you press **L**, lights not only come on in the "contents" box, but the same pattern of lights appears in the high (left-most) nibble of the address box. Moving the rotary switch to L (for Low nibble) and loading a value affects the low nibble (right-most) half of the 8-bit address in the same way. Once you have thus designated a full 8-bit address, move the pointer to the M position again to view the contents of *that same address*. By doing this, you have, in effect, moved to this address location, and can enter data there.

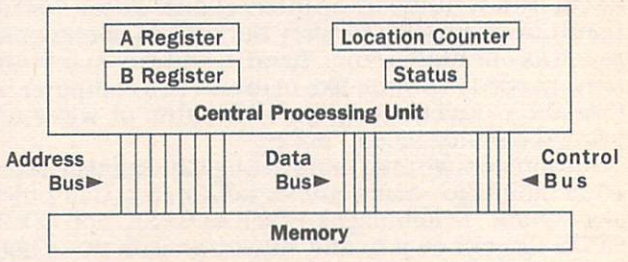
If you next move the rotary switch pointer to the A or B position and try to enter data, you will not be able to—because whatever goes in or out of these registers has to do so while the *NanoProcessor* is running instructions encoded into memory. You will also notice a small Output light (labeled "Out") at the upper left of the screen. We will explain the use of this in the *NanoAssembler* program next issue.

Your next job is to enter your first machine-language program on the *NanoProcessor*.

Programming The Machine

A CPU executes commands *sequentially*. As it runs a program, it steps through this sequence in much the same way you "incremented" through each memory location. However, the program may instruct the CPU to take other paths—"branching" to many different locations before completing its task. You are able to program this processor by entering three different kinds of data: 1) encoded commands; 2) pure numbers; and 3) addresses. As with any program, it is the *logic* of this sequence that determines what the processor will do.

Figure 2
Simplified Block Diagram of the NanoProcessor



NanoProcessor understands 16 different commands—its "instruction set." Although initially expressed in one nibble, some commands require additional memory locations to hold the data necessary to execute the command. Figure 3 lists these 16 commands, showing each corresponding binary code; how many nibbles in a program the instruction requires; its "mnemonic"; which (if any) flags in the status register the instruction affects; and a brief explanation of the command function. As you develop more complicated programs, you will have to understand and use more of these commands. But, for now, try a very short routine—one that simply adds two small numbers together.

Figure 3: Instructions Set

Dec.	Binary	Nibbles per instr.	Mnemonic	Flags* affected C Z	Function
0	%0000	1	ADD	Y Y	Add the contents of B register to the contents of A register—result in A.
1	%0001	2	LDA #	N Y	Load A with number following instruction.
2	%0010	3	LDA *addr*	N Y	Load A with number at location specified by *addr*.
3	%0011	3	STA *addr*	N N	Store the contents of A at location specified by *addr*.
4	%0100	1	TAB	N N	Transfer contents of A to B.
5	%0101	1	TBA	N Y	Transfer contents of B to A.
6	%0110	1	RRC	Y Y	Rotate A right one bit through carry.
7	%0111	1	RLC	Y Y	Rotate A left one bit through carry.
8	%1000	1	AND	Y Y	Logically AND A and B—Result in A.
9	%1001	1	OR	Y Y	Logically OR A and B—Result in A.
10	%1010	1	XOR	Y Y	Logically XOR A and B—Result in A.
11	%1011	3	BZ *addr*	N N	Branch to *addr* if Zero flag is set.
12	%1100	3	BNZ *addr*	N N	Branch to *addr* if Zero flag is not set.
13	%1101	3	BCS *addr*	N N	Branch to *addr* if Carry flag is set.
14	%1110	3	BCC *addr*	N N	Branch to *addr* if Carry flag is not set.
15	%1111	3	JMP *addr*	N N	Branch to *addr* unconditionally.

**Flags affected refers to whether or not the instruction has any effect on the flags in the status register. The C column stands for the Carry flag (did the operation result in a carry being generated?), and the Z stands for the Zero flag (did the operation result in a zero?). A Y appears in the column if the flag is affected by the instruction. An N indicates the flag is not changed by the instruction.*

Sample Program 1

Addr	Code	Mnemonic	Remark
0	%0001	LDA #3	;Get first number
1	%0011		
2	%0100	TAB	;Move to B
3	%0001	LDA #7	;Get second number
4	%0111		
5	%0000	ADD	;Figure sum
6	%1111	JMP 6	;Jump self to stop
7	%0110		
8	%0000		

Sample Program 2

Addr	Code	Mnemonic	Remark
0	%0010	LDA 240	;Get first number
1	%0000		
2	%1111		
3	%0100	TAB	;Move to B
4	%0010	LDA 241	;Get second number
5	%0001		
6	%1111		
7	%0000	ADD	;Figure sum
8	%0011	STA 248	;Put low nibble in memory
9	%1000		
10	%1111		
11	%1110	BCC 19	;Only one nibble answer
12	%0011		
13	%0001		
14	%0001	LDA #1	
15	%0001		
16	%1111	JMP 21	;All done
17	%0101		
18	%0001		
19	%0001	LDA #0	;Zero A
20	%0000		
21	%0011	STA 249	;Put high nibble in memory
22	%1001		
23	%1111		
24	%1111	JMP 24	;Jump self to terminate
25	%1000		
26	%0001		

Sample Program 3

Addr	Code	Mnemonic
0	%0001	LDA #2
1	%0010	
2	%0100	TAB
3	%1000	AND
4	%0110	RRC
5	%0011	STA 254
6	%1110	
7	%1111	
8	%0000	ADD
9	%0011	STA 254
10	%1110	
11	%1111	
12	%0000	ADD
13	%0011	STA 254
14	%1110	
15	%1111	
16	%0001	LDA #6
17	%0110	
18	%0011	STA 254
19	%1110	
20	%1111	
21	%0000	ADD
22	%0011	STA 254
23	%1110	
24	%1111	
25	%0000	ADD
26	%0011	STA 254
27	%1110	
28	%1111	
29	%0000	ADD
30	%0011	STA 254
31	%1110	
32	%1111	
33	%0001	LDA #13
34	%1101	
35	%0011	STA 254
36	%1110	
37	%1111	
38	%1111	JMP 38
39	%0110	
40	%0010	

Roundabout Addition

Sample Program 1 will add the numbers 7 and 3, and the answer will end up in the Accumulator. If you haven't already, turn on the power by pressing **P**. Now, press **B** for **B**egin, and confirm that the rotary is pointing at M (Memory). Now "key-in" this program with the following procedure:

1. Toggle the switches to the on and off positions corresponding to the bits of the number identified as Code in the program—up (or on) for 1, and down (or off) for 0. Notice that each binary code is preceded by a % (percent) sign to make it easy to distinguish binary numbers from decimal quantities (See "Numbers To Bits And Back" for details).
2. Check that the address indicated by the location counter is the correct one for that Code, and then Press **L** for **L**oad.
3. Press **I** for **I**ncrement. This will take you to the next address.
4. Repeat steps 1 through 3, loading the correct nibble into each address, and move on to the next set until you've loaded all the nibbles in the proper order.
5. Once you have completed loading the program, press **B** again to return to address 0. Then step through each memory location with the **I** key to be certain the program is entered properly.
6. Now press **B** for **B**egin once more, then **R** for **R**un. Note that you may **H**alt the program at any time (by pressing **H**) and continue again by pressing **R**.

Let's go over Sample Program 1 step-by-step to see exactly what it does when **L**oaded and **R**un. First it uses the "LoaD Accumulator immediate" instruction (abbreviated LDA #) to load the number stored at the address immediately following the instruction code (address 1) into the Accumulator. This number (in this case a %0011 or decimal 3) is one of the two to be added. At address 2 is an instruction to Transfer the number from the Accumulator into register B (TAB). Address

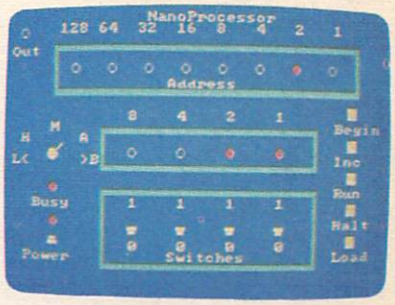


Photo 1: This shows the contents of the A register in the initial step of Sample Program 1. First, the program moves one number (3 or %0011) of an addition problem into A.

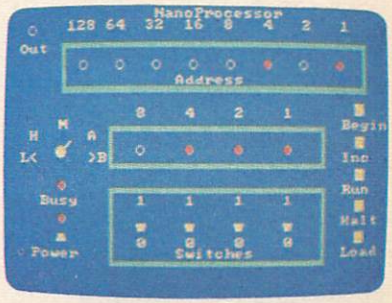


Photo 2: Next, after the first number moves to the B register, the second number (7 or %0111) is loaded in A.

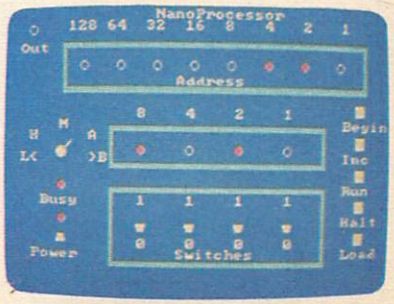


Photo 3. The A register now shows the result (10 or %1010) after the contents of A and B have been added together.

3 contains another LDA# instruction to Load a %0111 (7 decimal) from address 4 into register A. The instruction at address 5 actually ADDs the number in register B to the number in A, and places the answer in A. Address 6 contains a JuMP instruction (JMP *addr*), that tells the machine to jump to the address specified at the next two memory locations—7 and 8. All addresses are two nibbles, and the *NanoProcessor* follows a procedure standard to many microprocessors where the low nibble of the address is in the next location (7 in this case) and the high nibble in the following one (8). We call this a "jump self" because we specify address 6 (%0000 0110) as the place to jump to.

When you **R**un this program, the "busy light" remains on and both rows of lights flash different patterns as the CPU steps through the program. The *Nanoprocessor* has been made to **R**un slowly so that you can track each instruction as it is executing. When the program "hangs-up" at location 6, press **H** (for **H**alt) to make the busy light go off. Now turn the rotary switch to point at A. Here you find the answer to the addition problem: %1010 or 10 decimal. Keep the pointer in this position and run the program again, after pressing **B**egin. Watch the A register change values—first 3 (%0011), then 7 (%0111), then the answer, 10 (%1010). Photos 1 through 3 show this sequence.

CONTROL CAPSULE
NanoProcessor

Key	Function
B	Set address to zero.
I	Increment address by 1.
R	Run program.
H	Halt program.
L	Load location.
<	Move rotary switch counter-clockwise.
>	Move rotary switch clockwise.
P	Toggle Power switch.
E	End program (only when Power is off)
1-4	Toggle panel switch 1 = left-most bit, 4 = right-most bit.

CONTROL CAPSULE
NanoProcessor

Key	Function
OPTION	Save file.
SELECT	Load file.

CONTROL CAPSULE
NanoProcessor

Key	Function
F1	Save file.
F3	Load file.

CONTROL CAPSULE
NanoProcessor

Key	Function
CONTROL W	Save file.
CONTROL Q	Load file.

CONTROL CAPSULE
NanoProcessor

Key	Function
FN 6	Save file.
FN 7	Load file.

CONTROL CAPSULE
NanoProcessor

Key	Function
FCTN 6	Save file.
FCTN 8	Load file.

Moving On

In Sample Program 1, the machine added two numbers and got an answer that it could express in one 4-bit nibble. But, what if this answer had been larger than one 4-bit nibble—say, a number like 23 (%0001 0111)? Fifteen (%1111) is the largest number that one nibble can express. When a processor adds two numbers together whose answer is bigger than its registers can hold, the answer "overflows" the register. When this happens in *NanoProcessor*, a "carry flag" is set to 1 in a special Status register of the CPU. (This register is not directly accessible to the user.) The program has to contain commands that recognize the condition of this flag (either 1 when an overflow has occurred, or 0 when there is no overflow) and take appropriate action. You can determine which instructions cause changes in the carry flag by studying the C column (under "Flags affected") of Figure 3. If there is a Y in the C column, the instruction will affect the carry flag —i.e., set it to 1 if an overflow occurs, or reset it to 0 if no overflow occurs.

Sample Program 2 adds the numbers 11 (%1011) and 12 (%1100) to arrive at 23 (%0001 0111). Not only does the program have to check the carry flag, but because the answer doesn't fit in one register, it has to place the answer someplace else. The solution is to designate certain memory locations as data areas—two for input and two for output. Program 2 fetches the two numbers to be added from memory locations 240 (%1111 0000) and 241 (%1111 0001). These addresses are input areas. This means that before you **R**un the program, you must manually **L**oad the numbers to be added at these locations—place 11 at address 240, and 12 at address 241.

Similarly, the output area is at locations 248 (%1111 1000) and 249 (%1111 1001). The low nibble of the answer (%0111 in our example above) appears at 248, and the high nibble (%0001) at address 249.

This program also handles the overflow condition described above. If the answer does overflow a nibble, the program places a 1 in the accumulator and stores it as the answer's high nibble. If, however, the answer is less than 15 (and fits into one nibble), the program branches to another address, where it loads a 0 into A and stores that instead. This introduces one of 4 "conditional jump commands," which we will explore more fully in next issue's companion "utility," *NanoAssembler*.

Program 3 is a "mystery program" that actually accesses the "sound chip" we've built into the *NanoProcessor*. Watch next issue for an explanation of how this program works. Or perhaps, in the meantime, you will learn enough by playing with *NanoProcessor* to figure this one out yourself. The best way to learn the details of operating the the *NanoProcessor* is to use it and experiment by creating your own machine-language programs.

Saving and Loading

With *NanoProcessor*, you can Save and Load the entire 256 memory locations (%0000 0000 through %1111 1111) to disk (and/or tape on The C-64, Atari, and TI-99/4A). Use the Save command listed in your Control Capsule and type in a file name in response to the prompt. To Load, use the Load command and type in the name of the file you wish to load.

HCM Glossary terms: CPU, bus, machine language, binary numbers, Random Access Memory (RAM), byte, address, nibble, location counter, accumulator, register, instruction set, mnemonic, branch, jump, conditional jump, status register, zero flag, carry flag, overflow, weight (of bits).

HCM

For your key-in listings, see HCM PROGRAM LISTINGS Contents.

Electronic Typewriter

by Randy Thompson
HCM Staff

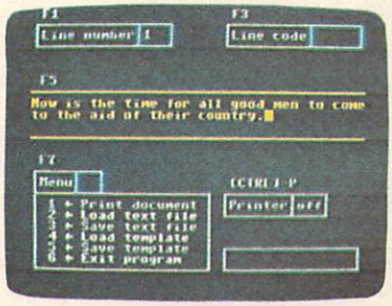
Turn your computer into an electronic typewriter with this super-easy-to-use "line processor."

NOTE TO TI-99/4A USERS:
Because the 99/4A has no keyboard buffer, *Electronic Typewriter* would work too slowly on that machine. However, the powerful **ACCEPT AT** statement in TI Extended BASIC allows us to bring you *TI CARD-TRIX* on page 21, another electronic productivity tool.

Using a full-blown word processor can be both frustrating and intimidating. When all you want to write is a simple letter or memo, most word processors are even more cumbersome than a typewriter. What the software industry needs is a typewriter simulator—something that enables you to just sit down and quickly put some words onto paper without limiting the computer's ability to remember text and correct information before it's output. We created the *Electronic Typewriter* to fulfill this need.

The "Line Processor"

The *Electronic Typewriter* is a combination computer/typewriter. It might be referred to as a "line processor." Instead of requiring that an entire document be entered, edited, and analyzed before printing, this program lets you enter and edit one line at a time. Once you're satisfied, simply press [RETURN] and that line immediately will be sent to the printer. Or, if you prefer, the line-print feature can be shut off so that you can first create all of your text and then print the entire document at a later time.



The main text-input screen and option boxes from the Commodore version.

Another useful feature included in this program is line coding. For each line of text, you may enter certain codes that center, indent, and print right-aligned text. You can save any combination of line codes to provide preset templates that will thereafter automatically format your text.

When you RUN this program, you are first asked to set the margins and line spacing. Margin settings are limited to 39 characters from both the left and right edge of the paper. Line spacing can be either single or double. These settings default to five-character-wide margins and double spacing. Once these settings are entered, the main editing screen will appear.

The editing screen has six basic parts: printer status, text entry, line number, line code, menu, and prompt window. A simple keypress will bring you to any of these six fields. (Refer to the Control Capsule for your computer to find the appropriate keys.)

Printer Access

The printer status box simply indicates whether the program is ready to access the printer. If the printer access is on, then every time you press [RETURN] from the text-entry section of the screen, your text will be sent to the printer.

Text Entry

Displaying one line at a time, this box on the screen is your window into the text file. All text is entered inside this window. Here all of the editing features of your computer (e.g., insert, delete, left and right cursor movement) are available. When you are satisfied with a line, press [RETURN] to advance to the next line. This program allows you to store up to 54 lines of text in memory.

Line Number

The line-number box informs you of the physical page line that you are working on. This number can range between 1 and 54. To change the current line number, move to the line-number field and enter the desired value. You may increment the line number from the text-entry box by pressing [RETURN], but to move back a line, or to jump ahead several lines, you will need to enter into the line-number section of the editing screen.

Line Code

Line codes allow you to optionally format any line of text. These codes are represented by the letters b, c, d, and e.

A b code is used to print a blank line. Any text that begins with a line code of b will never be printed. The c line code is used to center a line of text. This is great

for titles or page headers. To indent your text, use a code of d. Unlike the other line codes, the d code requires an added parameter—the number of spaces to indent. When you first enter a code of d, the computer moves the cursor over to the right and waits for you to input an indent value. If you do not enter anything, then the indent value defaults to zero. The line code e aligns text on the right edge (right justified, left ragged).

"What the software industry needs is a typewriter simulator —something that enables you to just sit down and quickly put some words onto paper . . ."

Line codes can be saved separately from the text. This is a very useful feature when you want to create a template for commonly used page formats. For instance, a letter template might have right-alignment codes (e) for the first three lines to position your name and address. The next line may have a blank line code, and so on. Once created, a template file can be used repeatedly. So, when you want to type a letter, just load your letter template and let the computer format your text automatically.

The Menu

The menu box allows you to choose from six options:

1) Print document
2) Load text file
3) Save text file
4) Load template
5) Save template
6) Exit program

Print Document

If you wish to print your text all at once, instead of line by line, then choose this option. Once selected, you will be asked to enter three items via the prompt window: line spacing, first line, and last line. The line spacing can be either single or double. The first line is the line number of the first line of text that you wish to print. The last line is the line number of the last line that you wish to print. When everything has been entered, the computer will prompt you to position the paper and press [RETURN]. Your text will now be sent to the printer regardless of the status of the "printer status" window.

Load/Save Text File

With these two options you can save and load text files. When you choose to save a text file, you are given the option to save your line codes with your text. This comes in handy when creating templates that require standard text such as "Dear," and "Respectfully."

Load/Save Template

These selections allow you to save and load the line codes that you enter. A template file consists of all 54 line codes that you input. Once you've created a template that you like and will use often, you should save it. By creating a library of such templates, you can save yourself a lot of time in formatting future documents such as letters, reports, and memorandums. Template files can be loaded and saved at any time, and do not destroy any text file that is present.

Exit Program

This option will "turn off" your *Electronic Typewriter*.

HCM Glossary terms: field, parameter, template.

For your key-in listings, see HCM PROGRAM LISTINGS Contents.

HCM

CONTROL CAPSULE

Electronic Typewriter

KEY	FUNCTION
CONTROL Q	Change line number.
CONTROL W	Change line code.
CONTROL T	Text entry.
CONTROL Z	Menu box.
CONTROL P	Toggle printer status.
CONTROL X	Erase data.
Text editing:	
CONTROL I	Insert letter.
CONTROL D	Delete character.
←	Cursor left.
→	Cursor right.
RETURN	Advance to next line.

CONTROL CAPSULE

Electronic Typewriter

KEY	FUNCTION
F1	Change line number.
F3	Change line code.
F5	Text entry.
F7	Menu box.
CTRL P	Toggle printer status.
F2	Erase data.
Text editing:	
SHIFT INST	Insert letter.
DEL	Delete (backspace).
Crsr Left	Cursor left.
Crsr Right	Cursor right.
RETURN	Advance to next line.

CONTROL CAPSULE

Electronic Typewriter

KEY	FUNCTION
CONTROL L	Change line number.
CONTROL C	Change line code.
CONTROL T	Text entry.
CONTROL M	Menu box.
CONTROL P	Toggle printer status.
CONTROL E	Erase data.
Text editing:	
CONTROL Insert	Insert letter.
CONTROL Backspace	Delete character.
Backspace	Backspace.
CONTROL ←	Cursor left.
CONTROL →	Cursor right.
RETURN	Advance to next line.

CONTROL CAPSULE

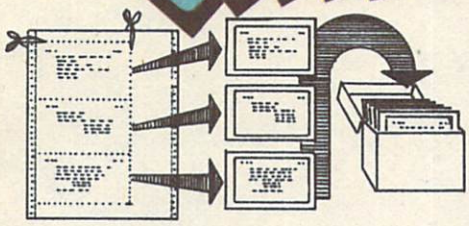
Electronic Typewriter

KEY	FUNCTION
F1	Change line number.
F2	Change line code.
F3	Text entry.
F4	Menu box.
CTRL P	Toggle printer status.
F10	Erase data.
Text editing:	
Insert	Toggle insert mode.
DEL	Delete character.
Backspace	Backspace.
←	Cursor left.
→	Cursor right.
RETURN	Advance to next line.

TI Card-Trix

Part 1

by Randy Thompson



Here's a real trick you can do with TI Card-Trix:

1. After you've created your electronic file cards on the computer, print them out on printer stock; then cut out each file as shown.
2. Tape or paste each file onto regular file card stock.
3. Store all cards in a file box.
4. You now can have *many* information "windows" opened up *simultaneously* on your *real* desk top for ease of use—a feat not practical even on a computer monitor.

In the next issue, Part 2 of this program will allow you to shuffle your file cards on the computer between many different cross-referenced file folders.

TI Card-Trix allows you to organize thoughts, book references, personal inventory—anything—on 3" by 5" cards. Such features as copy, search, and sort make creating the "cards" easy. And after editing a set (folder) of cards, you can print them out and/or save them to disk—your filebox for several folders of cards.

TI Card-Trix has seven main menu options:

1) Edit Cards
2) Search
3) Sort
4) Print
5) Save
6) Load
7) Exit Program

Edit Cards

Edit Cards immediately calls up the editing screen, which divides into four main fields: Index, Subject, Text, and Card number (represented by the # sign). Information for the Index can be both numbers and letters, but it is limited to 8 characters. If you wish to date your cards, or give them some other kind of identifying character sequence, the Index field is an ideal location. The Subject field can hold up to 28 characters, and may serve as a title for each of your cards. Most of your information will be placed in the Text field, where you have 9 lines, 28 characters wide. To move from one line to another within the text field, use [FCTN] E (↑) and [FCTN] X (↓).

The # field tells you on which of a maximum of 25 cards you are working. Every time you create a new card, the computer assigns it a number in this field. By changing this number, you change which card is currently on screen. For instance, changing the # field to a 15 is like putting the current card back into the folder and pulling out the 15th card so that you may work on it. You can edit cards in any order.

From the editing screen, you can Erase, Copy, and Paste cards. To flip through the cards one at a time, use the Forward and Backward functions. To access any of these features, simply enter the first character of the desired function at the CHOOSE ONE: prompt. For instance, to Copy the card that you are currently working on, enter a C. That card will now be copied into the hold buffer—a temporary storage location. Now, to Paste what you copied into the hold buffer onto another card, advance to the desired card and select P. The current card will now become identical to the card previously copied.

Search

This option allows you to search through your cards for any sequence of letters and/or numbers. You can search by Index, Subject, or Text. If the search yields a card with a corresponding character sequence, you are given the option to edit it, continue the search, or quit.

Sort

Here, you can have the computer sort your cards alphabetically by Index, Subject, or Text. Thus, you may save your folder of cards in a variety of different sequences. One folder might have cards sorted by Index, while another may be left in the order that was entered.

Print

To print out your cards, select this option. You can print out as many cards as you wish. When you enter the Print option, the computer asks you for the first and last card number of the cards to be printed. After you enter these numbers, a prompt will appear asking you to position the paper and press [ENTER]. Once you do this, the specified cards will print out.

Save

Here you can save your folder of cards. Saving different folders under unique names makes the cards easy to access and search through. This program stores in memory up to 25 cards per folder (or 15 completely filled cards). However, with memory expansion, this number can be raised. To do so, two numbers in line 210 have to be changed. To set the number of cards to 50, for example, change MX = 25 to MX = 50 and DIM C$(3,25) to DIM C$(3,50).

Load

This option loads a previously saved card folder.

Exit Program

Selecting this option allows you to quit the program. A prompt will ask whether you really want to exit. If you indicate No, the program will resume. Before indicating Yes, you should first save your folder of cards. All card data will be lost if you end the program before saving.

CONTROL CAPSULE

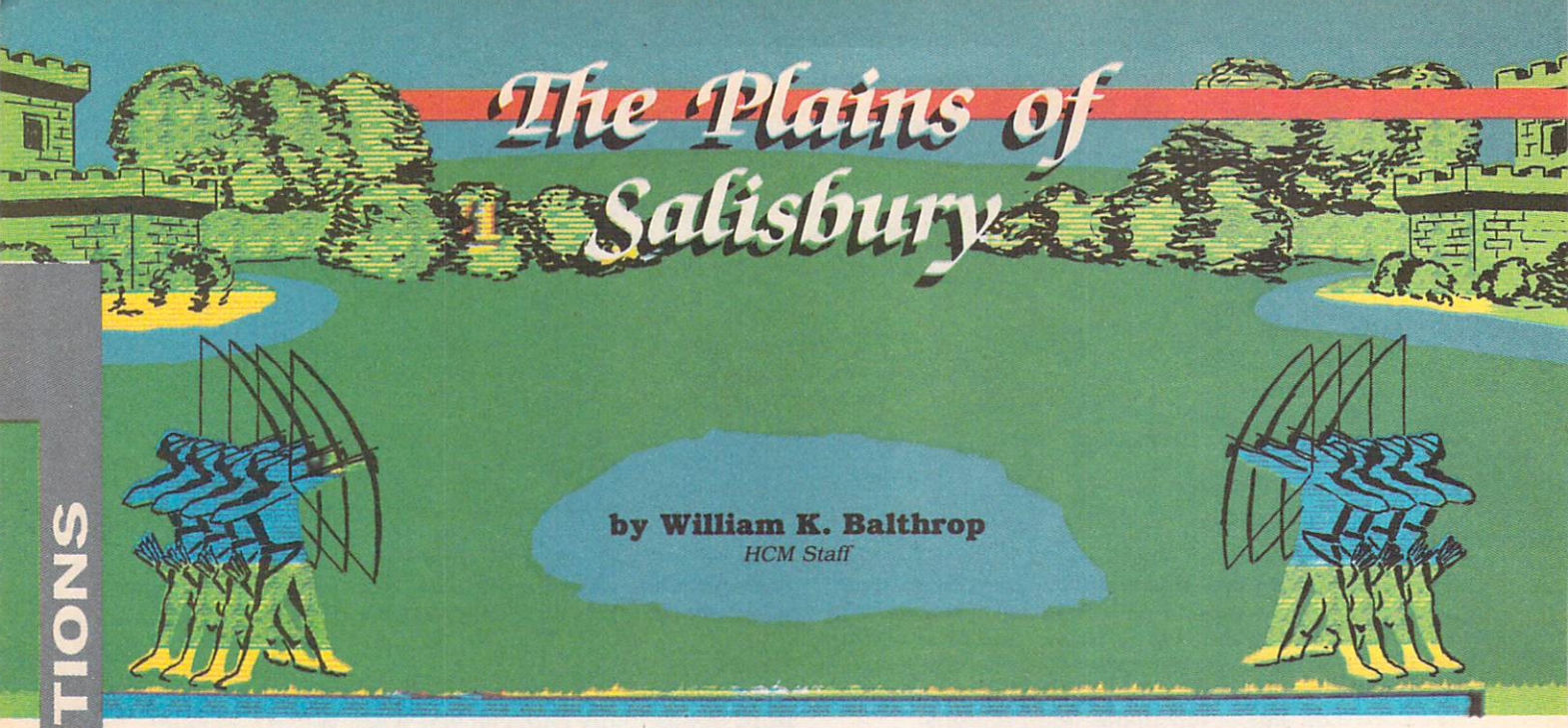
TI Card-Trix

KEY	FUNCTION
Edit Mode:	
I	Edit Index field.
S	Edit Subject field.
T	Edit Text field.
#	Move to another card.
E	Erase current card.
C	Copy current card to buffer.
P	Paste buffer contents to current card.
F	Skip Forward to next card.
B	Go Back to previous card.
R	Return to main menu.
Edit Text field:	
[FCTN] E	Move up a line.
[FCTN] X	Move down a line.

For your key-in listings, see HCM PROGRAM LISTINGS Contents.

HCM

SOFTWARE INSTRUCTIONS



The Plains of Salisbury

by William K. Balthrop
HCM Staff

All is not well in Camelot. The kingdom is in chaos as King Arthur prepares his gallant knights for his final battle with his arch-enemy (and son) Mordred.

Arthur sits alone at his dimly lit table, a large map spread before him. As he struggles to concentrate on tomorrow's coming battle at Salisbury, time weighs heavily on his mind. So many years have passed since he first drew the sword from the stone. At dawn, he will make a last valiant stand against his evil son. With Merlin gone, who will come to Arthur's aid? Will it be someone from the distant future? Could it be you?

This program is an exciting simulation of King Arthur's last battle on *The Plains of Salisbury*. The game requires two players, each controlling an army of gallant knights. Players move the knights and engage in combat until one player defeats all of the opposing player's troops.

The Playing Screens

Three maps of terrain comprise the battleground for this mini-war. Each map lies adjacent to the others. (Moving off the right edge of the first map brings you to the left edge of the middle map, and so on.) You may not move off the left edge of the first map or the right edge of the last map—or off the top or bottom of any map.

The program will ask you to designate a map layout. You can enter any ordered combination of the three maps to indicate the layout sequence: e.g., 123, 321, 213, 232, etc.

Each map contains 6 types of terrain that affect movement and troop defense. You may move your knights into all terrains except water. Each type of terrain has a different movement/defense factor—the higher the number, the more strength it takes for your knight to travel. The 6 terrain types are:

TERRAIN	MOVEMENT/DEFENSE UNITS
Roads	1
Open grasslands	2
Forest	3
Buildings	4
Forts	5
Water	No movement allowed.

Obviously, roads offer the least resistance to movement and the least protection from attack. On the other hand, a fort is the most difficult to move through, but it offers the highest level of protection. Forts also possess an endless supply of arrows. Each knight, however, is capable of carrying no more than 4 arrows at a time. When these are used up, the knight must return to a fort to get more before he can participate in the combat phase. A knight who remains in a fort will always have a supply of 4 arrows.

Movement Phase

Two phases make up each player's turn: movement and combat. During the movement phase, a player is given an opportunity to move his or her knights, starting with knight number 1 and continuing through knight number 6.

Every knight can travel up to 9 movement units each turn. You may move your knights either left, right, up, or down by using the 4 keys indicated in your machine's Control Capsule. If you don't wish to move, or if you wish to stop moving a knight before all of his movement units are gone, then press [ENTER] or [RETURN]. This will start the next knight's turn; or, if he's the last knight, this will start the combat phase.

The number of movement units expended when entering a terrain type can be seen in Chart 1. If you stick to the roads, you could move your knight up to 9 squares at a time (a square is one character on the screen). Traveling through open grasslands, knights can move only 4 squares in one turn. Moving in the open grasslands requires 2 movement units for each of the 4 squares; so if your knight has only 1 movement unit left, he cannot continue. At this point, press [ENTER] or [RETURN] to begin the next knight's turn. When a knight's movement factor is exhausted, you will be prompted to move the next knight.

Every time a knight moves, he expends strength (.1 strength units per movement unit). Each knight starts the game with 9.9 units of strength, shown simply as a 9 on the screen (a strength of .9 will show up as 0). If a knight's strength level drops below zero for any reason, the knight cannot do battle. Each knight will automatically receive .5 units of strength every turn. If a knight stays away from battle, and only moves 5 movement units per turn, then his strength level will stay even. Not moving will increase his strength by .5 every turn. Marching at full speed (using all 9 movement units) will drain .4 strength units per turn.

Hand-to-Hand

A knight entering a square occupied by an enemy knight will automatically initiate hand-to-hand combat. This is a fight to the death, eliminating the losing knight from play.

Once hand-to-hand combat begins, your knight's strength level determines how much strength you can drain from the enemy; thus, the stronger your knight, the better his odds are of winning. If the strength for both knights drops below 0 at the same time, both knights will be eliminated.

If a knight has engaged in hand-to-hand combat and wins, he may not move until his next turn. The losing knight, obviously, may never move again.

After a knight has moved into a new square, the program updates and displays the number of movement units remaining and the knight's strength level. If a knight runs out of strength while moving, he is out of the game. The vanquished knight, however, will remain on the map until the map has been updated.

Combat Phase

After completing the movement phase, the combat phase starts. If any of your knights are adjacent to an enemy knight (horizontally or vertically, *not diagonally*), you can now attack that knight with bow and arrow. The enemy will not be able to fire back at you until your turn is over, so you have nothing to lose except one arrow. Every knight may fire one arrow per turn in the combat phase.

Select the knight that is to fire an arrow by pressing the number corresponding to that knight. Then indicate the direction of fire by pressing the key indicated in the Control Capsule for your machine. After a knight fires an arrow, that knight may not attack again until your next turn. Hitting an enemy lowers strength by a factor based on a random number and the attacking knight's strength level.

Every knight begins with only 4 arrows. If a knight uses all 4 arrows without replenishing them, that knight will not be allowed to initiate combat. A knight can replenish his supply of arrows at one of the forts.

You can terminate a knight's combat phase at any time by pressing [ENTER] or [RETURN].

Save Exit Return

Pressing the appropriate Exit/Save option key (designated in your system's Control Capsule) calls up a short submenu. When you reach this menu, you can either save your game to disk (or cassette on the TI-99/4A, Commodore, and Atari machines), or exit without saving. After saving, you can also quit the game or return to where you left off.

When you exit the game you are informed of the winner at that time. If you've saved a game where each player has at least one knight left, the game can be continued later—but, as you exit, the computer tells you the winner *as if the game were over*. The winner is determined by the following scoring rules: 50 points for each enemy knight defeated, and 10 points for each arrow that hits an enemy.

Load a Game

After the title screen, the program will ask whether you want to load an old game. If you reply with a `Y` for Yes, then it will ask for the file name. The old game will load and commence where it left off at the time it was saved.

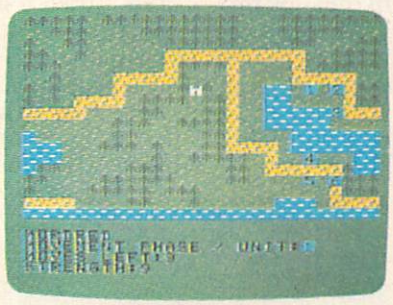
HCM

For your key-in listings, see HCM PROGRAM LISTINGS Contents.

CONTROL CAPSULE

The Plains of Salisbury

KEY	FUNCTION
Movement Phase	
I,J,K,M	Move up, left, right, down.
SPACEBAR	Enable screen scroll.
1	Screen 1.
2	Screen 2.
3	Screen 3.
RETURN	Terminate movement.
ESC	Exit/Save option.
Combat Phase	
1-6	Select unit to fire.
I,J,K,M	Fire up, left, right, down.
RETURN	Terminate combat.



This screen, from the TI-99/4A version shows a typical arrangement of knights moving across the landscape

CONTROL CAPSULE

The Plains of Salisbury

KEY	FUNCTION
Movement Phase	
Arrow keys	Move up, left, right, down.
A	Screen 1.
B	Screen 2.
C	Screen 3.
RETURN	Terminate movement.
ESC	Exit/Save option.
Combat Phase	
1-6	Select unit to fire.
Arrow keys	Fire up, left, right, down.
RETURN	Terminate combat.

CONTROL CAPSULE

The Plains of Salisbury

KEY	FUNCTION
Movement Phase	
Cursor keys	Move up, left, right, down.
F1	Screen 1.
F3	Screen 2.
F5	Screen 3.
RETURN	Terminate movement.
F7	Exit/Save option.
Combat Phase	
1-6	Select unit to fire.
Cursor keys	Fire up, left, right, down.
RETURN	Terminate combat.

CONTROL CAPSULE

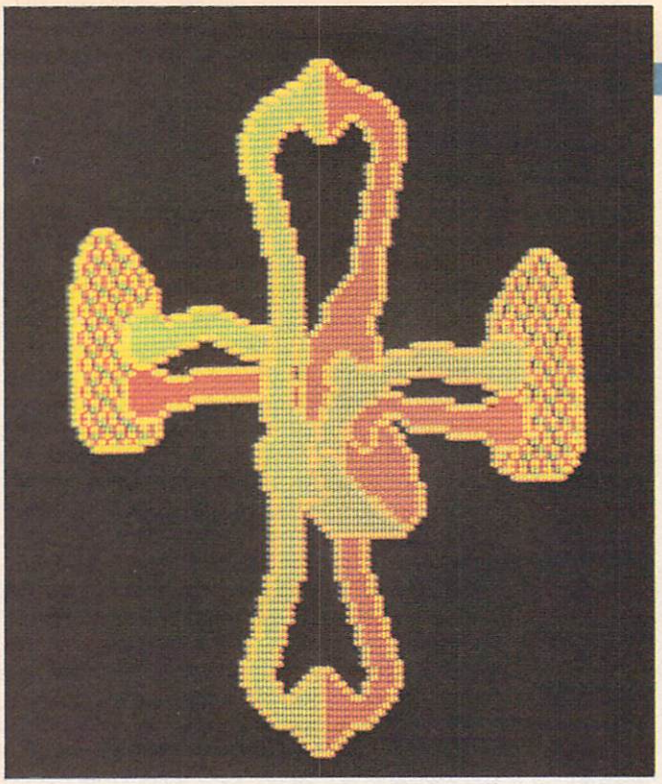
The Plains of Salisbury

KEY	FUNCTION
Movement Phase	
Cursor keys	Move up, left, right, down.
Fn 1	Screen 1.
Fn 2	Screen 2.
Fn 3	Screen 3.
ENTER	Terminate movement.
ESC	Exit/Save option.
Combat Phase	
1-6	Select unit to fire.
Cursor keys	Fire up, left, right, down.
ENTER	Terminate combat.

CONTROL CAPSULE

The Plains of Salisbury

KEY	FUNCTION
Movement Phase	
E,S,D,X	Cursor up, left, right, down.
CTRL 1	Screen 1.
CTRL 2	Screen 2.
CTRL 3	Screen 3.
ENTER	Terminate movement.
F9	Exit/Save option.
Combat Phase	
1-6	Select unit to fire.
E,S,D,X	Fire up, left, right, down.
ENTER	Terminate combat.



VITAL SIGNS

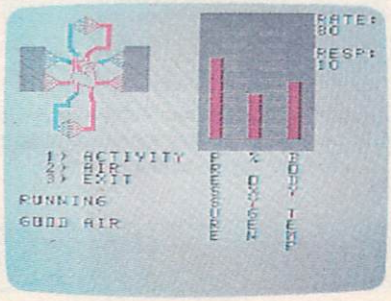
by William K. Balthrop
HCM Staff

Do you take your cardiovascular system for granted? With this simulation, you'll start paying attention to every breath you take and every beat of your heart.

An ambulance flashes a thick, red light. Another heart patient, with lungs nearly exhausted, exits the dense smog for the cool, conditioned air of the emergency room. Here, Intensive Care attempts to rescue this victim of long neglect—like restarting an engine that is terribly out of tune. But no amount of repair can make up for a lifetime's lack of care. For this person, it may be too late to start heeding those *Vital Signs*.

The heart and lungs play an extremely vital role in human existence: they supply life-giving oxygen to body tissues. If these two organs do not perform their job properly, you may end up looking at the clouds from the other side. Sometimes, to appreciate just how important this system is, we have to stop and attend to how it works.

Vital Signs is a program that provides a simplified simulation of the circulatory system. The many processes that occur in a living human body are far too complex for a computer program to handle (and remain small enough to publish in one issue of this magazine). For this reason, we have concentrated on a few key biological factors.



This screen photo from the C-64 version of Vital Signs shows the program in a Running mode—in which the player must adjust Heart and Respiration rates to compensate for the higher Activity level.

The Heart

The heart is responsible for pumping blood through the body. The blood carries—among other things—life-giving oxygen. The heart is really just a very complex pump. If we were to follow the path of a single blood cell through the body, the trip might go like this:

The blood cell's journey starts in the Right Atrium—one of four chambers in the heart. From here, the blood cell moves to the Right Ventricle (the second chamber), which then pumps the cell out of the heart and into a network of tiny capillaries in the lungs.

In the lungs, the blood cell picks up new oxygen and passes its load of carbon dioxide back into the small air sacs surrounded by the network of capillaries. From the lungs, the cell returns to the heart again, this time entering the Left Atrium (the third chamber). The Left Atrium sends the cell into the Left Ventricle (the fourth and last chamber), which is responsible for pumping fresh oxygenated blood cells to all organs and tissues. From here, the cell either goes through the upper circulatory system (arms and head), or through the lower circulatory system (abdomen and legs). After its journey through the body's tissues, the blood cell returns to the heart's Right Atrium for another trip.

If you listen to your heart, you will hear a short, hard "lub," then a long, soft "dub." The first sound is the heart contracting, pushing the blood out to the lungs and the rest of the body. The "dub," or second beat, is the heart relaxing, filling with blood for the next cycle.

The rate at which the heart beats is controlled by a natural pacemaker (or an artificial pacemaker surgically installed in people with heart problems).

In this program, *you* are the pacemaker. You can vary the heart rate from 0 to a maximum of 200 by using four keys on the keyboard. (See the Control Capsule for your machine.) Two keys increase or decrease the heart rate by one, while two other keys increase or decrease the heart rate by five.

The Lungs

The lungs are less complex than the heart, although they perform an equally essential role in the circulatory system. When you breathe in, your lungs fill with air. As blood circulates through the lungs, it picks up oxygen and gives off carbon dioxide, which you exhale.

Two major factors determine how much oxygen is transferred to the blood from the lungs: Respiration Rate and Air Quality (how much oxygen is in the air in proportion to other gases and pollutants).

The Respiration Rate is the speed at which you breathe—the number of breaths per minute. As you

breathe faster, more oxygen is placed in your lungs to be absorbed by the blood. Using two keys on the keyboard, you can increase or decrease the Respiration Rate from 0 to 30 breaths per minute.

Body Temperature

The amount of oxygen that the blood supplies to cell tissue determines the amount of energy available in the cell tissue. As you may know, this process of receiving and expending energy creates heat. This is why our bodies are warm. The more energy we expend, the more heat we generate (although our bodies usually regulate this temperature within a certain range). If the blood does not supply enough oxygen to the body tissues, then we create less heat, sometimes lowering the body's temperature below a tolerable level. If we have too much oxygen in our blood, then the body tries to burn it off, creating more heat and raising the body's temperature. The body automatically regulates its temperature by controlling the heart rate and respiration. In this progam, *you* are in charge of this process.

Body temperature can also be directly affected by the amount of blood flowing through our veins. The blood acts as a coolant, the capillaries in and near our skin as a radiator. The body can be cooled off by an increase in the flow of blood through these capillaries, or warmed up by a decrease in the flow of blood.

When the body starts to overheat due to exertion or a high external temperature, sweat cools the surface of the skin (the radiator for the blood). While the degree of sweating is not a direct factor in this program, it does have a limited influence on body temperature. When you choose either the Running or Swimming activities, the program simulates the heating of the body as energy is expended, and then the cooling of the body as the sweat glands start doing their job. You can see this process on the control-panel bar graph labeled BODY TEMP. The temperature increases for a little while, then it decreases as the sweat glands start working. The temperature graph ranges from 90 degrees to 107 degrees Fahrenheit.

"Sometimes, to appreciate just how important this system is, we have to stop and attend to how it works."

Percentage of Oxygen in the Blood

As mentioned earlier, the heart rate and the quality of the air (amount of oxygen available) control the amount of oxygen in the blood.

Another factor that determines the amount of oxygen in the blood is blood pressure. One method to increase blood *oxygen* is to increase the blood *pressure*. The higher the blood pressure, the more quickly blood flows. Consequently, higher blood pressure causes blood to receive oxygen from the lungs at a quicker rate. This program performs this function automatically. If the oxygen level drops below 50 percent, the blood pressure automatically increases to compensate. If the oxygen level exceeds 50 percent, the blood pressure drops.

If the blood pressure gets too high or too low, you will need to increase or decrease the oxygen level through the heart rate, the respiration level, or both to prevent the blood pressure from reaching more dangerous levels.

The oxygen level in the blood also has an effect on the body's temperature. A *high* oxygen level *increases* temperature, while a *low* oxygen level *decreases* temperature.

Blood Pressure

Blood pressure is the amount of force applied to the blood to push it through veins and arteries. High blood pressure can be quite serious if it persists over prolonged periods, and fatal if it's high enough over a short period. Although low blood pressure is not considered detrimental (it simply means the heart has to work less), extremely low pressure can be a problem.

As mentioned earlier, the oxygen level in the blood can affect blood pressure. The heart rate and the level of activity also affect blood pressure. Strenuous activities such as running or swimming increase blood pressure. Other factors affecting blood pressure—such as tension and disease—are not included in this program.

You can control your blood pressure directly by changing your heart rate, or indirectly by altering your respiration to change your blood oxygen level. The bar graph on the screen depicts a blood pressure range from 75 to 175. This value reflects the *systolic* pressure (or the pressure resulting as the heart contracts). This could be expressed, for example, as120/xx or 120 over xx, where xx is the *diastolic* pressure (the pressure from the expansion phase of the heartbeat). For simplicity, the program displays only the first number.

Using the Simulation

After the title screen, you see a control panel. The control panel is divided into several sections. In the upper-left corner is a graphics representation of the circulatory system. Below that are your Activity, Air, and Exit options. The area below these options is used to display the Activity and Air menus.

The bar graph in the center of the screen displays the body's blood pressure, the percent of oxygen in the blood, and the body's temperature. The height of these bars indicates their current level.

Pressure—Range is 75 to 175 (125 is normal).
% Oxygen—Range is 25% to 75% (50% is normal).
Body temp.—Range is 90 to 107 (98.6 is normal).

To the right of the bar graph are the Heart Rate (Rate) and Respiration Rate (Resp) indicators. These numeric readouts indicate beats-per-minute for the heart and breaths-per-minute for the lungs.

Options

You can select two options from the control panel. With the first option, you can alter the Activity level, thereby changing the demand for oxygen and affecting the body temperature. You can also select an Air Quality, which determines how much oxygen is in the air you breathe.

Activity—Your body's level of activity determines the amount of oxygen it actually uses. As you increase your

activity, your body burns more oxygen. This means that you need to breathe faster to get more oxygen to the lungs, or increase the heart rate to get more oxygen-carrying blood to your tissues, or both. You can select the activity level for the simulation with the Activity option (1) from the on-screen control panel:

A) Sleeping
B) Resting
C) Normal
D) Walking
E) Running
F) Swimming
G) Random

Option G causes the program to randomly change both the Activity level and the Air Quality. Your task is to regulate the Heart Rate and Respiration Rate to maintain a balanced system. If you are not careful, you might encounter a "blood clot," which will send your blood pressure soaring, or "lung cancer," which will reduce the amount of oxygen that your lungs are capable of supplying to the blood. The blood clot will be cured after a random amount of time, while lung cancer will be corrected after 50 cycles (beats of the heart simulator) through a lung transplant.

In *Vital Signs*, it is possible to get (simulated) lung cancer if you are using the Random option from the Activity menu. As established by research, the chance of getting lung cancer increases with a decrease in Air Quality.

"Vital Signs provides a hint of what it would be like if we had to control this process consciously, every minute of our lives."

Air Quality—the amount of oxygen in the air determines the amount of oxygen placed in your lungs with each breath. Four types of Air Quality are available:

A) Good air
B) Smoggy air
C) Smoking a cigarette
D) Smoking a cigarette with smoggy air

You can select the Air Quality with the Air option (2) from the control panel.

Keeping Score

As long as you maintain the system in a healthy condition (no warning lights), your score increases at a rate relative to your level of Activity and the Air Quality. Each healthy beat of the heart can add between 0 and 5 points to the score. The higher the level of Activity, and the worse the Air Quality, the more your score will increase. (When you are sleeping in good air, the score remains constant.) If, however, a warning light flashes in one of the three bar graphs, your score will decrement 40 points with each flash. You receive a final score display at the end of the game.

Staying Alive

Most of the time, if all goes well, our bodies take care of themselves—automatically regulating the entire cardiovascular system. *Vital Signs* provides a hint of what it would be like if we had to control this process *consciously*, every minute of our lives. So take advantage of this "lifelike" simulation to learn more about a system that most healthy people take completely for granted.

CONTROL CAPSULE
Vital Signs

KEY	FUNCTION
E	Increase respiration by 1.
X	Decrease respiration by1.
A	Decrease heart rate by 5.
S	Decrease heart rate by 1.
D	Increase heart rate by 1.
F	Increase heart rate by 5.
1	Select Activity option menu.
2	Select Air Quality menu.
3	Exit the program.

EXERCISE FOR HEALTHY HEART AND LUNGS

Heart disease is still the leading cause of death in the United States—killing almost a million people in 1982 alone. Cancer in its various forms, including lung cancer, comes in second—taking about half the number of lives attributed to heart disease. In many cases, the root causes of heart disease are not clear; but statistical evidence suggests that the risk of disease or death can be much lower for those that follow a low-fat diet and get adequate amounts of exercise.

Although fitness exists in various forms, the type that is important for the heart, lungs, and circulatory system is called *cardiovascular fitness*. Cardiovascular exercise improves the ability of the heart and blood vessels to supply oxygen to the entire body. It also enhances the body's capacity to utilize the oxygen in order to perform the work vital to the proper functioning of all organ systems. Oxygen is essential for the production of energy in every cell in the body. We rely on this energy for body maintenance, growth, and repair. The more oxygen that is supplied to and utilized by your cells, the greater your total work potential and cardiovascular capacity.

Five basic factors must be considered in developing a safe and effective cardiovascular conditioning exercise program: frequency, duration, intensity, type of exercise, and warm-up and cool-down periods.

[This material was condensed, by permission, from a reprint—entitled "The Principles of Conditioning"—of "Heart Briefs" (Spring, 1979), a publication of the American Heart Association, Alameda County Chapter, 11200 Golf Links Road, Oakland, CA 94605. "Heart Briefs" is copyright, 1979, American heart association. For more information on all aspects of the cardiovascular system and exercise, write or call the division headquarters of the American Heart Association in your area.]

For your key-in listings, see HCM PROGRAM LISTINGS Contents.

HCM

HCM Review Criteria

Each month, *Home Computer Magazine* (*HCM*) reviews products designed for the Apple II family, Atari 800 family and compatibles, Commodore 64, IBM PC and PCjr, and Texas Instruments 99/4A computers. *HCM* reviews take a detailed look at the quality, utility, and value of commercially available packages for these machines. Because our publishing charter forbids accepting outside advertising, we strive to make the scope and content of our review pages shine with a unique blend of humanistic frankness and objectivity.

Not only will you find all relevant information for making a wise purchase decision, but in some special cases we also provide nuggets of compu-prestidigitation.* For example, we frequently include essential documentation not furnished by the manufacturer. Additionally, each issue of *HCM* tries to review at least one outstanding product—a "Diamond in the Rough"—which, because of company size, marketing clout, or for some other reason, has not received the attention it deserves.

At the beginning of each review, a review-at-a-glance box provides the user with an instant assessment of the product. Each item will be evaluated, where relevant, with the criteria below.

HCM Review

Name: Old Art
Program Type: Recycled Graphics
Machine: Apple II family, Atari 800XL, C-64 IBM PC & PCjr, TI-99/4A
Distributor: Hit 'n' RUN Software, Inc.
Price: $99.99 (or trade for '72 Pinto)

System Requirements:
Disk Drive, Joystick, Trash Can optional

	Poor	Fair	Good	Excellent
Performance:			■	
Engrossment:			■	
Documentation:			■	

Products may also be evaluated in the following areas:

*** Flexibility—**
Can the product be adapted to the specific needs of the users?

*** Cost/Benefit—**
Is the product worth the user's investment in time and money?

*** Necessity—**
Is the product a solution for which a problem already exists?

*** Originality—**
Is it unique in concept, or simply a "me too" product?

*** Longevity—**
The "Boredom Factor." Does the program sustain interest?

*** Rewards—**
Are the audio-visual rewards motivating and appropriate?

*** Concept Presentation—**
Are the concepts presented clearly, logically, and in depth?

*** Special Effects—**
How does quality of sound and visual effects rate? Do they enhance or detract from the product or learning process?

*** Performance—**
How well the product performs as intended; how well it takes advantage of a specific machine's capabilities; how well it responds to the user's commands; how effectively the graphics, sound effects, music, or speech are integrated with the software.

*** Engrossment—**
Whether the game or activity has that intangible quality that holds players on the edge of their seats while the hours tick by unnoticed.

OR

*** Ease of Use—**
The degree to which a user can interact with the product without outside help: the ease and effectiveness of error-handling features; whether the actual reading level of the activity is appropriate for the suggested audience.

OR

*** Ease of Set-up—**
How well the product design facilitates easy installation.

*** Documentation—**
The quality of the printed matter that comes with the product; whether the instructions are clear and comprehensive; whether the machine configuration requirements are spelled out. Information such as how to load a program, use the keyboard, and restart an activity contributes to the documentation rating, as do tips on performance peculiarities.

Attention Software Authors & Peripheral Inventors:

* WANT TO BE DISCOVERED? *

Home Computer Magazine Wants To Give You A Chance!

We are looking for home computer products that have not received the attention they deserve. Each month, we will be singling out one such package for special review. If you have a unique commercial product of exceptional quality—but your advertising and promotion budget has not allowed you to capture major media attention—we want to see it. We will consider reviewing any product that meets our high standards.

We are an Equal Opportunity Reviewer!

In order to qualify for possible review, your product must:

1. Currently be available for purchase to readers of this magazine.
2. Make a unique and important contribution to the home computer industry.
3. Be of outstanding merit, quality, and value.
4. Be consistent with the type of machines and products we normally cover.

If you feel that your product qualifies, mail it to:

Home Computer Magazine
Attn: Editorial Submissions
1500 Valley River Drive, Suite 250
Eugene, OR. 97401

We reserve the right *not* to reply to each inquiry, so please do *not* contact us except to request return of your product. If you want your product to be returned, please include sufficient return postage.

***Compu-prestidigitation** (kóm´• pū •prĕs´•teh•dĭ•jeh•tā´•shŭn) —*n* 1. The magical quality of unexpected comprehension that results from presenting technical information about computers in a lively, entertaining, visually attractive and easy-to-understand format. 2. The magical tricks that make a computer sing, dance, and do all sorts of wonderfully useful things.

Home Print Studio

by the HCM Staff

With just a little imagination, a simple home computer, and one of these new type/graphic design programs, you can start up your own print studio at home.

PRODUCT REVIEWS

PRINTING SOFTWARE—A NEW VALUE?

An Overview

In the heightened competitiveness of today's software market, many developers are seeking new ways of using the computer to perform tasks traditionally done outside the home. This movement has spawned such valuable program genres as music performance and home accounting. It has also fostered applications of more questionable value—everything from biofeedback to personality analysis. Right now one of the most rapidly growing genres is what we call "home print studio" software. But, even though this new application promises much, the major question still to be resolved is, on which side of the fence does it fall? Do these programs replace traditional print and graphic sources, thereby adding great value to home computers? Or, do they fall alongside those of questionable utility?

Do these programs replace traditional print and graphic sources, thereby adding great value to home computers? Or, do they fall alongside those of questionable utility?

In a limited way, several of these programs *can* fill the order of a traditional print store. Print-studio aficionados have found that they can now use their computer to make cards, newsletters, posters, resumes, letterheads, charts, certificates, graphs, announcements, fancy memos, advertisements, invitations, banners, floor plans, needlework designs, and more. Of course, a computer printout may suffice for memos and posters—but those who prefer embossed, pastel-colored greeting cards with gold lettering face a tradeoff in quality for the convenience and freedom of creating their own print products at home.

For one thing, *freedom* to create near-professional-quality print products with a home computer will remain limited until color is more readily available. Currently, there are three ways to print in color: 1) Use a color printer with a program that has a color option; 2) interchange color printer ribbons while running out your document several times (once for red borders, again for blue text, again for green graphics, etc.); or 3) use colored paper (which is also possible together with the other two options). At this time, only two of the nine programs we review here accommodate color.

The *convenience* afforded by these programs really depends on how well they fulfill their claims of being quick and easy to use. And that hinges on two major conditions: the compatibility of a print program with a given printer, and the compatibility of the print program with other software used to create text and graphics.

Basically, these programs print in various type styles either by: 1) sending the printer certain codes that cause the printer to change the form in which it prints its normal character set—i.e., changing it to boldface, italic, expanded type, etc.; or 2) using bit-mapped graphics with the graphics options of some printers to create much more extravagant fonts. The first method actually just reconfigures the printer with codes for options that most printers (or word processors) provide anyway; so this type of program is not providing users with any added advantage. The second method, on the other hand, allows you to easily access a printer's bit-map graphics capabilities and print out your own files using the fanciest of fonts imaginable.

Our biggest problem in researching this review turned out to be making the programs work with supposedly compatible printers. Difficulties ranged from printer-switch settings that needed to be adjusted, to programs that used improper codes to access a printer's bit-map capabilities. These codes are so printer-specific that what works on one printer does not always work on another. Even though a program may list several printers as "compatible," minor differences can cause big headaches. In some cases, it is apparent that the developer neglected to adequately test for these differences before releasing a product. In others, the program simply had been released before some of our newer printers hit the market.

Software compatibility problems occur when, for example, you must use a program like a word processor to create a file, and a separate print program to produce the hard copy. If a file from the word processor is not compatible with your print program, you may end up with a garbled mess. And if you must reboot the word processor after running the print program, you lose your previously specified printing parameters.

We also found that, because the packaging and documentation of some of these programs used such phrases as "Epson-compatible" and "works with most word processors" rather carelessly, the documentation can't always be trusted. It is beyond the scope of this article to specify the multitude of possible printer, system, software, and interface compatibilities for the nine products reviewed here—so be sure that the print program you select works on your system before you buy it. If possible, test it on a system setup identical to your own at the store or at a user-group meeting to avoid disappointment.

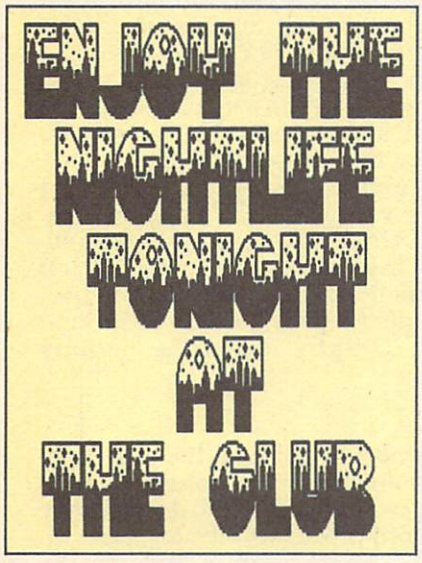
One other note: Many of these programs have a font-editor option that allows you to design your own character set pixel by pixel. However, you should be aware that there is no one-to-one correspondence between pixels on the screen and dots on a dot matrix printer. These relationships change due to varying screen resolutions and the way that systems make these pixel-to-dot translations. Although the representation is close, you won't get a one-to-one dot correspondence.

FEATURED PRINTERS

For this print studio review, in addition to the regular printer workhorses that we keep around the editorial department, we also tried out some new models provided for this review. The printers we used here include: the Apple Imagewriter, $595, $795 wide carriage (Apple Computer, Inc., 20525 Mariani Ave., Cupertino, CA 95014); the Texas Instruments 99/4A Impact Printer which is an Epson MX-80 (no longer being produced); the Epson HomeWriter 10, $269 and PIC interfaces, $60 each, tractor feeder $39.95, cut-sheet feeder $99.95; the Epson Spectrum LX-80, $299 (Epson America, Inc., 2780 Lomita Blvd., Torrance, CA 90505); and Okidata's Okimate 20, $268 (Okidata, Mt. Laurel, NJ 08054).

FONTRIX

Fontrix, by Data Transforms, is clearly the most sophisticated of the print-studio programs reviewed here, allowing you the greatest amount of freedom to design fonts, graphics, and entire pages. It brings the power of Macintosh's *MacWrite* and *MacPaint* programs to users of IBM computers and Apple II family machines, with just a little more work on the user's part. In fact, the manual and the program's capabilities are so overwhelming, it's easy to become intimidated by it all until you've had a chance to spend a good deal of time experimenting with it.



Fortunately, this is easy to do. A rather lengthy yet enjoyable tutorial in the manual guides you through each of the program's options as you create a birthday card and an invitation containing a map. You can take your pick from 12 fonts, 98 foreground and background colors and patterns, and a wide variety of drawing options. Commands such as Line, Box, Ellipse, and Airbrush (spray can) are highly reminiscent of *MacPaint*—minus the icons.

Two elements (at least) help make this program a typesetting/graphics standout: the additional font sets available separately (called *Fontpaks*), and a graphics option called Graffiles.

Although fonts are included with the program, they are limited to the all-too-common Roman, Script, and News style. *Fontpaks*, however, contain some of the most original fonts ever produced. The Skyline font from the *Incredible Novelty Fonts Fontpak* used in the printout here is just one example. Each package has ten complete character sets, most based on a certain theme—music, electronics, foreign alphabets, architecture symbols, and more. They are released periodically, and many are from *Fontrix* users who created them using *Fontrix*'s character editor.

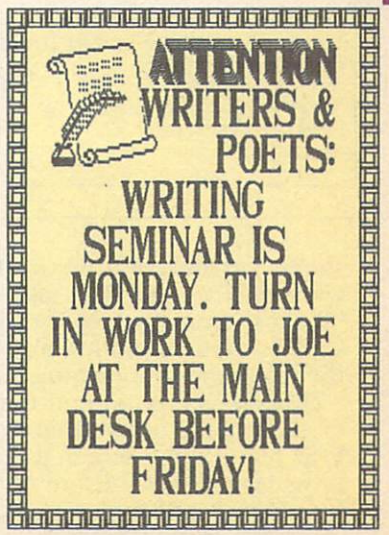
It's easy and fun to start with some of these sets and use the editor to alter them to your own taste. Like most other character editors, this one works by turning on and off screen dots that you specify by hitting a key. By changing the size of the cell, you can create different character sizes (maximum size is 32 x 32 screen dots). Each character set may have as many as 94 characters.

The Graffile concept is based on sectors. The high-resolution screen is about 5 x 6 *Fontrix* sectors, but you can create and scroll around on a Graffile that is *80 x 96* sectors. This allows you to create extremely intricate graphic and font designs—or combine them to, for example, throw together a newsletter with professional-looking headlines, charts, graphics, and justified columns of text. You can create these elements within the *Fontrix* program, or import text or graphics from outside programs and integrate it all on a Graffile, which can then be saved and printed out on your printer.

The only problem we had with this fine program occurred when we attempted to make a backup of the IBM version of *Fontrix as instructed*. After formatting a disk with the /s option (making the disk an MS-DOS startup), trying to copy the master provoked a message saying that there was insufficient space on the disk. Of the 24 files—only 20 would fit, leaving out 4 files. To use any of these missing fonts, you would have to delete some files from the backup and copy the desired files from the master copy.

THE PRINT SHOP

The Print Shop by Broderbund Software is the epitome of easy operation in a print program. Here, visually pleasing and informative menus rule, making the fine manual unnecessary. When the program encounters an illegal entry, it simply ignores it and returns you to the menu you were working from. Change your mind about something? Just back up through the menus and make your change, keeping the choices you've already selected, if you desire. You may also pause or stop printing at any time.



But what does *The Print Shop* offer? Its myriad options allow you to put together hundreds of combinations of ready-made graphics, fonts, and border designs for greeting cards, signs, letterheads, and banners. The package includes 50 pictures (like a birthday cake, Santa Claus, animals, hearts, etc.), 8 fonts (that can be set centered, left, or right, in solid, outline, or 3-dimensional letters), 9 border designs, and 10 background patterns, as well as colored pin-feed paper and envelopes. A separate package, *The Print Shop Graphics Library*, provides 120 more graphics pictures.

If you're the do-it-yourself type, the Graphic Editor option lets you make your own graphics from scratch, or alter *The Print Shop*'s graphics, through the use of the keyboard, a joystick, or a KoalaPad. Although you can't import graphics created in other programs, you *can* draw and erase, with the cursor as your guide and its X and Y coordinates displayed at the screen's bottom to aid in more precise drawing.

Screen Magic mode presents a series of constantly changing kaleidoscope and geometric patterns that may be "frozen," saved, and printed—with or without a text message and border. This mode can't be used with the others, and vice versa, because it functions in a lower-resolution mode.

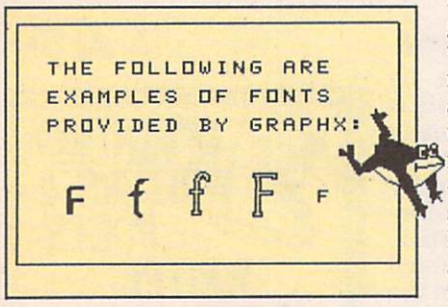
The Atari version of *The Print Shop* provides a pleasant surprise not found in many of the other programs here—it checks to see whether the printer supplies an automatic line feed. If it does, the progam accommodates it when setting up the disk. You don't even have to think about it.

Our biggest complaint with the program is that the actual printing takes *so long*. If you need multiple copies, you might consider using your *Print Shop* printout as a master and visiting a real print shop or a simple photocopy machine. Another disappointment is that you can only use one font and one picture in a document. You can repeat your graphic all over the page or pick one good spot for it, but you can use only one, until next time.

The best thing about this program? It's easy to use and the results look great.

GRAPHX

In the prolific outflow of printer programs for other home computers, the TI-99/4A has so far been largely ignored. The programs that *are* written for the TI machine are scant and few—often they neglect to make full use of the computer's considerable abilities. Fortunately, there are always exceptions, and TI-99/4A users need not feel slighted in the case of one such program. *Graphx*, a software package by R.L. and C.P. Davis, two talented "blokes" from Sydney, has made its way from Down Under. It is a program that will fill you once again with the excitement of owning a home computer.



Graphx has, as its name implies, a somewhat different focus than the other programs featured in this review. It is, above all, a screen-graphics program, yet it also provides most of the font and printing options offered by the best printing programs reviewed here.

Graphx is amazingly easy to use for a program that offers so many features. This ease-of-use is mainly due to concise help lines that appear at the top of each screen, logical menus, and a function-key overlay strip that identifies the keys used in the program. Freehand drawing and editing with either a joystick or the keyboard make creating graphics natural and comfortable. Additional program features include a multiple-speed option; an easy-erasing mode; a zoom function that magnifies portions of the screen for easy editing; circle, ellipse, and line functions; and copy and move functions that duplicate and shift images on the screen. A typewriter mode allows you to use the keyboard to add text and labels to your graphics.

The program includes an optional checkerboard pattern that you can temporarily substitute for the background. This pattern makes full-color drawing hassle-free by identifying eight-pixel cells that are limited to no more than two colors. Another function allows you to fill entire shapes with color rather than coloring one pixel at a time. As an added benefit to all of you assembly-language programmers, pictures may be saved and used as colorful and intricate backgrounds.

The program provides several printing options to be used on *Epson MX-80* or compatible printers. Four printing formats allow you to produce prints of two densities (single density and double density) and two sizes. The small-sized double-density option produces near-letter-quality print on the *Epson MX-80*.

The most useful aspect of the program is the Clipboard. It allows you to save and run sequenced images for an animated effect. It also provides four fonts: computer style, gothic, hollow lower-case, and hollow upper-case. These fonts may be edited to serve your particular needs—you may add new characters, alter characters, or fill in the hollow fonts. *Graphx* is a joy to use. Its multiple applications range from game programming and font and graphics production, to providing interactive, captivating entertainment. With its low cost and numerous features, *Graphx* deserves serious consideration.

FONTMASTER

Fontmaster is a unique entry among the other programs here because its creators have included a complete word-processing program on the disk. You may also convert outside word-processor files for use with this program.

Fontmaster's embedded-style typesetting commands allow you to: alternate among 8 fonts you've loaded in memory (from 16 that are on the disk); print in boldface, italic, inverse, expanded, or compressed type; print with sub- or superscripts; and format at 6 or 8 lines per inch. This program is very flexible, allowing you to change to several fonts within one word if you like. Although you can't change the size of the type, you can use the Fonteditor program and a little effort to change an existing font or create your own. Here, as in *Fontrix*, you get a grid in which you turn pixels on and off to design each character.

The program and the manual are presented in a straightforward manner—using this program involves little more than getting acquainted with the many varied word-processing commands summarized in a chart on the back cover of the manual. Unfortunately, a few are missing, such as the print commands. The rest of the manual also has a few problems, including grammatical errors. A "Figure 1" showing the word-processor screen is referred to early in the text, but it is buried way in the back of the book and is not listed in the Contents. In addition, the word processor's commands are listed according to command, not function, so it's tough to find what you want if you're in a hurry. It's actually more useful to stick to the nearly complete chart on the back cover.

Once you're in the word processor, you'll find that screen prompts appear whenever you choose an option, and a full screen of formatting parameters for your text is available by toggling the [←] key. Because this is an assembly-language program, you cannot just exit to BASIC to load another program (as with the Fonteditor) when you've finished with the word processor.

We found one flaw that seems to be a hardware problem: We could not get our text to print out on an *Epson MX-80* (the *TI-99/4A Impact Printer*) nor on the *Epson Spectrum LX-90*. According to a very helpful technical representative at Xetec, there are slight differences in Epson printers made for other companies, such as Texas Instruments. It seems that the *TI-99/4A Impact Printer* ignores a code sent to the printer that changes the line feed to its proper spacing. The technical representative was not sure whether Xetec's program is compatible with the Commodore interface on the new LX-90. He added that if one of *Fontmaster*'s print parameters is set at 80 columns, you'll get an extra line feed. It's therefore safer to change it to 81.

Shortly after this issue of *HCM* goes to press, a new version of *Fontmaster* is due to be released. Xetec said that its biggest change will be in accommodating newer printers and printer interfaces. It will also supposedly have an option to enter your own configuration codes if necessary; this would solve our problem with the line-feed code.

If the new version of *Fontmaster* performs as promised, its full-capability word processor, font sets, and font editor will offer much more than some programs that are priced $100 higher. Potential buyers should, however, be cautious and make sure that they first verify printer compatibility.

THE PRINTOGRAPHER

Although *The Printographer* does provide 9 font sets, it is chiefly a graphics dump that allows you to import or create a picture on the high- or low-resolution screen, add text, crop it, and easily print it all out. Images may be printed horizontally or vertically; in normal, inverse, or color mode; and from within your own program. You may also convert additional outside character

NAME	MACHINES	DISTRIBUTOR	PRICE	SYSTEM REQUIREMENTS	POOR FAIR GOOD EXCELLENT
Fontrix	Apple II family, IBM PC & PCjr, Tandy 1000	Data Transforms, Inc. 616 Washington St. Suite 106, Denver, CO 80203, (303) 832-1501	$75 Apple; $125 IBM	Apple: 48K, printer and interface. IBM: 256K, MS DOS 2.0 or later, color-graphics card, printer.	P: EU: D: CB:
The Print Shop	Apple II family, Atari, Commodore 64	Broderbund Software, 17 Paul Dr., San Rafael, CA 94903, (415) 479-1170	$49.95 Apple; $44.95 Atari & Commodore 64; $24.95 Graphics Library	Disk drive, printer. Also, printer interface for Apple & Atari.	P: EU: D: CB:
Graphx	TI-99/4A	Graphx, P.O. Box C568 Clarence St., Sydney, NSW Australia 2000	$50	Disk drive; joystick; RS-232 interface; printer; Mini Memory, Editor Assembler, or Extended BASIC cartridge.	P: EU: D: CB:
Fontmaster	Commodore 64	Xetec, Inc., 30100 Arnold Rd., Salina, KS 67401, (913) 827-0685	$39.95	Disk drive, printer.	P: EU: D: CB:
The Printographer	Apple II family	Roger Wagner Publishing, Inc., 10761 Woodside Ave. Suite E, Santee, CA 92071, (619) 592-3670	$39.95	48K, printer and interface.	P: EU: D: CB:
Printworks	IBM PC & PCjr, Tandy 1000	SoftStyle, 7192 Kalanianaole Hwy. Suite 205, Honolulu, HA 96825, (808) 396-6368	$69.95	128K, printer.	P: EU: D: CB:
Fancy Font	IBM PC & PCjr	Softcraft Inc., 222 State St., Madison, WI 53703	$180	128K, printer.	P: EU: D: CB:
Facelift	Apple II family, Commodore 64, IBM PC	Companion Software, Inc., P.O. Box 480741, Los Angeles, CA 90048	$29.95	Printer and interface, compatible word-processing program.	P: EU: D: CB:
Select-A-Font	IBM PC & PCjr	IBM Corp., Boca Raton, FL 33432	$19.95	128K, disk drive, printer.	P: EU: D: CB:

KEY: P: = Performance, EU: = Ease of Use, D: = Documentation, CB: = Cost/Benefit.

sets for use here—this program will accommodate fonts with the same format as that used in Apple Computer's *DOS Tool Kit*.

Users may experiment on 3 amusing cartoon graphics screens and a chart that are provided on the program disk. The Crop option displays the selected graphics and blinks an L-shaped cursor at its upper-left or lower-right corner. By moving the cursor up, down, left, or right, you can crop your picture just as you would snip a photograph's edges to emphasize its best elements. The Diamond and Cameo suboptions trim your graphic to form, naturally, a diamond- or cameo-shaped picture. It's an unusual feature with interesting possibilities that afford one picture many different looks.

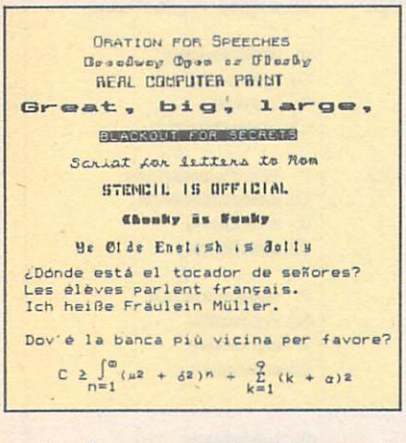
As for the fonts, they include the basic Roman, Old English, and Computer styles found everywhere, as well as a Greek character set. You may load 2 sets into memory for use at one time. Inverse mode is available to help make the text stand out from the picture.

The Magnify option increases the size of an image up to 99 times within *The Printographer* and up to 127 times when used in your own program. You probably wouldn't want to magnify an *entire screen* to such sizes, but to better see and print a few special graphics *characters*, it helps.

This is a good, solid screen-dump program, with a few unexpected extras (like the Crop and Magnify features) thrown in. The disk is set up to help you configure almost any printer, simply by following the screen instructions. The manual carefully explains each of the program options, and contains a bevy of appendices on special setups, charts, diagrams, and graphics routines for programmers interested in using *The Printographer*'s subroutines in their own programs.

PRINTWORKS

Printworks is strictly a font program, providing some ready-made standard and also foreign-language sets for quick printer dumps. In addition, a font editor lets you create your own characters.



Printworks is menu-driven and boots before your word-processing application, printing out your documents according to the options you specify. As mentioned in the overview, if you have a word processor that reinitializes the printer, you'll negate *Printworks*. However, tips for overcoming this are included in the manual.

Options available with *Printworks* allow you to print wide, condensed, small, tall, bold, italic, subscript, or superscript letters, to start with. Other options let you specify 6 or 8 lines per inch; margins; unidirectional printing, pica, elite, or proportional letter spacing; and variable line spacing.

Unfortunately, not all options are available with all of the printers that are listed as being "compatible" with this program. For example, some other options—such as download font (which is not necessary if you have a graphic printer), control code (prints printer-control codes so they can be used as regular characters), letter quality, left margin, quiet mode, and typewriter mode—were unavailable to us when using a PCjr with Epson's *Homewriter 10* printer.

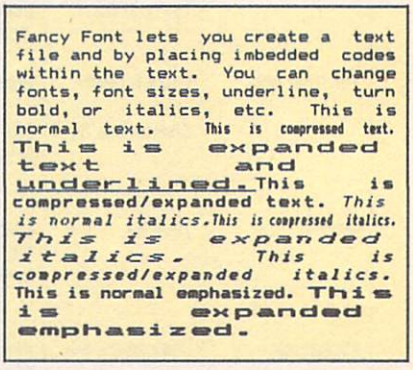
The Pivot Printing option rotates your text 90 degrees to print sideways on a page. This is quite handy for making banners or the ultimate endless spreadsheet. (In Condensed mode, however, it's possible to fit a 14-column, 8-character-per-column spreadsheet across the *width* of the page.)

Printworks' font editor screen displays a grid where you turn pixels on and off to create new characters or alter existing ones. The Replace mode is a nice touch here. With one command you can easily make your new character replace another in memory. Then you just call up the new character and begin editing where you left off—like when you want to turn an O into a C without repeating most of your work. Similar programs take you through elaborate copy, paste, and reloading procedures to do the same thing.

The Test Pattern option does what it says: it prints out all of the keyboard's characters using the options you've selected. This way you can determine whether "what you get is what you want" before you print out an entire file. It's a thoughtful addition.

FANCY FONT

This is another program that uses embedded commands to manipulate and format text. *Fancy Font* claims to work with almost any word processor—as long as it uses standard ASCII files. We tried it on several popular word processors, and found that unless the file is clean of tabs and other codes not entered via *Fancy Font*, it won't print at all.



The program comes in two versions: an MX version, which produces high-quality printouts at a resolution of 25,920 dots per square inch; and an FX version, which produces a higher quality printout at a resolution of 51,840 dots per square inch. Although *Fancy Font* is designed for use with Epson printers, it will work with other printers—but only in the MX version. Each version comes in 3 parts. Pfont is the part of the package that prints files, and is probably the one that most people will want to use. Efont and Cfont are supplied for those who may want to edit existing fonts or create their own.

Fancy Font provides more than 30 fonts (you can have up to 10 fonts active during a printout) ranging in size from 8 to 40 points. Font styles include Roman, Sans Serif, Script, and Old English. The program also features bold, italic, underline, and regular printing.

With *Fancy Font* we tried the *Epson Spectrum LX-80* printer, which works with the FX version of the program. The printer operated with no problems, although the first time we attempted to get a printout, we were unprepared for the extreme slowness of printing. The program directs the printer head to make from 6 to 12 passes in some cases. It took about 30 minutes to print out a sample of *Fancy Fonts*' features. If this is the price of being "fancy," it may be too much for some to pay . . .

Fancy Font lets you set up parameters to format your text (with indents, tabs, right or left justification, line width, etc.) by embedding codes in the text where

needed, or by placing them in the first command line. Placing the codes within your text file is a simple process, but it can become complicated very fast. As an example, here is the print command used to generate a sample file provided on disk with the program: `pfont sample.ff +fo romn12 romnb12 romni12`.

Once you become familiar with *Fancy Font*, you can print out near-letter-quality manuscripts, and enjoy a wide variety of fonts, print styles, and text formats. The ridiculously slow printing speed is a major handicap—although you can get draft-quality printouts (with all fonts) at a much faster speed.

In general, *Fancy Font* is an exercise in frustration. It offers many features, but it is much more complicated than it needs to be. Its manual is unclear, and assumes that the operator has a very good understanding of computing and coding. At $180, it is far and away the most expensive program we examined for this review, and frankly, we don't think that it warrants such a high price. Computer neophytes would be better off to look elsewhere for a typesetting program.

FACELIFT

Facelift, an attractively packaged font program by Companion Software, promises to enhance your computer's printing options by providing 92 different typestyles, sizes, and weights. It is available in several versions designed to work on the IBM PC, the Commodore 64, and Apple II family computers with Epson printers. It is supposed to enable you to create text in one of five fonts: condensed, babyface (an extremely small font), elite, pica, and trimline (a sans-serif font that looks like it has been squashed flat). These fonts can be used in conjunction with one or more commands that create italicized, heavy, double-strike, underlined, or wide-spaced print—at least that's the assertion. In reality, *Facelift* is an awkward program with minimal utility.

The program can be used only in conjunction with word processors, spreadsheets, or other programs with print functions. Because many of these programs already provide the same or similar printing options offered by *Facelift*, its value as an added utility is questionable. In addition, many programs that contain embedded commands will either over-ride or conflict with *Facelift* to produce unexpected results. The manual neglects to provide a list of compatible programs, but cautions that *Facelift* should be tested with the programs before use, and even suggests that the unexpected results of incompatibility might be "pleasant."

You may also expect to struggle through scant directions before getting *Facelift* to operate. The rewards for this struggle are few: *Facelift* sets the printer, not the computer, so once a typestyle is chosen, the entire file must be printed in that style—you can't underline, italicize, or otherwise alter particular words, lines, or sections.

If all this isn't enough to deter you from trying the program, there's more. The Apple version has a bug which makes it incompatible with *Epson MX-80* and *MX-100* printers. And that's not all—after several hours of concerted effort, we could get neither the Commodore nor the IBM versions to function. In a phone call, the manufacturers conceded that bugs might exist in these versions as well.

Despite its attractive package, the best we can say for *Facelift* is that it is appropriately named. Just as a facelift cannot cure old age—only hide the wrinkles—*Facelift*'s elegant wrapping does little more than conceal debilitating flaws.

SELECT-A-FONT

Select-A-Font gives your printer access to 9 different fonts, 3 type sizes, 9 character widths, and 2 print densities. *Select-A-Font* is designed to work with an IBM text editor or word processor like *Personal Editor*, but it will work with most word processors as long as they produce standard ASCII files.

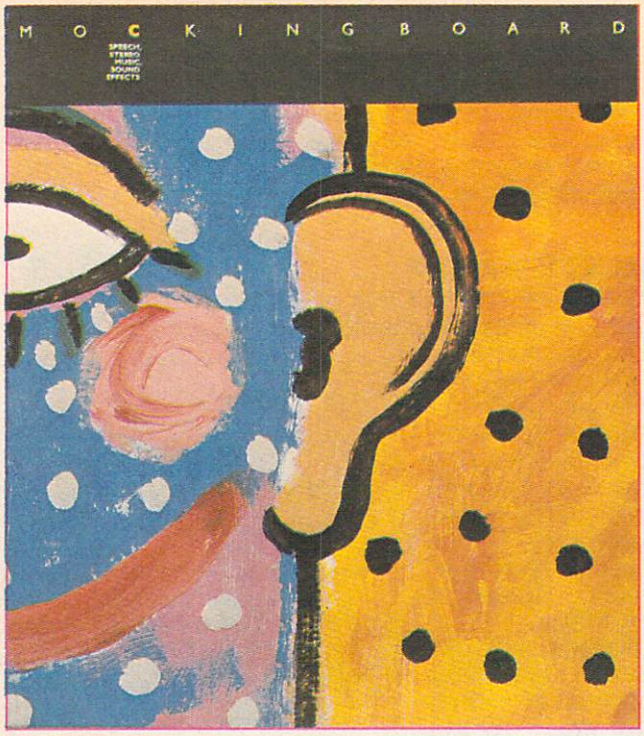
Select-A-Font uses embedded commands which you place inside your text next to the line or word that you want to alter. For example, to center a phrase in your text, enter .CE at the beginning of the line of text. The .CE command will center only one line at a time, so each subsequent line to be centered must have the command at the beginning.

The main thing that impressed us about this program is its simplicity of operation. The package doesn't come with an instruction manual, it's all on the disk (you can get a printout of the instructions if you wish). For some programs, this might be a problem, but *Select-A-Font*'s instructions are complete, and easily accessible. The instructions guide you through all of the various commands and operations, and you can also call up a help screen or use a quick-reference card.

We did discover one problem: changing the size of a font causes the line to become too long for the printer to print on one line. The program breaks the line, gives you an error message, and shows you the point in the line where you should have broken your sentence.

Operating this program was so easy that it was quite simply a lot of fun. And considering *Select-A-Font*'s low price and all of the features it offers, this is one program that ranks high on the cost/benefit curve.

HCM



Listen to the Mockingboard

A Review by Roger Wood
HCM Staff

The Apple II family's weakness is its lack of a built-in sound chip. Here's a product that fills the musical void and gives you a powerful speech system as well.

Apple is *the* pioneer in microcomputers for the home. But being first has its drawbacks. In particular, the Apple II family's lack of an internal sound chip severely limits its ability to produce musical sounds. Granted, some amazing effects can be created by "plucking the speaker." But when compared to the sophisticated sounds created by the Commodore 64, the TI-99/4A, or the IBM PCjr, Apple machines are at a distinct disadvantage. The *Mockingboard*, however, from Sweet Micro Systems, fills this Apple sound-void—plus, it provides a sophisticated, programmable speech system to boot.

Choose From Several Products

Mockingboard comes in 4 different configuration packages:

For Apple II, II+, and IIe
Mockingboard A - Sound, music board, no speech.
Mockingboard B - Speech upgrade (Used only in conjunction with Mockingboard A).
Mockingboard C - Both speech and sound.

For Apple IIc
Mockingboard D - Includes all the features of C, but with built-in speakers.

Products A, B, and C plug into one of the expansion slots (slot 4 is recommended, but not required) on your Apple II, II+, or *IIe* and require external speakers. *Mockingboard D* is a self-contained unit that interfaces with the Apple *IIc* modem port and comes in a *IIc*-colored box which includes speakers (see photos). Each product comes with a Demo/Utility disk. Although the software differs for each product (due to hardware differences in the computers), the programs operate almost identically.

In terms of setup, it's easy to start using all of the packages—just plug them in, place the *Demonstration* disk in the drive, and turn on the power. With A, B, and C you must provide 2-8 ohm speakers as well. For our simple tests, a couple of inexpensive Radio Shack speakers proved quite adequate.

Sweet Micro has an enhancement package for use with the A, B, and C packages: the *A-Max*. It includes two speakers and a jack for an earphone. It lists for $49.95. I found this product to be easy to install, and it works just fine. It takes the place of your internal Apple speaker and works with the *Mockingboard*, plus it adds an external stereo volume control. The price seems a bit steep, but it is a nice clean accessory for your machine.

Sounding Out the Mockingboard

After installing the *Mockingboard*, I first tested its music and sound capabilities. The board includes a sound chip capable of two different sounds—a musical tone and a noise waveform. (It has two speaker outputs for full stereophonic sound.) The musical tone and the noise waveform can be combined in a variety of ways and are variable over a wide frequency range, so countless stereo effects can be produced.

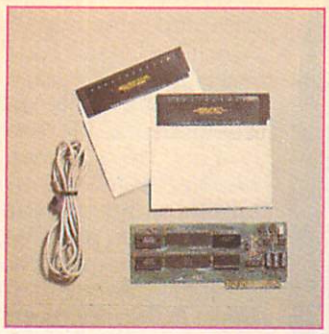
The *Demonstration* disk included with the package presents menu-driven programs giving quick and easy access to many sound effects and musical examples. The *Sound Utility* program (also accesssible from the menu) allows you to create your own effects and save the parameters to disk, or load the parameters used to make the sounds you hear on the demos. The program lets you modify the sound parameters by entering values on a relatively simple chart on the screen (see Figure 1).

The left column lists the parameters you may change. The Max Val column shows the upper limit of each parameter value so that you don't have to refer to the manual constantly when experimenting. The four columns on the right are for your entries. The All column affects all three voices, or channels, available for each speaker. Columns A, B, and C let you set separate values for each voice for several of the parameters.

The Tone Per Fine and Tone Per Course settings select the frequency of a musical tone. A table in the back of the manual gives the values for 8 octaves of notes. Because you can set the three voices for each speaker to different notes, you can use "frequency offsetting" to obtain many full synthesizer sounds. Noise Period determines the frequency of the noise, and the Amplitude on any channel can be either set to one of 15 levels or made variable—only when set to 16 (for variable) do the Envelope parameters have any effect.

Some of the parameter names are a bit cryptic—Enable, Envl Per Fine, and Envl Per Course—but the documentation clarifies them.

The Mockingboard D is a stand-alone unit that hooks up to the IIc Modem Port. Although there are certain software differences, the demonstration disk is functionally identical to the one supplied with the Mockingboard C.



Above is a picture of the Mockingboard C and its software. The product plugs into any slot, but the demonstration disk requires it to be plugged into slot 4.

Name:	Mockingboard
Product Type:	Speech and Music Board
Machines:	Apple *II*, II+, and *IIe* (Mockingboard A, B, and C), or *IIc* (Mockingboard D)
Distributor:	Sweet Micro Systems 50 Freeway Drive Cranston, RI 02920
Price:	Mockingboard A: $99 (no speech) Mockingboard B: $89 (speech upgrade for A) Mockingboard C: $179 (speech and sound) Mockingboard D: $195 (*IIc* only)
System Requirements:	For Mockingboard A, B and C: Apple *II*, II+, or *IIe*: 48K RAM, 1 Disk Drive, 2 external 8-ohm speakers. For Mockingboard D: Apple *IIc*.

	Poor	Fair	Good	Excellent
Performance:	■			
Ease of Use:	■	■		
Ease of Setup:	■	■	■	
Documentation:	■	■	■	■
Cost/Benefit:	■			

Enable determines whether a particular channel outputs noise, music, or both. Because 64 different on and off combinations are possible (2 ^6 = 64), Enable can vary from 0 to 63. Figure 1 demonstrates how you can view a menu of the functions of the 64 combinations to aid you in your selection.

The Envelope parameters are really my only complaint with the way that data is entered on the *Sound Utility* screen. I'm accustomed to the full Attack, Decay, Sustain, and Release (ADSR) settings used on many synthesizers and by the Commodore 64 Sound Interface Device (SID) chip. The terms Envl Per Fine, Envl Per Course, and Envl Shape don't correspond to the ADSR parameters normally specified with envelope generators. It turns out that there are only 16 different envelope shapes available on *Mockingboard*, and I found several that were not very useful. The Course and Fine adjustments determine how long it takes a particular envelope to run its course. Compared to a normal ADSR, the *Mockingboard* system seems limited and cumbersome.

To show the strengths and weaknesses of the *Mockingboard*, I've included a chart (Table 1) comparing its sound capabilities to those of the Commodore 64's sound chip.

Programmability

While looking at how the *Mockingboard* compares to the C-64, let's not forget "programmability." Without added software, the Commodore SID chip, though accessible from BASIC, is a tricky device for all but experienced programmers. Meanwhile, the *Mockingboard* for your Apple can be set up using rather friendly menu-driven programs. But, how workable is the *Mockingboard*'s sound in BASIC programs?

Sad to say, the package by itself isn't very easy to use for incorporating sound effects in BASIC. First, you must BLOAD certain utilities into memory before accessing any sound. Then, each individual tone must be translated into 16 different pieces of DATA and keyed-into your program. Finally, when you call the sound driver, your note is played. Alternatively, you can BLOAD each note and do a CALL to the driver, but this proves to be very slow. Sweet Micro does have an excellent *Developer's Toolkit* available which gives you many easier ways to put sound in your programs.

Talk Using English Words

When you compare *Mockingboard* to the Commodore's sound capabilities, remember that only *Mockingboard A* is strictly a sound board. With *Mockingboard*'s *B*, *C*, and *D* you also get a full-blown speech system—something not yet available on the unexpanded C-64. In fact, speech is this product's strongest suit.

The menu-driven *Text to Speech* program includes a short spoken message (and a little singing) by *Mockingboard* to give you an idea of what the robotic voice can do. Although the voice's main parameters are all programmable, it is still a synthesized voice with a definite robotic sound.

The program lets you create speech by simply typing in regular words. Many speech synthesizers let you input English words to access speech, but they are often limited to a fixed vocabulary. Others give you an unlimited vocabulary, except that the words are ex-

Figure 1: *The Sound Utility program lets you change all the parameters for creating music and sound effects. This particular setup creates a reedy synthesizer tone by frequency-offsetting three of the voices.*

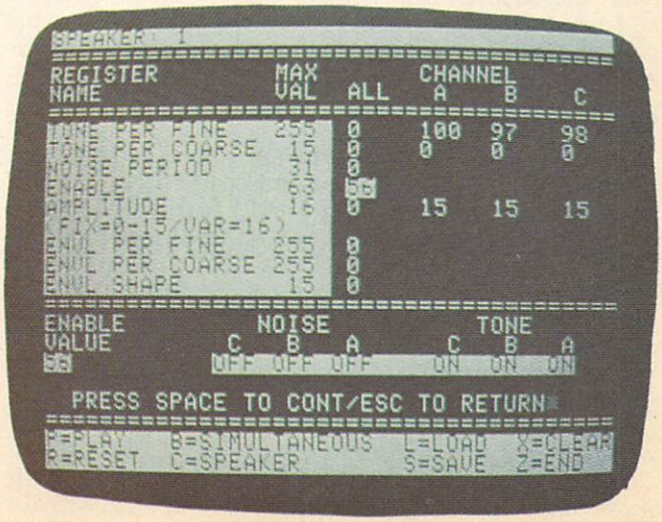


Table 1

MOCKINGBOARD VS. COMMODORE 64 SID CHIP

Feature	Mockingboard	C-64 SID
Number of Voices	6	3
Stereo Capability	Yes	No
Frequency Offset	Yes	Yes
No. & Types of Sound	2 (noise, music)	4 (triangle, sawtooth, pulse, noise)
Envelopes	16 fixed choices—with variable duration of envelope	Full ADSR (65536 combinations)
Filters	None	High-, low-, band-pass with resonance.
Ring Mod	No	Yes
Frequency Modulation	No	Yes

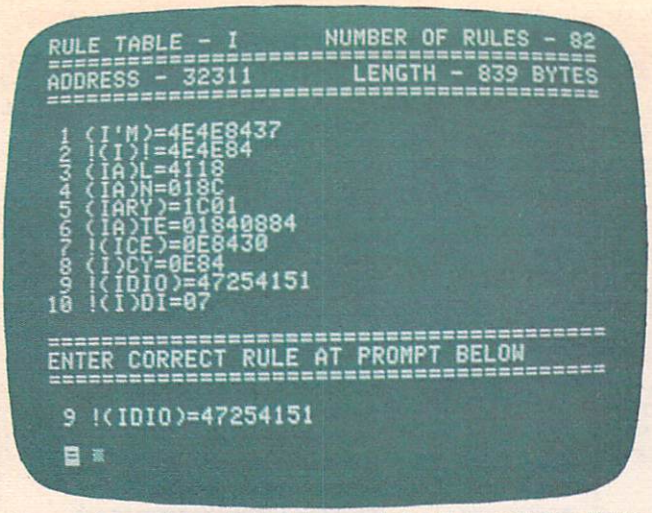


Figure 2: *A Speech Rule Table helps you define the way that particular combinations of lettters will be pronounced. This screen is the Rule Table for the letter I. The upper portion of the screen shows 10 different combinations and the phonemes used to pronounce those combinations. Here the letter combination IDIO is being edited at the bottom of the screen.*

pressed as larger vocabulary/phoneme codes. *Mockingboard* mixes these two in an ingenious way, giving the user the best of both worlds. If you want the *Mockingboard* to say hello, just type HELLO, and the word comes out. By putting a slash mark (/) in the middle of your words, you can make *Mockingboard* change inflection.

You may also call up from the menu a program that will take a regular disk-based text file as input and read it to you. Again, the limitation on pronunciation is the Rule Table.

Programming Speech With Rule Tables

Take a look at Figure 2. This is a picture of the Rule Table for the letter I. Whenever *Mockingboard* speaks a word, it starts by referencing the rule for the first letter. It then checks the Rule Table for that letter for the combination of letters in the word to be spoken. If the next letter does not match any combination in the first letter's Rule Table, it searches the second letter's Rule Table, and so on. Although the Phoneme Codes are a little strange at first, they are carefully detailed in the back of the manual. I found it fairly easy to successfully modify pronunciations of many words the first time I tried. The Rule Editor also lets you change parameters like Frequency, Speech Rate, Inflection, and Amplitude.

Again, the question of programmability comes to mind, but this time, thanks to disk drives, it really isn't very hard to incorporate speech into a program. Three separate files may be BLOADed into memory by your program, and then a simple assignment of words to a string (MB$), a CALL 26123 command, and the words placed into MB$ are spoken. Several easy-to-follow programs are included in the manual, so it's pretty easy to incorporate speech into a program right away.

What Next?

Even if you don't want to program your *Mockingboard*, there are many packages on the market that integrate this accessory into their programs. A partial list of some of the programs that use *Mockingboard* is included in Table 2.

Sweet Micro Systems has recently released 3 new packages to help software developers and home users alike take full advantage of their *Mockingboard*. One package is a *Foreign Language Rule Table* disk ($24.95) which includes French, German, and British English. Another package is the *Developers' Toolkit* ($29.95). This package includes utilities to move Rule Tables up into memory locations only available on computers with more memory—such as the Apple *IIc*, Apple *IIe* with an extended 80-column card, or a 64K Apple II+. One of the utilities included is called *AmperMock*. To install this Applesoft BASIC enhancement, you place the *Developers' Toolkit* disk in the drive and type BRUN AMPERMOCK. This makes all speech and sound commands accessible through the Ampersand (&) command. (For details on the Ampersand command, refer to an *Applesoft BASIC Programming Reference Manual*.) For example, to get *Mockingboard* to say a few words with *AmperMock*, all you have to type is &TALK followed by any string expression.

Finally, Sweet Micro has released a package called the *Speech Development System* ($39.95), for people who want to control *Mockingboard* even more completely, putting subtle nuances into each phrase spoken. This program uses a bar-graph scheme that makes changing parameters relatively easy, but the fine-tuning involved is a little time-consuming for the casual user.

All in all, the *Mockingboard* gives the Apple user something sorely lacking on an unexpanded Apple system—a high-quality sound and speech system. The documentation and optional software support make it a good educational and development tool for the serious computer user. The only drawback is that the package may be a little pricey for someone without a specific need for these computer applications.

Table 2

Some Mockingboard-Compatible Software

Distributor	Software Title
Avalon Hill	Tactical Armor Command
Datasoft	Zaxxon
D.T.I Data Trek	Maze Craze
Earthware	Zoomaster
Electronic Arts	Music Construction Set
Origin Systems	Exodus-Ultima III
Penguin Software	Adventure Magician
Strategic Systems, Inc.	Broadside

☞ COUNTERPOINT ☜

This bird can speak—but can it sing? Well, *Mockingboard* is no Prince, but it can at least carry a tune (even though it sounds like the tune is literally being carried in a bucket). Truth is, this Apple add-on brings a lot to a computer that, unaided, has some of the most limited sound ability of any home machine. Its producers are correct in stressing the importance of this addition, because it really does transform an almost mute system into a virtual jabberbox.

But what about *sound*? What about *music*? If you're looking—as I was—for the Apple equivalent of Commodore's SID, we are both in for a big disappointment. Even if there were software available to allow us full and easy access to the *Mockingboard* sound chip, we still could not produce the variety of sounds possible with SID. Lack of waveform, modulation, and envelope control dictate relatively bland results. Still, for Apple computer users, plugging in *Mockingboard* is like taking off a set of earmuffs.

As for speech, this system beats nearly all of its competitors. But despite *Mockingboard*'s current success in this field, anyone interested in computer speech should take into account developments on the horizon. Digital sampling techniques are already beginning to replace from-the-ground-up sound synthesis, leading to a more "natural" computer voice. Robotic speech, which once sounded futuristic, will increasingly seem old-fashioned. But for now (and who knows how long), this bird outsings all others in the Apple tree.

—Wayne Koberstein

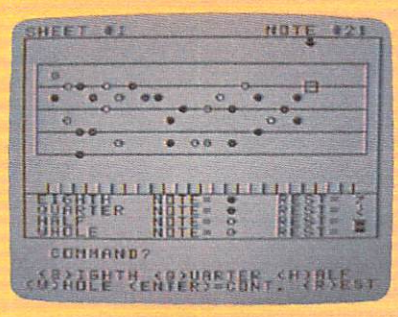
HCM Glossary terms: ampersand (&) command, driver, frequency offsetting, modem port.

HCM

MUSIC SYNTHESIZER

A Review by
Laile L. Di Silvestro
HCM Staff

HCM Review



Name:	Music Synthesizer
Program Type:	Music composition
Machine:	TI-99/4A
Distributor:	Asgard Software P.O. Box 10306 Rockville, MD 20850
Price:	$22.95 (disk), $19.95 (tape)

System Requirements: Disk drive, 32K memory expansion, Extended BASIC cartridge.

	Poor	Fair	Good	Excellent
Performance:				■
Engrossment:				■
Documentation:			■	

Is this the program that will finally reveal the wonders of the TI music chip?

Recently, the world of music has been opening up to home computers. Imaginative music construction, composition, and player programs have transformed computer consoles into a melodious realm of endless possibilities. Now Asgard has created *Music Synthesizer*, a program that seemingly promises to introduce TI-99/4A users to this realm. But is this program really a music synthesizer?

Creating Music

Music Synthesizer is a simple music composition program. Its screen-oriented editor enables you to place notes on a staff by choosing commands from concise menus. The songs thus created can have as many as three voices, and may be played and saved in whole or in part.

The program was intended for the musically inexperienced. Indeed, its stark simplicity implies ease of use. The black and white graphics on the screen contain nothing to confuse or distract. You have only to consider five horizontal lines (a musical staff), a menu, and a number indicating the note being worked on. The first menu enables you to place a note on the staff, play, edit, and view the screens. In addition, it provides save, new, and load functions.

Placing notes on the staff is, of course, the main purpose of the program. Logical commands make it easy but time-consuming, as several steps are necessary. For each note, you must enter L (for Leave note); you must maneuver the note using the E, S, D, and X keys; you must choose between whole, half, quarter, eighth notes, or a rest (if you choose rest, the process is one step longer); and, finally, you must press [ENTER]. If you want two or three voices to play simultaneously, these notes must be placed in a vertical row known as a "note column."

Contrary to the program's intentions, a fundamental knowledge of music *is* necessary to place notes on the staff. If you are truly inexperienced in music composition, you might find it more gratifying to just enter notes at random, play them, and smile in surprise.

Dismaying Flaws

Upon receiving *Music Synthesizer*, I eagerly sampled the music provided. I found myself dismayed at its limited nature. *Music Synthesizer* produces notes of three voices over a limited range of one-and-a-half octaves. It produces only one instrument sound and lacks the option of sharps and flats. It does not allow variations in tempo or volume. *Music Synthesizer* is not actually a synthesizer, as it does not *manipulate* sounds. Rather, it is an unsophisticated program that enables the TI-99/4A to generate simple *tones*.

One would expect more from a music program created for the TI machine. The 99/4A contains a well-contrived music chip that generates three tones and one noise simultaneously. The tones can be varied according to duration (1 millisecond to 4.25 seconds), frequency (110 to 44733 hertz), and volume (0 to 30). The noise may be one of eight provided—four white noises and four periodic noises. Of these noises, the frequencies of one white and one periodic noise may be altered. Thus, the TI-99/4A is capable of generating a great variety of sounds. It is a shame that *Music Synthesizer* does not make full use of these extensive capabilities.

The program has more problems: because it is written in Extended BASIC, the program is slow. For example, from the time the PLAY command is entered, to the moment when the song begins, there is at least a 30-second interval. During this wait, I found myself wishing for more stimulating graphics—perhaps some color or some movement to engage my interest. A more serious flaw is the program's inability to scroll while songs are playing.

> ***"One would expect more from a music program created for the TI machine."***

Poorly written instructions containing grammar and spelling errors add to the already numerous faults. It is rather surprising, therefore, that *Music Synthesizer* is one of the most highly priced programs of its kind. Its $22.95 price is at least $10 higher than comparable programs with greater application. (Watch for reviews in coming issues.)

Unfortunately, programs like *Music Synthesizer* by Asgard allow users to catch only a glimpse of the infinite potential and startling variety inherent to computerized music. As seen through *Music Synthesizer*, the musical world of computers appears quite barren and small.

HCM Glossary terms: frequency, periodic noise, synthesizer, white noise.

HCM

PRODUCT REVIEWS

Music Construction Set

A Review by

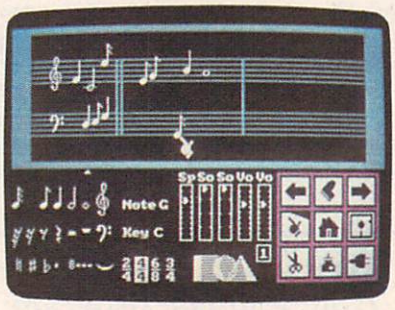
Steve Nelson
HCM Staff

You can have your computer belting out a tune before you know it with this easy-to-use product—but if you want to get serious, ease-of-use is not enough.

Even before Robert Moog built his first sound synthesizer, electronic devices were used to create music. Since then, these musical devices have continued to grow in sophistication—and now even home computers are flexing their musical muscles in a variety of ways.

Music Construction Set (*MCS*) by Electronic Arts is a program that lets you use your computer to compose music. After booting up the program, you are presented with a screen showing two staffs of music—a treble staff and a bass staff. Beneath these are the notes, rests, sound and volume controls, and other options for determining the musical output of your computer. *MCS* uses icons to represent the musical notations, as well as program functions such as accessing the disk, playing your composition, changing the sound, and editing.

Once booted up, *MCS* automatically goes into a demo mode that reveals just how musical your computer can be. The demonstration selections include: *Flight of the Bumblebee* by Rimsky-Korsakov, *Sonata in D Major, First Movement* by W.A. Mozart, and *Minuet* by Douglas Fulton. You can listen to these selections as they are, or you can copy and rearrange them to suit your tastes.



This photo from the IBM PCjr version shows the different icons and options that you can use to create music. Notice the pointing hand icon with note attached for placement.

Building a Song

MCS uses a cursor shaped like a hand with a pointing finger. You control it with the keyboard, a joystick, or a *KoalaPad*. Moving the cursor with the keyboard exclusively is quite slow and tedious. Using the option Keyboard Shortcuts with a joystick or a *KoalaPad* speeds up the composing process considerably. To make a composition, simply move the cursor to the musical notation that you want to place on the staff, select it, move it to the staff, and drop it into place. As soon as you set the note in position and release it, the computer plays it for you.

The musical notation—the different notes, rests, naturals, sharps, etc. that you can use to build your composition—determines the actual duration of each note and the space between notes. The program lets you choose five different note and rest lengths ranging from a whole to a sixteenth. A whole note (equal to four beats in any time signature) is twice as long as a half note, and four times as long as a quarter note, etc. You may further control duration with dots and ties—a dot after a note increases its length by one-half, and a tie binds two notes together.

For those of you who need a rest after all this music writing, *MCS* has provided a guessing game. Mystery Melodies reduces preset melodies (or your own) to their rhythmic core by placing all of their notes on one line. The result is a monotone melody that resembles the original song only in its rhythm. Your job is to figure out what the song is from that rhythm. As you may guess, the game is both difficult and inane. The memory could have been used to better ends.

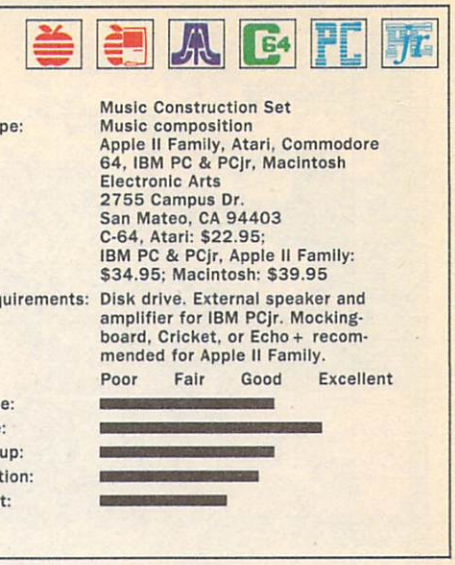
Different Voices

Each computer has different sound-producing hardware, thus each version produces somewhat different sounds. In the case of an unmodified Apple II Family computer or the IBM PC, no sound chip exists. Sound is generated by what is known as "plucking the speaker": PEEKing the hardware address of the speaker causes the speaker to move once. PEEKing it several times per second produces a musical note with a frequency dependent on how many times the speaker moves per second. *MCS* software is written in assembly language, and uses the IBM and Apple central processing units (CPU) to pluck the speaker so quickly that it appears to produce two tones at once. The display of music cannot scroll when two tones are being produced—there is simply not enough time for the CPU to produce the tones and still update the screen display.

If you have a *Mockingboard* (from Sweet Micro Systems), or *Echo II+* or *Cricket* (both from Street Electronics), you can play up to six notes simultaneously and scroll the music. With these Apple add-ons, you also have a choice of eight different musical sounds. The IBM PCjr, the Commodore 64, and the Atari versions of *MCS* are able to produce more tones simultaneously because they each contain dedicated sound chips. The Commodore 64 can scroll the music and play up to three notes simultaneously. In addition, you can determine the sound of the notes by choosing one of 13 musical settings, including eight different instruments and five special-effect sounds.

Using the PCjr, you can play and scroll up to three notes simultaneously. One thing not mentioned in the documentation is that an external speaker and amplifier are necessary to use *MCS* on the PCjr—that is, if you want to hear the music. The Atari version provides a 3-voice option and a 4-voice option, both with scrolling—and gives you a choice of 13 different sounds.

Name:	Music Construction Set
Program Type:	Music composition
Machines:	Apple II Family, Atari, Commodore 64, IBM PC & PCjr, Macintosh
Distributor:	Electronic Arts 2755 Campus Dr. San Mateo, CA 94403
Price:	C-64, Atari: $22.95; IBM PC & PCjr, Apple II Family: $34.95; Macintosh: $39.95
System Requirements:	Disk drive. External speaker and amplifier for IBM PCjr. Mockingboard, Cricket, or Echo+ recommended for Apple II Family.



Beat It

The documentation for *MCS* is brief. It includes short explanations of musical notations with some definitions and descriptions.

There is one section in the instruction manual that, if you have no musical background, is especially difficult to understand: the section on time signatures and beat counters. I had to read it several times before I really understood it, and even then I had a few implementation problems. A time signature is actually two figures, written like a fraction, that determine the number and duration of notes played in each measure. *MCS* uses the time signature to regulate the playback scrolling speed to match the tempo of a piece. *MCS* gives you four different time-signature settings: 2/4, 4/4, 6/8, and 3/4 time. The beat counter is a little box on the monitor screen that counts along as the song plays and changes color whenever there are too few or too many beats for the signature. If the number of notes doesn't match what the time signature specifies, the music's scrolling will not be accurate.

How does *MCS* sound anyway? Not bad, but not great either. The Apple version, with either the *Mockingboard*, *Echo II+*, or *Cricket*, produces the best sounds. The Atari and the Commodore versions are not far behind. The IBM versions bring up the rear, but have the best graphics of the bunch when used with an RGB monitor. The C-64 version is especially disappointing because of its poor use of the C-64's sound chip, and because of its lack of color.

"It's a well-done, tight little program that fulfills most of its promises —but falls short of being a truly useful music maker."

Music Tutor

I see *MCS* as a general introduction to music and your computer, rather than as a serious tool for making music. *MCS* does not allow you to alter sounds—you are limited to selecting from the preset instrument sounds, notes, or special effects. Be that as it may, *MCS* fills a gap between people who have absolutely no music or computer skills, and people who do. The program is very easy to use—although composing a complex song can be time-consuming, even with Keyboard Shortcuts.

MCS seems rather high-priced when compared to similar products, especially products written for the Commodore 64. So what does *MCS* do that makes it stand out? Well, not a whole lot. It's a well-done, tight little program that fulfills most of its promises—but it falls short of being a truly useful music maker. You can be as creative as you want in placing notes on the staff, but the actual process is kind of tiresome. Unless you can read music and really understand what you're doing, you are basically just setting down notes at random.

MCS is a few years old and, compared to what's available now, it seems to be somewhat outclassed. In the last few years, there has been an explosion of software and peripheral products that turn home computers into musical score writers and synthesizers. Electronic Arts needs to make this program a bit more flexible in order for it to stay competitive in such a rapidly expanding market.

Aside from this, *MCS* is a good starter program for learning the basics of computer sounds and musical composition—and it is one of the better music products available for the IBM PC. If you're searching for a program that turns your computer into a synthesizer, *MCS* isn't for you. On some computers—the IBM PC, in particular—no software package can overcome the limited hardware. *MCS* is primarily a *music* program, not a synthesizer package—which may explain its poor sound even on the C-64 and enhanced Apple *II* machines (with *Mockingboard* or *Cricket*!) that have adequate sound capabilities to exploit. But for those of you who want to experiment with music composition, *Music Construction Set* offers you a pleasant way to explore the world of music.

☞ COUNTERPOINT ☜

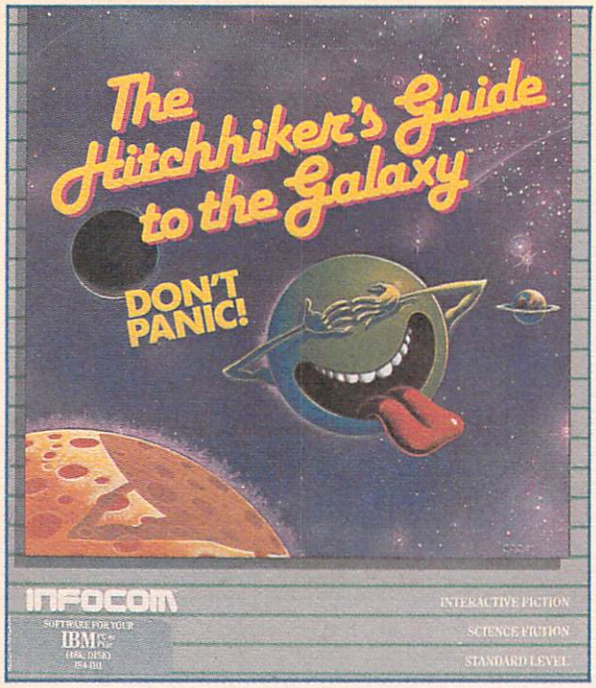
Computers and Music . . . two words we hear a lot these days, and with good reason. Currently, the most innovative and exciting trends in music are sponsored by the electronic revolution. Computers are generally used two ways in music: On a recreational level, a computer combined with special software can be turned into a simple synthesizer. On a more sophisticated level, a computer can control and program dedicated synthesizers and electronic musical instruments.

Music Construction Set definitely falls into the first category. This program strikes me as a perfect example of what I call a *playtool*. *MCS* is more than a toy, but less than a serious music composing and programming instrument. Probably the single most limiting factor is its sound—even on machines with better sound-producing hardware. (To my ears, *MCS* sounded best when used with the Atari and Commodore 64. Even then, the inescapable impression is one of a video arcade game.) For music newcomers seeking interactive entertainment from their computer, working with *MCS* could prove fun and educational. However, anyone with any musical background, as well as total beginners seriously interested in learning electronic music, will quickly outgrow it.

—Andy Widders-Ellis

HCM Glossary terms: synthesizer, sound chip, voices.

HCM



HITCHHIKER'S GUIDE TO THE GALAXY

A Review by Scott Darroch

You won't make it through this trip with just your thumb—you'll need the Guide and more, to survive this sardonically humorous galactic adventure . . .

What an excellent idea! Take a first-rate, out-of-this-world (and several other worlds as well) adventure filled with exotic locales and bizarre aliens, and create an interactive adventure game. This was the challenge undertaken by Infocom Systems when they translated *Hitchhiker's Guide to the Galaxy*, a popular book and PBS/BBC series, into an interactive text-adventure game. As in any attempt to transfer works of entertainment from one medium to another, there are inevitable tradeoffs and losses in translation—but also valuable gains. In this respect, *Hitchhiker's Guide to the Galaxy*, as adapted by author Douglas Adams and game designer Steven Meretzky, is something of a mixed bag.

For those of you who are unfamiliar with the adventure, a brief description is due: The place is Earth. The setting is a day—any day—in the present. Arthur Dent wakes up one morning to find that his house sits in the path of a proposed freeway, and is doomed to destruction. In the course of the day, as he battles the inevitable demolition of his house, Arthur discovers that the entire Earth is to be destroyed to make way for an intergalactic freeway being built by the Vogons. Arthur escapes the Earth's destruction and becomes a Galactic Hitchhiker. He explores the universe with Ford Prefect as a companion, destined for adventures that sparkle with sardonic humor.

Hitchhiker, the game, conforms closely to the book—with one important exception. *You* are Arthur Dent, Galactic Hitchhiker. A word of advice—don't panic!

". . . an astounding, exciting, and above all, humorous universe —where your most casual words might well instigate interstellar conflict . . ."

Playing Ease

Hitchhiker is offered in various levels of difficulty for players aged 9 and up: junior, standard, advanced, and expert. We tried the standard level and found it to be all the challenge a presumably sound-minded adult would desire. It is an intricate, detailed, and demanding game—one in which nearly every conceivable avenue of action has been foreseen, and every consequence described in colorful, descriptive prose.

Your progress in the game is assessed according to points and rounds. Points are scored by actions such as picking up an object or swallowing a tablet. A round consists of one action, command, or question.

As a text-adventure system, *Hitchhiker* operates well. It is equipped with a wide variety of recognized verbs and the ability to accept simple commands, direct requests to other characters, and understand multipart, complex sentences. If the program cannot understand an entry, it specifies whether it does not understand the entire statement, does not recognize a word, or does not understand a word as it has been used in the command—thereby prompting a revision. If you direct a useless or ridiculous action, the system displays a wide variety of droll, smart, and sometimes deadly, responses.

Hitchhiker also boasts a variety of features that simplify the game. Unless you ask *who*, *what*, or *where*, the system automatically prefaces each command sentence with "I want to . . ." to save time in entering commands. Articles (a, the, etc.) need not be entered. Game play is facilitated by the abbreviation of commonly used commands (Wait, Again, and Look are Z, G, and L respectively), and by special purpose commands including Inventory, Diagnose, Save, Restore, and Restart.

The only drawback to the address and command system is that you are not allowed to ask direct questions of Ford Prefect or to make inquiries using the future tense. This requires you to operate with direct commands, or to use one of the three standard question forms. Even a query such as "What happens now?" will be rejected. But, for the most part, the game moves along with few of the frustrating "I don't understand that" replies that can kill a text adventure quicker than an industrial-strength sleeping potion.

The commands Save and Restore come under heavy use, as the game is fatal in many situations—most of them unexpected. At certain points, you will be summarily executed if you have not found the solution to

a particular puzzle. If you save your positions and character as you move through the game, the Restore command will return you to the last-saved position, even if you have been killed or otherwise hopelessly mucked up.

Because the Restore command is limited to returning you to the last-saved position, the Script and Unscript commands are a welcome option. These commands direct your printer to start and stop printing a transcript of your adventure. For most of the situations that you will encounter, a transcript of previous actions and results will prove to be a pearl beyond price.

Some Words of Advice

When playing the game, you must understand its limits and operate within its constraints. It is very important to *look* at everything in the surrounding environment in order to fortify your character with knowledge of the specific details that you encounter. In addition to looking about, you may garner information by smelling, touching, speaking, tasting, and a variety of other actions. The system provides the information through entertaining descriptions and direct responses that both enlighten and challenge the player.

An important maxim to remember is "all things come to he who waits." At many points in the game, all you can do is pass time and try to fathom your latest situation. Many times, repeated use of a command will achieve the desired response, whereas one try will yield only gibberish or a frustrating reply. It also helps to take advantage of commands that produce diagnoses of your physical condition and inventories of your possessions. These checkups help to prompt actions you might never have considered (or been desperate enough to try) before.

Hitchhiker, even at the standard difficulty level, is a challenging series of mental mazes which may require many attempts to navigate—or even to proceed beyond a specific point. In fact, the single major flaw in the system is that the master map and line-by-line hint book (an ingenious device which reveals only one hint at a time) are aftermarket items not included in the software package. The *Hitchhiker* manual says that solving the game will probably take several days. We found that, without using the hint book, it took several days just to escape the earth's destruction (the real beginning of the adventure).

A more-than-passing familiarity with other Infocom games is also valuable. In fact, *HCM*'s most experienced Infocom gamer is the one who unearthed the most important solutions without reference to the hint book (see Counterpoint).

We have an additional hint for Commodore 64 owners. The documentation we received for the C-64 disk incorrectly directs users to load the game by entering `LOAD "GAME",8`. The manual should have stated `LOAD "STORY",8`. [According to Infocom, newly issued packages correct this mistake—Ed.] The documentation for all the other versions appears to be fine. Each version is remarkably similar, and equally humorous.

Don't Panic!

Hitchhiker's Guide to the Galaxy comes in an attractive package, complete with a sales brochure, peril-sensitive sunglasses, pocket fluff, a microscopic space fleet, destruction orders for both your house and Earth, but no tea. Also included is the essential "Don't Panic!" button—required equipment for this adventure.

The hand of *Hitchhiker*'s creator, Douglas Adams, is easily recognizable in the text descriptions and programmed responses. His imaginative and witty approach helps to make even unsuccessful forays into this beautiful and deadly universe an exhilarating and enlightening experience.

Name:	Hitchhiker's Guide To The Galaxy
Program Type:	Text adventure game
Machines:	Apple II Family, Atari, Commodore 64, IBM PC, PCjr, TI-99/4A
Distributor:	Infocom, Inc. 55 Wheeler St. Cambridge, MA 02138
Price:	$33.95
System Requirements:	Disk drive.

	Poor	Fair	Good	Excellent
Performance:				
Engrossment:				
Documentation:				

Can You Take It?

Anyone willing to withstand the rigors of matter transference and the near inevitability of being tortured by a Vogon poetry reading, will find in *Hitchhiker* an astounding, exciting, and above all, humorous universe. Here your most casual words might well instigate interstellar conflict between the V1'Hurg and G'Gugvunt peoples, laying waste to over 250,000 solar systems, and destroying one-quarter of a small galaxy. "Please choose your words more carefully."

Once beyond the confines of Earth, you will be able to hobnob with Trillian and Zaphod Beeblebrox (erstwhile President of the Galaxy), attempt to escape the Bugblatter Beast of Traal (the most ravenous creature in the universe), ponder the mysteries of the Physics of Infinite Improbability, and otherwise have the time of your life—or several lives. So, if you are ready for really high adventure, put on your gown, save your character often, stick out your thumb on the Interstellar Interchange—and don't forget your towel!

☞ COUNTERPOINT ☜

Hitchhiker's Guide to the Galaxy was the perfect book to integrate into Infocom's computer-adventure format. After spending many hours of my life playing Infocom's *Zork* series and reading all four books of Douglas Adams' *Galaxy* "trilogy" (sic), playing this adventure game was quite a pleasure. The humor portrayed in the book is certainly not lost in the game. The authors succeeded in keeping the computer program faithful to the book.

One thing *did* disappoint me—inflexibility. This program is very linear in its play. In order to advance in score and position, things must be done in the proper sequence at the proper time. After playing many of Infocom's other adventures, I found this program to be a bit too limiting in that aspect. I suppose that since books are linear by nature, this hardwired chain of events is unavoidable. In order to live through this galactic adventure, knowledge of the *Hitchhiker's Guide to the Galaxy* book is very helpful.

Overall, this program is very well done. When the second book of the *Hitchhiker* "trilogy" comes out in adventure form, you can be sure that I'll be playing it.

—Randy Thompson

HCM

THE MAGIC OF MUSIC VIDEOS

A Review of the Sight & Sound Music Video Kit

by Laile L. Di Silvestro
HCM Staff

This union of music and graphics places you in the video producer's chair.

The captivating dazzle of television's rock videos has won its way into millions of homes. Who hasn't at times dreamed of perching in the director's seat, at the composer's table, or even on stage? Sight & Sound's *Music Video Kit* for the Commodore 64 provides a chance for the producer in each of us to create sound track, text, and animated graphics for an infinite variety of music videos.

One Step Beyond

Music Video Kit not so much capitalizes on the popularity of television videos as it provides a welcome addition to the extensive line of music software already available for the Commodore 64. [See Vol. 5, No. 2 and Vol. 5, No. 3 for a thorough review of music software available for the Commodore 64—Ed.] One program, Ryo Kawasaki's *Rhythm Rocker*, goes one step beyond such programs by incorporating some exciting, but mostly random, graphics. *Music Video Kit* carries this trend even further by coordinating visual images with the music.

The package consists of two sections—the performer and the graphics editor. The performing mode enables you to piece together videos using backgrounds, actors, and music scores already provided. The second section instructs you on creating your own videos from scratch. In both sections, the graphic and musical performance may be recorded.

In concert with the philosophy behind other Sight & Sound software, *Music Video Kit* is designed to entertain and challenge both the novice and the advanced computer video producer. The software package encompasses three approaches to music video production. The first employs five prerecorded videos. These may be viewed passively or they may be altered. The second approach offers entertainment similar to that of a child's building kit: the components are provided—your role is to decide where they go. The third approach is the most creative as well as the most complex. It consists of an editor mode which enables you to form videos from scratch using animation techniques. Incorporation of Sight & Sound's *Music Processor* provides the option of composing your own sounds to embellish the visual effects.

These approaches reflect a fundamental dichotomy present in the software package: *Music Video Kit* appears to be marketed toward the voracious rock-video viewer who is not likely to delve into complex computing. The procedure for creating animated graphics, however, is intricate and toilsome, and the manual too scant to provide adequate guidance.

The Producer: Limited Choices

Rather than a multitude of lights, cameras, and instruments, with *Music Video Kit* your main tool of production is the joystick. In the first two sections, menus with concise instructions provide choices in backgrounds, actors, and musical scores. Equally lucid menus enable you to create, record, and play performances. Backgrounds range from abstract designs to outer-space scenes and stage settings. The selection of musical scores is relatively limited (only 17 are provided). This limitation may be overcome by using the music provided by Sight & Sound's *Music Processor*, *Music Video Hits*, and *On Stage*, or by creating your own music using the *Music Processor*. These programs are absolutely necessary if the attraction of *Music Video Kit* is to survive beyond the trial run.

". . . even the simplest computer video requires both a creative mind and a patient temperament."

A greater array of actors is available. The choices are fascinating and creative, featuring such oddities as a skating rabbit and a roving eyeball. With the joystick, you move each actor in and out, over and under the background and each other. The actors' movements are automatically timed with the music. The program is easily executed. Slow-down and freeze functions enhance this quality and make this aspect of the package suitable for most age groups.

Name:	Music Video Kit
Program Type:	Sound & graphics integration
Machine:	Commodore 64
Distributor:	Sight & Sound Music Software P. O. Box 27 New Berlin, WI 53151 (800) 558-0910
Price:	$39.95
System Requirements:	Disk drive, joystick

	Poor	Fair	Good	Excellent
Performance:				
Ease of use:				
Engrossment:				
Documentation:				

The Graphics Editor: A Painstaking Process

The Commodore 64 is equipped with the ability to create up to eight separate shapes known as sprites. Sprites are programmable objects that can be moved in an animated fashion over and under each other. You can create sprites with your joystick in the 21 by 24 pixel editing grid. Just move the cursor, via the joystick, over the area that you want to highlight. When you fire the joystick, the pixel under the cursor turns on if it was off when you fired, or off if it was on. The control keys may be pressed to alter the color of the areas thus drawn.

The fundamentals behind the creation of sprites are relatively simple, yet the actual process is cumbersome and the joystick often proves to be an unwieldy tool. The manual's neglect in explaining built-in limitations exacerbates the problems. It is essential that you understand the mechanics of sprites before you attempt detailed animation. Sprites may be created in either high resolution or multicolored mode. In hi-res mode, the sprite is limited to two colors—the "off" color and the "on" color. In multicolored mode, the sprite may have three colors. However, this increase in color is accompanied by a decrease in detail: In order to conserve memory, the pixels are arranged in pairs. Either both pixels are on, or both pixels are off. Thus, figures that look good in hi-res mode might appear bulky and muddled in multicolored mode—a fact not explained in the manual. Imagine the consternation this would produce in the uninformed!

Animation is effected through sequencing separate shapes. Producing intricate movements is a time-consuming and painstaking process, as each minor change in position must be represented by a separate shape. For the most basic videos, one or two separate motions are sufficient (such as opening and closing a mouth or extending and retracting a foot). Yet, even the simplest computer video created with the editor requires both a creative mind and a patient temperament.

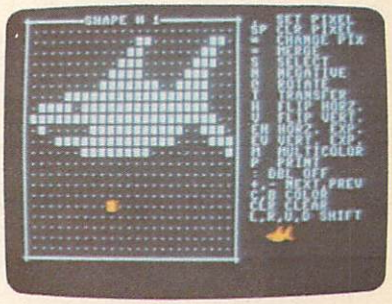


Photo 1. You can create figures by moving the cursor with your joystick and entering the appropriate command listed on the right.

Photo 2. By merging the first figure with this one, you can produce an animated effect.

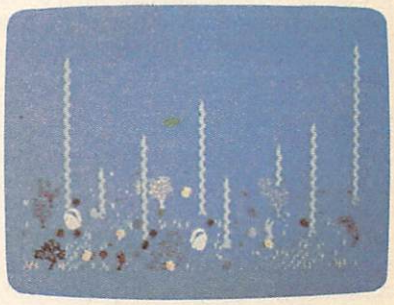


Photo 3. When complete, the animated figure can be transferred onto a preprogrammed background or one you create yourself.

Photo 4. A scene from a preprogrammed video. With imagination and diligence, you can create computer animations such as these.

Defects, Major and Minor

The program has several faults that may be attributed to its novelty. The rather lackluster prerecorded videos do little to inspire. Their elementary quality does not demonstrate the program's full potential, thus providing minimal motivation. Although a replay function is present, few will take advantage of it where these videos are concerned.

The performer section is also marred by both minor and major defects. Although the array of actors, backgrounds, and musical scores is large enough to allay boredom, its stark elementary character does not excite the imagination or dazzle the eye. A more serious problem is a difficulty in recalling the actors chosen to perform in the video. Although the manual vaguely indicates that the recall command `R` must be entered twice (to prepare for recall, and then to recall), it is normally necessary to enter the command several times, and even this does not always produce results.

Ironically, the true value of *Music Video Kit* lies in its difficult graphics-editor section. In this section, *Music Video Kit* has cleared the way for future programs by introducing the musically-inclined to a new level of computer interaction. A viable union of music and graphics in computer videos is no longer waiting around the corner. It is sparkling before our eyes. **HCM**

Apple Seedlings

Frequency Blender

by Roger Wood
HCM Staff

As nature always seems to work from the simple to the complex, the basic sine wave is the building block for all other waveforms. Learn how to blend these simple waves into complex patterns . . .

TUTORIALS

At the foundation of all physical vibration is the common sine wave. When this basic waveform is viewed on an oscilloscope, it resembles evenly spaced—but featureless—rolling hills. However, a pure sine wave, with a constant frequency and amplitude, is quite rare in nature. Most sound waves that we hear are invariably a simultaneous mixture of several frequencies at different amplitudes (heights).

When several sine waves blend together, what does the composite wave look like? You might be surprised at the result. The program listed here allows you to enter up to four frequencies, each with its own amplitude. The program then plots the sine waves of the four frequencies on the screen, followed by a fifth wave pattern. This fifth pattern is the algebraic sum, or "composite" of the first four.

Making Waves

When you RUN the program, it will ask you to enter first a frequency, and then an amplitude for the four sine waves. (For best results, the frequencies you enter should be close enough in value to be within a *viewable* range of each other, though this is not an absolute requirement.) Finally, you will need to specify the number of complete cycles of the lowest frequency wave that you wish to see on screen. This will determine the horizontal scale of the screen—the time scale. For example, if you choose waves with frequencies 100, 200, 300, and 400, and ask for five cycles of the lowest frequency wave, the program will display five of the 100 Hertz (Hz) waves, ten of the 200 Hz. waves, etc.

Any amplitude you enter is relative to the other specified amplitudes. The program totals all of the amplitudes, and then calculates how much "weight" each one has. If frequency 1 has an amplitude of 150 and frequency 2 has an amplitude of 50, then frequency 2 will have only one-third as much effect on the final wave as frequency 1.

One limitation in this program arises when you enter very high frequencies together with low frequencies, and ask to see numerous cycles of the lowest frequency. The display may show a lower-frequency sine wave instead of the expected high-frequency wave, which—if displayed properly—might resemble a thick white bar across the screen. This occurs because the computer is a digital device, not analog. The program takes 640 samples across the screen; if the number of cycles is greater than what can be displayed across one screen, the program may jump whole cycles in the sampling. This causes an erroneous sampling, and can produce a lower-frequency display.

[Note: Because of numerous requests to provide an Apple version of Frequency Blender, which appeared in the "IBMpressions" column of Vol. 5, No. 3, we are now offering such a version here in "Apple Seedlings"—Ed.]

Examples 3 and 4 illustrate the sort of screen display that you will obtain by using this program. For other interesting results, try these two examples:

Example 1:	**Example 2:**
FREQUENCY: 100	FREQUENCY: 100
AMPLITUDE: 100	AMPLITUDE: 100
FREQUENCY: 110	FREQUENCY: 200
AMPLITUDE: 100	AMPLITUDE: 100
FREQUENCY: 120	FREQUENCY: 300
AMPLITUDE: 100	AMPLITUDE: 100
FREQUENCY: 130	FREQUENCY: 400
AMPLITUDE: 100	AMPLITUDE: 100
Number of cycles of Lowest Frequency: 50	Number of cycles of Lowest Frequency: 10

Plotting and Calculating

The program to accomplish the seemingly complicated task of "melding" several waveforms into one is really quite simple.

Two arrays hold the frequency and amplitude values for the four sine waves input by the user: FR( ), and AMP( ). The value you enter for a frequency should always be positive, but the amplitude can be either positive or negative (see Example 4). A negative

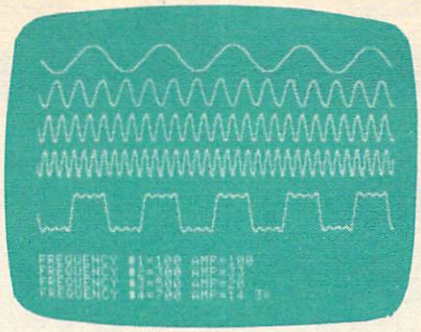
Example 3: *The following values entered into the program result in a close approximation of a square wave:*

FREQUENCY 1: 100
AMPLITUDE 1: 100

FREQUENCY 2: 300
AMPLITUDE 2: 33

FREQUENCY 3: 500
AMPLITUDE 3: 20

FREQUENCY 4: 700
AMPLITUDE 4: 14.3



Number of cycles of Lowest Frequency: 5

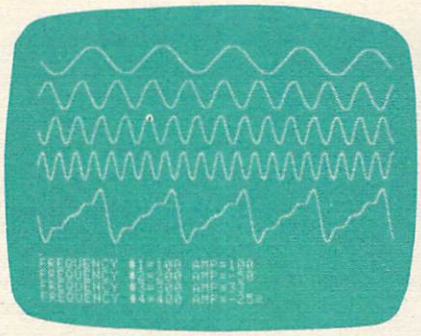
Example 4: *Here are the values to enter for the waveform pictured here. This approaches an alternating sawtooth wave:*

FREQUENCY 1: 100
AMPLITUDE 1: 100

FREQUENCY 2: 200
AMPLITUDE 2:-50

FREQUENCY 3: 300
AMPLITUDE 3: 33

FREQUENCY 4: 400
AMPLITUDE 4: -25



Number of cycles of Lowest Frequency: 5

amplitude creates a waveform 180 degrees out of phase with one of identical frequency and an identical (but positive) amplitude.

We define the function FN F( ) in line 280 to simplify the program. The parameter passed to this function is always a frequency value. The value returned is an offset that determines the vertical position of the next dot to be plotted by an HPLOT command.

Line 290 clears the screen, enables the high-resolution (hi-res) mode, and begins a FOR-NEXT loop (control variable J) for displaying the four sine waves.

Another FOR-NEXT loop (control variable I) controls the line plotting. You will notice that there is only one set of coordinates for the HPLOT command, with the word TO before it. This command plots a line *from* the last position plotted *to* the coordinates specified, thus ensuring a solid line. The control variable I determines the horizontal position of the line's destination. As the function in line 250 calculates each point of a wave, the program saves the value in the two-dimensional array FQ(,). The first dimension is the waveform number (1-4); the second dimension is a horizontal coordinate from 0 to 279.

It is important to note that the amplitude value affects only the display of the composite wave. Each of the four sine waves is displayed at the same height regardless of amplitude. The value returned by the function in line 280 is set to a standard size for easy viewing. An offset is then added to position the wave at its proper place on the display. Only the sign of the amplitude affects a sine wave's phase—i.e., negative waves are 180 degrees out of phase with positive ones.

Line 320 calculates the variable TA: the sum of all the amplitudes. The value added to TA is the ABSolute value of the amplitudes for each frequency. The ABS function converts a negative number to its positive counterpart (e.g. – 100 becomes 100). When we divide each frequency's amplitude by TA, we can determine that frequency's relative "weight" in the total amplitude of the composite wave. The control variable (I) of the FOR-NEXT loop in line 330 determines the horizontal position of the HPLOT statement. The vertical position is calculated by using the correct element of the FQ(,) array and multiplying by the correct weighted amplitude ((AMP(x)/TA)), where x is the number of each sine wave.

KEY VARIABLES

Here are the major variables used in *Frequency Blender*: **FR( )** = input frequency for each sine wave; **AMP( )** = input amplitude for each sine wave; **CY** = number of cycles of lowest frequency to be displayed; **FQ(,)** = two-dimensional array to hold values figured by function **FN F( )**; **FM** = highest frequency to be displayed; **TA** = sum of the four amplitudes; **OF** = offset used in drawing the four sine waves; **I** and **J** are loop-counter variables.

LISTING ANNOTATIONS

Frequency Blender (Apple II family)

Line Nos.

100-190	Program header.
200-220	Protect hi-res sceen.
230-270	Get input values.
280	Function to calculate vertical offset.
290-310	Calculate and display four sine waves.
320-330	Calculate and display composite wave.
340	Wait for keypress to continue.
350-370	Subroutine to display values as waves are displayed.

```
100  REM  ********************
110  REM  *  FREQUENCY BLENDER  *
120  REM  ********************
130  REM  COPYRIGHT 1985
140  REM  EMERALD VALLEY PUBLISHING CO.
150  REM  BY WILLIAM K. BALTHROP
160  REM  AND ROGER WOOD
170  REM  HOME COMPUTER MAGAZINE
180  REM  VERSION 5.5.1
190  REM  APPLE II FAMILY APPLESOFT
200  IF PEEK (104) = 64 THEN 230
210  POKE 104,64: POKE 103,1: POKE 16384,0
220  PRINT CHR$ (4);"RUN FREQ.BLEND"
230  TEXT : HOME : DIM FQ(4,279): FOR I = 1 TO 4: PRINT "ENTER FREQUENCY #";I;: INPUT FR(I): PRINT "ENTER AMPLITUDE:";: INPUT AMP(I): PRINT
240  IF FR(I) > 0 AND FM = 0 THEN FM = FR(I): GOTO 260
250  IF FR(I) < FM AND FR(I) > 0 THEN FM = FR(I)
260  NEXT
270  PRINT : PRINT "HOW MANY CYCLES OF THE LOWEST FREQUENCY DO YOU WISH TO SEE DISPLAYED: ": INPUT "CYCLES: ";CY: CY = CY * 2.85714289
280  DEF FN F(F) = SIN (I * (FR(F) * .01 / (FM / CY) * SGN (AMP(F)))))
290  HOME : HGR : HCOLOR= 3: FOR J = 1 TO 4
300  IF FR(J) > 0 THEN GOSUB 350: OF = 20 + (25 * (J - 1)): HPLOT 0,20 + (25 * (J - 1)): FOR I = 0 TO 279: FQ(J,I) = FN F(J): HPLOT TO I, FQ(J,I) * 10 + OF: NEXT I
310  NEXT J
320  TA = 0: FOR I = 1 TO 4: TA = TA + ABS (AMP(I)): NEXT
330  HPLOT 0,130: FOR I = 0 TO 279: HPLOT TO I, (FQ(1,I) * (AMP(1) / TA) + FQ(2,I) * (AMP(2) / TA) + FQ(3,I) * (AMP(3) / TA) + FQ(4,I) * (AMP(4) / TA)) * 25 + 130: NEXT
340  GET K$: RUN
350  VTAB 20 + J: PRINT "FREQUENCY #";J;"=";FR(J);" AMP=";AMP(J);
360  IF J < 4 THEN PRINT
370  RETURN
```

HCM

C64 Screen Dumping

```
100 REM ********************
110 REM * SCREEN DUMP  *
120 REM ********************
130 REM COPYRIGHT 1985
140 REM EMERALD VALLEY PUBLISHING CO
150 REM BY RANDY THOMPSON
160 REM HOME COMPUTER MAGAZINE
170 REM VERSION 5.5.1
180 REM COMMODORE 64 BASIC
190 FOR I=49152 TO 49367:READ D:C=C+D:P
    OKE I,D:NEXT
200 IF C<>30457 THEN PRINT"**  ERROR  IN
    DATA STATEMENTS  **":END
210 SYS 49152:PRINT"SCREEN DUMP ACTIVAT
    ED!!"
220 DATA 120,173,20,3,141,215,192,173
230 DATA 21,3,141,216,192,169,25,141
240 DATA 20,3,169,192,141,21,3,88
250 DATA 96,165,145,201,251,208,11,165
260 DATA 197,201,41,208,5,173,214,192
270 DATA 240,3,108,215,192,206,214,192
280 DATA 169,127,162,4,160,255,32,186
290 DATA 255,169,0,32,189,255,32,192
300 DATA 255,162,127,32,201,255,162,145
310 DATA 173,24,208,41,2,240,2,162
320 DATA 17,138,32,210,255,169,0,133
330 DATA 251,169,4,133,252,169,25,141
340 DATA 221,192,169,40,141,222,192,160
350 DATA 0,177,251,32,171,192,174,219
360 DATA 192,240,5,169,18,32,210,255
370 DATA 173,220,192,32,210,255,174,219
380 DATA 192,240,5,169,146,32,210,255
390 DATA 230,251,208,2,230,252,206,222
400 DATA 192,208,212,169,13,32,210,255
410 DATA 206,221,192,208,197,32,204,255
420 DATA 169,127,32,195,255,238,214,192
430 DATA 76,188,254,162,0,142,219,192
440 DATA 201,128,144,7,162,1,142,219
450 DATA 192,41,127,201,32,176,4,9
460 DATA 64,208,15,201,64,144,11,201
470 DATA 96,176,4,9,32,208,3,24
480 DATA 105,64,141,220,192,96,0,0
```

Whether it's a spreadsheet or an alien invasion, the ability to make a hardcopy of the computer's screen is a very useful feature. But with the Commodore, the program you are using determines whether a screen dump is possible, unless you have the short program listed here. This routine allows you to get a printout of your screen almost any time, simply by pressing **[CTRL] P**. It works in direct mode, as well as when a program is running.

To implement the screen-dump feature, you must first **LOAD** and **RUN** the accompanying program. Once **RUN**, the machine language for the dump routine is stored in memory at 49152 ($C000) to 49367 ($C0D7). Now, you can **LOAD**, **RUN**, **SAVE**, or just plain bang on the keyboard, and the screen-dump feature will remain intact. Whenever you want a screen dump, all you have to do is press **[CTRL] P**.

Every 60th of a second, the computer executes an **IRQ** (Interrupt **ReQ**uest) routine. This routine takes care of such things as reading the keyboard and flashing the cursor. In order for the screen dump routine to be able to check for a **[CTRL] P**, the normal **IRQ** routine has to be intercepted. Then, a **[CTRL] P** can be detected at any time during the computer's operation. Once **[CTRL] P** is pressed, a flag is set telling the computer to halt execution and print to the screen. This flag also keeps the print routine from being interrupted by another **[CTRL] P**.

The actual printing of the screen is accomplished in two steps. First, the program reads the Commodore's screen memory—located at 1024 ($0400) to 2023 ($07E7)—to get the screen code of a character. Second, the screen code is converted into ASCII and sent to the printer. During the conversion to ASCII another flag is used to determine whether the character should be printed in reverse. If the flag is set, the printer code for reversed characters is sent and the flag is cleared.

When using this routine, don't try to print a screen dump while accessing tape or disk. Both of these routines require accurate timing—timing that can be disrupted by the dump routine. Also, because the printer is opened with a logical file number of 127, any file using this same file number should be **CLOSE**d before you print a screen dump. Finally, this routine prints only the Commodore's text screen. If you are using the high-resolution screen, or if the text screen has been relocated, this screen dump will not work properly. Now, there's just one more thing to remember before using this routine—to turn on the printer.

—Randy Thompson

HCM Glossary terms: ASCII, interrupt request

Popping The Return Stack

TI Extended BASIC gives you the option of including your own error-handling routines in your programs. A common problem that you may encounter when doing so involves the pending of several **RETURN**s. When your BASIC program branches to a subroutine from a **GOSUB**, it places the return address in a buffer called the stack. The number of return addresses that can be placed in the stack is limited—if you place too many return addresses in the stack, you will get a **MEMORY FULL** error. To avoid this error, you must have the program **RETURN** from any subroutine called with a **GOSUB**.

```
100 !  *********************
110 !  *  ERROR RECOVERY   *
120 !  *********************
130 !  COPYRIGHT 1985
140 !  EMERALD VALLEY PUBLISHING CO.
150 !  BY WILLIAM K. BALTHROP
160 !  HOME COMPUTER MAGAZINE
170 !  VERSION 5.5.1
180 !  TI EXTENDED BASIC
190 !
200 ON ERROR 290
210 CALL CLEAR
220 A=A+1
230 PRINT A
240 GOSUB 250
250 GOSUB 260
260 GOSUB 270
270 PRIT "PRINT IS MISSPELLED"
280 PRINT "IT WON'T MAKE IT HERE"
290 PRINT "THIS IS THE ERROR TRAPPING
    ROUTINE. IT WILL POP THE":"PENDING
    RETURNS OFF THE":"STACK AND GOTO TH
    E START OF THE PROGRAM."
300 ON ERROR 320
310 RETURN 310
320 FOR TIMEDELAY=1 TO 500 :: NEXT TIME
    DELAY :: GOTO 200
```

A problem arises when your program branches to an error routine because of the **ON ERROR** statement: **RETURN**s remain pending in the stack. This could happen at almost any point in a program. To remedy this, you can simply exit the error routine with a **RETURN**. This method takes you back to the line that created the error, but it only works properly if the program has been able to correct the error in the error routine.

An alternative would be to use the **RETURN NEXT** statement to return the program to the statement immediately following the one which caused the error. This method works in many cases, but if an error is likely to repeat indefinitely, another solution is needed.

A third alternative is to use **RETURN** *line number*. When you specify a line number after the **RETURN** statement, program execution branches to that line. This method gives you absolute control over the exiting of the error routine. However, returning in this way pops only one return address off the stack. If the line on which the error occurred is embedded deeper than the line you return to, then return addresses remain on the stack. This may result in a **MEMORY FULL** message or in the program **RETURN**ing to an unpredictable location.

There is a way to clear the return stack, but you should use it with caution—if you don't do it properly, you may end up with a **RETURN WITHOUT GOSUB** error message. After clearing the return stack, you must go back to a place in the program that has no **RETURN** pending. You should not branch back to a subroutine. If you do, then the **RETURN** in that subroutine will cause an error message because the return stack has been cleared.

By telling the **RETURN** *line number* statement to **RETURN** to itself, you create a loop that repeats until the return stack is empty and a **RETURN WITHOUT GOSUB** error is generated. By trapping this new error, you can then branch back to a nonsubroutine part of your program with a clear stack.

The program listed here demonstrates this procedure. In line 200, the initial **ON ERROR** statement sets up the error trapping for the main body of the program. Lines 240, 250, and 260 place three return addresses on the gosub stack. Line 270 contains an intentional misspelling which causes a syntax error and sends the program to line 290 (the error routine). Line 280 can never be reached because of the error in line 270. Line 290 prints a message letting you know that it has entered the error routine. Line 300 sets up a new line to branch to in case of an error. Line 310 of the program loops back on itself continuously, taking a return address off of the return stack with every pass. When the stack becomes empty, an error is generated, and program flow branches to line 320. Line 320 simply contains a time-delay routine and a **GOTO** back to the start of the program. Lines 300 through 320 can be lifted and adapted to almost any error routine.

—William K. Balthrop

HCM Glossary terms: error routine, stack, subroutine, syntax error

Commodore Hornblower

Ring Modulation & Synchronization

Get in sync with the SID chip and ring out some wild effects on the C-64.

by Randy Thompson
HCM Staff

TUTORIALS

Zing! Beeeezh! Pop! Ffzowee! New computer buzz words? Nope, these words are simply a feeble attempt to describe some of the numerous sound effects that Commodore's SID chip is capable of producing. The most impressive sound effects produced on the Commodore 64 (such as those listed above) are usually created with the aid of one or both of the SID chip *special effects:* ring modulation and synchronization. Of course, to fully appreciate these sounds, you have to hear them. With this program, you will be able to create and hear these sounds—as well as sounds that the most expressive writer can't convey through words.

Ringmodu-what?

Ring modulation and synchronization are sound effects that allow you to combine the frequencies of two different voices. Synchronization correlates two frequencies so that at the start of each new cycle of the "master" frequency, the "slave" frequency also begins a new cycle. With the proper frequency settings, synchronization can produce some bizarre sounds. Ring modulation, my personal favorite, suppresses the two original frequencies and outputs their sum and difference. This produces discordant overtones, and results in a percussive, gong-like tone. Ring modulation works only when used with a triangle waveform.

This is the program's main screen. Notice that Sustain and ring modulation are both activated.

Don't worry—*understanding* these two effects is not a prerequisite to *using* them. Experimentation is the secret behind putting these two sound modifiers to work.

Adjusting The Controls

The *Ringsync* program is pretty much self-explanatory—all options and directions on accessing them are displayed on the screen (see screen photo). Press the A through G keys to play the notes A though G. If you want to play a C sharp, hold down the [SHIFT] key in conjunction with the C key. You can also move up and down an octave with the + and - keys. Toggle Sustain on and off by pressing S. Select special functions with the function keys. F1, for example, allows you to activate ring modulation. Remember, ring modulation will be active only if you are using a triangle waveform. You can set synchronization into action by pressing F3. To select a different waveform, press F7. As the waveform changes, the current waveform is printed to the screen along with its graphic representation. If you choose a pulse waveform, then you must also enter a pulse width. The pulse width can range between 0 percent and 99 percent. [For a more detailed description of waveforms, refer to "Commodore Hornblower" in *HCM* Vol. 5, No. 2—Ed.] Whenever you want to quit, press the ← key located in the upper-left corner of the keyboard.

Two Different Frequencies

As mentioned before, both ring modulation and synchronization are affected by two frequencies. In this program, these two frequencies are provided by voices 1 and 3. The keys A through G control the first frequency (voice 1). Pressing (F5) changes the second frequency (voice 3). When you press (F5), you are prompted to enter the new frequency. This frequency can range between 0 and 65535 (this is not a measure of cycles per second—or Hertz—but a number relating to the SID chip register). Press [RETURN] when you are finished setting voice 3. When you first RUN *Ringsync*, voice 3's frequency is set to a default value of 2000. To fully exploit the various capabilities of ring modulation and synchronization, try using a wide variety of frequency settings. Remember, the best sounds are always discovered through experimentation.

HCM Glossary terms: frequency, fundamental, harmonic, overtone, register, ring modulation, SID, synchronization, waveform.

KEY VARIABLES

The key variables in the *Ringsync* program are: **CR** = Control Register; **AD** = Attack Decay; **SR** = Sustain Release; **VL** = Volume; **OC** = Octave; **SU** = Sustain (on or off); **W** = Waveform; **NT%** = SID Frequency values; **L1** = Voice 1's low frequency; **H1** = Voice 1's high frequency; **L3** = Voice 3's low frequency; and **H3** = Voice 3's high frequency. The control register, **CR**, is used mainly to turn sounds on and off, but it is also used to set ring modulation and synchronization.

HCM

LISTING ANNOTATIONS

Ringsync (C-64)

Line Nos.	
100-180	Program header.
190-220	Initialize program.
230-420	Main control loop.
430-510	Change waveform.
520-540	Set ring modulation.
550-570	Set synchronization.
580-600	Change voice 3's frequency.
610-620	Print ON or OFF.
630-730	Display main screen.
740-800	Print current waveform.
810-860	Change pulse width.
870	Update control register.
880	Plot cursor at R,C.
890	Erase bottom of screen.
900-960	Input routine.
970-1130	Initialize variables and functions.

```
100
110
120
130
140
150
160
170
180
190
200

210
220

230
240
250
260

270

280

290

300

310

320

330
340
350
360
370

380
390
400
410
420
430

440

450
460
470

480

490

500

510
520

530
540
550

560
570
580
590

600

610

620
630
640

650
```

```
REM  ********** 
REM  * RINGSYNC *
REM  ********** 
REM  COPYRIGHT 1985
REM  EMERALD VALLEY PUBLISHING CO
REM  BY RANDY THOMPSON
REM  HOME COMPUTER MAGAZINE
REM  VERSION 5.5.1
REM  COMMODORE 64 BASIC
POKE 53280,15:POKE 53281,0
PRINT "{SHIFT CLR}{CTRL WHT}{2 CRSR}PRESS [SPACE] TO CHANGE WAVEFORM"
PRINT "{CRSR DOWN}  PREPARING SID POKE VALUES":PRINT "{CRSR DOWN}  PLEASE WAIT"
GOSUB970:GOSUB630:GOSUB230
POKE VL(0),0:POKE 198,0:PRINT"{SHIFT CLR}BYE BYE. SEE YOU NEXT TIME":END
IF PEEK(653)=0 THEN SH=0:GOTO250
SH=1
K=PEEK(197):IF K=64 THEN230
NT=(-1)*(((K=20)+(3*(K=18))+(5*(K=14))+(6*(K=21)))):IF NT THEN TN=NT:GOTO350
NT=(-1)*((8*(K=26))+(10*(K=10))+(12*(K=28))):IF NT THEN TN=NT:GOTO350
IF K=40 THEN OC=OC+1:IF OC>7 THEN OC=7
IF K=43 THEN OC=OC-1:IF OC<0 THEN OC=0
IF K=40 OR K=43 THEN FOR DE=1 TO 50:NEXT:IF SU GOTO350
IF K=13 THEN SU=NOT(SU):R=8:C=13:TF=SU:GOSUB610
IF K>2 THEN ON K-2 GOSUB430,520,550,580
IF K=57 THEN RETURN
GOTO230
IF NT=0 THEN NT=TN
NT=NT+SH:IF NT=13 THEN NT=1
POKE L1,NT%(NT,OC,0):POKE H1,NT%(NT,OC,1)
CR(1)=CR(1)OR1:GOSUB870
K1=PEEK(197):IF K1=K THEN390
IF SU THEN230
CR(1)=CR(1)AND 254:GOSUB870
GOTO230
GOSUB890:GOSUB880:PRINT " PRESS [SPACE BAR] TO CHANGE WAVEFORM"
PRINT" PRESS [RETURN] TO GO TO PLAY MODE"
IF PEEK(197)=1 THEN GOSUB890:RETURN
IF PEEK(197)<>60 THEN450
IF CR(1) AND 16 THEN CR(1)=(CR(1)AND 254):W=2:GOTO510
IF CR(1) AND 32 THEN CR(1)=(CR(1)AND 253):W=3:GOTO510
IF CR(1) AND 64 THEN CR(1)=(CR(1)AND 251):W=4:GOTO510
IF CR(1) AND 128 THEN CR(1)=(CR(1)AND 247):W=1
GOSUB870:GOSUB740:GOSUB870:GOTO430
IF CR(1)AND4THENCR(1)=CR(1)AND251:TF=0:GOTO540
CR(1)=CR(1)OR4:TF=1
GOSUB870:R=10:C=21:GOSUB610:RETURN
IF CR(1)AND2THENCR(1)=CR(1)AND253:TF=0:GOTO570
CR(1)=CR(1)OR2:TF=1
GOSUB870:R=11:C=21:GOSUB610:RETURN
R=12:C=18:TE=5:POKE198,0:GOSUB900
IF TE>65535 THEN S$="65535":TE=65535:GOSUB880:PRINTS$
P=INT(TE/256):POKE H3,P:POKE L3,TE-P*256:RETURN
GOSUB880:IF TF THEN PRINT"{CTRL RVS ON}ON":RETURN
PRINT"OFF":RETURN
PRINT "{SHIFT CLR}  READY"
PRINT "{CRSR DOWN}  PRESS LETTER TO PLAY THAT NOTE"
PRINT" PRESS SHIFT TO SHARP NOTE 1/2 STEP"
```

```
660
670
680
690
700
710
720
730
740
750
760
770
780
790
800
810
820
830
840
850
860
870
880
890
900
910
920
930
940
950
960
970
980
990
1000
1010
1020
1030
1040
1050
1060
1070
1080
1090
1100
1110
1120
1130
```

```
PRINT "{CRSR DOWN}  PRESS + TO GO UP AN OCTAVE"
PRINT "  PRESS - TO GO DOWN AN OCTAVE"
PRINT "{CRSR DOWN}  SUSTAIN IS OFF-{SHIFT C}PRESS S TO CHANGE"
PRINT "{CRSR DOWN}  RING-MODULATION IS OFF-{SHIFT C}F1 TO CHANGE"
PRINT "  SYNCHRONIZATION IS OFF-{SHIFT C}F3 TO CHANGE"
PRINT "  VOICE 3'S FRQ = 02000 -{SHIFT C}F5 TO CHANGE"
PRINT "{CRSR DOWN}  PRESS F7 TO CHANGE THE WAVEFORM";
IF W=1 THEN WV$="TRIANGLE {SHIFT M}{SHIFT N}"
IF W=2 THEN WV$="SAWTOOTH {SHIFT N}"
IF W=3 THEN WV$="PW$+"% PULSE {SHIFT L}{SHIFT O}"
IF W=4 THEN WV$="NOISE #1*%"
R=14:C=0:GOSUB880:PRINT "{CRSR DOWN}  PRESENT WAVEFORM IS A ";WV$
IF W=3 THEN GOSUB810:POKE PL(0),PL(1):POKE PH(0),PH(1)
RETURN
PW$=PW$:POKE198,0
GOSUB890:GOSUB880:PRINT " SELECT PULSE WIDTH OR PRESS [RETURN]"
PRINT" TO ACCEPT SHOWN VALUE.":PRINT" INPUT NEW PULSE WIDTH: ";S$;"%"
R=19:C=24:TE=2:GOSUB900
R=15:C=24:GOSUB880:PRINT S$;
PL(1)=FNL(TE):PH(1)=FNH(TE):PW$=S$:RETURN
POKE CR(0),CR(1):RETURN
POKE 781,R:POKE 782,C:POKE 783,.:SYS 65520:RETURN
R=17:C=0:GOSUB880:PRINT CL$:PRINT CL$:PRINT CL$:RETURN
GOSUB880:GOSUB950:FOR I=1TO TE:GOSUB930:IF K$=CR$ THEN920
NEXT:GOTO900
GOSUB960:RETURN
POKE204,0:POKE207,0:GETK$:IF(K$<"0" OR K$>"9")ANDK$<>CHR$(13)THEN930
POKE204,1:PRINTK$;:RETURN
P=1024+R*40+C:FOR I=P TO P+TE:POKE I,PEEK(I)AND127:NEXT:GOSUB880:RETURN
S$="":FOR I=P TO P+TE-1:S$=S$+CHR$(PEEK(I)):NEXT:TE=VAL(S$):RETURN
DIM PL(1),PH(1),CR(1),AD(1),SR(1),VL(1),NT%(12,7,1)
S=54272:L1=S:H1=S+1:L3=L1+14:H3=H1+14:SU=0:W=2
PL(0)=S+2:PH(0)=S+3:CR(0)=S+4:AD(0)=S+5:SR(0)=S+6:VL(0)=S+24
CR(1)=32:AD(1)=21:SR(1)=165:VL(1)=15
CL$="
                                                                                "
FOR I=S TO VL(0):POKE I,0:NEXT
DEF FN PF(FR)=FR/.06096:DEF FN FR(BA)=BA*(1.0594630?)
DEF FN HF(FR)=INT(FR/256):DEF FN LF(FR)=INT(FR-(256*(INT(FR/256)))):FQ=16.351597?:FOR OC=0 TO 7:FOR NT=1 TO 12
NT%(NT,OC,0)=FN LF(FN PF(FQ))
NT%(NT,OC,1)=FN HF(FN PF(FQ)):NEXT NT:NEXT OC
DEF FN H(V)=INT(V*4096/25600)
DEF FN L(V)=INT(V*4096/100-(256*INT(V*4096/25600)))
PL(1)=FNL(50):PH(1)=FNH(50):POKE PL(0),PL(1):POKE PH(0),PH(1):OC=4
POKE AD(0),AD(1):POKE SR(0),SR(1):POKE VL(0),VL(1):POKE CR(0),CR(1)
POKE L3,208:POKE H3,7:RETURN
```

IBM PRESSIONS

Shadow Graphics

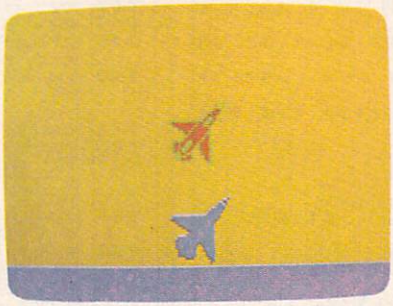
by William K. Balthrop
HCM Staff

3-D graphics can be simple when you apply a few tricks of the trade. Read on and discover keys to the mysterious world of computer graphics.

Have you ever gazed upon a good 3-D graphics program, wondering how the computer produces such a realistic scene? Quite often the method used to create the screen is much simpler than the innocent bystander might assume. Over the years, programmers have concocted a number of tricks to enable the two-dimensional computer screen to appear as a three-dimensional scene.

Adding a Third Dimension

One method of 3-D illusion involves shading. By giving an object a shadow, you convey a strong feeling of depth. This shadow, by its very nature, can provide the viewer with an added dimension.



This sample screen illustrates the illusion of depth that you can create by giving an object a shadow.

Take the problem of displaying a plane flying over the ground. It is an easy matter to show two of the plane's dimensions, such as its X and Y coordinates over the surface—its geographical location. But how can you show its altitude? You can't depict a change in altitude by moving the plane up and down the screen, because it's easy to lose the previous perspective. The plane could literally be at any altitude, regardless of its position on the screen, if there is no reference point.

The shadow makes a perfect reference point for an object. It provides the exact geographical location of the plane as long as we place the light source, such as the sun, directly above the plane.

In addition to locating the plane at a certain altitude, the shadow can indicate changes in altitude. Obviously, the higher the plane's altitude, the farther the shadow is from the plane.

You can also use the shadow as a reference point to determine whether the plane has landed—when both the shadow and the plane are at the same coordinates.

You can create the shadow effect on the IBM PC using BASICA, the IBM PCjr using Cartridge BASIC, or BASIC on the Tandy 1000. If the shadow is to be stationary, it might be most efficient to draw the shadow with the `DRAW` or `LINE` commands. If you need to animate the shadow, as you will with the plane, the best method is to use the `PUT` command. This allows you to quickly move blocks of graphics across the screen. The `PUT` command uses an array that stores an image from the screen. When you use this command, you must specify where on the screen you want the upper-left corner of the image to appear. For example,

```
PUT (100,100),A%
```

places the image stored in the integer array `A%()` on the screen, with the upper-left corner of the image at location 100,100.

Adding Color

It takes two bits of computer memory to determine the color of each pixel in medium-resolution mode (SCREEN 1). The pattern of those two bits determines the color of the pixel on the screen. (With two bits, four combinations are possible.) The actual color that appears also depends on the palette that you select. The chart below illustrates the bit pattern and color choices for the two available palettes:

Bits	#	Palette 1	Palette 2
00	0	BACKGROUND	BACKGROUND
01	1	GREEN	CYAN
10	2	RED	MAGENTA
11	3	BROWN	WHITE

A number of options can be used with the `PUT` command to get different special effects. The default mode is `XOR`. This option inverts the color of any pixel on the screen that matches a bit in the image when that bit is set to 1. For example, if you were to place a red pixel (10) on top of a brown pixel (11), the result would be a green pixel (01). Placing the red pixel on top of the green pixel will turn it back to brown, the original color. Thus, you can place an image on the screen using the `XOR` option, then place that same image on the screen a second time to restore the screen to its original state.

The one drawback to the `XOR` option is that the color of the image will change depending on what the image

is placed over. If this is an undesirable effect, you may prefer to use the PSET option of the PUT command. This option places the image on the screen exactly as it's recorded in the array. Unfortunately, it also erases any graphics that lie beneath the image. If, however, the destruction of the background is not a concern, PSET may be the best method to use.

Dealing with a Dilemma

In the program below, I have not defined any special background that needs to be preserved. This allows me to use the PSET option to place the images on the screen. This option is much quicker to use, and provides less "flicker" than the XOR option. A problem with flicker arises only when the plane and the shadow try to use the same area of the screen. You can alleviate this conflict by putting the shadow image on the screen first, and placing the image of the plane over it. The resulting image flickers back and forth when the plane is in motion, as first the shadow is displayed, then the plane. Because the plane is the last to be displayed, when it stops moving it will obscure the shadow wherever the shadow crosses the square area containing the shape.

"By giving an object a shadow, you convey a strong feeling of depth. This shadow, by its very nature, can provide the viewer with an added dimension."

The only alternative is to use the XOR option in place of the PSET. This presents problems of its own: the shapes flash all of the time, as the old location must be erased before the image can be placed at the new location. In addition, if you fly the plane so that it comes in contact with the shadow, a ghost of the shadow shows through the plane, distorting its color. Because of the additional work that the computer must do, the plane's motion is very slow. After examining both options, I decided to use the first method: PSET.

Try altering the program provided here to see what additional effects you can come up with. You might add a plane to the other side of the screen, control both planes with joysticks, and allow the planes to fire at each other. Now you're on your way to creating your own great 3-D game!

HCM Glossary terms: animate, bit, flicker, integer array, palette, perspective, pixel, shading.

CONTROL CAPSULE

Shadows

KEY	FUNCTION
↑	Increase altitude, move up.
↓	Decrease altitude, move down.
←	Move diagonal up and left.
→	Move diagonal down and right.

KEY VARIABLES

Here's a list of the more important variables in *Shadows*: **J( )** and **JS( )** = graphic arrays; **PX1** = X location of plane; **PX2** = X location of shadow; **PY1** = Y location of plane; and **PY2** = Y location of shadow.

LISTING ANNOTATIONS

Shadows (IBM PC & PCjr, Tandy 1000)

Line Nos.

100-230	Program header.
240	Set up the screen for medium-resolution graphics. Dimension two integer arrays to contain the plane and shadow images.
250	Fill the screen with the color brown.
260	Draw the plane on the screen using the DRAW command.
270	Draw the shadow on the screen.
280	Save the image of the plane in the J array, and the image of the shadow in the JS array, using the GET statement. This statement specifies the size of the image, and the destination (array name).
290	Scan the keyboard until a key is pressed.
300	Check for a legal key press, then branch to the appropriate routine.
310	Routine to move the plane up—increase altitude.
320	Routine to move the plane down—decrease altitude.
330	Routine to move the plane, and the shadow up and to the left.
340	Routine to move the plane, and the shadow down and to the right.
350	Place the shadow on the screen, then place the plane on the screen.

```
100 '  *********
110 '  *  SHADOWS  *
120 '  *********
130 '  COPYRIGHT 1985
140 '  EMERALD VALLEY PUBLISHING CO.
150 '  BY WILLIAM K. BALTHROP
160 '  HOME COMPUTER MAGAZINE
170 '  VERSION 5.5.1
180 '  IBM PCjr with CARTRIDGE BASIC or
190 '  IBM PC with BASICA and
200 '  COLOR/GRAPHICS ADAPTER and
210 '  COLOR MONITOR or
220 '  TANDY 1000 PC with GW BASIC
230 '
240 CLS:KEY OFF:SCREEN 1:COLOR 1,0:DEFINT J:DIM J(100),JS(100)
250 PAINT(170,199),3
260 DRAW "BM110,130C1GDGDG2DG3D14GUH2UHUH2NE6RBR3P2,1C1BL4G3LGL2GRFR3EU3BGP2,1C1BL3LUEU2EUEL3GD3FBU2P2,1C1BE3E2NE6H2LHLH2LER13BL3BDP2,1C1BR3BURE3RE2RERBG5P2,1C1DG2LUE2RG3BG6NG6DG3HE3"
270 DRAW "BM110,160C0GDGDG2DG3D14GUH2UHUH2G2D3GL3HL2U2HU3ER3E2H2LHLH2LER14E3RE2RERBG10P0,0"
280 GET (84,129)-(111,156),J:GET (84,159)-(111,186),JS:PY1=129:PX1=84:PY2=159:PX2=84
290 A$="":WHILE A$="":A$=INKEY$:WEND:A$=RIGHT$(A$,1)
300 K=INSTR("HPKM",A$):IF K=0 THEN 290 ELSE ON K GOSUB 310,320,330,340:GOTO 290
310 IF PY1>0 THEN PY1=PY1-1:GOSUB 350:RETURN ELSE RETURN
320 IF PY1<PY2 THEN PY1=PY1+1:GOSUB 350:RETURN ELSE SOUND 110,2:RETURN
330 IF PX1>0 AND PY1>0 THEN PY1=PY1-1:PX1=PX1-1:PY2=PY2-1:PX2=PX2-1:GOSUB 350:RETURN ELSE RETURN
340 IF PY2<172 THEN PY1=PY1+1:PX1=PX1+1:PY2=PY2+1:PX2=PX2+1:GOSUB 350:RETURN ELSE RETURN
350 PUT(PX2,PY2),JS,PSET:PUT(PX1,PY1),J,PSET:RETURN
```

HCM

Fast X-Ref

Here's a short cross-reference program that generates an indexed list of variables as they occur in Atari BASIC programs. Written entirely in machine language (except for the BASIC loader, of course), this routine can produce a variable cross-reference almost instantly. Furthermore, the program is not limited to the printer as an output device—when **RUN**, it can send the cross-reference to screen, disk, tape, or printer.

To load the cross-reference routine into memory, you must type-in and **RUN** the program listed on the next page. The computer then stores the machine-language program into memory, and prompts you to specify an output device. If you choose disk, the routine saves the cross-reference under the file name "D:X". After you select the output device, the BASIC loader program ends. Remember to **SAVE** the BASIC loader if you want to avoid keying it in again.

You can change the output device after you have **RUN** the BASIC loader, by **POKE**ing location 203 with the ASCII of the desired output device. For example, if you wish to output to the printer, enter:

POKE 203,ASC("P")

To redirect output to the screen, type:

POKE 203,ASC("E")

Of course, you can always reset the output device by re**LOAD**ing and **RUN**ing the BASIC loader.

This machine language program is stored in memory at page 6 ($0600-$06FF). Because this area of RAM is undisturbed by BASIC, you can **SAVE**, **LOAD**, and **RUN** almost any BASIC program, and still leave the cross-reference routine intact. Any time that you want a variable cross-reference of a program, simply **LOAD** that program into your computer and enter **X=USR(1536)**. The computer will begin generating a variable cross-reference immediately.

The program outputs the cross-reference in a standard format: Each variable is listed and immediately followed by a group of line numbers representing the program lines that contain the variable. Each line number is printed once for every time that the variable appears in the program line. So, if a variable is referred to twice in one program line, that line number is listed twice. Due to memory constraints, variables are not alphabetically sorted before they are listed. Instead, the variables are output in the order that you entered them.

The Atari stores variable names in a variable name table. This table contains all the variables that you have entered—even if they do not occur in a program. Because the **CLR** statement does not clear this table, and because there is no garbage collection (a method used by other computers to clear memory of unused variables to allocate space for new ones), the cross-reference may output variables that are not used inside a program. In this case the program prints the variable, but without any line numbers following it. When you **SAVE** programs to tape or disk, the variable name table is saved along with them. So, to truly clear a program of unused variables, you should first **LIST** the program to disk or tape, type **NEW**, and then **ENTER** it back into RAM.

This cross-reference program contains two main routines. The first routine finds each variable by searching through the variable name table. Upon finding a variable, the routine outputs that variable's name, then calls the second routine. The second routine searches through the BASIC program to find all occurrences of that variable. The Atari computer stores BASIC-program variables as token values consisting of the numbers 128 through 255. The position of a variable within the variable name table determines that variable's token value. The first variable listed in the table has a token value of 128, the next one a token value of 129. When the second routine searches for a variable, it simply checks for that variable's token value.

When it finds the variable, the routine outputs the line number of the program line in which the variable occured, and then continues the search. Once the entire program is searched, control returns to the first routine which finds the next variable. The cross-reference program terminates when it reaches the end of the variable name table.

Because certain reverse characters can have the same numeric value as a variable token, the cross-reference program ignores all **REM** and **DATA** statements and strings contained in **PRINT** statements. Avoiding **REM** and **DATA** statements is as easy as skipping over to the next program line. In order to avoid the strings contained in **PRINT** statements, the program searches for the string token (a numeric 15). The number that follows this token represents the length of the string and indicates the number of characters that should be ignored.

—Randy Thompson

HCM Glossary terms: BASIC loader, garbage collection, RAM, token, variable cross-reference.

LISTING ANNOTATIONS

Fast X-Ref (Atari)

Line Nos.	
100-190	Program header.
200	Store machine language.
210-270	Set output device.
280	Display instructions.
290-600	Machine language data.
610	ASCII of output device characters.

```
100 REM ********** 
110 REM * FAST X-REF *
120 REM ********** 
130 REM COPYRIGHT 1985
140 REM EMERALD VALLEY PUBLISHING CO.
150 REM BY RANDY THOMPSON
160 REM HOME COMPUTER MAGAZINE
170 REM VERSION 5.5.1
180 REM ATARI BASIC FOR THE 800, 800XL,
    130XE
190 POKE 82,0:? "■ESC CTRL <■RUNNING...
    ";
200 C=0:FOR I=1536 TO 1791:READ D:C=C+D
    :POKE I,D:NEXT I:IF C<>31423 THEN ?
    "■ESC CTRL <■*** ERROR IN DATA STA
    TEMENTS ***":END
210 POKE 204,58:POKE 205,88:POKE 206,15
    5
220 ? "■ESC CTRL <■SEND CROSS REFERENCE
    TO ";
230 ? :? :? "1) PRINTER":? "2) SCREEN
    ":? "3) DISK":? "4) TAPE":? :? "CHO
    OSE ONE (1-4)":;
240 OPEN #1,4,0,"K:"
250 GET #1,K:IF K<49 OR K>52 THEN 250
260 ? CHR$(K):IF K=51 THEN ? :? "X-REF
    FILE WILL BE SAVED AS 'D:X'":? :? "
    ENTER":? :? "TO GET A CROSS REFERENCE
    ":? :? "X=USR(1536)":END
270 FOR I=1 TO K-48:READ D:NEXT I:POKE
    203,D
280 ? :? :? "TO GET A CROSS REFERENCE,
    ENTER":? "X=USR(1536)":END
290 DATA 165,130,133,0,165,131,133,1
300 DATA 169,127,133,209,162,48,169,203
310 DATA 157,68,3,104,133,82,157,69
320 DATA 3,157,75,3,169,8,157,74
330 DATA 3,169,3,157,66,3,32,86
340 DATA 228,160,0,177,0,16,16,41
350 DATA 127,32,93,6,32,91,6,230
360 DATA 209,32,111,6,32,91,6,32
370 DATA 93,6,230,0,208,2,230,1
380 DATA 165,0,197,132,208,219,165,1
390 DATA 197,133,208,213,169,12,141,114
400 DATA 3,208,15,169,155,162,11,142
410 DATA 114,3,162,0,142,72,3,142
420 DATA 73,3,162,48,76,86,228,165
430 DATA 136,133,2,165,137,133,3,32
440 DATA 244,6,133,207,32,244,6,133
450 DATA 208,201,128,176,121,32,244,6
460 DATA 32,244,6,32,244,6,201,2
470 DATA 176,9,32,244,6,201,155,208
480 DATA 249,240,220,32,244,6,201,20
490 DATA 240,230,201,27,240,226,201,22
500 DATA 240,205,201,14,208,10,162,6
510 DATA 32,244,6,202,208,250,240,227
520 DATA 201,15,208,6,32,244,6,170
530 DATA 208,238,197,209,208,213,165,20
    7
540 DATA 133,212,165,208,133,213,32,170
550 DATA 217,32,230,216,160,255,132,175
560 DATA 230,175,164,175,192,8,240,187
570 DATA 177,243,72,41,127,32,93,6
580 DATA 104,16,237,169,160,164,175,200
590 DATA 145,243,208,228,160,0,177,2
600 DATA 230,2,208,2,230,3,96,0
610 DATA 80,69,68,67
```

ATARI ATRIUM

SOUND ON SOUND

Learn the secret of multi-track recording while laying down some musical tracks of your own.

by Scott Williams
HCM Staff

Many bands create a "big" sound by packing a recording studio full of glittering instruments and highly skilled musicians. The more musicians in the studio, the bigger (and better) the sound—right? This may be true in some cases, but today it's not uncommon to hear that same "big" sound performed by only one person. A one-man band? Not really. The secret to this trick is something called multi-track recording.

Multi-track recording is a technique that allows you to record complex music one instrument (or voice) at a time. As you listen to one recorded track played back, you can record another along with it—and so on, until all the tracks are full. When it's finished, you have a multi-voice composition.

Our program, *Sound on Sound*, is a 4-track recorder for the 4-voice Atari computer. You can record up to 1,000 notes in real time (as you play them) or in step mode (one note at a time). You can also save any tune you record to disk or tape.

When you run the program, the main menu appears on the screen, offering these 4 options:

1) COMPOSE MUSIC
2) PLAY MUSIC
3) SAVE/LOAD FILES
4) EXIT PROGRAM

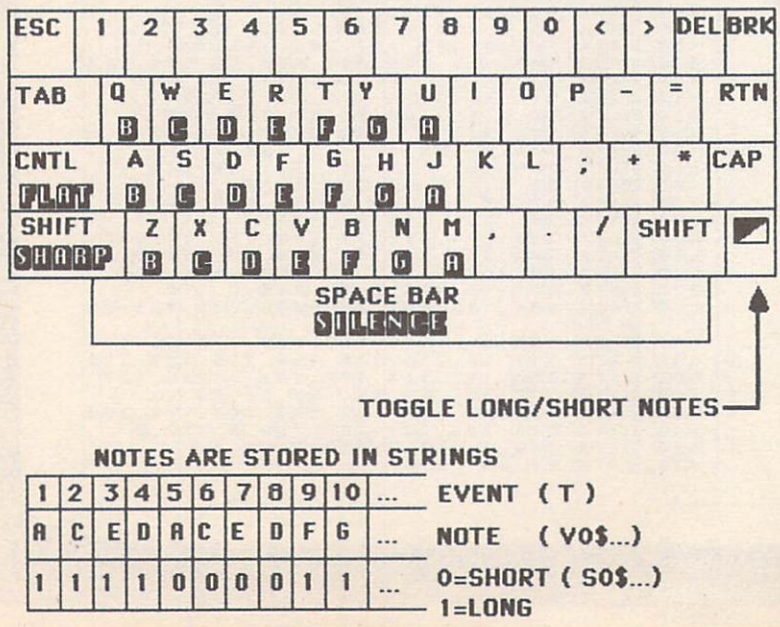
Compose Music

You might not be an accomplished musician, but with this option in *Sound on Sound* you're ready to piece together your own work of art. You can utilize any of the 5 composing tools (on the Compose Music submenu) to create your masterpiece:

1) RECORD IN REAL TIME
2) STEP MODE
3) SET TEMPO
4) CLEAR TRACK
5) SET VOICE TYPE
ESC) RETURN TO MAIN MENU

RECORD IN REAL TIME—This option simply records music exactly as you play it on the keyboard. In *Sound on Sound*, you can employ two kinds of notes—long and short. Long notes flow smoothly from one tone to the next. Short notes produce a distinct pause between each tone. A timer—displayed in the lower-left portion of your screen—runs while you play each note of your composition, as if you were playing a piano next to an inaudible metronome. The timer divides your composition into small segments or "events."

Figure 1.



To begin recording in real time, toggle the Atari key (in the lower-right section of your keyboard) until the desired note selection appears on the screen—either NOTES: LONG or NOTES: SHORT. Next, select a recording track (0,1,2, or 3). If you have recorded anything on the other three tracks, the program plays those voices back for you. You can, however, turn off these tracks, if you don't want to listen to them while you compose another track. (See SET TRACK section below.)

Now you're ready to start recording your composition. But remember that a note is recorded only as the timer advances, so be sure to hold the key down long enough to let the program record the note.

At any time, you can exit this option and return to the Compose Music menu by pressing (ESC).

STEP MODE—In this mode, you can scan forward or backward through a track to write or modify the notes in each event. Press (*) to count forward one event at a time, or press (+) to count backward one event at a time. When you release the key, a note from the event you have selected will play until you change it or turn it off. (Change the note by pressing another key, or turn it off by pressing the space bar.)

SET TEMPO—Select this option to adjust the play-back tempo. The current tempo setting is displayed along with a prompt asking you to enter the new tempo. The maximum tempo setting is 380 events per minute; the minimum is 1. Enter the new tempo, and press (RETURN) to return to the Play Music menu.

CLEAR TRACK—Use this option to completely clear a track. Simply select the track you wish to clear, and the program automatically clears each event on that track. Press (RETURN) if you decide not to clear any tracks.

SET VOICE TYPE—This option allows you to add distortion to the sound of any track (voice). When you select the track you wish to change, a prompt displays the current setting and asks you to enter a new setting. Because of the Atari sound-chip's design characteristics, only even numbers between 0 and 14 produce a tone. Odd-numbers silence the track. Experiment to create different sounds—you'll be surprised by some of the voices.

Play Music

With this main-menu option, you can listen to your newly created masterpiece. The following selections let you alter your composition as it plays:

```
1) SET TEMPO
2) SET TRACK
3) PLAY BACK MUSIC
ESC) RETURN TO MAIN MENU
```

SET TEMPO—This option performs the same function as the SET TEMPO option in the Compose Music Menu.

SET TRACK—This option allows you to turn any of the 4 tracks on or off. When you select this option, it displays the status of the 4 tracks at the bottom of the screen.

To change the track status from ON to OFF, press the number corresponding to the track. Press this key again to return the track to its original state. If you turn off a track, it will be disabled until you turn it on again. In other words, the tracks will remain off, even if you go back to the Compose Music option to create more music. Therefore, this option can be used to silence any unwanted recorded tracks during the creation of your composition. When you have achieved all of the desired settings, press (ESC) to return to the Play Music menu.

PLAY BACK MUSIC—Now you're ready to hear your composition. This selection plays back the tracks you select in the SET TRACK option. Watch out, Amadeus!

Save/Load Files

Select this option from the main menu to save your music to tape or disk or to recall previous work. When you select this option, you see another small menu:

```
1) LOAD DATA FILE
2) SAVE DATA FILE
ESC) RETURN TO MAIN MENU
```

If you select 1 or 2 from this menu, a prompt asks you to enter the desired file name. If you are using a cassette, simply enter C:. If you are using the default disk drive, enter the file name, using a maximum of 8 characters with no punctuation or spaces. To use a disk drive other than drive 1, enter D followed by the device number and a colon (:) before you enter the file name.

Exit Program

This cleverly named option from the main menu allows you to exit the program. But before you exit, the program asks you if you want to save your current recording as a file.

HCM Glossary terms: multi-track recording, track, real time, voice.

For your key-in listings, see HCM PROGRAM LISTINGS Contents.

LISTING ANNOTATIONS

***Sound On Sound* (Atari 800, 800XL, 130XE)**

Line Nos.	
100-190	Program header.
200-310	Initialization and main menu.
320-720	Compose music routine.
730-980	Play music routine.
990-1340	Load and Save data files.
1350-1380	Exit program routine.
1390-1610	Build sound strings, play music.
1620-1650	Clear sound string routines.
1660-1690	Program data.

KEY VARIABLES

These are the key variables for the Sound on Sound program: **V0$, V1$, V2$,** and **V3$** = Strings containing notes for each event; **S0$, S1$, S2$, S3$** = Strings containing flag for long or short notes; **PITCH( )** = List of note values for **SOUND**; **SHARP( )** = List of sharp notes for **SOUND**; **FLAT( )** = List of flat notes for **SOUND**; **KPRESS( )** = ATASCII to sound value array conversion; **CHAN( )** = On/off status of channels; **V( )** = Distortion setting for channels; **T** = Current event (time); **TMP** = Value used in a timing loop for the tempo; **TMPO** = Tempo value set by user.

HCM

TUTORIALS

Screen Dumping

```
100  REM  **********
110  REM  *  SCREEN  DUMP  *
120  REM  **********
130  REM  COPYRIGHT 1985
140  REM  EMERALD VALLEY PUBLISHING CO.
150  REM  BY RANDY THOMPSON
160  REM  HOME COMPUTER MAGAZINE
170  REM  VERSION 5.5.1
180  REM  APPLE II FAMILY APPLESOFT
190  HOME : PRINT "**********
     **********";
200  FOR I = 1 TO 21: PRINT "*" TAB( 40
     )"*": NEXT
210  PRINT "**********
     **********";
220  HTAB 8: VTAB 4: PRINT "===HCM SCRE
     EN DUMP DEMO===";
230  GOSUB 9000
240  HTAB 5: VTAB 12: PRINT "PRESS RETU
     RN TO PRINT THE SCREEN"
250  GET K$: IF K$ < > CHR$ (13) THEN
     250
260  CALL 768: HOME : PRINT "THAT'S ALL
     FOLKS": END
9000 REM INITIALIZE SCREEN DUMP ROUTINE
9010 REM TO USE ENTER: CALL 768
9020 FOR I = 768 TO 926: READ D:C = C +
     D: POKE I,D: NEXT
9030 IF C < > 15518 THEN PRINT "ERROR
     IN DATA STATEMENTS": END
9040 IF PEEK (48905) = 76 AND PEEK (4
     8911) = 0 THEN POKE 869,75: POKE 8
     73,184
9050 RETURN
9060 DATA169,0,133,54,169,193,133,55,16
     9,9,32,237,253,169,56,32,237,253,16
     9,48,32,237,253,169,78,32,237,253,1
     69,0,141,109,3,173,109,3,10,168,185
     ,111,3,133,26,185,112,3,133,27,169,
     0,141,110,3,172,110,3,177,26,201,22
     4,176,8,41,63,201
9070 DATA31,176,4,9,64,41,127,32,237,25
     3,238,110,3,173,110,3,201,40,208,22
     4,169,13,32,237,253,238,109,3,173,1
     09,3,201,24,208,189,169,189,133,54,
     169,158,133,55,96,0,0,0,4,128,4,0,5
     ,128,5,0,6,128,6,0,7,128,7,40,4,168
     ,4,40,5,168,5,40,6
9080 DATA168,6,40,7,168,7,80,4,208,4,80
     ,5,208,5,80,6,208,6,80,7,208,7
```

Adding a screen dump feature to your programs can be very useful. Almost any type of software can be made better by including a screen dump option. But because the order of lines as they appear on the Apple screen is not identical to their order in screen memory, developing an efficient screen dump through BASIC can become quite a task. Here is a dump routine already written for you—in machine language. The listing presented here provides a short subroutine that will store the machine language data into memory starting at 768 ($0300). Designed for use within other programs, this subroutine only has to be executed once to initialize the screen dump routine. Once the screen dump routine is initialized by a **GOSUB 9000**, a **CALL 768** will print the text screen.

To use this dump routine in one of your own programs, merge the desired program with lines 9000-9080 of the listing shown here. Now, before you **CALL 768**, make sure that you have previously executed a **GOSUB 9000**.

This machine language routine works in a fairly straight-forward manner. First, it sets the printer as the output device and turns off the screen echo. Next, the routine executes a loop to read the display value of each character from screen memory. As the routine reads the display value, it converts the value to ASCII and sends it to the printer. At the end of each screen row, the routine outputs a carriage return and then reads the next row. When the whole screen has been printed, output returns to the screen and the routine ends.

Because of the awkward arrangement of Apple's screen memory, we've employed a look-up table that makes this screen dump routine shorter and faster.

—Randy Thompson

HCM Glossary terms: display value, machine language, pointer, screen memory, subroutine.

Debugging the DOS Directory

The supplemental program disk that comes with the IBM PC and PCjr Disk Operating System (MS-DOS 2.10) contains a utility called DEBUG.COM. A simple but useful application of this utility is to directly inspect and modify the directory of a disk. Before you try any modifications, however, heed a word of warning: Don't use your only copy of some treasured disk—use a backup. If you make a mistake in writing a directory back to the disk, your files could be *unrecoverable*. With some caution and a little practice, however, you will be able to hide files, reveal files normally hidden from a directory search, and even make it impossible to delete a file, except from DEBUG.

To introduce you to this application, we have prepared step-by-step instructions on altering a boot disk to reveal a hidden system file: IBMBIO.COM. Before you begin, format a blank disk using the /S option to create a DOS boot disk.

(1) To run DEBUG, place a disk containing DEBUG.COM into drive A and type **DEBUG**, then press [**ENTER**]. The prompt becomes a hyphen (-) when DEBUG is running.

(2) Next, place the disk you formatted into drive A and type **L 100 0 5 1** to **L**oad **1** sector (sector **5**—the beginning of the directory), into memory at address $**100** of the current data segment (the $ indicates a hexadecimal number—all numbers in DEBUG are hexadecimal).

Address	Hexadecimal Dump of Memeory																ASCII of Memory
08F1:0100	49	42	4D	42	49	4F	20	20	43	4F	4D	27	00	00	00	00	IBMBIO COM'....
08F1:0110	00	00	00	00	00	00	00	00	60	54	07	02	00	80	12	00	........'T.....
08F1:0120	49	42	4D	44	4F	53	20	20	43	4F	4D	27	00	00	00	00	IBMDOS COM'....
08F1:0130	00	00	00	00	00	00	00	00	60	54	07	07	00	80	42	00	........'T....B.
08F1:0140	43	4F	4D	4D	41	4E	44	20	43	4F	4D	20	00	00	00	00	COMMAND COM ....
08F1:0150	00	00	00	00	00	00	00	00	60	54	07	18	00	80	45	00	........'T....E.
08F1:0160	00	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	.vvvvvvvvvvvvvvv
08F1:0170	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	F6	vvvvvvvvvvvvvvvv

Figure 1

(3) Now type **D 100** to **D**ump the first $80 bytes of the memory that you just loaded to the screen. Figure 1 illustrates your screen display at this point. It consists of three data fields: The left field contains the memory addresses of the information displayed in the next two fields. $08F1 is the data segment register contents—the only data segment you will encounter in this example. $0100, $0110, etc. are the offset addresses into the data segment. Because you specified **100**, DEBUG displays information starting from that address. The field in the center contains the next $10 bytes of memory. The right-hand field is the ASCII representation of these memory locations. For example, $49 is the ASCII value of capital I. The first 8 bytes constitute the file name and the next three are the extension. The next byte is the File Attribute—this is what you will alter to reveal the system file IBMBIO.COM to directory searches. Most files (COMMAND.COM for example), have a file attribute of $20, which simply means the file was closed the last time it was written to. The low nibble (the 4 right-hand bits in the byte) of the File Attribute determines how the system views the file. Specifically, if bit 0 (the right-most bit) is set, the file is read-only; if bit 1 is set, the file is hidden from directory searches; and if bit 2 is set, it is a system file (and hidden). The IBMBIO.COM file has all three of these bits set.

(4) To alter the File Attribute, first type **E 10B** (using DEBUG's **E**nter command) to change the appropriate memory location: $10B. The screen now displays **08F1:010B 27.** with the cursor just right of the period, waiting for your entry. If you type 21 and press [**ENTER**], the contents of memory location $010B change to $21. Verify the change by typing **D 100** to dump memory to the screen again.

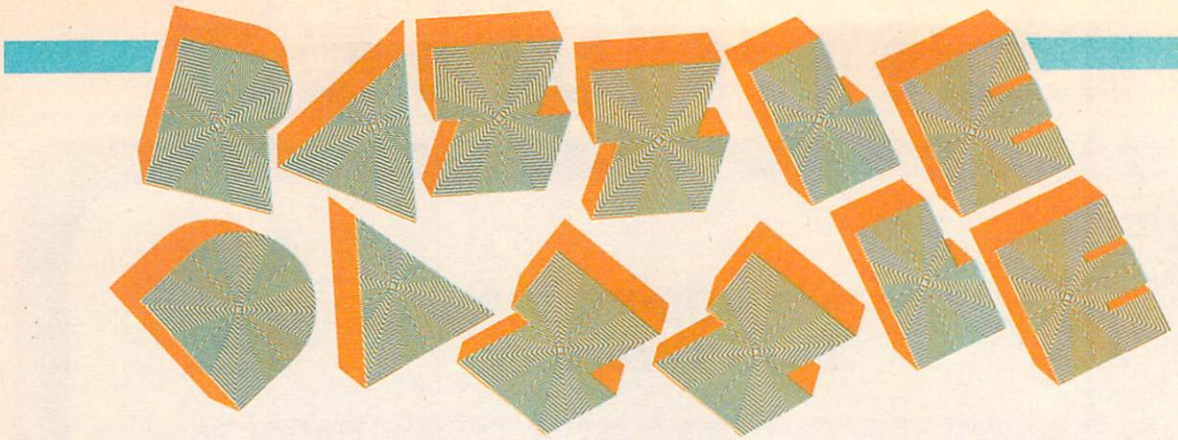
5) Next, make sure the disk you wish to modify has the write-protect notch uncovered. Then type **W 100 0 5 1** to effect the change to the disk. DEBUG now **W**rites the section of memory back to the disk, including the change in the File Attribute byte of the IBMBIO.COM file.

6) To view the modifications you have made, type **Q** to **Q**uit DEBUG and return to DOS. Now, type DIR and the computer displays the once-hidden file: **IBMBIO COM 4736 10-20-83 12:00p**.

Next issue, we will show you more about DEBUG, and demonstrate how to make a file impossible to delete, except from DEBUG.

—Roger Wood

HCM Glossary terms: ASCII, bit, byte, field, hexadecimal, nibble, offset.



Cyber-Abacus

by Scott Williams
HCM Staff

Produce dazzling results with this computer-age number cruncher

The computer is a multi-talented machine. It helps us write, it organizes our activities, it educates and amuses. When face-to-face with such versatility, however, we sometimes forget that the computer is foremost a number-cruncher. *Cyber-Abacus* is dedicated to this primary function—it might even replace that extra appendage on your desk: the calculator. As a simple utility program, *Cyber-Abacus* performs almost all the operations that are found on a standard calculator, as well as many that are not.

Getting Started

To begin crunching numbers, simply RUN the program and press [ENTER] when the title screen appears. The screen now displays printing instructions and prompts you to enter the printer device parameters. (Enter parameters according to your printer manual's specifications.) If you don't have a printer, or simply don't want a printout, press [ENTER].

After the program initializes, the computer displays the *Cyber-Abacus* screen—a calculator consisting of 5 fields. Enter the first value of your mathematical problem (the number to be operated on) into Field 1, located at the top of the screen. This field displays all results from a calculation or function after the operation is completed.

Sample Printout

```
F
VALUE OF PI
3.141592654
M
3.141592654
MOVE FIELD 1 TO MEMORY
C

CLEAR

3963
^
3
6.22404E+10
*M
1.95534E+11
MULTIPLY MEMORY
*
4
7.82136E+11
/
3
2.60712E+11
```

Directly below Field 1 lies the Operator field. The characters that you enter here determine the type of operation to be performed on the number in Field 1. Refer to Figure 1 for a list of characters that can be entered into this field and the functions that they perform. If you select an illegal operation, *Cyber-Abacus* displays a list of the possible legal operations in the lower part of the screen.

Below the Operator you find Field 2. Here, enter the second numeric value in your problem—the number that works with the Operator to act on the value in Field 1.

The fourth field displays printer status. If you have indicated a valid device name and parameter list, the field should display PRINTER ON. With this status, the program outputs any operation to the printer as it is performed (see Sample Printout). If you don't want a printout, switch to PRINTER OFF. You can toggle the printer status by entering P into the Operation field.

Field 5 displays the contents of memory. *Cyber-Abacus* memory works just like the memory on most calculators—it remembers any value placed into it, allowing you to perform other operations without losing the value. You can use many of the operators in Figure 1 to manipulate memory.

Functional Power

You can perform a number of functions on the value in Field 1 by selecting F at the Operator field. The program then displays a list of the 12 available functions at the bottom of the screen. To select a function, enter the letter located beside the function you desire. See Figure 2 for a list of the available functions and the keys used to select them.

HCM Glossary terms: device parameters, number crunching.

FIGURE 1

Operation	Function
+	Add Field 2 to Field 1
–	Subtract Field 2 from Field 1
*	Multiply Field 1 by Field 2
/	Divide Field 1 by Field 2
^	Raise Field 1 to the power of Field 2
C	Clear Field 1
Q	Quit. Exit the program
F	Select a function to perform on Field 1
M	Move Field 1 to memory
CM	Clear memory
+M	Add memory to Field 1
–M	Subtract memory from Field 1
*M	Multiply memory times Field 1
/M	Divide Field 1 by memory
R	Recall. Move memory to Field 1
P	Toggle printer ON/OFF switch

FIGURE 2

Key	Function	Result
A	NEG	Negative value
B	ATN	Arctangent (inverse of tangent)
C	COS	Cosine (in radians)
D	EXP	Exponential value
E	INT	Integer
F	LOG	Natural logarithm
G	PI	Places PI into Field 1
H	SIN	Sine (in radians)
I	SQR	Square root
J	TAN	Tangent (in radians)
K	ASN	Arcsine (inverse of sine)
L	ACS	Arccosine (inverse of cosine)

KEY VARIABLES

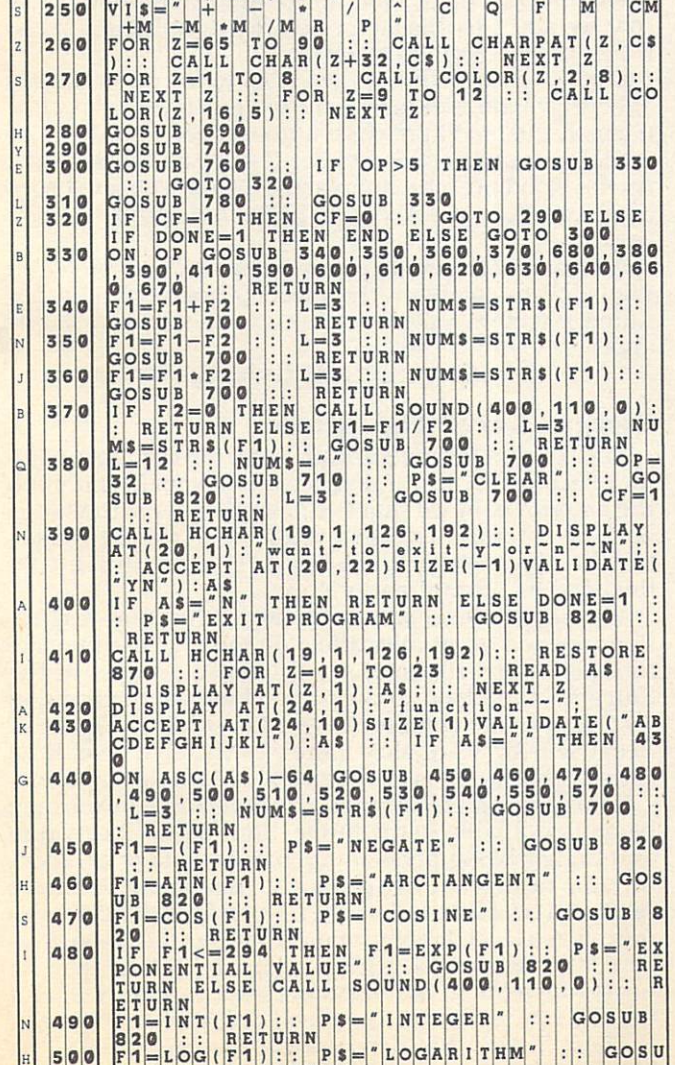
NUM$** is used to contain the numeric string in the numeric display routine. **P$ contains text to be sent to the printer. **CF** is a flag that indicates a clear-memory operation. **F1** contains the value of Field 1. **F2** contains the value of Field 2. **L** indicates the screen row for display. **MEM** contains the value of the memory field. **OP** contains the selected operation from the Operator field. **P** contains the status of the printer option ON/OFF switch.

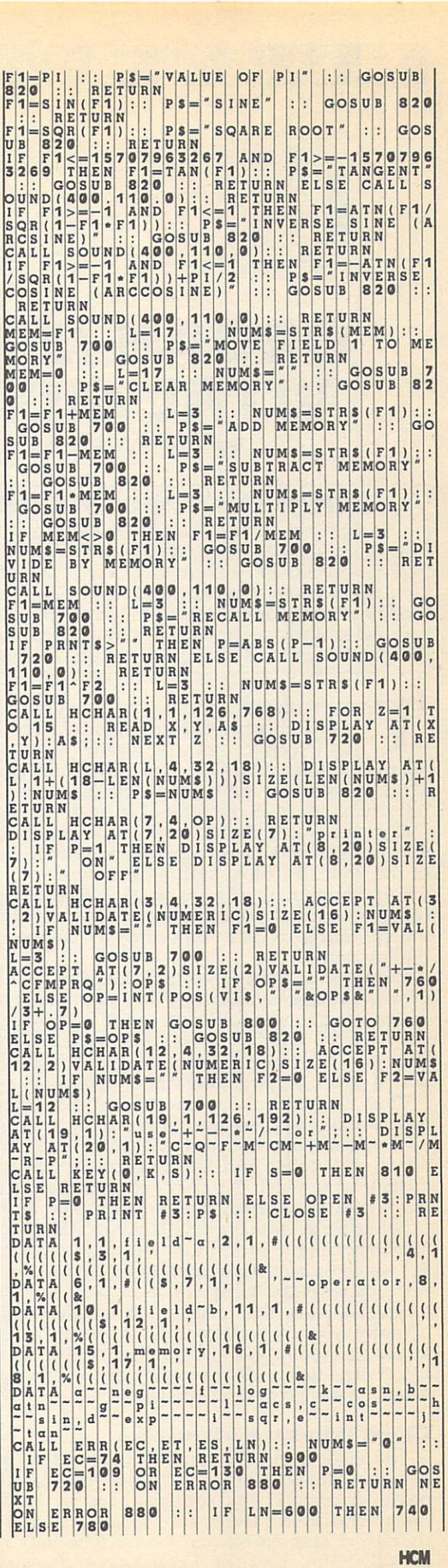
LISTING ANNOTATIONS

Cyber-Abacus (TI-99/4A)

Line Nos.

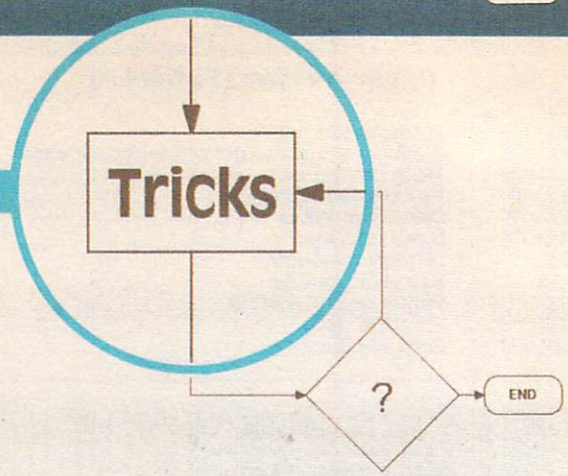
100-180	Program header.
190-290	Initilaize program, get printer parameters.
300-320	Main control loop.
330-680	Do selected operation.
690-730	Display routines.
740-790	Input routines.
800	Display message.
810	Key scan routine.
820	Printer output routine.
830-870	Program data.
880-900	Error routine.





TUTORIALS

Algorithm-A-Tricks



ORGAN-IZING VITAL SIGNS

by the HCM Staff

WHAT IS AN ALGORITHM?

An algorithm is simply a procedure—one that a program uses to complete a task or solve a problem. Flow diagrams and flow charts are handy tools for representing the steps in this procedure. Any program can be viewed as a collection of separate procedures. In this column, we focus on and explain one unusual or interesting algorithm that is found in one of the programs we publish each issue.

Here are the mathematical secrets behind the Vital Signs simulation of the circulatory and respiratory systems.

The circulatory and respiratory systems are the most complex facets of the human body—with the exception of the brain. Designing the algorithm that mimics these two important systems in *Vital Signs* was a challenging task (and one that took full advantage of all our brains' intricate construction).

Our goal was to design a simulation/game in which the user controls these normally-automatic functions of the body. To have both educational and strategic value, the simulation had to display a relatively accurate representation of circulatory and respiratory functions, incorporating realistic processes as well as variable parameters.

There are so many factors involved in the control of even the simplest circulatory or respiratory process, we might need a super computer with 100 megabytes of memory to take them all into consideration. We therefore focus on only the most critical of the functions which affect the circulatory and respiratory systems.

Thus, there are three functions in the simulation to be maintained. These are blood pressure, oxygen level in the blood, and body temperature. These functions affect each other and are additionally affected by 4 parameters, which the user can manipulate directly: heart rate, respiration rate, activity, and air quality.

Before we get into the details of how these parameters affect each of the three functions, take a look at Figure 1. This flow diagram gives a general picture of how heart rate and respiration rate are associated with the three functions. It also demonstrates how the three functions actually alter each other. It is important to note that, although respiration has no direct effect upon either blood pressure or body temperature in the algorithm, it has an *indirect* effect as shown.

Blood Pressure Algorithm

The algorithm that controls blood pressure is the simplest that we employ in *Vital Signs*. The heart rate obviously affects blood pressure because a faster heart rate pushes more blood through the circulatory system, thereby raising the blood pressure. The body's activity directly influences the heart rate's effect on blood pressure: Blood pressure increases with higher activity levels. We can therefore use the heart rate to offset the activity level in order to maintain pressure. (See Figure 2.)

To make the program more interesting, we add the possibility of a blood clot. Notice in the equation below that we add 1 to this factor before multiplying it in, so it has no effect when `BC=0`.

Finally, we add the oxygen level to the other factors affecting blood pressure. If the oxygen level falls below normal, the blood pressure automatically starts to increase, supplying blood to the tissues at a faster rate. As the blood pressure increases, the oxygen level goes back up. (See the Oxygen Level Algorithm section below.)

We use the following primary equation (depicted in Figure 2) to calculate the blood pressure. The other numeric factors scale the various factors relative to one another.

```
P=SQR(A*HR)*1.3485*(1+BC*.5)+OX
```

Oxygen Level Algorithm

The oxygen level algorithm is by far the most complex because of the number of factors that affect it. Both the amount of blood that the heart pumps to the lungs and the amount of oxygen available in the lungs control the oxygen level in the blood. These in turn are influenced by several parameters.

The oxygen level is the only algorithm that respiration rate and air quality affect directly. These two factors determine the amount of oxygen present in the lungs for the blood to pick up. Thus, air quality and respiration rate combine at the lower left in Figure 3. Because poor air quality increases the chance of lung cancer, this factor is also multiplied into the equation.

Heart rate and blood pressure determine how much blood is available to the lungs. Both higher heart rate and higher blood pressure increase the flow of blood, thereby raising the oxygen level. Increasing the heart rate has a twofold effect on the oxygen level: It influences the oxygen level directly; and, as it raises the blood pressure, it again increases the oxygen level. Notice that the heart rate adds to the blood pressure in Figure 2, and that it combines with the blood pressure in Figure 3.

These factors thus determine the oxygen level of the blood and the amount of blood being pumped to the

lungs. We also must consider how much of the available oxygen is actually used. It is important that there be a balance between the amount of oxygen supplied and the amount used. We therefore subtract the activity level value because higher activity levels cause more oxygen to be used. (See Figure 3.)

The equation we use to calculate the oxygen level is as follows:

```
CO = (SQR((RS*8*AIR*(1 - LC*.4))*SQR(HR^2 + P^2)) - A)*.02
```

Body Temperature Algorithm

The temperature algorithm actually changes slightly depending on the activity selected. If the activity is set to running or swimming we take a "sweat factor" into account. (See Figure 4.) Otherwise, the activity level directly increments the body temperature.

The inclusion of a perspiration factor adds realism to the simulation. The body relies heavily on the sweat glands to cool the outer layers of the skin when other methods of cooling the body are insufficient. This process in turn cools off the blood when it is near the skin.

Activity level always affects the body temperature. The more active the body is, the more energy must be expended, causing heat. The sweat counter (above the activity box in Figure 4) keeps track of the how long an activity level has continued. As the sweat counter increases, the body temperature drops. The counter increases over time, so its effect gradually increases as the activity continues. If the activity persists too long, the body eventually dehydrates (runs out of water), and body temperature starts to rise again.

The heart rate also directly controls the body's temperature. When the heart rate increases, the resulting increase in blood that reaches the skin tends to lower the temperature. The box at the top of Figure 4 represents this part of the algorithm. Conversely, lowering the heart rate increases the temperature.

In addition, oxygen level (as depicted in Figure 3) indirectly controls the body temperature. When the oxygen level decreases, the body cannot expend as much energy. Therefore, the body temperature decreases. (See the oxygen level algorithm section for details on how it is regulated).

Here are the equations (depicted in Figure 4) we use to calculate the body temperature:

```
CNT = CNT + 1
IF (CNT< =40 OR CNT> =100) AND A>4 THEN
  T = SQR((250 - HR) ^2 + (OX*3) ^2)*.07588 + 81.4
  + ((A*(CNT + 1)*.001)*((CNT>100)*1.1 + 1))
ELSE
  T = SQR((250 - HR) ^2 + (OX*3) ^2)*.07588 + 81.4 - A*.0001
```

Variables used in equations:

P	is the blood pressure.
A	is the level of activity.
HR	is the heart rate.
BC	is the flag which indicates a blood clot.
OX	is the deviation in oxygen level from optimum.
RS	is the respiration rate.
CO	is the change in oxygen level.
AIR	is the air quality.
LC	is a flag which indicates lung cancer.
HR	is the heart rate.
CNT	is the sweat counter.

HCM Glossary terms: algorithm, automatic function, parameter, scale.

HCM

Figure 1.
Effect Of Body Functions On Other Functions

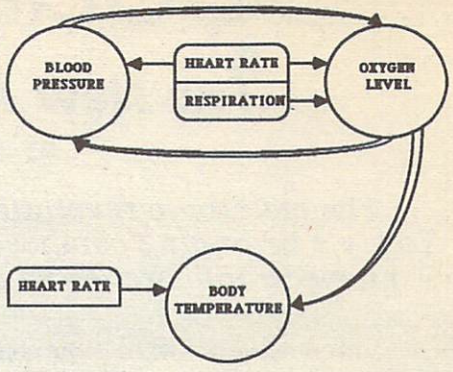


Figure 2.
Control Algorithm For Blood Pressure

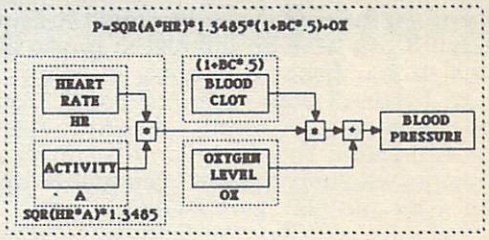


Figure 3.
Control Algorithm For Oxygen Level

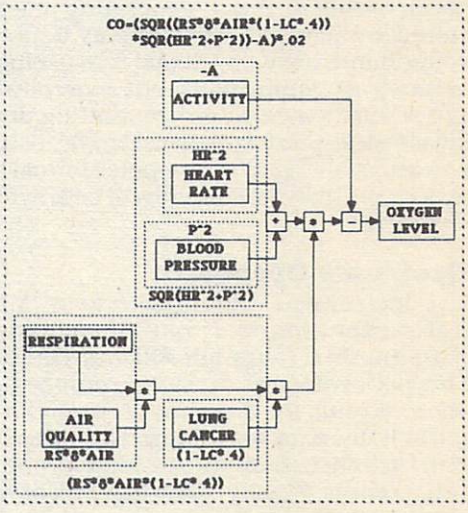
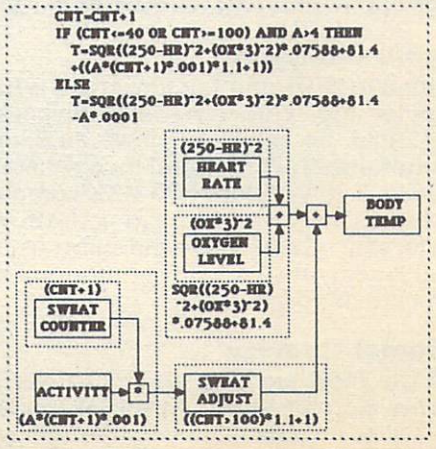


Figure 4.
Control Algorithm For Body Temperature



TUTORIALS

Soundbytes

The New Age of Music-Making

by Andy Widders-Ellis
HCM Staff

The electronic revolution brings musical power to the people. Today's beginning composer can use his or her computer to conduct an electronic orchestra or band—at a cost starting under $500.

Music and computers—what is the connection? In the past decade, as computers have established themselves in millions of homes, they have also emerged as a major force in music. Computers are removing the years of physical training that formerly stood between the average person and the act of making music. If you have an interest in music but have hesitated to get involved because of this perceived "physical barrier," your time has come. Computer-assisted composing, recording, and synthesizer sound creation is a reality. Today, the world of music is opening to people who may not yet even realize that they, too, can work and play with sound.

Synthesizers Get Smart

Until recently, the synthesizer—fast becoming the dominant instrument in contemporary music—has served as an excellent sound-generating machine, but has required a skilled performer to play it. Computers, on the other hand, have developed into useful tools for creating musical compositions, but generally have *not* been very sophisticated sound-generating devices. As prices fall for these two forms of electronic instruments, new ways are evolving to create potent music-making *systems* that combine the strengths of both synthesizers and computers.

The Electronic Orchestra

Musical Instrument Digital Interface (MIDI) is a method of transmitting data from one computer or synthesizer to another. Through MIDI, a computer can memorize performances or store compositions produced by electronic instruments. A user can then arrange and edit this data with special software. Other programs can turn these computer-assisted compositions into printed scores. Finally, MIDI can link several electronic instruments to the computer to perform numerous parts. Today's beginning composer can thus use his or her computer to conduct an electronic orchestra or band—at a cost starting under $500.

The Music Computer

In addition to MIDI, another new trend is to combine the computer and synthesizer into one package. The Yamaha CX5M, for example, places an 8-voice, polyphonic, multi-timbral FM digital tone generator under the control of a 48K under-$500 MSX computer. With this machine, you can record and perform your compositions (in stereo), and create and store custom voices. You can also generate hardcopy printouts of music and voice data on standard paper.

The "Sound Camera"

One of the most exciting and controversial developments in electronic music—resulting from recent advances in digital recording—is sound-sampling. A sound (such as a bowed violin string) can be translated into binary code and stored in a microchip or on floppy disk. This technique has led to keyboards and drum machines that can either re-create the sound of traditional instruments very accurately—or use virtually *any* sampled sound as a new "instrument." Even trained ears sometimes cannot distinguish between these re-created sounds and the "real thing." Sound-sampling is now beginning to find its way into the home computer field. An example of this technology is the Decillionix DX-1. This under-$400 hardware/software package for the Apple II family allows digital recording and playback of any sound.

The New Folk Instrument

Computers have created an interesting situation for music makers. Do we seek the experience of producing music physically? Or do we simply "dream it up" and program electronic instruments to perform for us? Neither choice precludes the other. But the second option promises to have an impact not unlike the invention of other "folk instruments" of the past—such as the autoharp, the accordian, and the electric organ. It means that music is now more *accessible* to more people. This new music medium based on computers and electronics is indeed the folk instrument of our time.

All indications are that the computer is emerging as the ideal means of expressing the musical *vox populi*. The rules may have changed, but the game remains the same—human communication. Man's most universal language can now benefit from man's most powerful electronic tool.

Home computer owners can participate in this music revolution in many ways: Are you a budding musician? With your computer you can study sight-reading, theory, ear training, and composition, by tapping the wealth of music instruction software currently available. How about a musical "pen-pal" network, employing computers and modems? (Imagine "playing" your music for some friends who are sitting in their livingroom, thousands of miles away.) You can learn the rudiments of keyboard, guitar, or several other instruments from your friendly computer-tutor. Study the masters by entering their compositions part-by-part into computer memory. Experiment, orchestrating these parts with a synthesizer through MIDI. Re-arrange Beethoven! Digitally record environmental sounds, edit, and arrange them into sonic events and tone poems. The possibilities are almost endless, yet as near as your computer keyboard.

For in-depth information and recorded examples of the changing face of music, read our new sister publication *Music & Electronics: The Magazine of Creative Discovery Through Sound Technology*.*

HCM Glossary terms: MIDI, modem, MSX, polyphonic, multi-timbral, FM digital tone generator, voice, synthesizer, digital sound-sampling.

*See sample contents & order form on pages 63 & 64.

HCM

MUSIC & ELECTRONICS™

The Magazine of Creative Discovery Through Sound Technology

VOLUME 1 NUMBER 1

$3.50 in U.S.A.

THE NEW MUSIC MAKERS

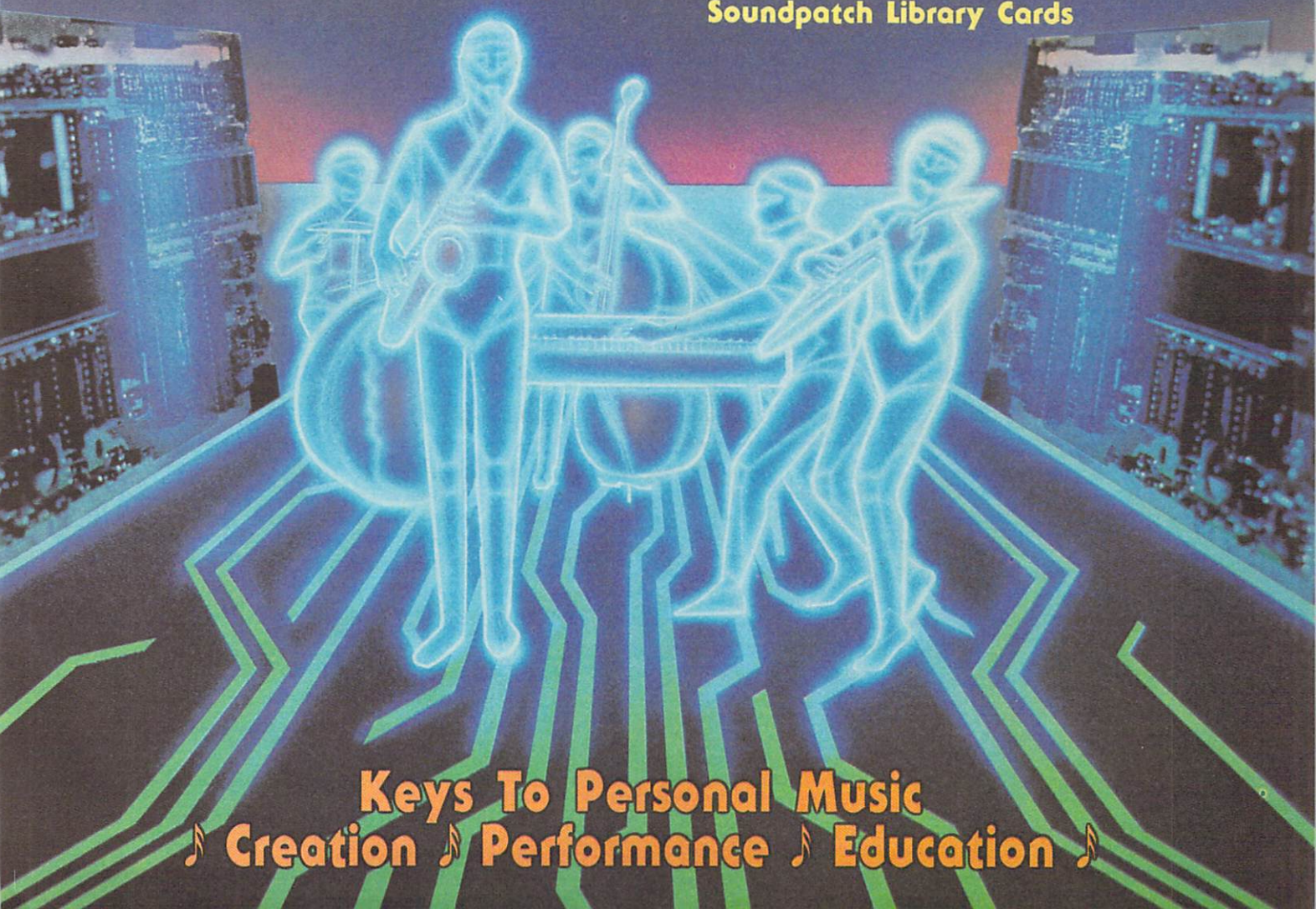
The Compact Home Music Studio

Home Computers: New Dimensions in Musical Participation

Synthesizer Soundings & Digital Orchestration

Stretch Your Tape Recording Dollar:
Affordable Multi-Track Cassette Equipment & Techniques

Featuring a Bound-In Sound Demo Record
PLUS—A Free Set of Soundpatch Library Cards



Keys To Personal Music
♪ Creation ♪ Performance ♪ Education ♪

MUSIC & ELECTRONICS™

The Magazine of Creative Discovery Through Sound Technology

Music & Electronics is devoted to "the musician in all of us." Each issue explores the many facets of personal music creation, performance, and education through the magic of computers and electronics. Editorial content spans a full range of subjects—from creating the Electronic Orchestra by interfacing musical devices with home computers, to analog and digital recording techniques, to computer-assisted music lessons. Bound into each issue is a sound demo record, that adds an extra dimension to articles and reviews, plus an assortment of Soundpatch Library Cards for re-creating unique sounds on current musical devices.

Electronic Music Lab™
Basic Principles of Music Synthesis
What is this thing called sound? What distinguishes one sound from another? How is sound created electronically? We answer these and many more questions in this basic primer on the physical phenomenon of sound and synthesis. Our clear explanation of this medium takes the mystery—but not the wonder—out of the new musical technology.

Synthesizer Soundings™
Hands-on Lessons
Here we demonstrate how anyone can "program" the most popular synthesizers. It's not just a matter of moving switches and pushing buttons. M & E's mandate is to teach the art of sound design.

The Digital Sampler™
From Physics to Numbers and Back
Digital recording is a fundamentally new way of capturing sound. Home computers can play a role in handling and manipulating this musical data. Learn how to work with this new technology, as M&E takes you along the digital path.

The M&E Soundpatch Library™
Ready-to-Use Synthesizer "Patches"
Looking for new and useful sounds on your electronic instrument? Each issue contains easy tear-out cards with original patches for popular synthesizers. When appropriate, soundpatch data from the magazine will also be available on tape or disk.

Computer Home Companion™
Home Computers and Music Magic
Home computers—some of which have small built-in synthesizers—are opening up the floodgates to popular participation in the creation and performance of music. Some software teaches music theory—while other programs convert the computer into an instrument itself. M & E also shows readers how computers can record and control musical devices.

The Home Music Studio™
The Art of Analog & Digital Recording & Mixing
Miniaturization and cost-reduction of electronic instruments, computers, and recorders have made compact, affordable home studios possible. In this series, we help you develop your own recording system—providing tips on design, construction, and operation, as well as advice on selecting the proper equipment. Learn multi-track and MIDI recording techniques, signal processing tricks, and how to achieve professional results from your personal studio.

M&E Rhythm Beat™
Programming Digital Drum Machines
Drumming has changed radically since Ringo struck up the band. Today, even those who never mastered the sticks can program a machine to both imitate drummers and play humanly impossible "licks." The digital drum machine is an extension of the traditional drum set—for which rhythmic skills are essential. Learn the rudiments of rhythm; enjoy lessons in style; explore the world of the big beat.

MIDI Melting Pot™
Plugging into the Industry's Common Communications Standard
MIDI (Musical Instrument Digital Interface) is an industry-standard interface that sends musical information in digital form between synthesizers, signal processors, and computers. This interface can link many electronic music devices in one large ensemble, or connect the synthesizers and signal processors to a computer for digital recording and editing. Learn how MIDI is indeed the key to the Electronic Orchestra.

Composer's Workshop™
Creating & Arranging a Musical Idea
Electronic instruments make it easy to write and arrange music, even for those with little or no previous experience. Let M & E show you how to generate and develop ideas—bringing your music out of the shower, into the streets.

In Session™
The Making of a "Track"
Listen to the bound-in sound recording, then follow us as we document the entire recording and mixing process step-by step, using text, diagrams, and color photographs.

Pro-Formance Playbill™
M & E's beat covers the exciting world of electronically-produced music through interviews with major artists plus reviews of concerts and new releases.

M & E Review Roundup™
M & E provides incisive reviews of electronic musical instruments and equipment plus in-depth analysis of the latest software products.

Ask M&E™
Here our readers get to sound-off with any questions related to electronics and music.

And There's More:

- **Create-A-Sound Contest™**
- **Compact Disk Corner™**
- **New Products Soundcase™**
- **Notes to the Editor™**
- **M & E Glossary & Abstracts™**

—Plus a few special surprises . . .

YES! Enclosed is $18 (Canada: $23) in U.S. funds.
Please enter my 1 year (6 issue) charter subscription to
MUSIC & ELECTRONICS™
The Magazine of Creative Discovery Through Sound Technology

Please Print
Name _____
Address _____
City _____ State _____ Zip _____
☐ Check or Money Order Enclosed
MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK
Charge my ☐VISA ☐MasterCard
Account No. | | | | | | | | | | | | | | | | |
Date Expires _____
Tel. No. _____
Signature _____

Clip or photocopy this coupon or use the bind-in order form at center of magazine.

✷ CHARTER SUBSCRIBER SPECIAL ✷
SAVE 33%
SPECIAL BONUS PACKAGE
For Charter Subscribers!
ACT NOW to receive your
FREE PREMIUM—a stereo cassette tape ($5.95 premium value)
Featuring:
- Unique performances from the latest electronic musical instruments
- Demonstrations of step-by-step recording techniques
- Lessons on creating original synthesizer sounds
- An audio odyssey through M & E's popular columns: "Composer's Workshop" and "In Session"

✷ The Vol. 1 No. 1 Charter Issue will be mailed in November ✷

Enclose payment or credit card information & mail with completed form to:
Emerald Valley Publishing Co. ● P.O. Box 70288 ● Eugene, OR 97401
Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:
1-800-828-2212 In Oregon, Alaska, Hawaii Tel. (503) 485-8796
Satisfaction Guaranteed—or the unfilled portion of your subscription will be refunded, less the cost of any premiums you have received. Prices subject to change without notice.

HCM Glossary

VOLUME 5 NUMBER 5

accumulator – The A register where, in most processors, the solutions to math and logic operations are placed.

address – The label referring to the hardware location of a piece of data.

algorithm – A set of rules or procedures used to solve a problem.

ampersand (&) command – An Applesoft BASIC extension command.

animate – To create the illusion of motion.

application – A program which performs one or more specific tasks.

ASCII – (American Standard Code for Information Interchange) The computer code most commonly used to represent upper- and lower-case letters, numbers, symbols, and punctuation marks.

automatic function – A function or operation that the computer carries out automatically and over which the user has no control.

BASIC loader – A BASIC program that stores, or "loads," a machine language program into memory.

binary numbers – A base-2 numbering system in which the only symbols used are 0 and 1.

bit – (binary digit) The most basic unit of information that the computer uses. Each bit is an electronic impulse, that, combined with other such impulses fed into the computer's circuitry, forms letters and numbers.

branch – A departure from the sequential execution of program instruction—usually due to the test of a condition.

bus – A set of wires or traces on a printed circuit board that that carry signals.

byte – A sequence of 8 bits used to represent one character.

carry flag – A flag in the status register that is set to one when an operation results in an overflow or carry.

character set – The set of characters that a computer can display on a screen.

conditional jump – A departure from the sequential execution of program instructions, due to the outcome of a test of a condition. (See "branch.")

CPU – (Central Processing Unit) The central brain and controller of the microcomputer.

device parameters – Instructions that tell the computer which output device to send information to and how to send that information.

digital sound sampling – A method of recording sound by translating it into binary code using computers. This data is stored in microchips or on disk for use in electronic instruments.

display value – The number that represents a character in the Apple computer's screen memory.

driver – The software that allows communication between a computer and a peripheral.

element – A specific item in an array.

error routine – A segment of a BASIC program that is designed to handle user-initiated problems that the programmer has predicted might occur in a program, so that the program will not stop running (causing data in memory to be lost).

field – A specified area for a particular category of data.

flag – A variable or a memory location that contains a value that represents a condition that the program needs to test.

flicker – The oscillation of a screen or an object on a screen at a speed visible to the human eye.

FM digital tone generator – A device that uses frequency-modulated (FM) sine waves to produce sound.

frequency – The rate of repetition of a waveform. It is perceived as pitch in the audio range.

frequency modulation – (FM) A technique of varying the frequency of a waveform (carrier) with the frequency of another waveform (modulator).

frequency offsetting – (detuning) Changing the frequency of an oscillator in relation to that of another oscillator, within a synthesizer patch.

fundamental – The principal frequency of a waveform. In the audio range, it is the perceived pitch.

garbage collection – A process used by many computers to clear out old or unused variables to make room for new ones.

graphics image file – A file that contains a bit-for-bit image of a screen or a section of a screen.

graphics mode – An optional mode, usually selected through software, that determines what parameters the computer uses when sending data to the screen.

harmonic – A frequency that is an integer multiple of the fundamental.

hexadecimal numbers – A base-16 numbering system using decimal digits 0 through 9 and the letters A through F.

high bit – The left-most bit of a binary number; e.g., the eighth bit of a byte.

high resolution graphics page – An area of memory that has been reserved for the display of graphics.

instruction set – The full complement of encoded instructions that a CPU can directly implement.

integer array – An array of integer numbers.

interrupt request – A signal that directs the computer away from the program sequence.

jump – A departure from the sequential execution of program instructions.

limit checking – A test, performed by programs, that prevents the program or the user from exceeding predefined boundaries, limits, or rules.

location counter – A register in the CPU which points to an address in memory where the next instruction or piece of data will be accessed.

loop – A sequence of instructions, in a program, that repeats until a set of conditions is satisfied.

loop counter –a variable used to count the number of iterations in a loop

machine language – The native language of the microprocessor in a computer expressed in terms of binary ones and zeros.

MIDI – (Musical Instrument Digital Interface) Specified protocols for transmitting digital information from one synthesizer, sequencer, computer, signal processor, etc., to another.

mnemonic – A code or symbol that helps people remember something specific, often made up of letters from the word or phrases it represents.

modem – (MODulator/DEModulator) This device converts digital information into analog information so that it can be sent across telephone lines.

modem port – A serial (RS-232C) port on the Apple *IIc*, specifically set up for intercomputer communications, but often used for interfacing peripherals.

MSX – A Z-80 based computer produced by several Japanese manufacturers according to agreed-upon standards.

multi-timbral – An instrument capable of producing two or more musical timbres (tone colors) simultaneously.

multi-track recording – A method of using a recorder with many synchronized channels to record and play back sounds, either individually or together.

nibble – 4 binary digits (bits) of data—a half byte.

number crunching – Processing data through the use of multiple complex operations.

offset – A value that locates a particular character in a string by determining a certain number of characters from the base character.

overflow – A situation in which the results of an operation cannot be expressed in the number of bits in a register.

overtone – A frequency component of a waveform, that is higher than the fundamental.

palette – The determinant of the colors available for a graphics display screen.

parameter – A variable used to control a particular process.

periodic noise – One of 4 pulse waves produced by the TI-99/4A sound chip.

perspective – The true relative position of an object perceived by the viewer.

pixel – The smallest dot that a computer is capable of generating on a display. The number of pixels your computer generates on the screen determines the video resolution. The more pixels packed onto one screen, the higher the resolution.

pointer – An address that gives the location of the next item of data to be accessed.

polyphonic – Music that contains two or more independent voices sounded simultaneously.

program flow – The order in which a program executes—not necessarily the way it appears when listed.

RAM – See Random Access Memory.

Random Access Memory (RAM) – The set of hardware locations in a computer where programs and data are stored.

real time recording – a method of recording musical data into a sequencer or computer precisely as it occurs.

redefined graphics character – A character whose shape has been changed by a program that rearranges the pixels which make up the character.

register – A small computer storage location in an integrated circuit.

resequence – To alter the line numbers of a BASIC program listing so that they are spaced equally.

reset – In binary, to zero a bit or memory location.

ring modulation – Mathematically combining two frequencies by outputting their sum and difference and suppressing the original frequencies. Most frequencies created through ring modulation are nonharmonic.

scale – To adjust a series of values so that they are in a predetermined proportion to one another.

screen code – The number that represents a character in the Commodore's screen memory.

screen memory – The memory locations in a computer that hold the data for the current screen display.

shading – Providing a two-dimensional object on the screen with a shadow so that a third dimension can be visualized.

SID – Sound Interface Device chip in the Commodore 64.

sine wave – A single frequency oscillation which produces a pure, fundamental tone without harmonics.

sound chip – An integrated circuit used to create sound.

sound sampling – See digital sound sampling.

split screen – A screen that has two or more areas isolated by either software or hardware.

stack – An area of memory reserved for the temporary storage of data in a linear fashion, in which items are added or retrieved off of one end.

status register – One of the internal registers in the CPU where certain conditions are recorded.

string – A consecutive set of similar data items—usually bits or characters.

string array – An arrangement of strings that the computer can easily search through.

subprogram – A self-contained routine used from within another program.

subroutine – A routine that is part of a larger program. In a BASIC program it is called by a GOSUB.

synchronization – The linking of two oscillators such that the start of the cycle of one oscillator triggers the start of the second oscillator's cycle. One oscillator acts as the master to which the other oscillator is slaved.

syntax error – An incorrect command that the computer rejects as unrecognizable.

synthesizer – An electronic musical instrument designed to generate, modify, and control the waveforms used to create sound.

template – In file management programs, a file used to define a format for other files.

token – An abbreviation of a basic statement, function, or (on the Atari computer) variable.

track – An independent storage area for recorded signals which can be monitored individually or in synchronization with other tracks. On magnetic recording tape, a track is assigned its own linear path. In a sequencer, a track occupies its own set of memory locations.

trap – The trapping of a specific event that causes the program to branch to a specific routine.

variable cross-reference – An indexed list of variables and their locations in a program.

variable-length element string array – A string array that contains lists of ASCII characters. These lists can be any length up to the maximum designed into the software (BASIC, etc.). All of the memory is not reserved for the array until that memory is required when more characters or lists are placed into the array.

voice – 1. An independent audio signal produced by a synthesizer, that can be simultaneously sounded with other voices: e.g., "My JX-8P is a 6-*voice* synthesizer." 2. A specific timbre, or "patch," created by programming a synthesizer: e.g., "*Voice* #22 is a bamboo flute."

waveform – A signal with periodic fluctuations, created by an oscillator.

weight (of bits) – In the binary number system, each bit's decimal equivalent.

white noise – Random frequencies dispersed uniformly across the audio spectrum.

zero flag – A flag in the status register that is set to one when an operation results in a zero.

HOME COMPUTER product news

Each month we publish items of interest and news of recently or soon-to-be released computer products. Our publication of information from manufacturers of computers, peripherals, software, and accessories is not to be construed as product endorsement. Prices quoted are the manufacturers' suggested retail prices and are subject to change.

Send press releases to:

Product News Editor
Home Computer Magazine
1500 Valley River Drive., Suite 250
Eugene, OR 97401

❖❖❖❖

Cassettes for Computer Literacy

Audio Tape as a Computer Education Tool

FlipTrack has released a "How to Operate" series of audio cassette courses designed to teach anyone to operate a home computer. The cassettes are available for the Commodore 64, VIC-20, TI-99/4A, and Atari 600XL/800XL computers. The tutorials start with instructions on setting up the computer system and then progress into programming commands; calculations; and use of color, graphics, and sound. At various stages in the lesson the tape may be flipped over for an excursion into optional activities that provide more practice and instruction. The Commodore 64 version

retails for $29.95, the Atari and VIC-20 versions for $19.95, and the TI-99/4A version for $16.95.

FlipTrack Learning Systems
999 Main, Suite 200
Glen Ellyn, IL 60137
(312) 790-1117

❖❖❖❖

Saving Real Time

High-Speed Indexing for dBASE Users

Fox & Geller has announced the release of Quickindex, a program that provides high-speed indexing to dBASE users. It is available for IBM computers and operates in conjunction with Ashton-Tate's dBASE-II and dBASE-III programs. Utilization of an advanced algorithm for indexing and memory management enables Quickindex to produce index files at speeds up to 10 times that of dBASE alone. A copy-protected version retails for $69. A nonprotected version costs $99.

Fox & Geller, Inc.
604 Market St.
Elmwood Park, NJ 07407
(201) 794-8883

❖❖❖❖

In Perspective: Stretching The Image

Graphic Distortion on the Mac

T/Maker Company has announced the release of Click Art: Effects, a graphics tool package for the Apple Macintosh. It operates as a desk accessory within MacPaint, making it possible to rotate, slant, distort or add perspective to MacPaint images. A selected image or portion of an image may be rotated one degree at a time or slanted backward, forward, up, or down. The distortion function makes an image pliable, allowing

the user to pull portions of it in any direction. The perspective function reduces an image's size with respect to a vanishing point. Click Art: Effects is available for $49.95.

TMaker Company
2115 Landings Dr.
Mountain View, CA 94043
(415) 962-0195

❖❖❖❖

Ed-Time For Gonzo

And Help for Students Taking the SAT

Simon & Schuster has introduced the second program in its Muppet Institute of Technology series, as well as a program aid for the college entrance exam. The Great Gonzo in Wordrider, like other programs in the MIT series, is an educational game for children. It is a strategy adventure game that aids the development of reading, vocabulary, and word-usage skills. In this program, children combine adjectives and nouns to help the Great Gonzo rescue his love, Camilla the Chicken, from the Swedish Chef. The Great Gonzo in Wordrider is available for the Commodore 64 ($29.95) and the Apple II family of computers ($34.95).

Simon & Schuster has also released Lovejoy's Preparation for the SAT—a comprehensive package offering information and drills that help students prepare for college. The program includes tutorials, tips, practice tests, techniques for test taking, and a copy of Lovejoy's Concise College Guide book. Versions are available for the IBM PC and PCjr, Commodore 64, and Apple II family of computers for $69.95.

Simon & Schuster
1230 Avenue of the Americas
New York, NY 10020
(212) 245-6400

❖❖❖❖

Encore On the Mac

Music Construction for Macintosh

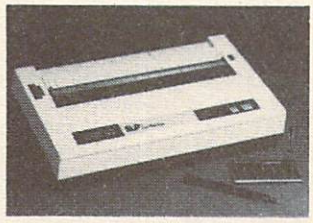
A state-of-the-art music composition program is now available for musically-inclined Apple Macintosh users. Deluxe Music Construction Set, produced by Electronic Arts, is an advanced version of Music Construction Set. With it, beginners can learn standard notation and basic composition skills. For the expert, the set provides professional composition capabilities, enabling users to compose music, view it on the screen, listen to it, and print it as sheet music. The set includes musical notation tools, playback equipment, and printing controls. It is $50.

Electronic Arts
2755 Campus Dr.
San Mateo, CA 94403
(415) 571-7171

❖❖❖❖

Super Printer

It Fits in a Briefcase or a Desk Drawer



Model SLP (Super Little Printer) is the latest release from Axiom Corporation. This compact printer measures 13.1 inches wide by 7.5 inches deep by 2.8 inches high and tips the scale at 6.6 pounds. It offers near-letter-quality printing and features higher draft speed mode, superscript, subscript, underlining, dot-addressable and IBM PC-compatible graphics, and detachable tractors. A choice of parallel, serial, or Commodore direct-connect interfaces is available. Model SLP costs $299.

Axiom Corp.
1014 Griswold Ave.
San Fernando, CA 91340
(818) 365-9521

❖❖❖❖

Passport To Commodore MIDI

Program Links With Music Keyboards

Passport Designs has developed a MIDI version of Broderbund's The Music Shop for the Commodore 64. The program is designed to benefit both the accomplished musician and the user who has never before played a note. The program allows you to compose, store, edit, and print sheet music in piano or single-staff formats. Its composing and editing features are enhanced by eight voices that can be assigned to four different MIDI channels or keyboards. The Music Shop includes: a user interface with pull-down menus; windows and dialog boxes controlled by a joystick and the MIDI keyboard; legible on-screen notation; easy editing with cut, paste and copy features; and broad music capabilities. It retails for $99.95.

Passport Designs, Inc.
625 Miramontes St.
Half Moon Bay, CA 94109
(415) 726-0280

❖❖❖❖

What You See is What You Get

Atari Opens its Eyes.

Digital Vision has introduced Computereyes, a video-acquisition system for Atari computers. The system incorporates a peripheral scanning device and a software program that allow users to employ video cameras to transfer real-world images onto Atari's high-resolution graphics display. The package includes an interface module, software support on disk, and an owner's manual, retailing for $129.95. A package that also includes a video camera is available for $399.95.

Digital Vision, Inc.
14 Oak St., Suite 2
Needham, MA 02192
(617) 444-9040

❖❖❖❖

Get More For Your Modem

Economical Access To Modem Net

A complete modem system that includes modem, software, and cables is now available. The Smart Communications System from 1-800-FLOPPYS is compatible with most personal computers. The system provides a 30-day trial period, a one-year warranty, a toll-free technical support line, a user manual, Newsnet time, and membership to the Delphi on-line data base. Apple computer users also receive an Apple Bulletin Board System. The Smart Communications System retails for $99.

1-800-FLOPPYS
Southfield, MI
(800) 356-7796

❖❖❖❖

Software For Kids

Bundle of Programs for C-64, 99/4A

KIDware has introduced a bundle of educational programs for children. More than 33 TI-99/4A programs and 60 Commodore 64 programs are now available to educate and amuse children from 1 to 16 years old. Each program makes full use of the graphics and music capabilities of both computers. The programs are available on cassette tape for the TI-99/4A, and on both tape and disk for the Commodore 64. All are packaged in sets of two, which sell for $9.95 (tape) and $11.95 (disk).

KIDware
P.O. Box 9762
Moscow, Idaho 83843
(208) 882-3830

❖❖❖❖

Two For TI One

Special Deal for TI-Runner Owners

EB Software has announced a special software offer for TI-99/4A owners. For a limited time, purchasers of the award-winning TI-Runner game will receive two additional games for the same price. A free disk will contain Galactic Battle, an exploration/battle game in which up to 9 players compete for universal superiority, and Spy Adventure, an adventure game involving the intrigues of secret agents. Editor/Assembler and 32K memory expansion are required for these programs. The three-game package retails for $24.95.

EB Software
12912 Villa Rose
Santa Ana, CA 92705

❖❖❖❖

Speaking In Foreign Tongues

C-64 and Atari Links To Languages

Artworx has announced the release of Linkword, a series of programs designed to facilitate the learning of foreign languages. Four packages cover Spanish, French, German, and Italian. Each includes a program disk and an audio tape to teach grammar and accurate pronunciation of a 400-word basic vocabulary. Linkwork employs a memory technique that associates foreign words with acoustically-similar English words. This system enables users to learn the basics of a language in 10 hours. Linkword is available for Apple, IBM PC, Commodore, and Atari computers. The Apple and IBM versions retail for $29.95. The Commodore and Atari versions are priced at $24.95.

Artworx Software Company, Inc.
150 North Main St.
Fairport, NY 14450
(800) 828-6573

❖❖❖❖

Speaking of Voices . . .

Music and Speech for Apple, Atari, Commodore

Covox has announced the release of its Voice Master speech system. The system provides digital speech, voice recognition and music synthesis capabilities to Commodore 64/128, Atari, and Apple II family computers. It consists of a hardware module, a headset/microphone, system software on disk, user's manual, and accessory cables. The system enables users to record their own voices as digital information. Storage and playback features provide endless possibilities, such as voice-controlled blackjack games and talking alarm clocks. The system sells for $89.95.

Covox, Inc.
675-D Conger St.
Eugene, OR 97402
(503) 342-1271

❖❖❖❖

Getting Ahead to BASIC

BASIC for the Atari

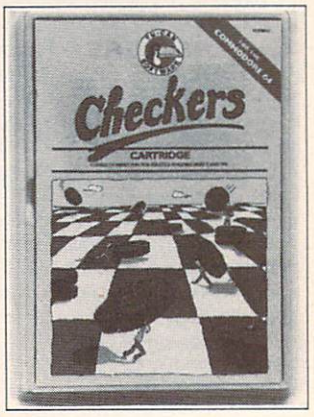
Optimized Systems Software has released the first programming language designed specifically for the Atari 130 XE, allowing programmers to take advantage of all of the XE's 128K of memory. BASIC XE can provide a programming space of over 62,000 bytes and a storage field of over 35,000 bytes. Although extended memory can be accessed only on the Atari130 XE, BASIC XE runs on any Atari XL or XE computer. Atari BASIC and BASIC XL are upward-compatible with BASIC XE. The package includes reference manual, OSS SuperCartridge, and extension disk, and retails for $79.

Optimized Systems Software, Inc.
1221B Kentwood Ave.
San Jose, CA 95129
(408) 446-3099

❖❖❖❖

A New Mode for an Old Classic

Checkers Enters the Computer Age



Checkers is now available in cartridge format for Commodore 64 users. Yu-Can Software has released a checkers program offering 4 levels of skill to challenge both novices and experts. Users play against the computer. The program provides beginners with on-screen instructions at the start of each game. Checkers can be played using either joystick or keyboard. It retails for $29.95.

Yu-Can Software
34 Maple Ave.
Armonk, NY 10504
(914) 273-6480

❖❖❖❖

Learning at Play, the Muppet Way

A Playful Approach to Education

Sunburst Communications has released two additions to its line of "learn-as-you-play" games. Getting Ready to Read & Add and Teddy's Playground, which run on Apple II family computers, are compatible with Muppet Learning Keys: Kid's Computer Keyboard. Both programs are designed to enhance understanding of shapes and colors, as well as to improve letter and number recognition. The programs incorporate varying levels of challenge that allow parents or teachers to set a pace appropriate to each child's skill level. The program packages, which include backup disk and teacher's guide, retail for $55.

Sunburst Communications, Inc.
29 Washington Ave.
Pleasantville, NY 10570
(914) 769-5030

❖❖❖❖

Front Page Graphics

A Collection of Newspaper Art

Springboard Software has announced the release of The Clip Art Collection, Volume I. It is a supplement to The Newsroom, a program that makes it possible to publish a newspaper with Apple II family computers as well as with the IBM PC and PCjr. The Clip Art Collection contains over 600 items, featuring both cartoon and realistic characters in categories ranging from famous people to classic cars. Each piece of clip art can be used as is, combined with other pieces, or altered to create new characters. The Clip Art Collection retails for $29.95.

Springboard Software, Inc.
7807 Creekridge Circle
Minneapolis, MN 55435
(612) 944-3912

❖ ❖ ❖ ❖

Getting It Letter Perfect

Inexpensive Correspondence on the TI

Micro-Biz Hawaii has announced the release of Basic Letter, a program which allows TI-99/4A users to write, edit, and print correspondence in personal letter format. The program is especially suited for form letters which may be stored—10 to a disk or cassette. The program also permits the preparation of envelopes by printing addresses on standard paper and providing markers indicating where the paper should be folded. Basic Letter may be obtained from the producer for $10.

Micro-Biz Hawaii
98-1409 D. Kaahumanu St.
Aiea, HI 96701
(808) 488-9491

❖ ❖ ❖ ❖

Counting Apples on Atari

Cartridge Math Tutor

Simage has released Eli's Ladder, a cartridge designed for use with the Atari 2600 home computer. Eli's Ladder is an educational product that teaches basic mathematical concepts from counting apples to advanced addition and subtraction. The product offers 20 sequential learning levels, 8 speeds, and 3 game types. The use of random numbers insures game variety. The cartridge costs $29.95.

Simage
110 Corte Ramon
Greenbrae, CA 94904
(415) 461-4348

❖ ❖ ❖ ❖

Tame Your Appliances

Electrical Control Interface

Powerhouse, a new computer interface, enables Commodore 64 and Apple II family computers to automatically control electrical devices in any house, store, or office. Developed by X-10 (USA) Inc., the interface is a small peripheral that sends signals over AC wiring to control up to 72 lights & appliances plugged into System X-10 modules, which are in turn plugged into 120 volt outlets. Powerhouse is actually a self-contained micro computer backed up by a battery that can sustain it without AC power for 100 hours. Powerhouse is programmed from the computer console. Graphics lead the user from room to room, and explain how lights & appliances may be selected and controlled.

Once programming is complete, Powerhouse may be disconnected from the computer, freeing the computer for other uses. The X-10 interface, software disk, and connecting cables retail for $120.

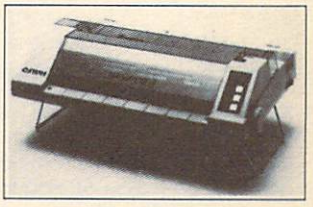
X-10 (USA) Inc.
185A Legrand Ave.
Northvale, NJ 07647
(201) 784-9700

❖ ❖ ❖ ❖

An International Printer

IBM PC-Printing In Nine Languages

ProWriter, a compact, lightweight dot matrix printer that enables IBM PC users to produce near-letter-quality documents in nine languages, is being offered by C. Itoh Digital Products. Its size (3.9 inches high, 15.2 inches wide and 11.3 inches deep) renders ProWriter versatile and portable while accommodating several useful features. A unique horizontal print head allows convenient front paper loading. A paper-saving feature prints documents on the first sheet fed into the printer. The printer is software-programmable in 9 foreign languages, making it ideal for international business uses. It prints at 105 characters per second and features a Centronics parallel interface. An Apple version will be available later this year. ProWriter retails for under $300.



C. Itoh Digital Products, Inc.
19750 South Vermont Ave., Suite 220
Torrance, CA 90502
(800) 423-0300

❖ ❖ ❖ ❖

A New Tongue for Atari

Pascal on the Atari XL, XE

Kyan Software has introduced Kyan Pascal, a programming package for the Atari XL and XE computers. The package includes a comprehensive tutorial manual and features a HELP screen, command menus, and a library of compiler error messages—all designed to assist the beginning programmer. Additional features include a 6502 machine-code compiler, a full-screen text editor, full-pass error detection, built-in assembler, and DOS 2.5 operating system. The package costs $69.95.

Kyan Software
1850 Union St. #183
San Francisco, CA 94123
(415) 775-2923

❖❖❖❖

Electronic Adventure By The Book

Interactive Fiction

Synapse Software has announced the release of Essex, the second in its Electronic Novels series. Essex, like all programs in the series, is published as a hard-cover book that introduces the characters and sets the scene. A software disk then hurls the player into deep space to conduct an intergalactic search-and-rescue mission. Confined on a spaceship, surrounded by a crew of unlikely characters, the player must find a scientist capable of preventing the destruction of the universe. The Apple and IBM versions sell for $44.95, and Commodore and Atari versions are priced at $39.95.

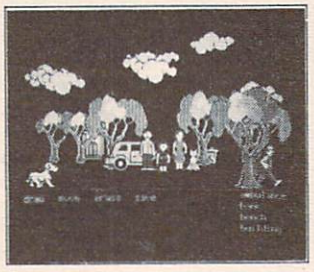
Synapse Software
17 Paul Dr.
San Rafael, CA 94903
(415) 479-1170

❖❖❖❖

New Theories Mean New Education

Research Develops Active Learning Aids

Compu-Teach has introduced a new line of educational software for use on IBM PC, PCjr, and Apple computers. The 12 programs, for use by 4- to 10-year-olds, are based on research completed at Yale University on learning theory and artificial intelligence. They are highly interactive programs that teach basic educational skills—reading, arthmetic and spelling. Each program features several skill levels or changeable parameters



that tailor the game to a child's learning level, simple commands, color graphics, and active participation by the child.

Compu-Teach, Inc.
240 Bradley St.
New Haven, CT 06511
(800) 44TEACH

❖❖❖❖

Why Pay a Stockbroker?

Wall Street Wiz On a Disk

Synapse has released Synstock, a program for the IBM PC that analyzes stock portfolio information, stock patterns and market trends. With a modem, the program can automatically log-on and download stock data from Compuserve, Dow Jones News/Retrieval, and Warner Computers. The data can be displayed in four-color charts and graphics in different formats, including averages, relative strengths, and volume indicators. Synstock supports portfolios of up to 52 stocks each; and each stock's database can hold up to 550 days of trading information. It is now available for the IBM PC, PC XT, PC compatibles with a minimum of 192K of memory, and the PCjr (128K minimum). DOS 2.0 or later version and a Graphics Adapter Card are required. Suggested retail for the IBM version is $99.95.

Broderbund Software
17 Paul Dr.
San Rafael, CA 94903
(415) 47901170

❖❖❖❖

Paint It, Then Print It

New Graphics for the TI Machine

Navarone Industries has introduced Paint 'N Print, a new graphics cartridge for the TI-99/4A computer. By manipulating a joystick or a trackball, the user can create, save and print complex images. A magnification feature permits precise control over each pixel for fine resolution work. The program will print in full color or shaded black and white on the Axiom GP-100 printer. Paint 'N Print is $39.95.

Navarone Industries, Inc.
19968 El Ray Ln.
Sonora, CA 95370
(209) 533-8349

❖❖❖❖

Interfacing Breadboard

Constructing Control Circuits

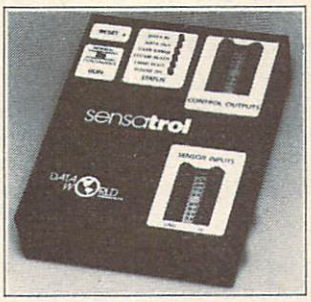
Group Technology has unveiled the BG-Board Interface Breadboard, a versatile interface that teaches the skills of interfacing by enabling the user to construct circuits and control the flow of information between the computer and external devices. Through BASIC programs, the user can employ IBM PC, Commodore 64, and VIC-20 computers in controlling and monitoring home appliances, analytical instruments, temperature control systems, security systems, and voice synthesizers. Texts and experiments provide instruction and guidance, and can be adapted to individual or classroom use. The IBM version retails for $355 (assembled) or $280 (kit). The Commodore 64 and VIC-20 versions retail for $334.95 (assembled) or $259.95 (kit).

Group Technology, Ltd.
P.O. Box 87
Check, VA 24072
(703) 651-3153

❖❖❖❖

From Sensing To Controlling

Environmental Monitoring & Control



Data World Products has introduced Sensatrol, a sensor/controller interface for environmental sensing and energy control. It allows users to measure weather conditions, control thermostats and monitor many types of environmental conditions on any computer. It has an easy-to-understand command structure in ASCII and communicates in any computer language. Extensive instructions and suggested applications are included in the package, which retails for $385.

Data World Products
P.O. Box 33
Francestown, NH 03043
(603) 588-3746

❖❖❖❖

Mastering Musical Instruments

C-64, Apple II Teach How To Play

MasterSoft has released two more programs in its Mastery in Music series. Trumpet Master and Clarinet Master convert the Commodore 64 into a generator of random music. Users may play along with millions of random exercises, or they may alter them by changing the tempos, manipulating the beat, or substituting the notes. Exercises include scales, thirds, and intervals in every major key. Trumpet Master and Clarinet Master will soon be available for Apple II family computers. It retails for $49.95.

MasterSoft
P.O. Box 1027
Bend, OR 97709
(503) 388-7654

❖❖❖❖

It's a Matter of Time

Educational Time Travel for Kids

Learning Well has announced the release of a new addition to its educational software for the Apple II family of computers. Time Capsule is a game designed to enhance reading skills by whisking players through 10 time periods in the past, the present, and the future. Adventures in each time period are described in colorful narratives. By answering related questions that test reading skills, the players control the motion of their capsule. Time Capsule is available in three reading levels, each retailing for $49.95.

Learning Well
200 South Service Rd.
Roslyn Heights, NY 11577
(516) 621-1540

❖❖❖❖

Body Cycles By Computer

Charting Biorhythms On The 99/4A

A new program from Custom Computer Programming converts your TI-99/4A into a biorhythm analyzer. The Biorhythm program computes and graphs monthly biorhythm charts based on birthdates. The program package includes a brief description of the theories behind biorhythm analysis. It is available directly from the producer for $12.95.

Custom Computer Programming
Suite 165
99 Tidemill Lane
Hampton, VA 23666

❖❖❖❖

All That Desktop Jazz

Jazz Program Integrates Office Functions

Lotus Development Corporation has introduced Jazz, a multi-function business software package for Apple's Macintosh 512K personal computer. The package incorporates five functions: worksheet, graphics, word processing, data base, and communications. HotView, a feature associated with the word processing function, allows integration with the worksheet, data base, and graphics functions. The program is easy to operate and is suitable for both the novice and the experienced user. Jazz has a retail price of $595.

Lotus Development Corp.
55 Cambridge Parkway
Cambridge, MA 02142
(617) 577-8500

❖❖❖❖

High-Tech Armchair Quarterbacks

Predict Your Favorite NFL Winners with IBM PC

A new prediction program for NFL football fans has hit the software market. Scorecast from Tradewind is a database program with a 5-window format and "Z-bar" cursor control. Users have access to a database preload of over 20,000 statistics, including final scores, rushing, first downs, turnovers, penalties, punts, passes, and returns. Also provided are current season schedules, on-screen graphics, and forecasting formulas. 23 on-screen bar charts analyze every game before and after it has been played. Users can pick their own favorites or let the program select a choice from formulas and calculations based on comparisons between any two teams. First National 800 Data Bank, an on-line data retrieval system, updates Scorecast files on Tuesday nights. Scorecast operates on the IBM PC, PCjr, and compatibles (Compaq, Tandy 1000/1200, and AT&T 6300). The program retails for $49.95 (the unprotected version is $69.95).

Tradewind Software
P.O. Box 26165
Honolulu, HI 96825
(808) 395-6700

❖❖❖❖

HOME COMPUTER

VOL. V NO. 5 ★★★★ INTERNATIONAL EDITION

Corporate Cleanup

Big Blue Announces New Strategic Moves

Boca Honchos Head North More Price Cuts Expected As Lowe Disavows PC2

It looks like a late spring cleaning. IBM is making moves to tighten product development efforts among its various divisions.

The company recently decided to move its once-independent Boca Raton division from Florida to New Jersey, and to consolidate responsibility for its communications products in North Carolina. The firm hopes to integrate the highly successful PC division into the larger corporate structure.

As these changes were announced, IBM forecasters also revised their 1985 growth rate predictions—reducing their estimates from 30% to 20%. Forecasters therefore reduced their predictions for IBM earnings. And while the PC product line is a bright spot in the IBM picture, it's undermining sales of the company's mid-range computers.

Some market analysts believe that this decline demonstrates that IBM's influence on the PC standard is no longer absolute. Others expect the company will adjust its PC product strategy to revive overall sales. IBM has cut the prices of several of its products and is planning special promotions.

Meanwhile, dealers report that the PC AT's once-booming sales are decreasing now that adequate inventory exists. Buyers are turning to the PC XT—responding to price cuts to the XT line and President Lowe's announcement that the PC2 does not exist, never did exist, and that no new PC desktop models will be introduced this year.

Lowe reportedly made this announcement so that rumors of an impending PC2 release would not harm sales of PC-compatibles. Analysts say, however, that Big Blue's main concern is not for its competitors, but for moving its own multi-billion dollar inventory excess. They say the products which have been suffering the most from the rumor of the PC2 introduction has been IBM's own product line.

"QUOTABLES"

"I'm using electrons to activate brain circuits to get you high now. Fortunately, that's still legal."

—Timothy Leary, LSD-dropping guru of the '60s, discussing his new "womb-to-tomb" computer game.

What's News—

* * *

Lotus dropped VisiCalc, the pioneering computer spreadsheet it acquired with its purchase of Software Arts. The firm has also cut the price of Spotlight, another Software Arts product, to position it against Borland's best-selling SideKick.

* * *

Tandy is aggressively pushing its MS-DOS series of computers with a $300 price cut and the availability of an add-on hard disk drive for the Model 1000. A Bernoulli Box with removeable mass-storage media has been licensed from Iomega for the Tandy line.

* * *

Texas Instruments is now offering Font Gen, software that allows creation of custom fonts for Omni 800 family printers. TI continues to hype discounts on slow moving TI Pros to members of 99/4A users groups.

* * *

Microchip technology enters new territory with high-tech items such as computerized running shoes that measure distance and calorie expenditure, and teddy bears that talk, move, and sing. No computerized toilet paper has as yet been announced.

* * *

Commodore has completed a major reorganization and expansion of its customer service and support capabilities. A Customer Service Support Line (800-247-9000) is now available as part of the new service network.

* * *

Tandy is rumored to be considering the introduction of an Apple look-alike for under $500. This would be a major policy change for the firm, but based on its IBM-clone success, the move is not unprecedented. Tandy already conveniently sells Apple and IBM software.

Restructuring The Core

Apple Stakes New Growth On Same Old Seeds

Can New Mac Attack Crack Corporate Market?

A bushel of setbacks continues to sour Apple Computer Corp., as crucial products are delayed and the company reorganizes its operations and product lines.

Apple has announced that the release of the missing links in its Macintosh Office product line—a file server and external hard-disk drive—will be delayed. Originally intending to manufacture the items itself, Apple has turned to third-party manufacturers to supply the hard-disk and file-server hardware.

Finally, Apple's President Scully, with the backing of the Board of Directors—in what's been heralded as a "palace revolt"—has revamped the company's corporate structure along more traditional lines. The move put Steven Jobs, Chairman of the Board and head of the MacIntosh Division, in the dubious position of "product visionary," and removed him from day-to-day operations. This opens the way for a new Mac architecture, reportedly opposed by Jobs. Analysts expect a more powerful Mac with an open bus and color graphics—as well as a more powerful Apple II with a 16-bit processor.

Commodore's Gamble

Firm's Recovery Rests On Amiga's Success

Shelf Space A Question As Atari Joins Fray

Commodore is gambling heavily on its new, long-awaited Amiga computer, unveiled July 23, hoping that this latest techno-marvel will revive the computer market and stem Commodore's recent tide of red ink.

The machine, based on the 68000 microprocessor, will hit the market in two versions—a 256K unit priced at $1,295, and a 512K model with RGB monitor at $1,995. Both feature 3-D animation, 3½-inch disk drive, stereo sound, and 4,000+ colors. An IBM PC emulation package is forhtcoming.

With Atari's recent release of its 520ST, both firms are scrambling for shelf space. Most retailers, however, are leary of the new products, prefering to hold off stocking them until proven in the market place—thus putting both Commodore and Atari in a "Catch 22" situation. Without the support of the largest retail chains, both firms will be hard-pressed to demonstrate market acceptance.

Industry analysts cite Atari's financial instability, and Commodore's "image" problem as major hurdles that must be straddled—and don't expect to see Commodore's name mentioned in any advertising for the new Amiga.

INDUSTRY JOURNAL™

© COPYRIGHT 1985 EMERALD VALLEY PUBLISHING CO. EUGENE, OREGON NO CENTS

Bigger Blooms for Apple

More Memory and Speed Enhance Apple II Line

Are Imminent Price Cuts Too Little, Too Late?

While Apple is struggling to push the Macintosh into the business world, the company is also refocusing its Apple II family for a rosier future.

New add-on boards from third-party manufacturers are being designed to expand Apple II's memory, increase operating speed, and add sound and music. Apple itself is reportedly working on a RAM board (the Slinky) that will expand the memory of the Apple II to 1.5 megabytes in 256K increments and plug into the IIe. Also rumored to be in the works are double-sided, double-density 3½-inch disk drives.

The ROM memory upgrade and 65C02 enhancement that Apple added to the line in March led to the present surge of Macintosh-like programs for the IIe and IIc, incorporating such popular features as mouse editing, pull-down menus, and advanced graphics.

Analysts feel, however, that it's going to take more than the much-talked-about $300 retail price cut to satisfactorily position the Apple II family against newer machines, such as the Amiga and the Atari ST.

Atari Straddles Fence

Smaller Version Of ST Slated For Mass Market

Software The Main Key To Sparking Future Sales

In an effort to keep both ends of the marketplace satisfied, Atari will be selling a less-powerful version of its new 520ST. The new 260ST system, with 256K-bytes of RAM and a built-in 3½-inch disk drive, is designed for sales in mass-market outlets at under $500.

The 520ST will be distributed through computer specialty stores, with an eye toward vertical markets. Atari claims it will have 30 to 40 vertical application programs available for the 520ST by September. VIP Professional, a Lotus 1-2-3 clone from VIP Technologies, is slated for ST debut at under $100.

Moody Market Blues

Computer Industry Slides Further Into Slump

Software Makers Eye Video Market As A Hedge

The atmosphere pervading the home computer market continues to be one of "gloom and doom." Revenues have plummeted over 20% since April 1984. Faint rumblings about a "saturated first-time buyers market" have surfaced—suggesting that the lion's share of new sales will come from owners upgrading to new machines, or from new business with government agencies.

In an effort to hedge against the market doldrums, some software manufacturers are also producing educational video tapes. The 20 million VCR owners in the U.S. is the attraction.

International Computing

Apricot Cuts Price and Pits Model Against U.S. Makers

Apple Chases Rising Sun, While IBM Hops "Down Under"

While U.S. consumers wait for the long-delayed arrival of Atari's ST, Commodore's Amiga, and IBM's PC2, foreign firms are making their move. The Apricot F1, a British machine, was cut in price by 33% to make it more attractive to U.S. consumers.

Meanwhile in Australia, IBM is marketing an English version of the JX, a machine that was previously introduced in Japan. And in an attempt to correct what has been termed a "classic failure," Apple is stepping up efforts to strengthen its weak foothold in the Japanese market with a KanaMacintosh, expected to arrive by the year's end.

THROUGH THE LOOKING GLASS

Optics May Eventually Enhance Electronics, Bringing Real Miracles To Home Computing

Electric Dreams, a hit movie about a computer falling in love, helped promote the popular notion that a home computer can do almost anything. But these inflated expectations may have led millions of new computer owners to the brink of disappointment—and to the discovery that their machines aren't able to soak up limitless databases or write hit songs on their own. Why not? Alas, fancy electronics may not be enough—even with promised advances—to perform such miracles.

What could be faster or more powerful than electrons streaming through microcircuits? The answer is *light*. New computers that use micro-optics rather than electronics to both store and greatly accelerate the flow of digital information are already in prototype. Light-based technology—ranging from optical disk drives to tiny light "gates"—may eventually replace electronic components at all levels of computer hardware. And with such light-based computers, true miracles may yet come to pass—even for home users—within the next 10 years.

At present, the most promising application of optics is in the area of computer "logic elements." Microscopic optical logic gates ("bi-stable semiconductors" built from gallium arsenide) can replace many of the logic gates in conventional microcircuits. With these components, it is much easier to construct super-large arrays that process information *in parallel*—rather than *serially*, as most computer circuits do. Optical computers should also have much faster "clock rates," based on the incredible switching speeds of optic gates. At their best, electronic gates can switch about 10^9 times a second, whereas optic gates can probably achieve up to 10^{15} switches-per-second.

Inevitably, the first applications of this technology will probably be industrial and military. "Star Wars" computers will have to think fast if they're going to shoot live missles out of the air. But will anyone think to build a computer that falls in love?

HCM ONE-LINERS

Here they are . . . the best of the one-line programs that we have received since printing the fourth "HCM One-Liners" column in *Home Computer Magazine* Vol. 5, No. 4. Although many interesting programs were submitted, we have selected what we felt were the best 6 of those that arrived prior to this issue's press date (one for each brand of computer covered in our magazine, including a TI BASIC "10-Liner"). If you have not yet submitted your masterpiece, it is not too late! As long as we keep getting great one-liners written in any computer language, we'll keep filling this page for you. Our prize winners this issue will each receive a check for $50 for sharing their ideas with our readers.

Shape-Make an Apple
[Applesoft BASIC on the Apple *IIe, IIc*]

Dear Sir:

The *Shape Maker* draws a variety of colorful geometric shapes and designs on your screen with the touch of a key. After the program has finished drawing an image, press any key, and it will automatically draw a new shape within seconds. Remember to type in the program without using spaces.

Kevin Cooney
Middleton, MA

```
1  L=80:FORI=0TO1STEP0:H
GR:HCOLOR=3:HOME:FORJ=0
TO80STEP5:L=L-(RND(1)>.
5)*(L-INT(RND(1)*160)):
K=160-L:HPLOTJ,80TOK,KT
O80,JTOL,KTO160-J,80TOL
,LTO80,160-JTOK,LTOJ,80
:NEXTJ:VTAB23:PRINT"PRE
SSANYKEY";:GETA$:NEXTI
```

A Spritely Design
[TI Extended BASIC on the TI-99/4A]

Dear Sir:

This program is called *MoTIon Art*. It creates a moving pattern of 26 sprites on your screen—a string of multi-colored squares swirl from the edge of the screen into its center. Type in the program until you hear the Input beep, press [ENTER], then [FCTN] 8, and finish typing the program in. You can change the vertical and horizontal motion of the sprites while the pattern is in motion—just type in new sets of numbers after each beep.

Bob Munro
Exton, PA

```
1  CALL CHAR(43,RPT$("F"
,16)):: INPUT "V,H":V,
H :: CALL SCREEN(2)::F
OR X=1 TO 13 :: CALL SP
RITE(#X,43,X+2,192,24,-
V,H,#28-X,43,17-X,192,2
24,-V,-H):: NEXT X ::G
OTO 1
```

Tune Up Your Keyboard
[Commodore 64 BASIC on the C-64]

Dear Sir:

By pressing different keys, you can play your own musical phrases and tunes with this program. Try this sequence of keys to play "Twinkle Twinkle:" RRGGHHG XXTT66R.

With this sequence, play "Reveille":

UU+ U+(CLR) U+(CLR) U+(CLR)
U+(CLR) +(CLR)Q(CLR)+U UU+

Fred McGorsky
Randville, MA

```
1  S=54272:X=P{SHIFT E}(
197):P{SHIFT O}S,240:P{
SHIFT O}S+1,X:P{SHIFT O
}S+24,15*-(X<>64):P{SHI
FT O}S+6,255:P{SHIFT O}S
+4,33:G{SHIFT O}1
```

A Three-Voice Tune
[TI BASIC for TI-99/4A]

Dear Sir:

This program plays a short musical tune on the TI-99/4A. Input a numerical value to tell the computer how fast to play the tune. Smaller numbers play the tune faster.

Scott Williams
Marimont, CT.

```
1  INPUT "TIME:":T
2  A$="DABCDCBCDABCDCBCD
ABCDCBCDABCDCBCDABCDCBC
DABCDCBCDABCDCBCDABCDCB
CACEACEBEABDABDCDACFAEA
DCAAAAAAA"
3  B$="IIIIIIIIHHHHHHHG
GGGGGGGFFFFFFFIIIIIIII
HHHHHHHHGGGGGGGGFFFFFFF
FIIKKKKKIIIKKKKIIIIKKKK
KKBBCCCCD"
4  C$="KKKKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKKKKKKK
KKKKKKKKKKKKKKKKKKKKKKK
KIIKKKKKIIIKKKKIIIIKKKK
KKDDFFEEI"
5  DIM N(11)
6  READ N(0),N(1),N(2),N
(3),N(4),N(5),N(6),N(7)
,N(8),N(9),N(10),N(11)
7  DATA 110,262,311,349,
392,415,440,466,494,523
,466,20000
8  FOR Z=1 TO LEN(A$)
9  CALL SOUND(T,N(ASC(SE
G$(A$,Z,1))-64),5,N(ASC
(SEG$(B$,Z,1))-64),2,N(
ASC(SEG$(C$,Z,1))-64),0
)
10  NEXT Z
```

[NOTE: Because of built-in line length limitations in TI BASIC, we are now accepting "Ten-Liners" as entries for this column—Ed.]

Drawing With the PC Pen
[BASICA on the IBM PC, Cartridge BASIC on the IBM PCjr]

Dear Sir:

I call this one *PC Pen*. It brings a new dimension to on-screen drawing with the IBM PC. You can make line drawings with this program using the rubberband technique to place lines on the screen. Press C to change the color of the lines, and O to reset their origin. To exit the program, press X.

Bob Langill
Hillsboro, OR

```
1  SCREEN 1:CLS:C=1:WHIL
E A$<>"X":WHILE A$="C":
C=(C+1)AND 3:A$="":WEND
:WHILE A$="O":A=X:B=Y:A
$="":WEND:WHILE LEN(A$)
=2:B$=RIGHT$(A$,1):LINE
(A,B)-(X,Y),0:X=X+(B$="
K")-(B$="M"):Y=Y +(B$="
H")-(B$="P"):LINE (A,B)
-(X,Y),C:A$=INKEY$:WEND
:A$=INKEY$:WEND
```

Shape Up Your Atari
[Atari BASIC on the Atari 800, 800 XL, and 130 XE]

Until this issue, *HCM* didn't cover Atari Computers. Therefore, we had no Atari One-Liner entries at press time. However, our staff has generated an Atari program that might help you get started producing your own Atari One-Liners. This program draws a variety of random geometric shapes on your screen. To type in this line, use command abbreviations and don't include spaces, otherwise the line will be too long. Double-check the line before pressing [RETURN]—which translates the shorthand commands back to their normal size (making the line longer than usually allowed).

```
1GR.24:A=159:C.3:B=INT(
RND(0)*12)/2:F.Z=0TO99S
TEP.1:PL.COS(Z)*A+A,SIN
(Z)*50+96:DR.SIN(Z*B)*A
+A.COS(Z*B)*90+96:N.Z
```

All One-Liner submissions are subject to the same publishing criteria as Letters to the Editor (explained in the magazine's masthead, page 6). If you have written a great One-Liner in any language on any computer covered by *HCM*, send it addressed to: Letters to the Editor, 1500 Valley River Drive, Ste. 250, Eugene, OR 97401. You too may win a cash prize and be immortalized in print!

CONTENTS

Apple Note: HCM Programs RUNing on the Apple II+ require 64K.
Atari Note: All HCM programs RUN on the Atari 800, 800-XL, 65-XE, and 130-XE. Shorter HCM programs should also RUN on other Atari models with less memory.
Commodore Note: All HCM Programs also RUN on the C-128 in C-64 mode.
Tandy Note: All programs for the IBM PC/PCjr commencing with the Vol. 5 No. 5 issue are designed to RUN without modification on the Tandy 1000.
Franklin Note: See page 9.

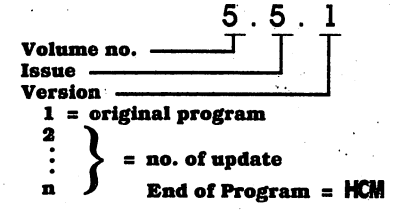
WHAT IS A PROGRAMMER'S WINDOW?

Home Computer Magazine strives to serve those eager to expand their programming knowledge by including specific information about the software published in these pages. This information is contained in a separate section: the "Programmer's Window." For each main program included in this section there are 5 different pages—one for each brand of machine we cover (Apple II family, Atari 800-compatible family, Commodore 64, IBM PC & PCjr, and TI-99/4A). Each page contains 4 separate windows as follows:

1) **Design Focus** (flow chart or diagram of a specific procedure or program structure)
2) **Remarks** (text explaining an aspect of the program version)
3) **Directory of Variables** (definition of all variables)
4) **Listing Annotations** (line-numbered program guide)

Program Identification

Each program header (the first few lines of the program) contains information giving the language the program is written in (e.g., TI Extended BASIC, Applesoft, etc.) and any special system components that are required (memory expansion cards, etc.). The first two digits of the version number tell you in which volume and issue of *HCM* the program *initially* appeared. The third digit of the version number indicates the version of the program. When a program initially appears, in *HCM*, it is version 1. Any subsequent revisions to the program, if later published in the magazine or in the software available on magnetic media from *HCM*, will bear a revised version number.



Your Guide to Typing in Programs from *HCM*

Within these pages is a software bonanza: entertainment, education, home and business applications, utilities, and tutorials—just for you. All you need to do is type them into your computer. *HCM* has taken most of the strain out of this process:

- Typeset listings with numbers in boldface.
- A bold, double vertical bar separating the line numbers from the program statements in BASIC listings.
- A vertical background grid to aid entry of the spaces.
- An error-reducing Bug-Out Code

Looking at the Character Reference Chart (Figure 1, below) will show you how each character actually appears in the program listings. By checking any questionable characters with this chart, you can reduce typing errors to a minimum.

Figure 1: Character Reference Chart

Before You Begin

Since *HCM* publishes for several different computers, the first thing you should do is make sure that you are looking at the listing designed for your machine. If, for example, you have an Apple *IIe*, make sure you look for the following black bar above the listing: APPLE II Family The computer model name will likewise appear on each subsequent page of each listing, so always look for the name before you begin typing from a new page of listings.

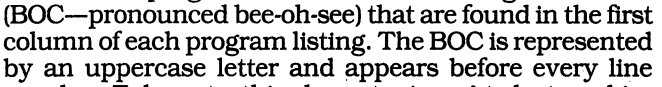
Before you begin typing-in the program, you will want to set up a system to save your program. Whether you are using a cassette or diskette storage system, now is the time to be certain it is properly connected, powered up, and loaded with a blank cassette or an initialized disk. As you type-in your program, you should get in the habit of saving your work after every 20 or so lines.

One of the most common errors in entering a listing is typing one symbol for another. These transpositions include substituting the letter O for the number 0, the letter l for the number 1, the letter S for the symbol $, and the uppercase B for the number 8. The last error is especially likely when working in hexadecimal numbers which are composed of the numbers 0-9 and the uppercase letters A-F.

The listings in *HCM* are always the same number of characters wide, but the number of characters put on any line of the video display will vary from computer to computer. Don't try to make your listings *look like* the type-set listing—instead make sure you type the listings character for character and space for space.

The Bug-Out Code (BOC)

All *HCM* programs are listed with Bug-Out Codes (BOC—pronounced bee-oh-see) that are found in the first column of each program listing. The BOC is represented by an uppercase letter and appears before every line number. Take note: this character is **not** to be typed-in. One of the main reasons that we are using letters instead of numbers is to make sure that the BOC is not entered as part of a line number.



To use BOCs for keeping bugs from creeping into your typed-in programs, see pages 77-79.

A Special Note to TI Users:

Due to architecture and language constraints with the TI-99/4A, users must have both Extended BASIC and a disk drive in order to use the BOC. We are currently developing a BOC program that will work with the Editor/Assembler Cartridge, and hope to have it soon.

A Special Note on Atari Listings

Atari has more special graphic characters available from the keyboard than any other computer that we cover. Reproducing these characters can often take up to three different key strokes. Instead of forcing our readers to translate each of these special characters into the correct key press, we include key-stroke intructions that are placed between two hands with pointing fingers. For example, when you find ☞CTRL ,☜ in an *HCM* listing, you will know to hold down the [CTRL] key and press the [,] key. The result of this key press is the ♥ symbol.

These instructions will include a number if the operation is to be repeated. For example, ☞5 CTRL A☜ tells you to hold down the [CTRL] key and press the [A] key 5 times.

Whenever you see the [CTRL] or [SHIFT] keys between two hands, hold them down while you press the key that follows them. The [ESC] key is an exception. This key represents a separate key press. So, when you see a ☞ESC CTRL +☜ you are to first *tap* the [ESC] key, and *then* hold down [CTRL] and press the [+] key.

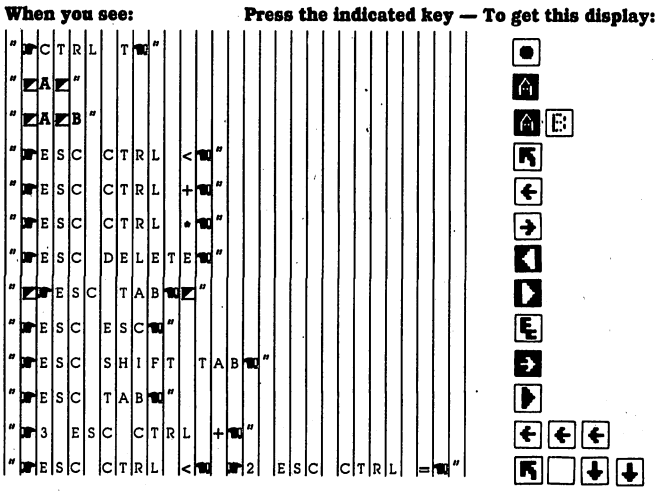
The Atari has one special key that none of the other computers have. This key looks like this ◩ and is used to print reverse characters. Whenever you come across ◩ in a listing, simply press this key.

When you come to the hand symbols, remember:

- **Each operation is enclosed in its own set of hand symbols.**
- **If any key action requires you to press [CTRL] or [SHIFT], press the control key or the shift key *first* and hold it down before pressing the second key.**
- **Both the [ESC] and ◩ keys are single key presses and should be pressed independently of the keys that follow them.**
- **Everything between a pair of hand symbols is set in a different typeface.**

In Figure 2 below, we have included a chart showing you a representative sample of the symbols that appear when you use keystrokes enclosed by the hand symbols.

Figure 2: Atari Special Symbols



A Special Note on C-64 Listings

Commodore uses more than 90 special symbols to represent various keyboard operations: for instance, the symbol [♥] in a program represents the operation of holding down the [SHIFT] key and pressing the key which has CLR on its upper half (second key from the right on the top row). This operation clears the screen.

Rather than reproducing these symbols, *HCM's* listings include key-stroke instructions, between two hands with pointing fingers. For example, when you find ☞SHIFT CLR☜ in an *HCM* listing, you will know to hold down the [SHIFT] key and press the key with CLR on it.

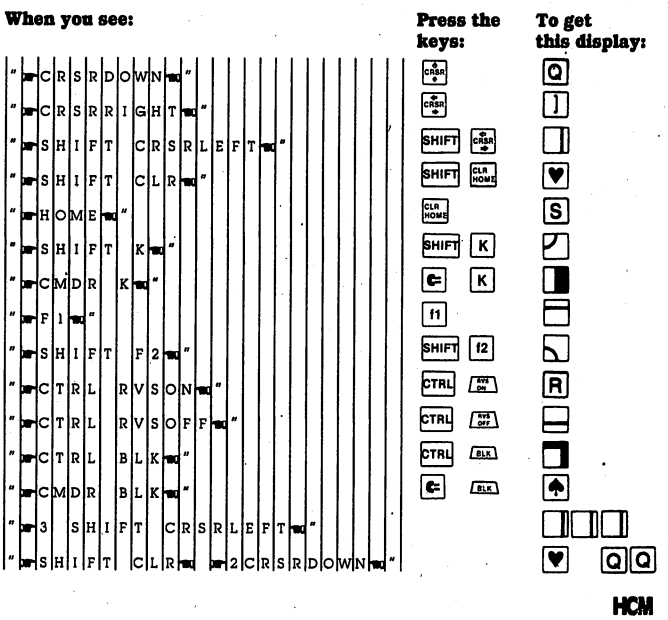
A number is included if you need to repeat the operation: ☞8 SHIFT CRSRLEFT☜ tells you to hold the [SHIFT] key down and press the cursor left key (on the bottom right of the keyboard) 8 times.

When you come to the hand symbols, remember:

- **Each operation is enclosed in its own set of hand symbols.**
- **If any key action requires you to press two keys, press the control key, the Commodore key, or the shift key *first* and hold it down before pressing the second key.**
- **Everything between a pair of hand symbols is set in a different typeface.**

In Figure 3, we have included a chart showing you a representative sample of the symbols that appear when you use keystrokes enclosed by the hand symbols. (Notice that the hand symbols always appear within quotation marks—as in a print statement.)

Figure 3: C-64 Special Symbols



HCM BUG-OUT

Bug-Out is an error detection program for catching type-in mistakes. It is available for every computer brand *HCM* covers—Apple, Atari, Commodore, IBM, and Texas Instruments. When you use this utility, typos are easily found and corrected.

Before you type-in another *HCM* program, type-in the *Bug-Out* program specified for your computer. Because a properly typed-in *Bug-Out* routine is essential for it to accurately detect typing errors in other programs, be extra careful to ensure accuracy. Once you have it entered, save *Bug-Out* to tape (an option for Atari and Commodore only) or disk.

Comparing Two Sets of BOC's

When you look at our listings, you will notice an upper-case letter placed in the left-most column at the beginning of each program line. Separated from the line numbers by a bold vertical bar, this letter is the correct BOC. Do **not** type these letters in. The BOC is a quality-control character. Each program line is carefully dissected, and mathematically compacted into a single-character representation.

These letters will help you detect key-in errors *after* you have a listing fully entered. When you RUN the *Bug-Out* program as directed below, it will generate another series of BOCs either on the screen, or to a printer. Compare the codes generated by the *Bug-Out* program with the codes published in the left-most column of our listings. If a published BOC for a line is different from a BOC for your typed version of the same line, you will know that line contains a typing error.

How To Do It

1. The first step is to type-in the desired program.
2. After typing the program in, SAVE it as usual.
3. Then also SAVE the program as an ASCII text file—this is the format needed for *Bug-Out* to do its job. Always use a different file name to distinguish between the program file SAVEd in step 2 and this text file. We suggest you add a suffix like .T (or _T on the 99/4A) to the end of the text file name for added clarity. The process of saving programs as ASCII text files on each machine is detailed on page 78 (see "Turning Programs Into Text Files").
4. After you've SAVEd your program as an ASCII text file, make sure that the disk or tape containing the text file is inserted, then RUN the *Bug-Out* program. Once RUN, *Bug-Out* will ask for the name of the ASCII file and whether you want the program's output to go to the screen or the printer. After all this has been entered, the computer will print out its list of BOCs and the corresponding line number. For example:

```
N 100
S 110
Q 120 . . .
```

5. Carefully go through the program listing in the magazine to find, and take note of, all the published BOCs that are different from the BOCs generated by the *Bug-Out* program. Every line that you find with a different BOC code has been typed incorrectly, and should be carefully examined and corrected.

To correct your mistakes (if any), LOAD (OLD on the 99/4A) the program version (not the text file) that you keyed in previously. Now, make the necessary changes to the incorrect program lines and repeat the previous 5 steps until all the BOC codes match. Once they all match, your program should be error free.

REM statements that are not typed correctly will result in erroneous BOCs. If the only differences between a typed-in program and the magazine listing are in REM statements, the program will still RUN as intended. So you needn't waste time concerning every REM statement before running your *HCM* programs. ***Continued***

TURNING PROGRAMS INTO TEXT FILES

The Apple method of making a program into a text file requires merging your typed-in program with a short *Capture* program (also included in this issue). Our version is based on the *Capture* program found on page 140 of the *BASIC Programming with ProDOS* manual published by Apple. If you do not have either *Apple Programmer's Assistant(APA)* or *Renumber* program in DOS 3.3, you must type-in *Capture* at the same time as the program you are **SAVE**ing as a text file.

In either case, with *Capture* (at lines 1 through 10), and the program you wish to capture as a text file in memory, just type **RUN**. The *Capture* program then **LIST**s the program in memory to disk as a text file under whatever name is **INPUT** when *Capture* first starts running. Because *HCM* programs always begin with line 100, the *Capture* program always **LIST**s starting at line 100, and does ***not*** **LIST** lines 1 through 10, which do the actual capturing.

Disk users enter:
LIST "D:PRGNAME.T"

Tape users enter:
LIST "C:PRGNAME.T"

Disk users enter:
OPEN 8,8,8,"PRGNAME.T,S,W" : CMD 8 : LIST
Wait till cursor returns to the screen and enter:
PRINT#8 : CLOSE 8

Tape users enter:
OPEN 1,1,1,"PRGNAME.T" : CMD 1 : LIST
Wait till cursor returns to the screen and enter:
PRINT#1 : CLOSE 1

Just **SAVE** program with the ASCII option like this:
SAVE"PRGNAME.T",A

You can **LIST** your program to disk from Extended BASIC by typing the following, with your newly typed-in program residing in memory: **LIST "DSK1.PRGNAME_T"**

Special Note to Apple Users:

Applesoft BASIC has an idiosyncracy concerning REM and DATA statements. When you press [RETURN] after entering one of these statements in a BASIC program, an extra space is added to the line between the word REM or DATA and the rest of the line. The Apple version of our Bug-Out program takes this one extra space into account. Therefore, to ensure identical BOCs, always make sure your REM and DATA statements are typed *exactly* as they appear in the listing. Although REM statements do not affect a program's performance, DATA statements are often a source of typing bugs. Therefore, after either typing in or editing a DATA statement, be sure that it has the same number of spaces as in the magazine.

BUG-OUT

APPLE // Family

```
100 REM  **********
110 REM  *  BUG-OUT  *
120 REM  **********
130 REM  COPYRIGHT 1985
140 REM  EMERALD VALLEY PUBLISHING CO.
150 REM  BY THE HCM STAFF
160 REM  HOME COMPUTER MAGAZINE
170 REM  VERSION 5.5.1
180 REM  APPLE II FAMILY APPLESOFT
190 D$ = CHR$ (4): C$ = CHR$ (13): SP$ = CHR$ (32)
200 TEXT : HOME : PRINT TAB( 10);"HCM BUG-OUT PROGRAM LISTING TO DISK": PRINT : PRINT "PLACE DISK WITH ENOUGH": PRINT "ROOM FOR TEXT FILE IN DRIVE 1.": PRINT : INPUT "PRESS RETURN WHEN READY";A$
210 ONERR GOTO 380: PRINT D$;"OPEN";: INPUT "";F$: PRINT D$;"DELETE";F$: PRINT D$;"OPEN";F$;",D1": PRINT "OUTPUT TO 2.";: PRINT "SCREEN": PRINT "SLOT 1": PRINT "1" AND K$ > "2" THEN 260
220 GET K$: IF K$ = "1" THEN 290
230 PRINT D$;"PR#1": PRINT "READ";F$
240 GET C$: PRINT C$;: IF C$ = C$ THEN 300
250 IF C$ = CHR$ (13) GOTO 340
260 A$ = A$ + C$: GOTO 300
270 IF A$ = "" THEN 300
280 IF LEFT$ (A$,1) = SP$ THEN A$ = RIGHT$ (A$, LEN (A$) - 1)
290 LN = VAL (A$): B$ = STR$ (LN): GOSUB 500
300 PRINT SP$ + B$ + SP$ + OT$: A$ = "": GOTO 290
310 POKE 216,0: EN = PEEK (222): EL = PEEK (218) + PEEK (219) * 256
320 FOR I = EN TO EN + 5 THEN 430
330 PRINT D$;"CLOSE";F$: PRINT D$;"PR#0": END
340 ON (EN = 6) + 2 * (EN = 8) + 3 * (EN = 16)) GOTO 440,450,460
350 IF LEFT$ (A$,1) = SP$ THEN A$ = RIGHT$ (A$, LEN (A$) - 1)
360 LN = VAL (A$): B$ = STR$ (LN): GOSUB 500
370 PRINT SP$ + B$ + SP$ + OT$: A$ = "": GOTO 290
380 POKE 216,0: EN = PEEK (222): EL = PEEK (218) + PEEK (219) * 256
390 IF EN < > 5 THEN 430
400 PRINT "** DONE **  WITH "; FL$
410 PRINT D$;"CLOSE";F$: PRINT
420 PRINT D$;"PR#0": END
430 ON (EN = 6) + 2 * (EN = 8) + 3 * (EN = 16)) GOTO 440,450,460
440 PRINT FL$ + " IS NOT ON DISK": GOTO 470
450 PRINT "I/O ERROR-CHECK DRIVE": GOTO 470
460 PRINT "ILLEGAL FILE NAME-TRY AGAIN"
470 CALL - 328: PRINT "PRESS ANY KEY TO CONTINUE": GET K$: RUN
480 PRINT "ERROR #";EN;" DETECTED AT LINE";EL: PRINT "TRY STARTING AGAIN"
490 END
500 CK = 0: C1 = 0: FOR I = 1 TO LEN (A$): MID$ (A$, I, 1)) : C1 = C1 + ASC ( MID$ (A$, I, 1)): NEXT : CK = CK + CH: CK = CK - INT (CK / 26) * 26: IF CK = CH THEN RETURN
510 IF I = 2 THEN CK = CK + CH: GOTO 530
520 CK = CK + CH
530 NEXT I: CK = ABS (CK) * C1: I = INT (CK / 26): OT$ = CHR$ (65 + CK): RETURN
```

HCM

BUG-OUT

ATARI 800/800XL/130XE

```
100 REM  **********
110 REM  *  BUG-OUT  *
120 REM  **********
130 REM  COPYRIGHT 1985
140 REM  EMERALD VALLEY PUBLISHING CO.
150 REM  BY THE HCM STAFF
160 REM  HOME COMPUTER MAGAZINE
170 REM  VERSION 5.5.1
180 REM  ATARI BASIC FOR THE 800, 800XL,
         130 XE
190 DIM FILE$(16),IN$(255),BL$(7):BL$="
200 ? "ESC CTRL <
    HCM BUG-OUT
    or ENTER DEVICE & FILENAME:
210 ? FILE$;:INPUT FILE$:IF FILE$="" THEN 330
220 TRAP 340
230 OPEN #1,4,0,FILE$:? "LIST TO SCREEN
    OR PRINTER (S/P)?":POKE 764,255
240 K=PEEK(764):IF K<>62 AND K<>10 THEN
    240
250 ?  :? "READING...":TRAP 330
260 INPUT #1;IN$:P=1
270 IF IN$(P,P)<>" " THEN P=P+1:GOTO 270
280 CK=0:CK1=0:FOR I=1 TO LEN(IN$):CK1=
    CK1+ASC(IN$(I,I)):IF I/2=INT(I/2) T
    HEN CK=CK-ASC(IN$(I,I)):GOTO 300
290 CK=CK+ASC(IN$(I,I))
300 NEXT I:CK=ABS(CK)*CK1:I=INT(CK/26):
    IF K=10 THEN LPRINT CHR$(CK+65);BL$
    (1,6-P);IN$(1,P):GOTO 260
310 PRINT CHR$(CK+65);BL$(1,6-P);IN$(1,
    P):GOTO 260
320 CLOSE #1:? :? "DONE":POKE 764,255:E
    ND
330 ? :STATUS #1,ST:? "**I/O ERROR**":S
    T=ST/128:IF ST=1 THEN SOUND 1,50,10,
    8:FOR I=1 TO 100:NEXT I:SOUND 1,0,0,
    0:CLOSE #1
340 GOTO 200
```

HCM

BUG-OUT

COMMODORE 64

```
N  100 REM ************
Z  110 REM *  BUG-OUT  *
F  120 REM ************
M  130 REM COPYRIGHT 1985
R  140 REM EMERALD VALLEY PUBLISHING CO.
O  150 REM BY THE HCM STAFF
N  160 REM HOME COMPUTER MAGAZINE
V  170 REM VERSION 5.5.1
G  180 REM COMMODORE 64 BASIC
B  190 PRINT "{SHIFT CLR}{CTRL RVSON}              HCM BUG-OUT      "
F  200 PRINT "           PROGRAM MUST HAVE BEEN SAVED"
L  210 PRINT "           AS A SEQUENTIAL TEXT FILE."
Z  220 F$="":INPUT "NAME OF LISTED FILE";F$:IF F$="" THEN END
Y  230 PRINT:PRINT "TAPE OR DISK (T/D)?"
I  240 GET K$:IF K$<>"T" AND K$<>"D" THEN240
Q  250 IF K$="D" THEN270
Q  260 OPEN 1,1,0,F$:GOTO290
P  270 OPEN 15,8,15:OPEN 1,8,2,F$+",S,R":INPUT#15,E,E$
U  280 IF E THEN PRINT "** "E$" **":FOR I=1 TO 1000:NEXT:CLOSE1:CLOSE15:GOTO190
X  290 PRINT:PRINT"LIST TO SCREEN OR PRINTER (S/P)?"
N  300 GET K$:IF K$<>"S" AND K$<>"P" THEN300
E  310 PRINT:PRINT "READING...":IF K$="P" THEN OPEN 4,4
H  320 GET#1,CH$:IF CH$<"0" OR CH$>"9" THEN320
P  330 IN$=CH$
P  340 GET#1,CH$:IN$=IN$+CH$
X  350 IF CH$=CHR$(10) THEN CLOSE1:CLOSE4:CLOSE15:PRINT:PRINT"DONE":END
N  360 IF CH$<>CHR$(13) THEN340
Y  370 P=1
F  380 IF MID$(IN$,P,1)<>" " THEN P=P+1:GOTO380
C  390 CK=0:C1=0:FOR I=1 TO LEN(IN$)-1:C1=C1+ASC(MID$(IN$,I,1))
A  400 IF I/2=INT(I/2) THEN CK=CK-ASC(MID$(IN$,I,1)):GOTO420
D  410 CK=CK+ASC(MID$(IN$,I,1))
J  420 NEXT
P  430 CK=ABS(CK)*C1:I=INT(CK/26):CK=CK-I*26
Q  440 IFK$="S"THENPRINTCHR$(CK+65)LEFT$("      ",6-P)LEFT$(IN$,P),:IN$="":GOTO340
B  450 PRINT#4,CHR$(CK+65)LEFT$("      ",6-P)LEFT$(IN$,P):IN$="":GOTO340
```

HCM

BUG-OUT

IBM PC/PCjr, TANDY 1000

```
I  100 '  ************
W  110 '  *  BUG-OUT  *
Q  120 '  ************
X  130 '  COPYRIGHT 1985
W  140 '  EMERALD VALLEY PUBLISHING CO
H  150 '  BY THE HCM STAFF
U  160 '  HOME COMPUTER MAGAZINE
E  170 '  VERSION 5.5.1
X  180 '  DOS 2.1 AND EITHER
D  190 '  IBM PC WITH BASICA
U  200 '  OR IBM PCjr WITH CARTRIDGE BASIC
H  210 '  TANDY 1000 WITH GW BASIC
F  220 CLS:SCREEN 0:WIDTH 40: SP$=CHR$(32):CR$=CHR$(13):LOCATE 3,10:PRINT "HCM BUG-OUT PROGRAM"
Q  230 PRINT "FILE MUST BE A BASIC PROGRAM SAVED WITH THE A(SCII) OPTION"
K  240 LOCATE 6,3:PRINT "Input File Name: ";:INPUT FL$:IF FL$="" THEN 220 ELSE FL$=LEFT$(FL$,12)
N  250 LOCATE 7,3:PRINT "Which disk drive? A";
I  260 LOCATE 7,21,1:DR$=INKEY$:IF DR$="" THEN 260 ELSE IF INSTR("AaBb"+CR$,DR$) THEN IF DR$=CR$ THEN DR$="A" ELSE 270 ELSE 260
A  270 PRINT DR$:LOCATE 9,3,0:PRINT "Output to (1) Screen":LOCATE 10,13:PRINT "(2) Line Printer"
F  280 LOCATE 12,3:PRINT "Your Choice: "
N  290 LOCATE 12,16,1:DEV$=INKEY$:IF DEV$<>"1" AND DEV$<>"2" THEN 290 ELSE PRINT DEV$:IF DEV$="1" THEN DEV$="SCRN:" ELSE DEV$="LPT1:"
I  300 OPEN DR$+":"+FL$ FOR INPUT AS #1
O  310 OPEN DEV$ FOR OUTPUT AS #2
T  320 IF EOF(1) THEN 350
N  330 LINE INPUT #1,C$ :GOSUB 370
D  340 N=INSTR(C$,SP$):PRINT #2,LEFT$(C$,N);CHR$(65+CK):GOTO 320
I  350 PRINT "THAT'S ALL":END
V  360 ' CHECK SUM
E  370 CK=0:CK1=0:FOR I=1 TO LEN(C$):CHK=ASC(MID$(C$,I,1)):CK1=CK1+CHK:IF I/2=INT(I/2) THEN CK=CK-CHK ELSE CK=CK+CHK
N  380 NEXT I
R  390 CK=ABS(CK)*CK1:I=INT(CK/26):CK=CK-I*26
G  400 OT$=CHR$(65+CK):RETURN
```

HCM

BUG-OUT

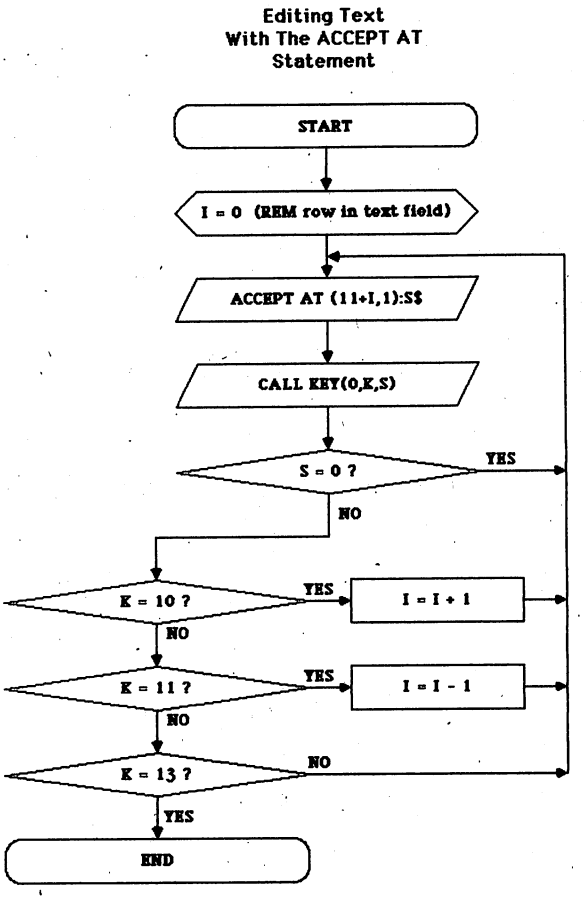
TI-99/4A

```
P  100 REM ************
Q  110 REM *  BUG-OUT  *
P  120 REM ************
M  130 REM COPYRIGHT 1985
H  140 REM EMERALD VALLEY PUBLISHING CO
A  150 REM BY THE HCM STAFF
R  160 REM HOME COMPUTER MAGAZINE
G  170 REM VERSION 5.5.1
T  180 REM TI EXTENDED BASIC ONLY
D  190 SP$=CHR$(32):: CR$=CHR$(13)
X  200 DISPLAY AT(2,5)ERASE ALL:"HCM BUG-OUT PROGRAM"
N  210 DISPLAY AT(4,1):"YOU MUST ""LIST"" THE PROGRAM TO DISK." :: DISPLAY AT(6,1):"THEN RUN ""BUG-OUT"""
P  220 DISPLAY AT(8,1):"FILE NAME: DSK1." :: ACCEPT AT(8,12)SIZE(-15):FL$
F  230 ON ERROR 520
R  240 OPEN #2:FL$,INPUT ,DISPLAY ,VARIABLE 80
Q  250 DISPLAY AT(10,1):"SEND OUTPUT TO: 1. SCREEN" :: DISPLAY AT(11,17):"2. PRINTER"
M  260 DISPLAY AT(12,1):"1" :: ACCEPT AT(12,1)VALIDATE("12")SIZE(-1):DV :: DV=DV-1
C  270 IF DV=0 THEN 310
T  280 ON ERROR 530
T  290 DISPLAY AT(13,1):"PRINTER:" :: DISPLAY AT(14,1):"RS232" :: ACCEPT AT(14,1)SIZE(-28):DV$
W  300 OPEN #1:DV$,OUTPUT
N  310 ON ERROR STOP
Z  320 IF EOF(2)THEN 500
A  330 LINPUT #2:C$
E  340 IF C$="" THEN 320
Y  350 N=POS(C$,SP$,1):: OT$=SEG$(C$,1,N):: LNNUM=VAL(OT$):: IF LEN(C$)<80 THEN 400
A  360 FLAG=1 :: IF EOF(2)THEN FLAG=0 :: GOTO 400 ELSE LINPUT #2:D$
Z  370 M=POS(D$,SP$,1):: IF M>5 THEN 390
Y  380 ON ERROR 540 :: LN=VAL(SEG$(D$,1,M)):: IF LN>LNNUM THEN 400
Y  390 C$=C$&D$ :: FLAG=0
M  400 CK$=C$ :: M1=POS(C$," REM ",N):: M2=POS(C$," ! ",N):: IF M1<1 AND M2<1 THEN 450
A  410 IF M1=N THEN CK$=SEG$(C$,1,N+4):: GOTO 450
A  420 IF M2=N THEN CK$=SEG$(C$,1,N+2):: GOTO 450
H  430 IF M1>N THEN CK$=SEG$(C$,1,POS(C$," REM ",1)+4):: GOTO 450
Y  440 IF M2>N THEN CK$=SEG$(C$,1,POS(C$," ! ",1)+2)
W  450 CK=0 :: CK1=0 :: FOR I=1 TO LEN(CK$):: CHK=ASC(SEG$(CK$,I,1)):: CK1=CK1+CHK :: IF I/2=INT(I/2)THEN CK=CK-CHK ELSE CK=CK+CHK
M  460 NEXT I
C  470 CK=ABS(CK)*CK1 :: I=INT(CK/26):: CK=CK-I*26
W  480 OT$=OT$&SP$&CHR$(65+CK)
S  490 PRINT #DV:OT$ :: IF FLAG=0 THEN C$="" :: GOTO 320 ELSE C$=D$ :: FLAG=0 :: GOTO 350
A  500 CLOSE #2 :: PRINT "ALL DONE" :: IF DV=1 THEN CLOSE #1
H  510 END
H  520 DISPLAY AT(10,1)BEEP:"FILE NOT FOUND" :: GOSUB 550 :: GOTO 200
P  530 DISPLAY AT(14,1):"PRINTER NOT RECOGNIZED" :: GOSUB 550 :: RETURN 280
X  540 CALL ERR(EN,ET,ES,EL):: IF EN=74 THEN RETURN 390 ELSE PRINT "ERROR NUMBER:";EN;" IN LINE ";EL :: STOP
M  550 FOR DE=1 TO 500 :: NEXT DE :: RETURN
```

HCM

Card-Trix

DESIGN FOCUS



LISTING ANNOTATIONS

Line Nos.

100-200	Program header.
210-220	Initialize program.
230	Display title screen.
240-270	Main menu.
280-350	Edit screen menu.
360	Edit Index field.
370	Edit Subject field.
380-440	Edit Text field.
450-480	Enter card number.
490-520	Erase card.
530-550	Get a Y or N response.
560	Increment card number (Forward).
570	Decrement card number (Back).
580-600	Copy card.
610-630	Paste card.
640-700	Sort cards.
710-790	Search cards.
800-950	Print cards.
960	Error routine.
970-980	Update edit screen.
990-1010	Save card folder.
1020-1040	Load card folder.
1050-1100	Get device and file name.
1110-1170	Draw edit screen.
1180	Read keyboard.
1190-1210	Exit program?
1220	Character data.
1230	Box coordinates for edit screen.

REMARKS

One of the strongest features of TI Extended BASIC is the ACCEPT AT statement. ACCEPT AT enables your program to accept data at any location on the screen, *beep* when ready to accept the data, erase the screen or just the data field prior to input, limit the length of the data, and limit the type of characters that you can input. The versatility and control that the ACCEPT AT command provides for data entry made the TI machine a perfect choice for our *Card-Trix* program.

We did, however, run into a slight problem, when using ACCEPT AT for entering data into *Card-Trix*'s Text field. The ACCEPT AT statement allows you to enter only one screen line of data at a time. Because the Text field on a *Card-Trix* card contains 9 lines, we had to find a way to move from one line to another. The solution to this problem was using the CALL KEY function in conjunction with the ACCEPT AT statement.

There are three different ways to exit an ACCEPT AT—the [ENTER] key, FCTN E, and FCTN X. By putting a CALL KEY statement directly after the ACCEPT AT, we can tell which of the three keys was used to exit from the ACCEPT AT statement. The Design Focus on the left provides a flow chart of the Text editing portion of *Card-Trix*. This routine is located in program lines 380-440.

When you first enter the Text edit routine, the variable I is set to zero, signifying that the cursor is located on the first row of the Text field. Next, an ACCEPT AT statement executes using I as an offset to the vertical position of the cursor. Now, the CALL KEY subprogram is called, returning the ASCII code of the key used to exit the ACCEPT AT. If for some reason the CALL KEY is not quick enough to catch the exit key, the program loops back to the ACCEPT AT statement.

If you press a FCTN X (ASCII 10), then I is incremented, thus moving the cursor down a line in the Text field. If you enter a FCTN E (ASCII 11), then I is decremented, moving the cursor up a line. Finally, if you press an [ENTER], then the program exits the routine. Additional limit checking, such as making sure that I never equals anything less than zero or greater than 8, is not shown in the flow chart.

HCM Glossary terms: field, subprogram.

DIRECTORY OF VARIABLES

Variables	Functions
A	Utility variable.
A$	Used in search routine.
B	Utility variable.
C$(,)	Array used to hold card data.
E	Used to return file name error.
E$	String of editing command characters.
I	Loop counter.
J	Loop counter.
K	ASCII of keypress.
MX	Maximum number of cards in a folder.
N	Current card number.
S	Status variable in CALL KEY statement.
S$	Utility string.
X	Utility variable.
Y	Utility variable.

Electronic Typewriter

REMARKS

When a program is running, the Commodore 64's cursor normally flashes only when the machine is executing an INPUT statement. Although convenient, the INPUT statement is not the best method of obtaining information from a user—you could move the cursor off the input line during an INPUT, scroll, or even clear the screen! In addition, the INPUT command ignores commas and colons. Its drawbacks are numerous.

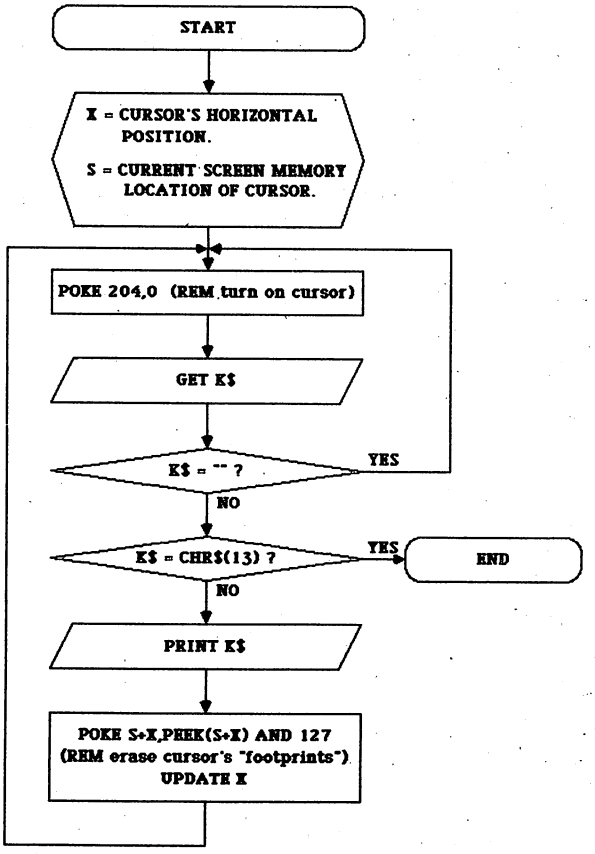
Because of the limitations of BASIC's INPUT command, the Commodore version of *Electronic Typewriter* uses the GET statement to receive all of its data. Now, what about the cursor? Won't it refuse to appear if you use a GET? It is true that the cursor cannot normally be used in anything but INPUT statements, but there is a devious way to "get" around this. By POKEing location 204 with a zero just before your GET command, you can fool the computer into displaying the cursor.

This method of turning on the cursor is not completely error free, however. There are some things you should watch for. When you turn the cursor on through POKEing 204, the cursor does not turn itself off when moved. In other words, letters are often left in reverse and little cursor footprints are left all over the screen. The input routine located in lines 920-1200 of *Electronic Typewriter* takes care of these problems. To erase the cursor's footprints, screen memory is POKEd to turn off the reverse status of the last position of the cursor. By clearing the high bit of a character's screen code, you turn off its reverse status. See the Design Focus for a more detailed look. Although this technique has its drawbacks, it provides a friendlier and safer method of input.

HCM Glossary terms: high bit, screen code.

DESIGN FOCUS

Turning On The Cursor
During a GET



LISTING ANNOTATIONS

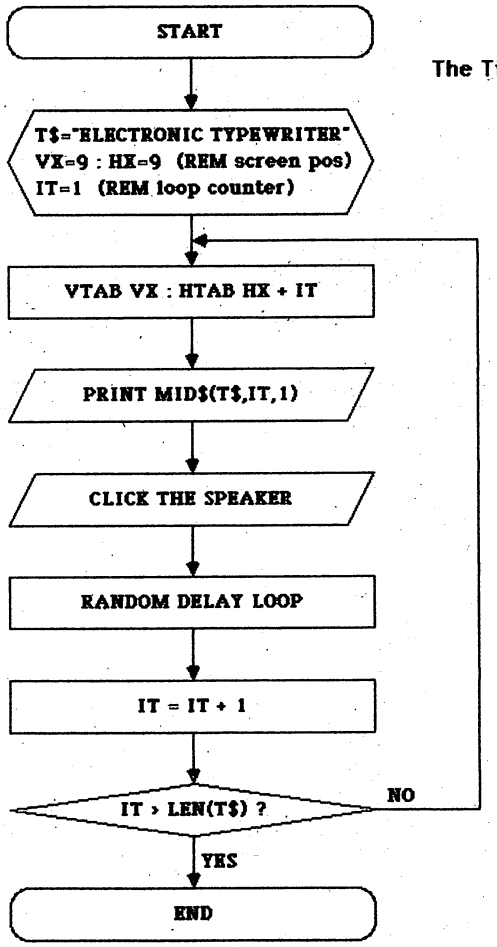
Line Nos.	
100-190	Program header.
200-270	Display title screen.
280-360	Initialize program.
370-380	Main control loop.
390-440	Line number.
450-560	Line code.
570-730	Text.
740-780	Menu.
790-850	Update the editing screen.
860-870	Beep beep sound.
880-890	Ding sound.
900-910	Thwack sound.
920-1200	Input routine.
1210-1240	Set/reset printer status.
1250-1270	Read disk error.
1280-1310	Display message and beep.
1320-1330	Plot cursor at X,Y.
1340-1380	Erase data.
1390-1440	Exit program?
1450-1500	Get a yes or no response.
1510-1660	Set margins and spacing.
1670-1820	Load text file.
1830-2000	Save text file.
2010-2110	Load template.
2120-2220	Save template
2230-2300	Get file name and device.
2310-2480	Print whole page.
2490-2980	Draw edit screen.
2990-3030	Sound data.

DIRECTORY OF VARIABLES

Variables	Functions
B	Input routine parameter.
BL$	Character string of 80 spaces.
C$	Used to hold line code character.
E	Set to disk error number, if any.
E$	Disk error descriptor.
F1$	String containing line number.
F7$	String containing last menu choice.
I	Loop counter.
IO	Flag used by input routine.
J	Loop counter.
K	ASCII of keyboard input.
K$	Keyboard input string.
L	Line length for input routine.
LM	Left margin.
LN	Line number for text.
M	Current editing screen mode.
MX	Maximum number of text lines.
PR	Printer status. 0/1 = OFF/ON.
RM	Right margin.
S	Utility variable.
S$	Utility string.
SP	Line spacing.
T$	Utility string.
TP$( )	Template array.
TX$( )	Text array.
X	X coordinate of cursor.
Y	Y coordinate of cursor.

Electronic Typewriter

DESIGN FOCUS



The Title Screen

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-260	Initialize program.
270-280	Main control loop.
290-430	Initialize variables.
440-490	Display title screen.
500-660	Set margins and spacing.
670-820	Draw edit screen.
830-890	Update screen.
900-1100	Text.
1110-1170	Toggle printer status.
1180-1280	Line code.
1290-1440	Menu.
1450-1590	Print document.
1600-1730	Load text file.
1740-1880	Save text file.
1890-1970	Load template.
1980-2040	Save template.
2050-2080	Exit program?
2090-2120	Erase data.
2130-2260	Print a line of text.
2270-2310	Sound routines.
2320-2390	Print messages.
2400-2570	Number entry.
2580-2600	Turn printer on.
2610-2620	Turn printer off.
2630-2920	Get file name.
2930-3130	Error routines.
3140-3200	Get a yes or no response.

REMARKS

The first thing you see when you RUN the Apple version of *Electronic Typewriter* is the title screen. When executed, the title screen seems to magically type the name of the program onto the screen. Although this part of the program may seem trivial, it does add a touch of class and entertainment to the program.

The title screen is created in lines 440-490. This routine is actually very simple. A string, T$, is set equal to the words "Electronic Typewriter." Once a screen position is defined, a simple FOR loop is used to print T$ onto the screen one letter at a time. As each letter is plotted onto the screen, using the Apple's VTAB and HTAB commands, a sound is produced to simulate the *thwack!* sound of a typewriter. After the title is "typed" onto the screen, you are prompted to press [RETURN] and continue with the program. The Design Focus shown on this page provides a flow chart of this routine.

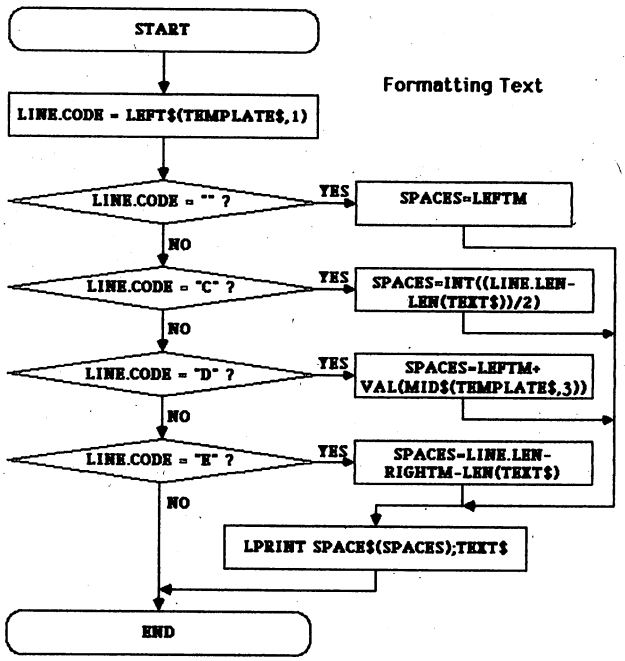
HCM Glossary terms: loop, string.

DIRECTORY OF VARIABLES

Variables	Functions
BE$	Bell sound, CHR$(7).
BL$	80 blank characters.
C	Selects print mode from line code.
C$	First character of line code.
CN	Value of number input by user.
CN$	String equivalent of CN.
CN$( )	Array used to build CN$.
D$	Used for disk commands.
DI,IT	Loop counters.
DR$	Disk drive number.
E	Error code.
ESC$	Escape character, CHR$(27).
FL$	File name.
FL$( )	Array used to build FL$.
FX$	Control keys used.
HC	Horizontal screen position during input.
HX	General use horizontal screen position.
IN$	Character input.
L	Line length in text input.
L1	First line for document printout.
L2	Last line for document printout.
LM	Left margin.
LN	Document's current line number.
M	Current editing screen mode.
MN$( )	Menu selections.
MX	Maximum number of text lines.
N1	Vertical location during number entry.
N2	Horizon location during number entry.
N3	Maximum number of chars. for input.
PR	Printer status.
PX	Control code selector.
RM	Right margin.
RS	Disk error program "resume" selector.
S$,S2$,T$	Utility strings.
SD	Memory location of sound routine.
SP	Line spacing.
SV	Saves document line number.
SV$	Flag to save template with text.
TP$( )	Template array.
TX$( )	Text array.
VC,VX	Vertical screen positions.
X	Utility variable.
XT	Pointer for truncating blanks from file.

Electronic Typewriter

DESIGN FOCUS



REMARKS

Lines 700-800 in the IBM version of *Electronic Typewriter* are the work horses behind printing formatted text. These few lines are responsible for interpreting line codes and sending the proper output to the printer. To further illustrate this process, we have provided a simple flowchart (see the Design Focus on this page). As you can see, the only real trick behind printing a formatted line is calculating the number of spaces to be used as an indent. For an explanation of variables, refer to the chart below.

LISTING ANNOTATIONS

Line Nos.	
100-210	Program header.
220-280	Display title screen.
290-320	Initialize program.
330-550	Set margins and line spacing.
560-570	Main control loop.
580-810	Text mode.
820-890	Line number.
900-1070	Line code.
1080-1090	Thwack sound.
1100-1110	Ding sound.
1120-1180	Display a message on the screen.
1190-1200	Beep beep sound.
1210-1240	Toggle printer status.
1250-1290	Get an ENTER and continue.
1300-1420	Define a few constants.
1430-1490	Menu.
1500-1630	Box display routines.
1640-1980	Input routine.
1990-2010	Cursor right.
2020-2040	Cursor left.
2050-2070	Toggle insert mode.
2080-2100	Backspace.
2110-2130	Delete a character.
2140-2160	Trap for file name error.
2170-2220	Display file name error.
2230-2310	Get file name.
2320-2360	Erase all data.
2370-2420	Get a yes or no response.
2430-2720	Print document.
2730-2840	Load text file.
2850-3020	Save text file.
3030-3100	Load template.
3110-3180	Save template.
3190-3250	Exit program?
3260-3560	Display edit screen.
3570-3640	Update edit screen.

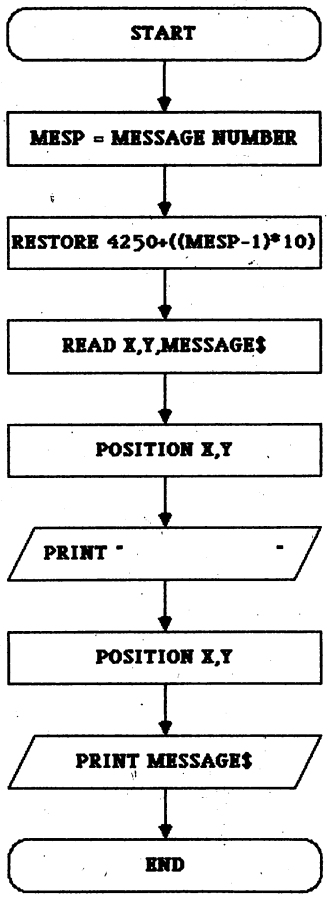
DIRECTORY OF VARIABLES

Variables	Functions
T$,S$,X$	Utility strings.
I,J	Loop counters.
MAX.LINE	Maximum number of lines.
LINE.LENGTH	Maximum number of chars. per line.
TEXT$( )	Text array.
TEMPLATE$( )	Template array.
RIGHTM	Right margin.
LEFTM	Left margin.
CUR.LINE	Current line.
MODE	Current mode.
ROW,COL	Used to specify a box.
WDTH,HEIGHT	Used to specify a box.
MIN.CHARS	Specifies range in input routine.
MAX.CHARS	Specifies range in input routine.
PROW,PCOL	Used to specify a "prompt" position.
LINE.CODE	Template code of current line.
SPACES	Number of spaces from edge of paper.
STATUS	Keyboard status.
LENGTH	Length of current input field.
EMPTY.LINE$	Used when saving text and template.
BORDER.COLOR	Color of border.
PON,POFF	Printer status flag.
MESSAGE.ROW	Row message will be printed at.
MESSAGE.COL	Column message will be printed at.
MESSAGE.LENGTH	Length of message box.
PSTATUS.ROW	Row position of printer status box.
PSTATUS.COL	Column position of printer status box.
CODE.ROW	Row position of line code box.
CODE.COL	Column postion of line code box.
LNUM.ROW	Row positon of line number box.
LNUM.COL	Column positon of line number box.
TEXT.ROW	Row position of text window.
TEXT.COL	Column position of text window.
MENU.ROW	Row position of menu box.
MENU.COL	Column postion of menu box.
CHANGE.LINE	Edit mode, F1.
CHANGE.COL	Edit mode, F2.
TEXT, MENU	Edit modes, F3 and F4.
ERASE.DATA	Edit mode, F10.
ULC,URC,LLC	Used to specify the sides of a box.
LRC,H,V	Used to specify the sides of a box.
SPOS	String position in the insert routine.
INSERT	Insert mode flag.
CHARS	Number of characters in string.
YES	Flag.
PROMPT$	Used to specify file operation.
F$	File name.
QT	Used to mask quotation marks.
BLACK,BLUE	Colors used in SCREEN mode 0.
GREEN,CYAN,RED	Colors used in SCREEN mode 0.
MAGENTA,BROWN	Colors used in SCREEN mode 0.
WHITE,GREY	Colors used in SCREEN mode 0.
YELLOW,LBLUE	Colors used in SCREEN mode 0.
LGREEN,LCYAN	Colors used in SCREEN mode 0.
LRED,LMAGENTA	Colors used in SCREEN mode 0.
HWHITE	Colors used in SCREEN mode 0.

Electronic Typewriter

DESIGN FOCUS

Print a Message To The Screen



LISTING ANNOTATIONS

Line Nos.	
100-180	Program header.
190-210	Display title screen.
220-300	Main control loop.
310-420	Input a two digit number.
430-560	CIO routine.
570-640	Get file name.
650-840	Disk I/O.
850-880	Set I/O for cassette.
890-920	Toggle printer status.
930-1030	Menu.
1040-1140	Line number.
1150-1340	Erase data.
1350-1700	Text mode.
1710-1800	Update line.
1810-1990	Line code.
2000-2230	Print document.
2240-2420	Load text file.
2430-2630	Save text file.
2640-2760	Load template.
2770-2900	Save template.
2910-2980	Exit program?
2990-3150	Dimension strings.
3160-3180	Bell sound.
3190-3240	Print a series of messages.
3250-3690	Display edit screen.
3700-3790	Pick function input.
3800-4160	Output to printer.
4170-4550	Print a message.
4560-4780	Set margins and spacing.
4790-4870	Format indent.

REMARKS

One of the Atari computer's best features is the unique ability to use calculated GOTOs, GOSUBs, and RESTOREs. On most computers, commands such as RESTORE require that a constant be used as a parameter. The Atari version of *Electronic Typewriter* uses the calculated RESTORE command in lines 4170-4550 to print messages on the screen. All you have to do is set the variable MESP to point to one of the messages that are stored in DATA statements in lines 4240-4550, then GOSUB 4170. For an illustrated view of how this routine works, refer to the Design Focus on this page.

HCM Glossary terms: parameter.

DIRECTORY OF VARIABLES

Variables	Functions
A,P,Z	Utility variables
ADD	Address of string to be saved or loaded.
ADH	High byte of ADD.
ADL	Low byte of ADD.
BLANK$	A string of blanks.
C	Column during input.
CFLAG	Flag to specify cassette I/O.
CHAR	Last function key presses.
CHN	I/O channel.
CYCLES	Loop counter.
DEFAULT	Default settings.
EXT$	File name extension.
FILE$	File name.
FILLER$	Utility string.
HI	Upper limit for numeric input.
IN$	Input string.
INDENT	Indent value.
INDENT$	ASCII of indent value.
IOCB	I/O commands.
K	Last keypress.
L	Loop counter.
LCODE$	String that contains template.
LINES	Utility variable.
LNUM	Current line number.
LNUM$	String equivalent of LNUM.
LMAR	Left margin.
LO	Lower limit for numeric input.
LSPACE	Line spacing.
MAX	Length limit for current text line.
MAXLEN	Maximum line length.
MAXLINES	Maximum number of lines.
MESH	Number of last message.
MESL	Number of first message.
MESP	Pointer to message.
MESSAGE$	Current message.
ND	Last line for document printout.
NMH	High byte of NUM.
NML	Low byte of NUM.
NUM	Number of bytes.
OUT$	Output string.
PFLAG	Printer status.
R	Row during input.
RING	Flag used for bell sound.
RMAR	Right margin.
RWF	Read write flag.
SETTING( )	Used for setting margins and spacing.
ST	First line for document printout.
TEXT$	String that contains text.
TFLAG	Flag to save template with text.
WAIT	Loop counter.
WORK$	Utility string.
X	Column during input.
XBLANK	String of blank chars.
Y	Row during input.

The Plains of Salisbury

REMARKS

It is often necessary to write a separate input routine on the TI-99/4A computer when using BASIC, because the standard INPUT statement is inadequate. (If you have Extended BASIC, the ACCEPT AT statement is capable of handling most input needs.) TI BASIC offers a powerful but under-used function that can simplify input routines: POS. With this function, you can locate the position of a string of characters within another string.

Writing your own input routine can eat up a lot of memory, and slow down a program. In special cases, the POS function can help alleviate some of these problems. Often, we need the program to allow the user to enter only certain characters. These characters may be scattered throughout the character set, making limit checking very difficult and slow. By placing a list of legal keys in a string variable, we can use the POS function on each key entered to see if it is on the list.

Another benefit of the POS function, is that it will tell you the position of the character in the list that matches the character entered. This number can be used with an ON GOTO or an ON GOSUB statement to control program flow. This greatly decreases the amount of code needed to make a separate test for each legal character. An alternative to this method is to use the POS function to test for illegal keys (e.g., in an input routine for a file name).

HCM Glossary terms: character set, limit checking, program flow.

DESIGN FOCUS

USING POS()
TO VERIFY CHARACTER ENTRY

```
260 D$="EXSD"&CHR$(13)&CHR$(15)
*
1210 CALL KEY(0,K,S)
*
1260 K=POS(D$,CHR$(K),1)
1270 IF K=0 THEN 1210
1300 ON K GOTO 1310,1330,1350,1370,2020,1390
*
```

```
POS(D$,CHR$(K),1)
```

- 1 — CHARACTER POSITION WITHIN D$ TO START THE SEARCH FOR CHR$(K)— START SEARCH AT FIRST CHARACTER POSITION
- CHR$(K) — THE POSITION OF THIS CHARACTER WITHIN D$ WILL BE RETURNED— K IS THE VALUE RETURNED FROM CALL KEY
- D$ — D$ CONTAINS A LIST OF ALL LEGAL CHARACTERS; EXSD, CHR$(13)=[ENTER], CHR$(15)=[FCTN 9]

DIRECTORY OF VARIABLES

Variables	Functions
M$( )	Screen messages.
P$( )	Players' names.
L( )	Knight data.
M( )	Players' scores.
R( )	Player's knights remaining.
A$	Utility for input and display.
B$	Contains the map arrangment.
C$	Contains knight status in combat.
D$	Movement phase legal keys.
E$	List of legal terrain types.
F$	File name.
A	Utility.
A0, A1, A2, A3, A4, A5, A6, A8	Constants 1, 2, 3, 4, 5, 6, and 8.
B	Utility.
C	Character on the screen.
C1	Character code for player's knight.
D	Counterattack factor.
E	Knight off screen flag.
F	Defense factor during combat phase.
FL	Load game flag.
G	Enemy knight; hand-to-hand.
H	Movement factor cost.
J	Movement units remaining.
K	ASCII value of key pressed.
N	Knight # during movement phase.
P	Current player (offensive).
P2	Current enemy (defensive).
Q	Current screen being displayed.
S	Screen knight is on.
U	Direction to move.
V	Utility; loop counter.
X	X coordinate where arrow is fired.
Y	Y coordinate where arrow is fired.
Z	Utility loop counter.

LISTING ANNOTATIONS

Line Nos.	
100-180	Program header.
190-250	Initialize program variables.
260-270	Set up two user functions.
280-440	Title screen, character graphics.
450-510	Get option to load an old game.
520-550	Get players' names.
560-620	Get the map arrangement.
630-730	Set up for the start of a game.
740-880	Main control loop.
890-940	Start of the movement phase.
950	Start knight's turn.
960-1190	Update and check knight's vital statistics.
1200-1370	Scan keyboard, determine direction.
1380-1390	Branch to Save/Exit/Return submenu.
1400-1560	Routine to move knight up or down.
1550-1820	Routine to move knight left or right.
1830-1870	Calculate the movement factor.
1880-2020	Check for hand-to-hand combat.
2030-2060	End of a knight's movement phase.
2070-2470	Hand-to-hand combat.
2480-2520	Prepare for combat phase.
2530-2660	Select a knight for combat.
2670-2700	Change screens if necessary.
2710-2800	Select a direction to fire the arrow.
2810-2890	Determine the X and Y coordinate to fire.
2900-3260	Fire the arrow.
3270-3330	Display a different map screen.
3340-3430	Display knights on the current screen.
3440-3580	Display a map.
3590-3610	Character graphics data.
3620	Character color data.
3630-3680	Data for map 1.
3690-3740	Data for map 2.
3750-3800	Data for map 3.
3810	Data for knight's X and Y coordinates.
3820-3850	Key-scan routine.
3860-3900	Control routine to display screen text.
3910-3940	Routine to display one line of text.
3950-3970	Time-delay routine.
3980-4040	Save/Exit/Return option routine.
4050-4160	Exit routine; display score screen.
4170-4180	Return to game.
4190-4300	Save a game to tape or disk.
4310-4400	Load a game from tape or disk.

The Plains of Salisbury

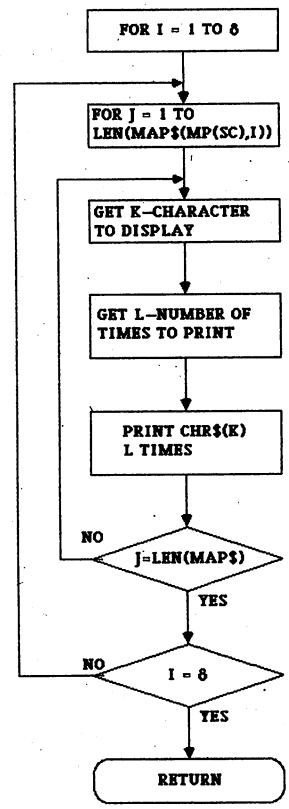
REMARKS

The Commodore 64 offers more than one way to place graphics on the screen. In *The Plains of Salisbury*, we needed to display three full screens of graphics. The easiest method would have involved simply printing the redefined graphics characters to the screen. This was not desirable, however, because it would have consumed an excessive amount of program memory. So we designed a more memory efficient algorithm to display the screens.

The flow chart on this page shows the structure of the algorithm. The program reads DATA into a string variable, then takes two characters at a time from the string. The first character indicates which graphics character to display. The second character determines how many times that graphics character repeats. This method, while slower than printing entire screen lines at one time, does conserve quite a bit of memory.

HCM Glossary terms: algorithm, string, redefined graphics characters.

DESIGN FOCUS



SCREEN GRAPHICS ALGORITHM TO SAVE MEMORY

DIRECTORY OF VARIABLES

Variables	Functions
BL$	Contains CHR$(144).
CL$	Contains CHR$(147).
CR$	Contains CHR$(13).
DN$	Contains CHR$(17)+CHR$(17).
E$	Used to read the error channel.
FL$	File name.
J$	Used in retrieving the value for terrain.
K$	Returns character entered.
KE$	Used in INPUT routines.
MP$	Used to input map arrangement.
Z$	Used to display "'A KILL."
GR$( )	Contains the graphics characters.
MA$( )	Contains the map arrangement.
NA$( )	Players' names.
SP$( )	Used for printing spaces.
TX$( )	Contains screen message text.
AR( )	Knights' status during combat phase.
CO( )	Determines player colors.
MP( )	Map arrangement.
PL( )	Contains information on each knight.
S( )	Players' scores.
TR( )	Used to restore the PL( ) array.
WI( )	Number of knights still alive.
DV	File device number.
E	Contains the error condition.
FI	Constant for 56334.
HO	Enemy player (defensive).
I	Utility; loop counter.
J	Utility; loop counter.
K	Utility.
K%	Character returned from the screen.
KE	Key pressed.
KN	Current knight.
L	Determine screen positions.
M	Utility.
N	Enemy knight number.
OK	Constant value of 1.
Q	Points to locations 217 to 242 to clear the screen link list.
SA	Secondary address; file operations.
SC	Current screen.
TD	Time delay loop counter.
TE	Current player.
VL	Flag used when changing screens.
X	Flag indicates if a knight is hit.
Y	Loop counter.
Z	Loop counter.
ZZ	Indicates a game was just loaded.

LISTING ANNOTATIONS

Line Nos.	
100-210	Program header.
220-340	Display the current map area and the knights.
350-490	Initialize program.
500-540	Title screen.
550-560	Load old game option.
570-600	Get players' names.
610-650	Get map arrangement.
660-680	Determine player's turn.
690-770	Update and check knight's vital statistics.
780-800	Get input and branch to appropriate routine.
810-840	Change screens for display only.
850	Start next knight.
860-900	Update knight.
910-1020	Move the knight.
1030-1200	Check for hand-to-hand combat and adjust for terrain type.
1210-1240	Check border limits.
1250-1420	Hand-to-hand combat.
1430-1460	Combat phase menu.
1470-1530	Pick knight for combat. Check for eligibility.
1540	Change screen if necessary.
1550-1580	Determine direction of fire.
1590-1790	Fire arrow and calculate the consequences.
1800-1810	Save/Exit menu.
1820-2010	Save game.
2020-2080	Exit game, display report screen.
2090-2300	Load old game.
2310-2360	Get a key from the keyboard.
2370-2400	Error routine.
2410-2700	Program data.

The Plains of Salisbury

REMARKS

The Apple II version of *The Plains of Salisbury* offers a solution to the inadequacy of the INPUT statement. In almost any BASIC language for most applications the INPUT statement provides virtually no automatic limit checking; you must write elaborate routines to scan the data after you have finished typing in the entire line and pressed [RETURN]. The optimum input routine would check for legal characters as they are entered or prohibit the use of illegal characters altogether.

If all of the legal or illegal characters fell into one or two ranges, the problem of sorting them out would be fairly simple (e.g., all letters between A and F, or X to Z). Suppose, however, that you wanted to allow a scattered list of keys to be entered (e.g., I, J, K, M, [RETURN], and [ESC]). This is a common problem in many game programs where only a few select keys are used, while many are not.

By keeping a list of either the legal or the illegal keys in a string, you can easily check any key pressed against the list to determine whether you should allow it or not. In *The Plains of Salisbury*, a simple FOR-NEXT loop searches the string.

You can then use the position of the key pressed within the string with an ON GOTO or ON GOSUB statement to further simplify a program. To change the keys used, simply change the list in the string. For a more versatile input routine, compile several lists of keys for different parts of the program. You can then pass the appropriate list to the input routine to do the proper checks.

HCM Glossary terms: application, limit checking.

LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-240	Relocate program above hi-res screen.
250-330	Initialize the program and shapes.
340-350	Get sound effects option.
360-380	Load an old game option.
390-400	Get players' names.
410-450	Get map arrangement.
460-470	Set up for a new game.
480-520	Main control loop.
530-1080	Movement phase.
530-610	Get knight's vital statistics.
620-710	Get input and determine direction.
720-770	Move knight up or down.
780-860	Move knight left or right.
870-910	Get movement factor for this move.
920	Sound routine.
930-1070	Hand-to-hand combat phase.
1080	End of movement phase.
1090-1340	Combat phase.
1090-1160	Get knight for combat.
1170	Change screen routine.
1180-1240	Determine direction of fire.
1250-1340	Fire arrow and determine outcome.
1350-1380	Routine to change maps.
1390-1410	Key input routine.
1420-1450	Routine to display knights on the screen.
1460-1510	Save/Exit option menu.
1520-1600	Exit program, display report screen.
1610-1900	Program data for screen display.
1910-1990	Load an old game.
2000-2070	Save a game.
2080-2190	Error routine.
2200-2270	Program data for sound and graphics.

DESIGN FOCUS

USING A FOR NEXT/LOOP TO VERIFY KEY INPUT

```
270 D$="IMJK"+CHR$(27)+CHR$(13)
*
*
*
620 GOSUB 1390          <-- GO GET A KEY
630 IF A$=" " THEN HOME : GOSUB 1350 : GOTO 570
640 FOR K=1 TO 6 : IF MID$(D$,K,1)<>A$ THEN NEXT
650 IF K>6 THEN 620     <-- KEY NOT IN LIST
660 ON K GOTO 670,680,690,700,710,920
*
*
*
```

D$=

1	2	3	4	5	6	POSITION IN STRING
I	M	J	K	CHR$(27) [ESC]	CHR$(13) [RETURN]	CHARACTER IN STRING

[RETURN] PRESSED
A$=CHR$(13)—SIXTH POSITION
PROGRAM WILL BRANCH TO LINE 920

[J] PRESSED
A$="J"—THIRD POSITION
PROGRAM WILL BRANCH TO LINE 690

[I] PRESSED
A$="I"—FIRST POSITION
PROGRAM WILL BRANCH TO LINE 670

DIRECTORY OF VARIABLES

Variables	Functions
A$	Utility and input variable.
D$	Legal key list for inputs.
M$( )	Map segment labels.
P$( )	Players' names.
B( )	Contains physical order of the screens.
C%( )	Knight status during combat phase.
L( )	Contains data on both players' knights.
M( )	Players' scores.
R( )	Number of knights still alive.
S%( )	Screen memory coordinates.
A	Attacked knight in hand-to-hand combat.
B	Left screen edge.
C	Type of terrain under a knight.
D	Right screen edge.
E	Flag indicates the screen has changed.
F	Attack factor in combat phase.
G	Flag indicates hand-to-hand combat is to commence.
H	Movement factors needed for this move.
J	Number of movement factors left.
K	Indicates which key was pressed.
N	Indicates the current knight.
OFF	Constant 2051. Turn sound off.
P	Current player (offensive).
P2	Defensive player.
PD	Prodos flag.
PRNT	Constant 2048. Display characters.
Q	Current screen.
S	Screen current knight is moving toward.
SOUND	Constant 2054. Sound effects routine.
U	Indicates direction of movement.
V	Loop counter.
X	Horizontal direction to fire.
Y	Vertical direction to fire.
Z	Utility loop counter.

The Plains of Salisbury

REMARKS

Quite often, it is a challenge to get a large amount of graphics onto the screen in a short period of time. We can solve this problem on the IBM PC by using the BLOAD command. It is impractical, however, to publish an entire BLOAD file in the magazine, so we have come up with an alternative.

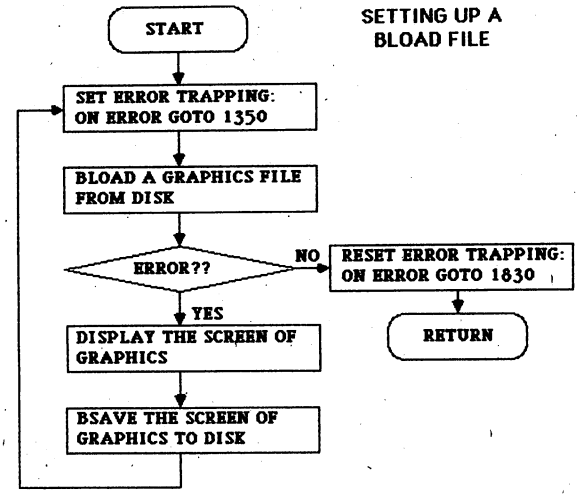
If a graphics image file is found during BLOADing, the program loads that file and continues. If no file is found, an error is generated. The program traps for that error condition and branches to a routine that uses another algorithm to paint the screen. The program then uses the BSAVE command to save the screen's image to the disk. In this program, there are three screens of graphics which must be saved to the disk. Unless you change data disks, the saving process will not be repeated, and your screens will be drawn very quickly.

HCM Glossary terms: graphics image file, trap.

LISTING ANNOTATIONS

Line Nos.	
100-240	Program header.
250-270	Title screen and program initialization.
280-340	Graphics initialization.
350-360	Option to load an old game.
370	Get players' names.
380-430	Get map layout.
440-460	Main control loop.
470-890	Movement phase.
470-520	Check each knight's vital statistics.
530-570	Determine direction of movement.
580-640	Move knight up or down.
650-710	Move knight left or right.
720-730	Change screens if necessary.
740-790	Replace terrain under knight's last position.
800	Routine to change screens.
810	End of movement phase.
820-830	Routines to update knight's vital statistics on the screen.
840-890	Hand-to-hand combat routine.
900-1070	Combat phase.
900-910	Check to see which knights have a supply of arrows.
920-930	Get knight for combat.
940	Change screen if necessary.
950-1000	Input direction to fire arrow.
1010-1050	Fire arrow and calculate consequences.
1060-1070	Combat display and exit routines.
1080-1250	Exit/Save option.
1080-1090	Get option.
1100-1170	Get file name for Load and Save options.
1180-1200	Save game to disk.
1210-1240	End of game report, and option to Play again.
1250	Error routine for Save option. Return to top of Save routine.
1260-1280	Load old game option.
1290	Error routine for Load option. Return to top of Load routine.
1300	Get single character and beep routine.
1310-1340	Load the maps from disk.
1350-1370	Control routine to display knights on a new screen.
1380-1500	Knights' display subroutines.
1510	Read the map array.
1520-1630	Routine to draw the maps and save them if the maps are not already on the disk.
1640-1820	Program data.
1830-1880	Error routine.

DESIGN FOCUS



DIRECTORY OF VARIABLES

Variables	Functions
A$	Returns character from keyboard.
B$	Contains legal file name characters.
C$	Knight status during combat.
F$	File name.
M1$, M2$, M3$, M4$, M5$, M6$	Used to DRAW the numbers 1 through 6.
MAP$	Contains map arrangement.
NAM$( )	Players' names.
KNIGHT( )	Keeps track of each player's knights.
SB1( ), SB2( ), SB3( ), SB4( ), SB5( ), SB6( )	Blue knights' shape arrays.
SBUILD( )	Building shape array.
SCN( )	Battle field contents for three screens.
SCORE( )	Player scores
SFORT( )	Fort shape array.
SGRASS( )	Grass shape array.
SR1( ), SR2( ), SR3( ), SR4( ), SR5( ), SR6( )	Red knights' shape arrays.
SROAD( )	Road shape array.
STREE( )	Tree shape array.
SWATER( )	Water shape array.
A	Utility variable.
ALIVE	Number of knights left for each player.
ATT	Attack factor.
C	Terrain type under a knight.
CATT	Counterattack factor in combat.
EL	Line where error occurred.
ER	Error codes for error routine.
FL	Flag for old game.
K	Key selected during movement phase.
LIM	Limit check value for edge boundaries.
LM	Indicates either Load or Save mode for the file name input routine.
MLEFT	Number of movement factors remaining.
MV	Direction of movement.
N	Current player's knight.
P	Current player (offensive).
P2	Defensive player.
Q	Current screen being displayed.
S	Screen knight is moving toward.
SMD	Indicates either movement or combat phase to control returns.
TD	Time delay loop counter.
TN	Temporary storage for current knight.
TQ	Temporary storage for current screen.
X	Knight's horizontal position.
XP	Knight's horizontal pixel position.
XO	Horizontal X offset for a screen.
Y	Vertical position of a knight.
YP	Vertical pixel position of a knight.
Z, ZZ	Utility variable.

The Plains of Salisbury

REMARKS

As you may know by now, the Atari BASICs that run on the 800, 800XL, and 130XE do not directly support string arrays. Don't lose faith too quickly, however, because it is possible to build your own array from a single string. Strings on the Atari can be any length—up to the limitations of your system's memory, of course. Therefore, if you plan the string requirements ahead of time, it is a simple matter of doing a few calculations, and *presto*, you have a functional string array.

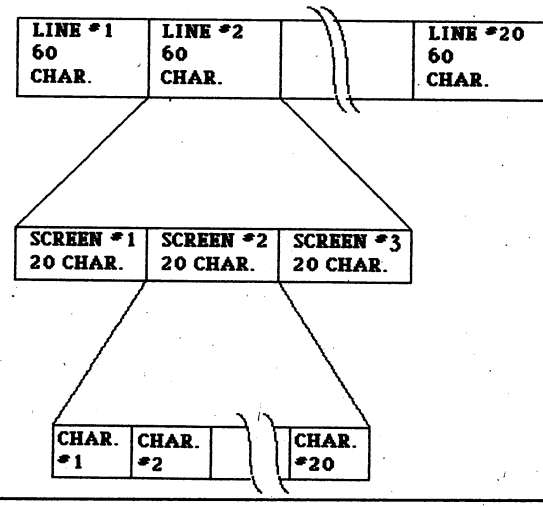
The Plains of Salisbury requires a two-dimensional string array to store a screen image. This storage is necessary to speed up the re-painting of the character graphics screens during the game. Because the Atari does not support string arrays, we store the data in a single, long string. By extracting substrings from the single string, we simulate a multiple element array.

The Design Focus on this page illustrates how the program stores the array in a single, long string. One drawback to this method is that all elements of the array must be of the same length for you to use a simple mathematical algorithm to determine the array element locations. This particular array is designed to display three screens of graphics in graphics mode 1 with a split screen. A variable-length element string array would be much more complex, requiring a numeric array to keep track of each element's start and stop locations.

HCM Glossary terms: string, string arrays, elements, variable-length element string array, graphics mode, split screen.

DESIGN FOCUS

CREATING A STRING ARRAY FOR SCREEN GRAPHICS



LISTING ANNOTATIONS

Line Nos.	
100-210	Program header.
220-270	Initialize program.
280-290	Load old game option.
300-330	Get players' names.
340-380	Get map arrangement.
390-410	Control loop for setting up graphics.
420-480	Main control loop.
490-560	Subroutine to set up characters.
570-1250	Movement phase.
570-660	Adjust and check knights' vital statistics.
670-750	Get input and determine direction.
760-880	Move knights left or right.
890-930	Move knights up or down.
940-990	Movement factors needed for this move.
1000	Control the displaying of the knights.
1010	End of the movement phase.
1020-1140	Hand-to-hand combat routine.
1150-1190	Change screens for display only.
1200-1250	Display knight's status on the screen.
1260-1560	Combat phase.
1260-1340	Pick a knight for combat.
1350-1360	Change screens if nessesary.
1370-1430	Determine the direction to fire.
1440-1560	Fire arrow and calculate the consequences.
1570-1640	Save/Exit option menu.
1650-1730	Save game routine.
1740-1830	Exit game and display report screen.
1840-1980	Get file name routine.
1990-2070	Load an old game routine.
2080-2170	Display knights.
2180-2280	Input routine.
2290-2320	Screen display routine.
2330-2420	Build the terrain string array.
2430-2520	Graphics character data.
2530-2740	Screen image format data.
2750-2770	Data for initial knights' positions.

DIRECTORY OF VARIABLES

Variables	Functions
A$	Input string for input routine.
CL$	Contains CHR$(125); clears the screen.
DEV$	Contains the file name parameters.
E$	Contains CHR$(155).
LN$	Used to read a line of screen data.
MAP$	Map arrangement.
N$	Players' names.
S$	Contains screen display format image.
Q$	Contains the terrain values image.
FA( )	Knight status during combat phase.
L( )	Contains information on knights.
M( )	Players' scores.
R( )	Number of knights alive for each player.
SO( )	Screen offsets for calculating positions.
A	Utility variable.
ADD	Memory address pointer.
AE	File OPEN parameter.
B	Utility variable.
BRI	Used to set the color brightness.
C	Counterattack factor.
COL	Used to set color.
DIR	Indicates direction of movement.
DX	Horizontal direction of arrow fire.
DY	Vertical direction of arrow fire.
FL	Load game flag.
FDX	Arrow target; X coordinate.
FDY	Arrow target; Y coordinate.
G	Combat flag.
GT	Enemy encountered in hand-to-hand combat.
H	Movement factor cost.
KN	Player's knight number.
LIM	Used for map border limit checks.
LS	Load/Save mode flag.
ML	Maximum string length in input routine.
N	Knight number index into L( ) array.
NM	Number of movement factors remaining.
P	Current player (offense).
P2	Enemy player (defense).
Q	Screen map pointer.
S	Screen to which knight moves.
SQ	Real screen being displayed.
TD	Time delay loop counter.
TE	File OPEN parameter.
TQ	Temporary screen map pointer.
X	Horizontal screen coordinate.
Y	Vertical screen coordinate.
Z	Utility, loop counter.
ZZ	Utility, loop counter.

Vital Signs

DESIGN FOCUS

HI RESOLUTION BAR CHART

ASCII CHARACTER	CHARACTER CODE
128	0000000000000000
129	00000000000000FF
130	000000000000FFFF
131	0000000000FFFFFF
132	00000000FFFFFFFF
133	000000FFFFFFFFFF
134	0000FFFFFFFFFFFF
135	00FFFFFFFFFFFFFF
136	FFFFFFFFFFFFFFFF

CHARACTER TO ERASE IN CASE BAR MOVES DOWN

```
3690 CALL HCHAR(INT((100-PB)*.1+1),18,128)
3700 CALL HCHAR(INT((100-PB)*.1+2),18,
     INT(128+(PB-INT(PB*.1)*10)*.69))
```

BLANK CHARACTER

CALCULATES CHARACTER FOR THIS BAR POSITION

* PB IS SCALED TO BE BETWEEN 0 AND 100, AND IS USED TO DETERMINE BOTH THE POSITION OF THE CHARACTER AND THE ASCII CHARACTER TO USE

REMARKS

If you think that the TI-99/4A graphics character set is limited when generating high-resolution graphics such as bar charts, take another look. In *Vital Signs* three bar charts display blood pressure, oxygen level, and body temperature. To create the chart, we had to redesign only 8 characters—the rest of the magic is all in the display algorithm.

The Design Focus on this page displays the hexadecimal codes that we used to redefine characters 128 through 136. The hexadecimal code is displayed next to the ASCII character number. These codes allow a vertical graduation of one pixel in the bar graphs. The two program lines appearing in the Design Focus are from the routine that plots the bar graph for blood pressure. Before the program reaches line 3690, it scales the value of PB to a figure between 0 and 100.

Lines 3690 and 3700 contain the expression (100-PB)*.1 which further scales PB to a value between 1 and 10 so that a vertical column can be plotted using the HCHAR statement. Line 3700 contains the value 18 which indicates column 18. In the same line, 2 is added to the expression to define the upper limit of the graph at the screen's second row. The rest of the line calculates which of the 8 graphic characters is required at this level of the bar. In line 3690, 1 is added to the expression to clear the space above the character in case the bar is going down.

HCM Glossary terms: hexadecimal, ASCII, algorithm, pixel.

LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-380	Initialize program.
390-820	Display the playing screen.
830-1040	Main control loop.
1050-1280	Routines to keep track of vitals when they extend beyond normal limits.
1290-1580	Update vitals.
1590-1710	Change Activity when operating at random.
1720-1850	Change Air Quality at random.
1860-1920	Get lung cancer.
1930-1960	Get a blood clot.
1970-2100	Lung cancer cured.
2110-2190	Blood clot cured.
2200-2310	Key-input routine.
2320-2610	Routines to increase and decrease heart and respiration rates.
2620-2810	Routine to change Activity.
2820-2960	Routine to change Air Quality.
2970-3220	End-of-game routine.
3230-3490	Update status of conditions (Activity, Air Quality, etc.)
3500-3590	Display heart and respiration rates.
3600-3630	Clear lower-left corner of the screen.
3640-3910	Display bar graphs.
3920-3900	Routines to display text on the screen.
4000-4030	Single-key input routine.
4040-4110	Graphics character data.
4120	Character color assignments.
4130-4140	Screen-graphics format.
4150-4230	End-of-game messages.
4240-4300	Initialize game variables for the start of a new game.

DIRECTORY OF VARIABLES

Variables	Functions
A	Used in defining character color.
A0	Current Activity option.
A1	Current Activity.
A2	Activity level.
B	Used in defining character color.
BC	Blood-clot flag.
CNT	Used to control body temperature.
CO	Change in oxygen level.
D	End-of-game flag.
HR	Heart rate.
K	(Return) key pressed.
LC	Lung cancer flag.
LCC	Counter to determine how long lung cancer has resided in lung.
LZ	Air Quality lung damage.
OB	Used in the display of oxygen level.
OC	Oxygen out-of-limits counter.
OX	Oxygen level.
P	Blood pressure.
PA	Pressure change due to oxygen.
PB	Used in pressure display.
PC	Pressure out-of-limits counter.
R1	Air Quality option.
R2	Air Quality factor.
RS	Respiration rate.
S	Key-scan status.
SC	Score.
T	Body temperature.
T1	Used in the temperature display.
TC	Temperature out-of-limits counter.
TOX	Temporary oxygen variable.
TP	Temporary pressure variable.
TTA	Temporary temperature variable.
X	Horizontal screen coordinate.
Y	Vertical screen coordinate.
Z	Utility and loop counter.

Vital Signs

REMARKS

In the C-64 version of *Vital Signs*, lines 500 to 610 hold what is commonly called the "main control routine." Main control routines may consist solely of a main menu routine, or they can encompass many operations. It is best to keep the routine simple—just a series of branches to more complex subroutines—to make it readable.

This routine in *Vital Signs* is responsible for the execution and maintenance of the rest of the program. Its purpose is to control the sequence of events. Among other tasks, the routine checks to see if the player has either lost, or decided to quit the game.

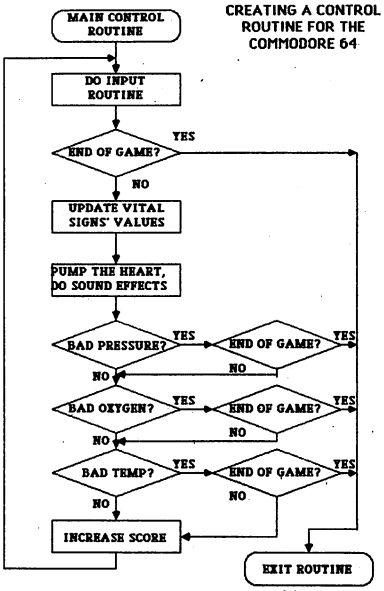
The main control routine's first operation is a branch to the input subroutine, which is responsible for getting the player's input and carrying out commands. After returning from the input routine, the main routine checks to see if the player has decided to exit the game.

The next operation is a branch to the update vital statistics subroutine. This routine calculates the new values for blood pressure, oxygen level, and body temperature. After exiting this subroutine, the main routine sets a flag to indicate the first position of the graphic heart on the screen, and then branches to a routine that draws the heart in its second position—thereby performing the simple animation of the beating heart. The main routine then calls a sound routine, and branches again to the heart graphics routine, repeating the process.

The last section of the main control routine branches to three subroutines to check the status of the vital signs. Any one of these routines could set a flag ending the game, at which time the program adds up the score.

HCM Glossary terms: flag.

DESIGN FOCUS



LISTING ANNOTATIONS

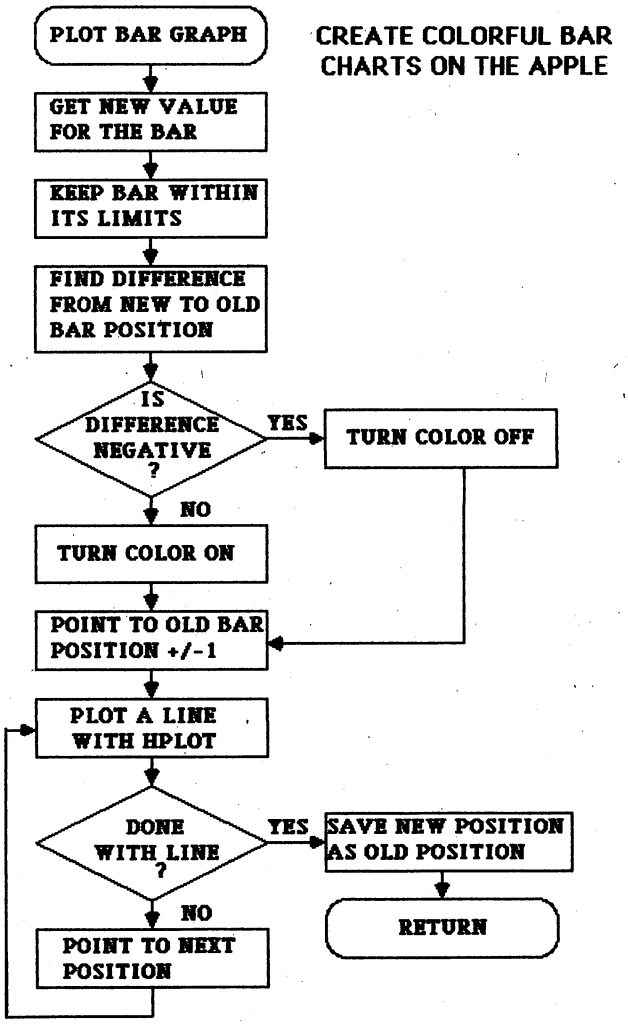
Line Nos.	
100-200	Program header.
210-250	Title screen.
260-490	Display the playing screen.
500-610	Main control loop.
620-660	Check for pressure out-of-limits.
670-700	Check for oxygen out-of-limits.
710-740	Check for temperature out-of-limits.
750-950	Update vital signs.
960-1010	Check for lung cancer.
1020-1070	Check for a blood clot.
1080-1290	Cure lung cancer and blood clot.
1300-1380	Scan keyboard and branch to appropriate routine.
1390-1500	Change heart and respiration rates.
1510-1530	Heartbeat.
1540-1610	Change Activity option.
1620-1660	Change Air Quality option.
1670-1800	End-of-game routine, option to play again.
1810-1890	Display the current conditions: Activity, Air Quality, etc.
1900-1910	Display the heart and respiration rates.
1920	Clear a part of the screen.
1930-2150	Update vital signs.
2160-2230	Display end of game messages.
2240-2260	Initialize program variables.
2270-2330	Relocate and initialize the sprite table.
2340-2630	Program data.
2640-2690	Sound routines.

DIRECTORY OF VARIABLES

Variables	Functions
AC$( )	Activity display text.
AR$( )	Air Quality display text.
A$	Used in the display routine.
BL$	Contains spaces for clearing the screen.
DW$	Contains formatting characters.
HM$	Used to place cursor at top of text area.
K$	Contains input character.
Q1$	Dummy variable to step through data.
AC( )	Activity level values.
AR( )	Air Quality level values.
A0	Activity option.
A1	Activity level.
A2	Activity value.
BC	Blood clot flag.
CL	Address for Voice 1 frequency low byte.
CNT	Counter for sweating.
CO	Change in oxygen level.
D	Flag to indicate end of the game.
F	Target body temperature.
HR	Heart rate.
K	ASCII value of K$.
L2	Lung cancer time counter.
LC	Lung cancer flag.
LZ	Chance of getting lung cancer.
M1	Used in sound routine.
M2	Used in sound routine.
OB	Used in the oxygen bar graph display.
OC	Oxygen out-of-limits counter.
OX	Oxygen level.
P	Blood pressure.
PA	Effect of oxygen on blood pressure.
PB	Used in the blood pressure display.
PC	Blood pressure out-of-limits counter.
Q1, Q2, Q3	Utility, used in address calculations.
R1	Air Quality option level.
R2	Air Quality value.
RS	Respiration rate.
SC	Player's score.
T	Body temperature.
T1	Used in temperature calculation.
TC	Temperature out-of-limits counter.
TP	Target blood pressure.
TX	Target oxygen level.
W	Determines which sprite shape to display for the heart.
X	X screen coordinate.
Y	Y screen coordinate.
Z	Utility, loop counter.

Vital Signs

DESIGN FOCUS



LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200	Title screen.
210-230	Reload program, graphic above hi-res screen.
240-280	Initialize program, graphics, and sound routines.
290-450	Draw the playing screen.
460-520	Main control loop.
530-590	Check for vitals beyond limits.
600-720	Update blood pressure, oxygen level, and body temperature.
730-810	Select a random Activity and Air Quality.
820-940	Routine for lung cancer and blood clots.
950-1060	Scan the keyboard and branch to the appropriate routines.
1070-1140	Change Activity routine.
1150-1210	Change Air Quality routine.
1220-1300	End-of-game routine, option to play again.
1310-1380	Display current conditions; Activity, Air Quality, lung cancer if present, and blood clot if present.
1390-1450	Misc. screen handling routines.
1460-1520	Display the bar graphs.
1530-1590	End-of-game messages.
1600-1650	Program data.

REMARKS

In accord with the old adage, "a picture is worth a thousand words," bar graphs can greatly enhance certain programs. Bar graphs on the Apple II family of computers are simple to make, using the Hi-resolution graphics page and the HPLOT command. Three useful subroutines, starting at line 1460 of the *Vital Signs* program, plot the three bar graphs which display blood pressure, blood oxygen level, and body temperature.

The key to these routines is that the program always plots from the old bar position to the new position—only the color changes as the bar moves up and down. The program compares the bar's last position with it's new position. It then turns the color either on or off, depending on whether the bar is to increase or decrease. A FOR-NEXT loop then counts from the last position to the current position, drawing a line to fill the bar in as it goes.

The flow chart in the Design Focus on this page illustrates the logical flow of this routine.

HCM Glossary terms: Hi-resolution graphics page.

DIRECTORY OF VARIABLES

Variables	Functions
AC$( )	Activity display text.
AR$( )	Air Quality display text.
A$	Used for input routines.
AC( )	Activity values used in calculations.
AR( )	Air Quality values used in calculations.
OLD( )	Used in the plotting of the bar graphs.
A0	Current Activity option.
A1	Current Activity level.
A2	Current Activity value used in calculations.
BC	Blood clot flag.
CH	Amount and direction of change in one of the bar graphs.
CNT	Counter to keep track of sweating.
CO	Amount of change in the oxygen level.
D	Flag that the game has terminated.
HR	Heart rate.
LC	Lung cancer flag.
LX	Counter to time lung cancer.
LZ	Air Quality variable.
N	Loop counter.
OC	Oxygen out-of-limits counter.
OX	Level of oxygen in the blood.
P	Current blood pressure.
PA	Change in blood pressure due to oxygen level.
PC	Blood pressure out-of-limits counter.
R1	Current Air Quality.
R2	Current Air Quality value used in calculations.
RS	Current respiration.
SC	Player's score.
SOUND	Constant for 786. Address of sound routine.
T	Current temperature.
TC	Temperature out-of-limits counter.
TP	Target blood pressure.
TTA	Target body temperature.
TX	Target oxygen level.
VL	Value used to position the bar graph in the bar graph routine.
X	Utility.
Y	Utility, used in the screen display routines.
Z	Loop counter, utility.

Vital Signs

REMARKS

In designing the IBM version of *Vital Signs*, we encountered a challenge in simulating the effects of heavy exercise such as running. In the real world, if the body temperature exceeds a certain level, or if the external ambient temperature becomes high enough, we start to sweat. The evaporation of the sweat causes our skin to cool, which in turn cools the blood traveling through the capillaries near the skin.

In order to simulate the body's ability to cool itself, we have given the program the ability to "sweat." If the Activity level is set to running or swimming, the body begins to perspire. The cooling-off of the blood is not instantaneous, however. At first, the body temperature starts to rise as a large amount of energy is expended and converted to heat. After 40 cycles (or 40 beats of the heart), the sweat glands start to work, and the body temperature begins to drop. If this same Activity is continued for 100 or more cycles, the temperature starts to rise again as the body dehydrates.

In the IBM version, the routine that controls this perspiration cycle starts at line 460. A flow chart of this routine appears in the Design Focus on this page. The routine uses the variable **SWEAT.COUNT** to keep track of the number of cycles the program has remained at the same high Activity level. If the Activity is running or swimming, the counter increments by one; otherwise, it resets back to zero.

Line 470 calculates the base body temperature. Two factors determine this value: heart rate and blood oxygen level. After it calculates the base temperature, the program tests **SWEAT.COUNT** to see whether or not the temperature is increasing—either because the sweat glands haven't yet started to work, or the body is dehydrating. After the counter passes 100, its effect on temperature is only one-tenth of what it is before the counter reaches 40.

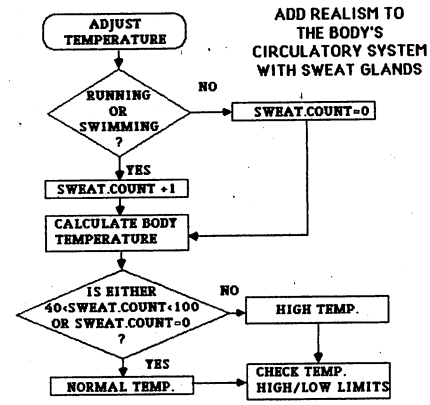
After the program performs the final calculation to adjust the body temperature, it checks the high and low temperature limits.

HCM Glossary terms: reset.

LISTING ANNOTATIONS

Line Nos.	
100-240	Program header.
250-350	Initialize program.
360-440	Get a character from the keyboard and make the heart beat.
450-840	Update vital statistics.
850-930	Change Air Quality option.
940-970	Display respiration and heart rates.
980-990	Display the main menu.
1000-1110	Main game input routine. Get key and branch to appropriate routines.
1120-1220	Change Activity level option.
1230-1280	Display current conditions on the screen.
1290-1420	Initialize bar graph screen area.
1430-1550	Display the bar graph levels.
1560-1710	Set up system variables.
1720-1830	Draw the heart graphics to be placed in an array.
1840-1890	Print end-of-game message from problems.
1900-1970	End the game and find out if player wants to play again.

DESIGN FOCUS



DIRECTORY OF VARIABLES

Variables	Functions
ACTIVITY$( )	Text for Activity levels.
AIR.OPTION$( )	Text for Air Quality levels.
A$, K$, S$	Used for input and display.
ACTIVITY.LEVEL( )	Values used for activity calculations.
AIR.QUALITY ( )	Array for Air Quality calculations.
CON( ), EX( )	Graphics image arrays used with PUT.
LUNG( ), OX( )	Graphics image arrays used with PUT.
PS( ), TP( )	Graphics image arrays used with PUT.
ACTIVITY	Current Activity.
ACTIVITY.OPTION	Current Activity option.
AIR.OPTION	Current Air Quality option.
BLACK, GOOD.AIR	Constant for 0.
SLEEPING	Constant for 0.
BLOOD.FLAG	Indicates there is a blood clot.
BROWN, WALKING	Constant for 3.
SMOKE.SMOG	Constant for 3.
TEMP.PROBLEMS	Constant for 3.
COL	Screen column.
D, I, TD	Utility, loop counter.
DONE	Indicates end of program.
G.ROW	Used in moving the bar graph.
GREEN, SMOGGY	Constant for 1.
PRESSURE.PROBLEMS	Constant for 1.
RESTING	Constant for 1.
HEART.CHANGE	Amount of change in heart rate.
HEART.RATE	Current heart rate.
LAST.ROW	Used in bar graph display.
LUNG.COUNTER	Timer for lung cancer.
LUNG.DAMAGE	Alters chances for lung cancer.
LUNG.FLAG	Indicates lung cancer.
NEW.ROW	Used in moving the bar graph.
NORMAL,RED,	Constant for 2.
SMOKE.CIG	Constant for 2.
OXYGEN.PROBLEMS	Constant for 2.
OC, OX.OB	Oxygen out-of-limits variables.
OX.TMP	Amount of change in oxygen level.
OXYGEN	Current percent of oxygen in blood.
OXYGEN.OLD	Used to display bar graph.
P	Used to determine which key pressed.
P.OB	Blood pressure out-of-limits flag.
PA	Effect of oxygen on blood pressure.
PC	Blood pressure out-of-limits counter.
PRESSURE	Current blood pressure.
PRESSURE.OLD	Used in the bar graph display.
RANDOM	Constant for 6.
RESP	Current respiration.
RESP.CHANGE	Amount of change in respiration.
RUNNING	Constant for 4
SCORE	Player's score.
SWEAT.COUNT	Amount of time sweating.
SWIMMING	Constant for 5.
T.OB, TC	Temperature out-of-limits variables.
TEMP	Body temperature.
TEMP.OLD	Used to display temperature graph.
TEMP.TMP	Target temperature.

Vital Signs

REMARKS

Atari BASIC offers a unique feature not available in most BASIC languages: the "calculated" GOTO or GOSUB. All BASICs place the line number to be branched to at the right of the GOTO or GOSUB; but Atari BASIC adds the ability to use a numeric expression in place of the line number. The computer calculates the actual line number as the program RUNs.

This method can add efficiency and clarity to program code. If line numbers of major routines are given descriptive variable names, these names in the GOTO or GOSUB statement make a program more readable. In addition, calculated GOTOs and GOSUBs allow branching to one of several statements.

A problem arises, however, when you wish to resequence the program (using a utility such as *Monkey Wrench* from Eastern House). It is impossible for these utilities to take variables and expressions into consideration when resequencing programs. But when expressions calculate one of several possible branches, you can solve the resequencing problem with a simple procedure.

First, place the first line number of the possible branches immediately after the GOTO or GOSUB statement so the utility will renumber that line. A multiplier in the expression that follows can then form an offset for the other possible line numbers. Notice that the numeric difference between line numbers of the different branches must always be the same. For example, if you use line number increments of 10, and each routine takes up only one line, then the expression of the calculated GOTO can include a multiplier of 10. The Design Focus on this page shows how this was done in *Vital Signs*. A1, which determines the routine branched to, is in a range from 0 to 6.

HCM Glossary terms: resequence.

LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-440	Initialize program and graphics.
450-700	Display playing screen.
710-860	Main control loop.
870-920	Check for blood pressure out-of-limits.
930-970	Check for oxygen out-of-limits.
980-1020	Check for body temperature limits.
1030-1190	Update values of the vital signs.
1200-1340	Change Activity at random.
1350-1470	Change Air Quality at random.
1480-1520	Get lung cancer.
1530-1550	Get a blood clot.
1560-1620	Cure cancer.
1630-1670	Blood clot cured.
1680-1920	Scan keyboard and branch to routines.
1930-2020	Change Activity routine.
2030-2080	Change Air Quality routine.
2090-2220	End-of-game routine. Play again option.
2230-2470	Display current options: Activity, Air Quality, etc.
2480-2520	Display the heart and respiration rates.
2530-2560	Clear a part of the screen for messages.
2570-2730	Display the bar graphs.
2740-2770	Single key input routine.
2780-3180	Character graphics data.
3190-3240	End-of-game messages.
3250-3320	Initialize arrays.

DESIGN FOCUS

CALCULATED GOTO'S AND GOSUB'S:
AN ALTERNATIVE WAY TO CONTROL PROGRAM FLOW

```
2230 REM **** UPDATE CONDITIONS ****
2240 GOTO 2250+A1*10 <-- CALCULATED GOTO
2250 A$="SLEEPING":GOTO 2320
2260 A$="RESTING":GOTO 2320
2270 A$="NORMAL":GOTO 2320
2280 A$="WALKING":GOTO 2320
2290 A$="RUNNING":GOTO 2320
2300 A$="SWIMMING":GOTO 2320
2310 A$="RANDOM"
2320 POSITION 1,15:? BLANK$:POSITION 3,15:? A$
```

DIRECTORY OF VARIABLES

Variables	Functions
A$	Utility, for message displays.
BLANK$	Used to clear a line.
AC( )	Activity level values for calculation.
AR( )	Air Quality values for calculation.
A	Utility.
A0	Activity option.
A1	Activity level.
A2	Activity level value.
BC	Blood clot flag.
CH	Used in locating top of memory.
CNT	Timer for perspiration to control body temperature.
CO	Change in oxygen level.
CSET	Location of character set.
D	End-of-game flag.
HR	Heart rate.
I	Loop counter.
K	ATASCII value of key pressed.
LC	Lung cancer flag.
LCC	Lung cancer time counter.
LZ	Chance of getting lung cancer.
OB	Used to display oxygen bar graph.
OC	Oxygen out-of-limits counter.
OX	Oxygen level.
P	Blood pressure.
PA	Change in blood pressure due to oxygen level.
PB	Used to display the blood pressure bar graph.
PC	Blood pressure out-of-limits counter.
R1	Air Quality level.
R2	Air Quality value used for calculations.
RS	Respiration rate.
SC	Player's score.
T	Body temperature.
T1	Used to display the body temperature bar graph.
TC	Temperature out-of-limits counter.
TOX	Target oxygen level.
TP	Target blood pressure.
TTA	Target body temperature.
Z	Utility.

NanoProcessor

REMARKS

The fact that math in Commodore 64 BASIC is entirely decimal based (as it is in most BASICs), creates difficulties in programs such as *NanoProcessor* where all math is binary. This problem becomes especially apparent when we create strictly binary routines such as the Rotate Left through Carry (RLC) instruction.

Fortunately, rotating a binary number left multiplies that number by 2. First, we set TEMP to zero, and then test the carry flag. If it is set, we make TEMP = 1. We next check if multiplication by 2 will result in overflow. If not, we clear the carry flag and simply multiply the A register by 2 and add TEMP. Otherwise, we set the carry flag and perform the same multiplication, but subtract 16.

The Design Focus on this page details the specifics of the algorithm for the RLC routine.

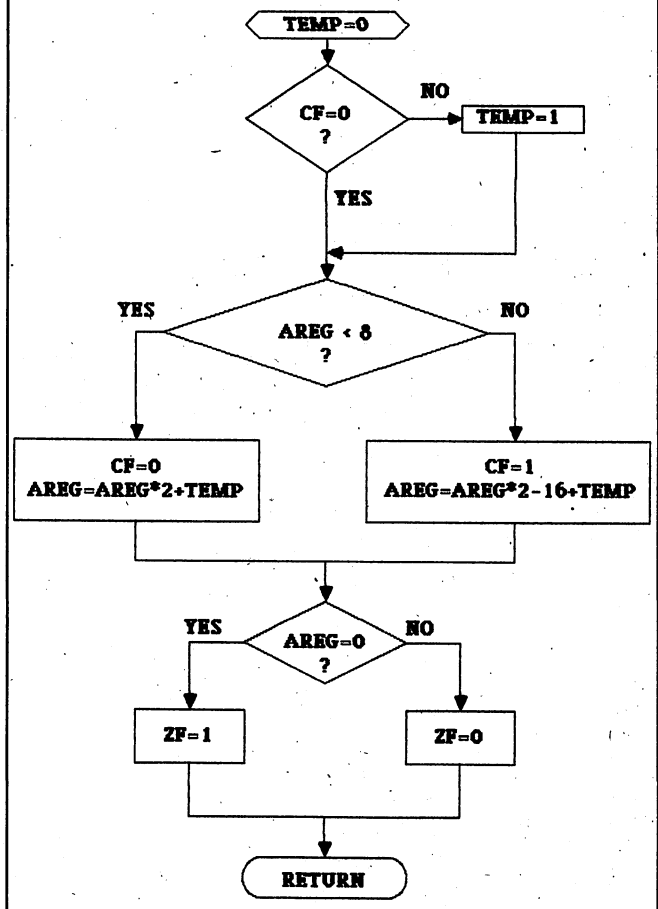
HCM Glossary terms: accumulator, binary, carry flag, zero flag.

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-250	Initialize variables, branch to main screen.
260-320	Initial key input routine.
330-390	Main Loop.
400-490	Parse keyboard input.
500-540	Place values in TEMP variable.
550-630	Key input routine.
640-710	Save file routine.
720-790	Load file routine.
800-830	Tape or Disk option.
840-860	Repaint screen routine.
870-880	Convert to binary.
890-930	Write to screen.
940-1000	Convert address.
1010-1030	Lights off (power-down).
1040-1060	Start (go to address zero).
1070-1190	Increment address.
1200-1220	Run program.
1230-1360	Load from switches.
1370-1380	Rotate switch counter-clockwise.
1390-1400	Rotate switch clockwise.
1410-1460	Common rotate routines.
1470-1520	Rotate subroutines.
1530-1610	Draw rotary switch subroutines.
1620-1670	ADD routine.
1680-1690	LoaD Accumulator immediate routine.
1700-1720	LoaD Accumulator from memory routine.
1730-1800	STore Accumulator in memory routine.
1810-1820	Transfer A to B routine.
1830-1840	Transfer B to A routine.
1850-1910	Rotate A Right through Carry routine.
1920-1980	Rotate A Left through Carry routine.
1990-2030	AND A with B routine.
2040-2100	OR A with B routine.
2110-2170	XOR A with B routine.
2180-2210	Branch on Zero routine.
2220-2250	Branch on Not Zero routine.
2260-2290	Branch on Carry Set routine.
2300-2330	Branch on Carry Clear routine.
2340-2420	Unconditional JuMP routine.
2430-2440	Delay routine.
2450-2500	Randomize registers.
2510-2860	Program initialization.
2870-2920	DATA for array and music.
2930-2940	End routine.
2950-2980	Error handling routine.
2990-3000	Get key press.
3010-3070	Music subroutine.

DESIGN FOCUS

ROTATING LEFT THROUGH CARRY IN BASIC



DIRECTORY OF VARIABLES

Variables	Functions
D$( )	Convert decimal to binary string array.
AD( )	Memory array.
NT%( )	Tone array.
AD$	Binary string of address.
AR$, BR$	Binary strings of A and B registers.
FL$	File name variable.
PL$	Temporary variable for display.
AR, BR	Decimal value of A and B registers.
CA	Current address.
CF	Carry flag.
CH	Character to be displayed.
CT	Count variable.
D	Delay counter.
DV	Device number for I/O.
DV$	Device name (T/D).
E	Error number.
E$	Error name.
I,IT,JT	Loop counters.
JN$	Two cursors down.
KE, KEY, KEY$	Key-press variables.
M	Variable containing toggle switch positon.
RF	Run flag.
RO$	Clear screen character.
SA	Save or Load flag.
SW	Current rotary switch position.
TEMP	Temporary variable.
ZF	Zero flag.
Z	Sound chip address.

NanoProcessor

REMARKS

Updating the address lights in the TI version of *NanoProcessor*, involves displaying a decimal number in binary. We first convert the current address (CA) into a string (AD$) of ones and zeros which correspond to the "on" and "off" bits of the binary equivalent. This conversion is simplified by storing each of the 16 possible binary arrangements of a nibble in a 16 element array: D$( ).

The FOR-NEXT loop in lines 1390-1460 contains the logic that displays the lights. The starting and ending limits of the loop counter IT (6 and 27) delineate the left- and right-most columns of the light display. IT is used as the column in the HCHAR statement of line 1450. To determine which of the two possible characters (on or off) is displayed, the loop counter is first divided by 3. Then in line 1410 we test whether or not that bit is a 1. If it is, we set CH2 to the character of the "on" light, otherwise we set it to the "off" character. The Design Focus on this page shows the details of this logic.

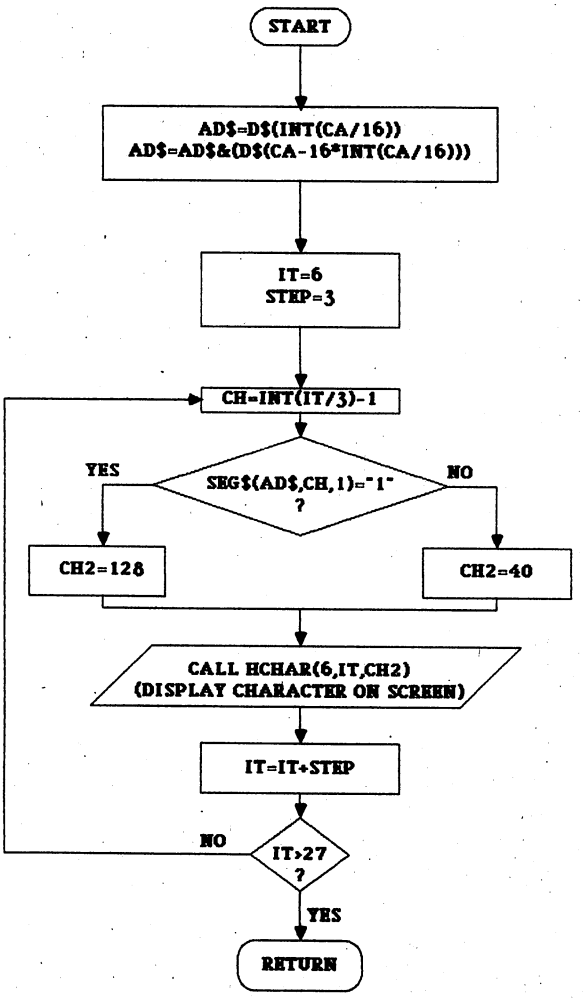
HCM Glossary terms: array, binary, element, loop counter, nibble.

LISTING ANNOTATIONS

Line Nos.	
100-180	Program header.
190-260	Program initialization.
270-420	Check for Power, End, or switches.
430-500	Main input loop.
510-700	Program-running routine.
710-800	Switch-setting routines.
810-890	Keyboard input.
900-1010	Save file routine.
1020-1100	Load file routine.
1110-1250	Return to main screen.
1260-1470	Convert to binary and display.
1480-1560	Turn lights off and Power down.
1570-1630	Go to first address.
1640-1810	Increment address.
1820-1870	Run program.
1880-2000	Load data from switches.
2010-2120	Convert address subroutines.
2130-2180	Rotate switch routine.
2190-2250	Get panel switch settings.
2260-2290	Get rotary switch setting.
2300-2610	Rotate switch.
2620-2710	Output light and sound subroutines.
2720-2770	ADD routine.
2790-2820	LoaD Accumulator immediate routine.
2830-2900	LoaD Accumulator from memory.
2910-3030	STore Accumulator in memory routine.
3040-3050	Transfer A to B routine.
3060-3070	Transfer B to A routine.
3080-3160	Rotate A Right through Carry routine.
3170-3260	Rotate A Left through Carry routine.
3270-3320	AND A with B routine.
3330-3380	OR A with B routine.
3390-3440	XOR A with B routine.
3450-3490	Branch on Zero routine.
3500-3540	Branch on Not Zero routine.
3550-3590	Branch on Carry Set routine.
3600-3640	Branch on Carry Clear routine.
3650-3720	Unconditional JuMP routine.
3730-3770	Set up for logic instruction.
3780-3840	Check zero flag and return.
3850-3930	Delay loop and randomize registers.
3940-4060	Initialize arrays.
4070-4180	Display screen.
4190-4260	Program DATA.
4270-4280	End program.

DESIGN FOCUS

DISPLAYING DECIMAL ADDRESS AS BINARY QUANTITY

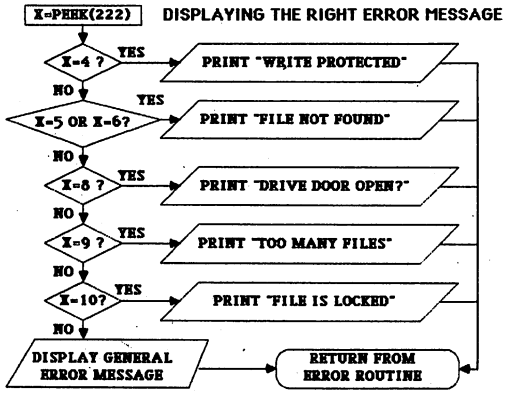


DIRECTORY OF VARIABLES

Variables	Functions
D$( )	Convert decimal to binary string array.
AD( )	Memory array.
TN( )	Tone array.
AD$	Binary string of address.
AR$, BR$	Binary strings of A and B registers.
CH$	Temporary string variable.
FL$	File name variable.
PL$	Temporary variable for display.
AR, BR	Decimal value of A and B registers.
C	Key-press choice.
CA	Current address.
CF	Carry flag.
CH	Character to be displayed.
CH2	Alternate character to be displayed.
D	Delay counter.
IT	Loop counter.
K	ASCII of key input.
M	Variable containing toggle switch positon.
RF	Run flag.
S	Key-press status.
SW	Current rotary switch position.
TEMP	Temporary variable.
ZF	Zero flag.

NanoProcessor

DESIGN FOCUS



LISTING ANNOTATIONS

Line Nos.

100-220	Program header.
230-280	Initialize hi-res and error handling.
290-330	Main program loop.
340-720	Program initialization.
730-870	Program DATA.
880-1150	Draw screen routine.
1160-1200	Draw boxes and output lamp.
1210-1300	Draw memory lamp and rotary switch.
1310-1320	Draw Busy lamp.
1330-1340	Draw power lamp and switch.
1350-1360	Draw data lamps.
1370-1380	Draw switches.
1390-1480	Indicate button pushes.
1490-1520	End program routine.
1530-1590	Power routine.
1600-1630	Start (go to address zero).
1640-1710	Increment address.
1720-1780	Run program.
1790-1800	ADD routine.
1810-1820	LoaD Accumulator immediate routine.
1830-1840	LoaD Accumulator from memory routine.
1850-1860	STore Accumulator in memory routine.
1870-1890	Output light and sound chip routines.
1900-1910	Transfer A to B routine.
1920-1930	Transfer B to A routine.
1940-1950	Rotate A Right through Carry routine.
1960-1970	Rotate A Left through Carry routine.
1980-1990	AND A with B routine.
2000-2010	OR A with B routine.
2020-2030	XOR A with B routine.
2040-2060	Branch on Zero routine.
2070-2090	Branch on Not Zero routine.
2100-2120	Branch on Carry Set routine.
2130-2150	Branch on Carry Clear routine.
2160-2170	Unconditional JuMP routine.
2180-2190	Check zero flag and return.
2200-2210	Logical instruction set up.
2220-2260	Halt running program.
2270-2450	Load from switches.
2460-2480	Rotate counter-clockwise.
2490-2510	Rotate switch clockwise.
2520-2540	Update switches.
2550-2630	Load file.
2640-2740	Save file.
2790-2800	Delay loop.
2810-2840	Convert nibble to binary.
2850-2860	Convert binary array to nibble.
2870-2880	Randomize registers.
2890-2960	Display Data.
2970-2990	Convert address to binary and display.
3000-3010	Convert binary address to decimal.
3020-3280	Input file name routine.
3290-3340	Keybard scan.
3350-3460	Error handling.

REMARKS

The Apple version of *NanoProcessor* uses a combination of logic and simple math to display disk error messages. It is important that error messages are clearly stated so that the user will know just what action to take if a disk error occurs.

In line 3360 we first PEEK the error number from location 222. We then conduct a series of tests and multiplications in an ON-GOTO statement to control program flow so that the proper error message is displayed.

We check the most likely disk error numbers against the one that has occurred. When Applesoft BASIC evaluates an equality relationship, the result is expressed as either a one for true, or a zero for false. Because no more than one of the tests can prove true, when we multiply result by an assigned integer (from 1 to 5), program control GOes TO one of 5 locations to display the proper message. If none of the tests proves true, the control falls through to line 3370 where a general error message is displayed. The Design Focus on this page details the logic in a flow chart.

DIRECTORY OF VARIABLES

Variables	Functions
D$( )	Convert decimal to binary string array.
ET$( )	String array for input routine.
FG( )	Status of address lamp array.
AD( )	Memory array.
TN%( )	Tone array.
DT( )	Data lamp array.
AL( )	Switch position array.
A( ), B( )	Arrays for bits in A and B registers.
NB( )	Temporary array for memory contents.
RG( )	Register array.
RS( )	Rotary switch condition array.
SW( )	Current rotary switch position array.
A$	String containg letter A—CHR$(65).
BL$	Bell tone—CHR$(7).
CM$	Comma—CHR$(44).
CR$	Carriage return—CHR$(13).
DR$	Drive number.
ESC$	Escape string.
FL$	File name variable.
IN$	One character of input.
LF$	Left cursor—CHR$(8).
LG$	Legal input string.
PL$	Temporary variable for display.
X$	Utility string.
Z$	String containg letter Z—CHR$(90).
AD	Address pointer.
AR, BR	Decimal value of A and B registers.
B1, B2, B3, B4	Temporary variables for drawing boxes.
CHAR	Address of character define routine.
BI, D, DI, HZ	Loop counters.
IT, I, J, JI ,S	Loop counters.
IN, NB, P, XT	Utility variable.
ER	Error number.
HCHAR	Address of character display routine.
K	Utility variable.
M	Variable containing toggle switch positon.
OFF	Address to turn of hi-res PRINT routine.
OT	Flag for condition of the output lamp.
PRNT	Address of PRiNT to hi-res routine.
RF	Run flag.
RS	Temporary variable for rotary switch.
SOUND	Address of sound routine.
TEMP	Temporary variable.
TP	Tone address in music routine.
X	Temporary variable when doing logic instruction.

REMARKS

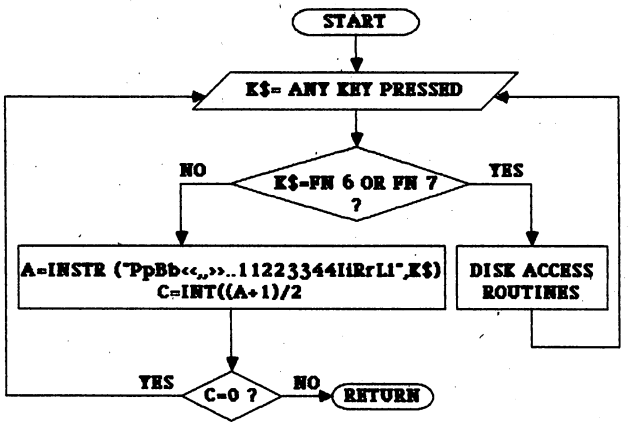
On the standard IBM PC and PCjr keyboards, there is no indication whether the keyboard is in CapsLock mode or not. This means that your input routines must be set up for both uppercase and lowercase characters on any given keypress. The *NanoProcessor* uses a unique "pairing" algorithm to take care of this.

Program lines 610-620 first check whether any key has been pressed, and if so, whether one of two function keys (Fn 6 or Fn 7) has been pressed. If any other key has been pressed, the INSTR function makes A equal to the position of the character in the string "PpBb<<,,>> . . 11223344IiRrLl" (the keys used to access the various buttons and switches on the front panel). Notice that each character is either paired with itself, or with its uppercase or lowercase rendition. We set C equal to the integer of (A+1)/2 at the end of line 630. This number is then used to control program flow in the main loop in lines 430-460.

HCM Glossary terms: algorithm, string.

DESIGN FOCUS

ACCEPTING BOTH UPPERCASE AND LOWERCASE INPUT



LISTING ANNOTATIONS

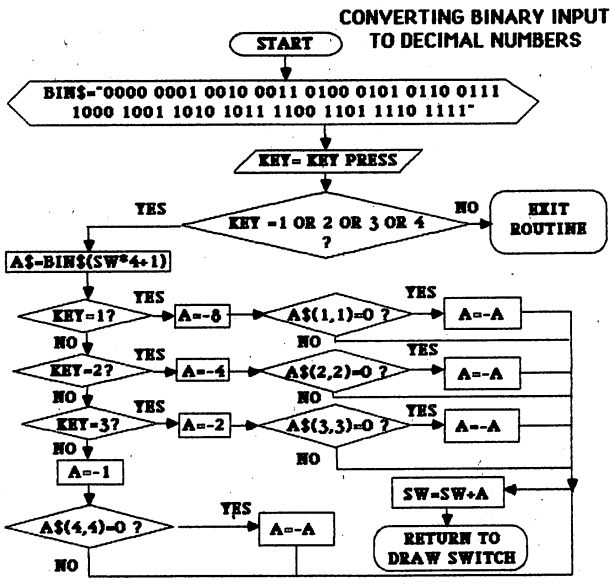
Line Nos.	
100-230	Program header.
240-300	Program initialization.
310-400	Initial key-press routine.
410-470	Main loop.
480-540	Run program routine.
550-590	Set TEMP to proper value.
600-650	Key input routine.
660-700	Save file routine.
710-750	Load file routine.
760-840	Repaint screen.
850-860	Convert to binary.
870-910	Write to screen.
920-1000	Convert address.
1010-1050	Begin (return to address zero).
1060-1180	Increment address.
1190-1210	Set Run program flag.
1220-1380	Load address contents from switches.
1390-1420	Rotate switch counter-clockwise.
1430-1440	Rotate switch clockwise.
1450-1620	Common rotate switch routines.
1630-1690	Draw rotary switch.
1700-1720	ADD routine.
1730-1750	LoaD Accumulator immediate routine.
1760-1800	LoaD Accumulator from memory routine.
1810-1840	STore Accumulator in memory routine.
1850-1880	Output light and sound chip routines.
1890-1900	Transfer A to B routine.
1910-1920	Transfer B to A routine.
1930-1980	Rotate A Right through Carry routine.
1990-2040	Rotate A Left through Carry routine.
2050-2090	AND A with B routine.
2100-2150	OR A with B routine.
2160-2220	XOR A with B routine.
2230-2260	Branch on Zero routine.
2270-2310	Branch on Not Zero routine.
2320-2350	Branch on Carry Set routine.
2360-2380	Branch on Carry Clear routine.
2390-2420	Unconditional JuMP routine.
2430-2470	Routines for rotate routines.
2480-2510	Randomize registers.
2520-3080	Print screen.
3090-3150	Program DATA.
3160-3280	File name input routine.
3290	Initialize error routine.
3300-3340	Disk error routine.
3350-3360	Error message DATA.

DIRECTORY OF VARIABLES

Variables	Functions
D$( )	Convert decimal to binary.
AD( )	Memory array.
NT$( )	Note array.
AD$	Binary string of address.
AR$, BR$	Binary strings of A and B registers.
CH$	Temporary string variable.
FL$	File name variable.
G$	Temporary variable for DRAW strings.
IN$	Input for file manager.
J1$, J2$	Temporary variables for logic.
K$	Key input variable.
PL$	Temporary variable for display.
AR, BR	Decimal value of A and B registers.
A	Temporary variable.
BDOT( )	Graphic array.
C	Key-press choice.
CA	Current address.
CF	Carry flag.
CH	Character to be displayed.
COL	Column where input is accepted.
CT	Counter variable.
DE, TD	Delay counters.
DN( )	Graphic array variable.
ERCD,ERM$	Error variables.
IT, I, Z	Loop counters.
L	Line number of error.
LDOT( )	Graphic array variable.
M	Variable for toggle switch position.
MAXLEN	Maximum length of input.
MD1	Mode flag (save or load).
N	Counter variable.
OFFLIT( )	Graphic array variable.
ONLIT( )	Graphic array variable.
P1, P2, P3	Graphic array variables.
P4, P5	Graphic array variables.
PT	Pointer to current character for input.
Q$	Temporary string variable.
R	Temporary variable.
RF	Run flag.
ROW	Row where input will begin.
S$	Secondary key input variable.
SC( )	Array of toggle switch status.
SELECT$	String of acceptable input characters.
SW	Current rotary switch position.
TEMP	Temporary variable.
UP( )	Graphic array variable.
ZF	Zero flag.

NanoProcessor

DESIGN FOCUS



LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-320	Program initialization.
330-390	Main loop.
400-500	Loop to run program.
510-630	Print screen routine.
640-760	Initalize variables.
770-890	Initialize arrays.
900-940	Array DATA.
950-1040	Plot point in Player Missile area.
1050-1180	Convert and print address.
1190-1350	Convert contents to binary and display.
1360-1440	Convert switches to binary and display.
1450-1580	Key input routine with branch to routine.
1590-1700	Work rotary switch.
1710-1850	Work toggle switches.
1860-2030	Power on.
2040-2120	Start (make address zero).
2130-2210	Clear or set run flag.
2220-2300	Load from switches.
2310-2340	ADD routine.
2350-2390	LoaD Accumulator immediate routine.
2400-2460	LoaD Accumulator from memory routine.
2470-2520	STore Accumulator in memory routine.
2530-2610	Output light and sound chip routines.
2620-2640	Transfer A to B routine.
2650-2670	Transfer B to A routine.
2680-2720	Rotate A Right through Carry routine.
2730-2770	Rotate A Left through Carry routine.
2780-2870	AND A with B routine.
2860-2920	OR A with B routine.
2930-2990	XOR A with B routine.
3000-3020	Branch on Zero routine.
3030-3050	Branch on Not Zero routine.
3060-3080	Branch on Carry Set routine.
3090-3110	Branch on Carry Clear routine.
3120-3180	Unconditional JuMP routine.
3190-3220	Set up AR$ and BR$.
3230-3270	End program.
3280-3400	Save file.
3410-3530	Load file.
3540-3600	Error handling.
3610-3640	Open NanoProcessor.
3650-3730	Close NanoProcessor.
3740-3810	Save or load screen.
3820-3850	Increment address and check for wrap.

REMARKS

One of the major tricks involved in writing the *NanoProcessor*, is converting the binary input of the toggle switches into decimal numbers for Atari BASIC to work with.

During inititialization, we set the BIN$ string equal to the 16 combinations of binary numbers from 0000 to 1111. Then, if a key designating a switch (1-4) is pressed, the KEY variable is set to that number, and the program branches to line 1710. Here, we set A$ to the current binary equivalent of the switches by using SW (which contains the current decimal value that the switches represent) as an offset into BIN$. The KEY variable then controls the branch to one of 4 lines (1740, 1760, 1780, or 1800)—each of which causes a different switch to toggle.

For example, if the 1 key is pressed, the left-most bit needs to change. In preparation, A is set to −8. If the left-most bit of A$ is a one, then the switch needs to toggle down (into the zero position). In this case A is left at −8 and the program branches to line 1810 where A is added to SW. If, however, the left-most character of A$ is presently a zero, the switch needs to be toggled up. Therefore, A is set equal to +8 to increment SW by 8.

HCM Glossary terms: binary, offset, string.

DIRECTORY OF VARIABLES

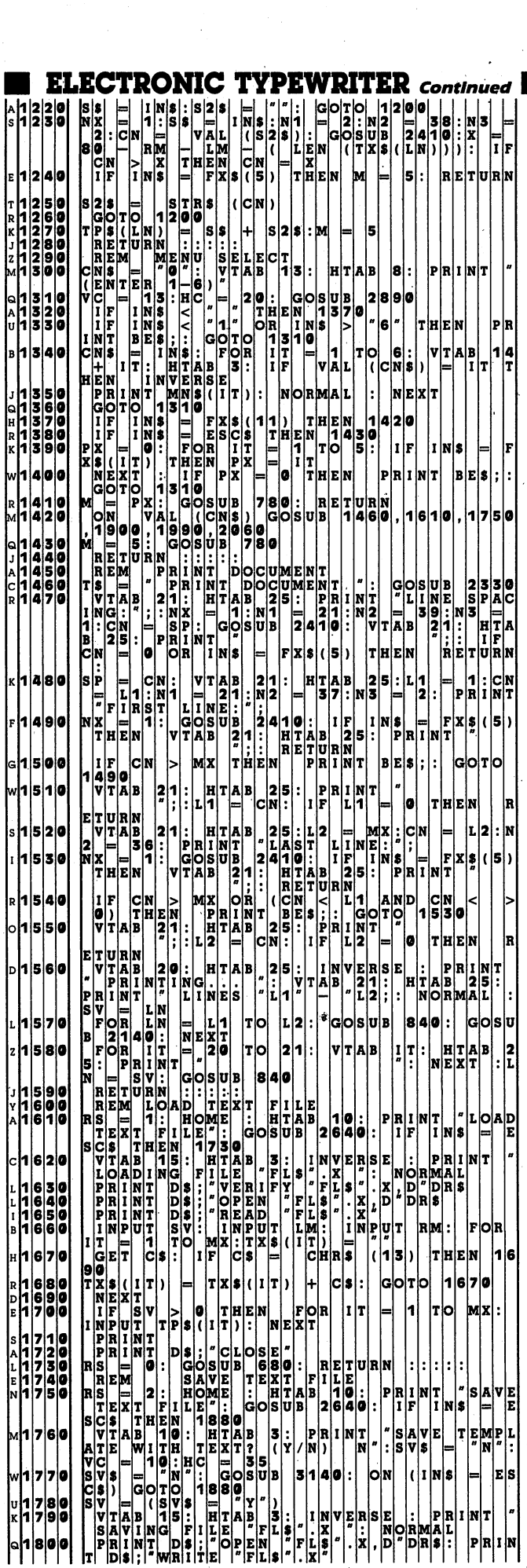
Variables	Functions
ADDR( )	Memory array.
NT( )	Tone array.
BIN( )	Binary array.
RSX( ), RSY( )	Horizontal and vertical location for print to screen routine.
A$	Utility string variable.
AREG, BREG$	Binary strings of A and B registers.
BIN$	Binary string for display.
F$	File name variable.
LON$	string containing Light on condition.
PC$	Binary string of address.
RSC$	String of Print to screen temporary variables.
SW$	String for rotary switches.
A, B, C	Utility variables.
AREG, BREG	Decimal value of A and B registers.
CFLAG	Carry flag.
DELAY	Delay counter.
INC	Loop counter.
KEY	Key input variable.
LOC	Location of Player Missile POKE start.
LON	Light on flag.
MD	Save or Load flag.
PINC	Loop counter variable for POKEing Player Missile information.
PMBAS	Player Missile Base address.
POWER	Power on/off flag.
PPC$	Address location string.
MEM$	Memory location string.
PMEM$	Memory location temporary string.
PV	POKE value variable.
RF	Run flag.
S	Key-press status.
RSW	Current rotary switch position.
SW	Decimal value of toggle switches.
TEMP	Temporary variable.
XPOS	X position of cursor.
YPOS	Y position of cursor.
ZFLAG	Zero flag.

ELECTRONIC TYPEWRITER


```
100 REM ELECTRONIC TYPEWRITER
110 REM BY DAVID HOWARD
120 REM COMPUTE! MAGAZINE
130 REM P.O. BOX 5406
140 REM GREENSBORO, NC 27403
150 REM
160 REM COPYRIGHT (C) 1985
170 REM COMPUTE! PUBLICATIONS, INC.
180 REM ALL RIGHTS RESERVED
190 REM
200 REM PUBLISHED IN COMPUTE! MAGAZINE
210 REM
220 REM
230 REM
240 REM
250 REM
260 REM
270 GOSUB 1140: GOSUB 1190: GOSUB 1300: GOSUB 2100: GOSUB 910
280 REM
290 REM
300 REM
310 REM INITIALIZATION
320 REM
330 REM
340 REM
350 REM
360 REM
370 REM
380 REM
390 REM
400 REM
410 REM
420 REM
430 REM
440 REM
450 REM
460 REM
470 REM
480 REM
490 REM
500 REM
510 REM
520 REM
530 REM
540 REM
550 REM
560 REM
570 REM
580 REM
590 REM
600 REM
610 REM
620 REM
630 REM
640 REM
650 REM
660 REM
670 REM
680 REM
690 REM
700 REM
710 REM
720 REM
730 REM
740 REM
750 REM
```

Continued

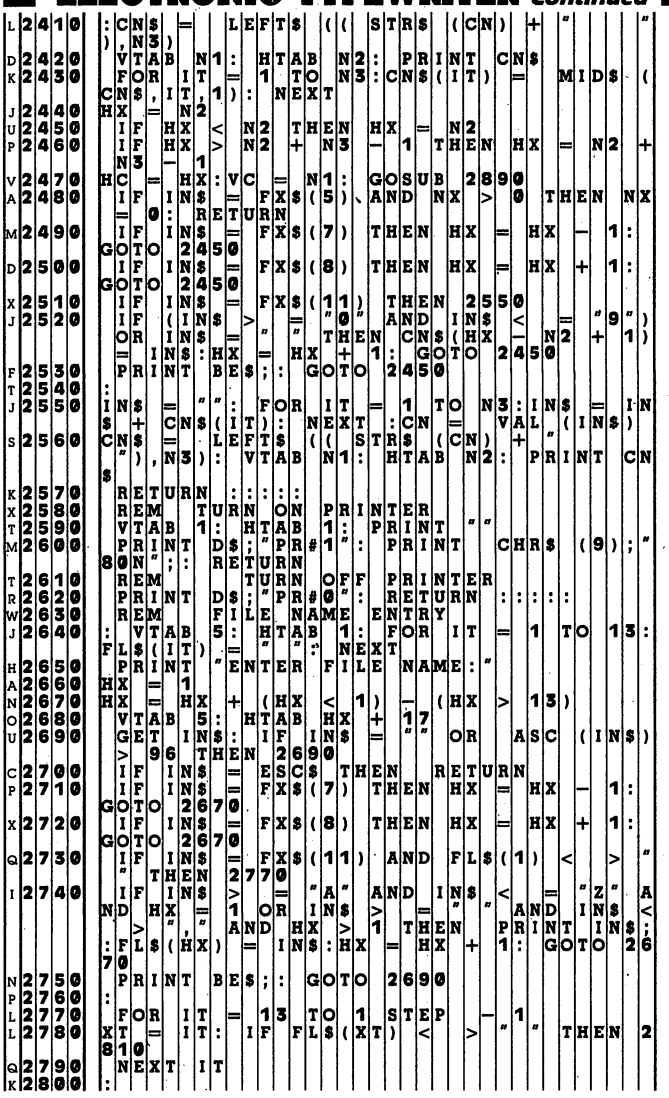
ELECTRONIC TYPEWRITER *Continued*

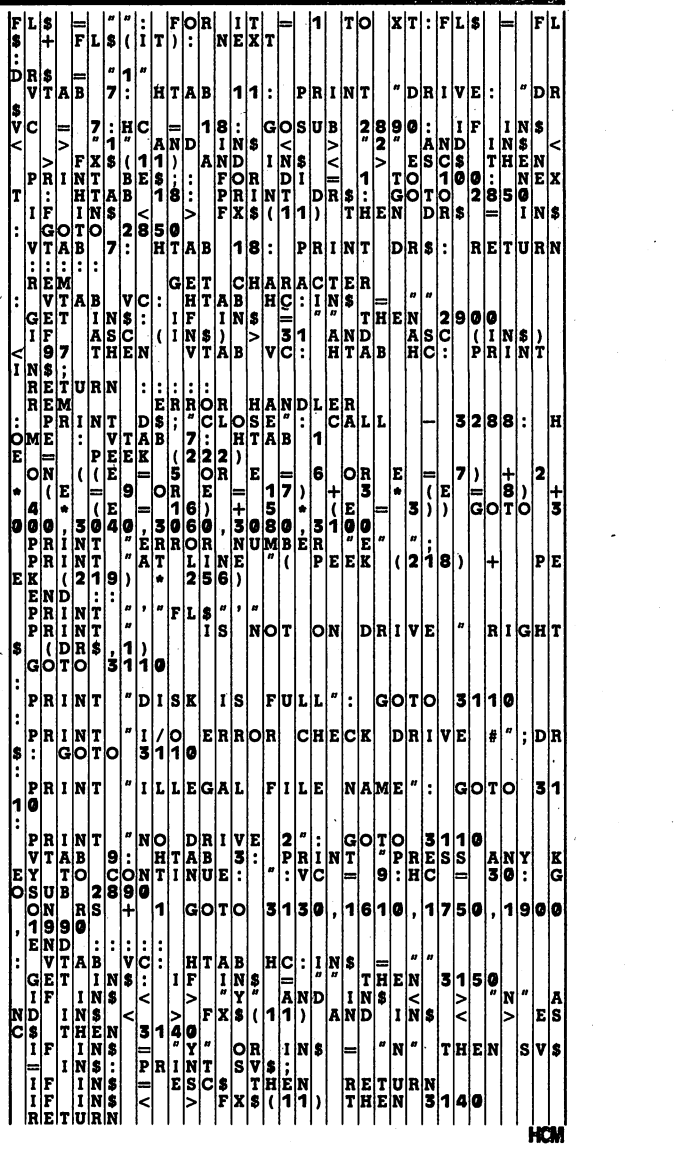


Continued

ELECTRONIC TYPEWRITER *Continued*

APPLE // Family

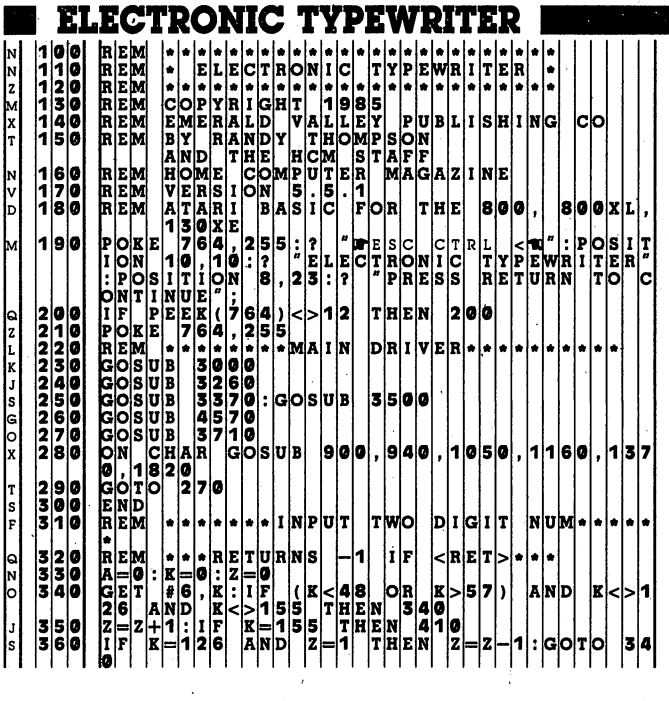


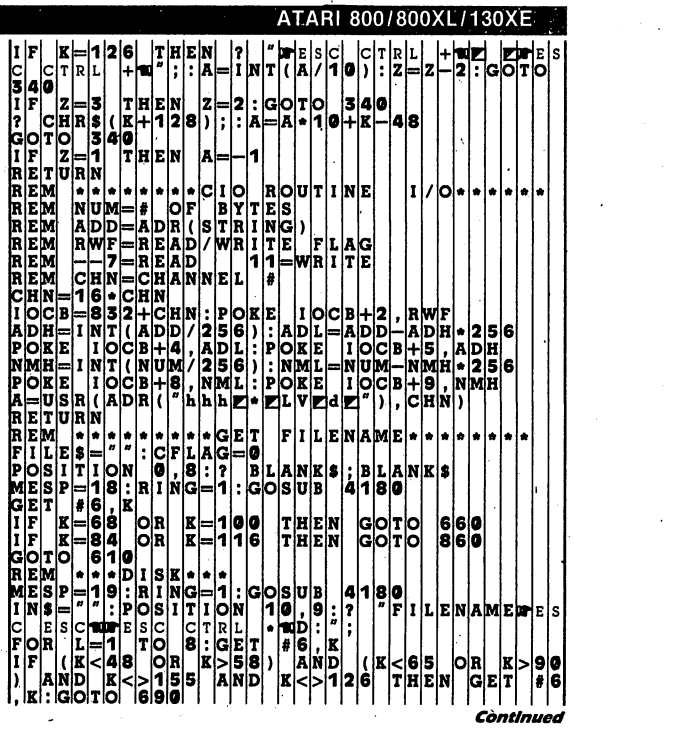


HCM

ELECTRONIC TYPEWRITER

ATARI 800/800XL/130XE

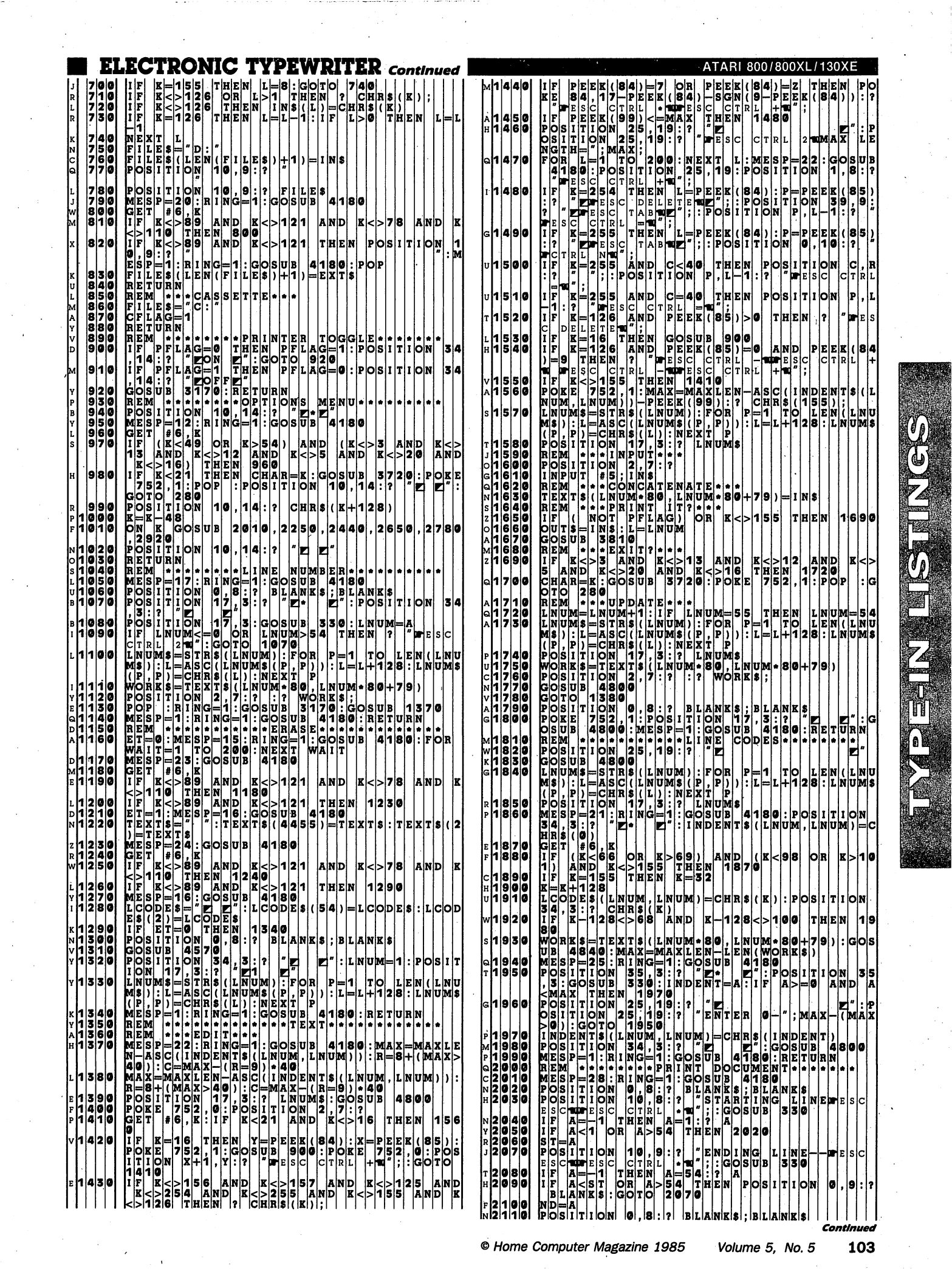




Continued

ELECTRONIC TYPEWRITER *Continued*

ATARI 800/800XL/130XE



TYPE-IN LISTINGS

Continued

ELECTRONIC TYPEWRITER *Continued*


```
A 2120 POSITION 10,8:? "LINE SPACING{ESC ESC}{ESC CTRL +}";:GOSUB 330
S 2130 IF A=-1 THEN A=2:? A
A 2140 IF A<1 OR A>2 THEN 2120
Y 2150 LSPACE=A
L 2160 FOR L=ST TO ND
H 2170 POSITION 0,8:? BLANK$;BLANK$
Q 2180 POSITION 17,3:? LNUM$
T 2190 OUT$=TEXT$(L*80)
K 2200 GOSUB 3810
U 2210 NEXT L
Z 2220 POSITION 0,8:? BLANK$;BLANK$
Z 2230 MESP=1:RING=1:GOSUB 4180:RETURN
P 2240 REM *******LOAD TEXT FILE******
F 2250 MESP=29:RING=1:GOSUB 4180:FOR WAIT=1 TO 70:NEXT WAIT
V 2260 POSITION 0,8:? BLANK$;BLANK$
C 2270 EXT$=".TXT":GOSUB 580
E 2280 IF CFLAG THEN POSITION 9,8:? "CUE TAPE AND PRESS <PLAY>":POSITION 9,9:? "PRESS <RETURN> TO BEGIN"
W 2290 TRAP 2370:CLOSE #2:OPEN #2,4,0,FILE$
C 2300 MESP=2:GOSUB 4180
X 2310 GET #2,TFLAG
S 2320 NUM=4455:ADD=ADR(TEXT$):RWF=7:CHN=2:GOSUB 440
Q 2330 IF TFLAG THEN NUM=54:ADD=ADR(LCODE$):RWF=7:CHN=2:GOSUB 440
I 2340 IF TFLAG THEN NUM=54:ADD=ADR(INDENT$):RWF=7:CHN=2:GOSUB 440
H 2350 GET #2,RMAR:GET #2,LMAR:CLOSE #2
X 2360 GOTO 2380
K 2370 MESP=8:GOSUB 4180:FOR WAIT=1 TO 150:NEXT WAIT
I 2380 POSITION 0,8:? BLANK$;BLANK$
M 2390 POSITION 34,3:? "◪  ◪":LNUM=1:POSITION 17,3:? "◪1  ◪"
T 2400 LNUM$=STR$(LNUM):FOR P=1 TO LEN(LNUM$):L=ASC(LNUM$(P,P)):L=L+128:LNUM$(P,P)=CHR$(L):NEXT P
C 2410 TRAP 32767:CLOSE #2:MESP=1:RING=1:GOSUB 4180:RETURN
W 2420 RETURN
Z 2430 REM *******SAVE TEXT FILE******
Q 2440 MESP=30:RING=1:GOSUB 4180
K 2450 POSITION 0,8:? BLANK$;BLANK$
W 2460 POSITION 2,8:? "SAVE TEMPLATE WITH TEXT?(◪Y◪/◪N◪)"
R 2470 GET #6,K:IF K<>89 AND K<>121 AND K<>78 AND K<>110 THEN 2470
B 2480 TFLAG=0:IF K=89 OR K=121 THEN TFLAG=1
K 2490 EXT$=".TXT":GOSUB 580
Q 2500 POSITION 0,8:? BLANK$;BLANK$
G 2510 IF CFLAG THEN POSITION 9,8:? "CUE TAPE AND PRESS <PLAY/RECORD>":POSITION 9,9:? "PRESS <RETURN> TO BEGIN"
A 2520 TRAP 2600:CLOSE #2:OPEN #2,8,0,FILE$
S 2530 MESP=3:GOSUB 4180
X 2540 PUT #2,TFLAG
W 2550 NUM=4455:ADD=ADR(TEXT$):RWF=11:CHN=2:GOSUB 440
A 2560 IF TFLAG THEN NUM=54:ADD=ADR(LCODE$):RWF=11:CHN=2:GOSUB 440
A 2570 IF TFLAG THEN NUM=54:ADD=ADR(INDENT$):RWF=11:CHN=2:GOSUB 440
Q 2580 PUT #2,RMAR:PUT #2,LMAR:CLOSE #2
K 2590 GOTO 2610
T 2600 MESP=7:GOSUB 4180:FOR WAIT=1 TO 150:NEXT WAIT
S 2610 POSITION 0,8:? BLANK$;BLANK$
E 2620 TRAP 32767:CLOSE #2:WORK$=TEXT$(LNUM*80,LNUM*80+79):POSITION 0,8:? WORK$;
D 2630 MESP=1:RING=1:GOSUB 4180:RETURN
S 2640 REM *******LOAD TEMPLATE******
I 2650 MESP=31:RING=1:GOSUB 4180:FOR WAIT=1 TO 70:NEXT WAIT
V 2660 POSITION 0,8:? BLANK$;BLANK$
W 2670 EXT$=".TEM":GOSUB 580
A 2680 IF CFLAG THEN POSITION 9,8:? "CUE TAPE AND PRESS <PLAY>":POSITION 9,9:? "PRESS <RETURN> TO BEGIN"
J 2690 TRAP 2740:CLOSE #2:OPEN #2,4,0,FILE$
E 2700 MESP=2:GOSUB 4180
S 2710 NUM=54:ADD=ADR(LCODE$):RWF=7:CHN=2:GOSUB 440
J 2720 NUM=54:ADD=ADR(INDENT$):RWF=7:CHN=2:GOSUB 440:CLOSE #2
H 2730 GOTO 2750
P 2740 MESP=8:GOSUB 4180:FOR WAIT=1 TO 150:NEXT WAIT
B 2750 POSITION 0,8:? BLANK$;BLANK$
G 2760 TRAP 32767:CLOSE #2:MESP=1:RING=1:GOSUB 4180:RETURN
J 2770 REM *******SAVE TEMPLATE******
I 2780 MESP=32:RING=1:GOSUB 4180:FOR WAIT=1 TO 200:NEXT WAIT
V 2790 POSITION 0,8:? BLANK$;BLANK$
N 2800 EXT$=".TEM":GOSUB 580
N 2810 IF CFLAG THEN POSITION 9,8:? "CUE TAPE AND PRESS <PLAY/RECORD>":POSITION 9,9:? "PRESS <RETURN> TO BEGIN"
C 2820 TRAP 2870:CLOSE #2:OPEN #2,8,0,FILE$
J 2830 MESP=3:GOSUB 4180
D 2840 NUM=54:ADD=ADR(LCODE$):RWF=11:CHN=2:GOSUB 440
G 2850 NUM=54:ADD=ADR(INDENT$):RWF=11:CHN=2:GOSUB 440:CLOSE #2
X 2860 GOTO 2880
Q 2870 MESP=7:GOSUB 4180:FOR WAIT=1 TO 150:NEXT WAIT
T 2880 POSITION 0,8:? BLANK$;BLANK$
J 2890 TRAP 32767:CLOSE #2:WORK$=TEXT$(LNUM*80,LNUM*80+79):POSITION 0,8:? WORK$;
F 2900 MESP=1:RING=1:GOSUB 4180:RETURN
A 2910 REM **********EXIT*************
V 2920 MESP=10:RING=1:GOSUB 4180
A 2930 GET #6,K
G 2940 IF K<>89 AND K<>121 AND K<>78 AND K<>110 THEN 2930
L 2950 IF K=78 OR K=110 THEN 2980
I 2960 MESP=11:RING=1:GOSUB 4180
N 2970 FOR WAIT=1 TO 150:NEXT WAIT:GRAPHICS 0:END
E 2980 POSITION 25,19:? "◪              ◪":RETURN
T 2990 REM ********SDIM MODULE*********
A 3000 DIM TEXT$(4455),LCODE$(54),INDENT$(54)
Z 3010 DIM MESSAGE$(20),IN$(80),WORK$(80),OUT$(80)
M 3020 DIM TTL$(80)
P 3030 DIM FILE$(25),EXT$(4),XBLANK$(40)
G 3040 DIM BLANK$(40),LNUM$(2),FILLER$(80)
A 3050 DIM SETTING(5)
F 3060 TEXT$=" ":TEXT$(4455)=TEXT$:TEXT$(2)=TEXT$
J 3070 LCODE$="◪ ◪":LCODE$(54)=LCODE$:LCODE$(2)=LCODE$
B 3080 INDENT$="{CTRL ,}":INDENT$(54)=INDENT$:INDENT$(2)=INDENT$
B 3090 XBLANK$="◪ ◪":XBLANK$(40)=XBLANK$:XBLANK$(2)=XBLANK$
N 3100 BLANK$=" ":BLANK$(40)=BLANK$:BLANK$(2)=BLANK$
E 3110 FILLER$=" ":FILLER$(40)=FILLER$:FILLER$(2)=FILLER$
W 3120 OPEN #6,4,0,"K:":OPEN #5,9,0,"E:"
G 3130 POKE 752,1:POKE 82,0
J 3140 LNUM=1:MAXLINES=54
X 3150 RETURN
B 3160 REM *****BELL*****
V 3170 FOR L=15 TO 0 STEP -1:SOUND 0,10,10,L:FOR P=1 TO 4:NEXT P:NEXT L
R 3180 RING=0:RETURN
Y 3190 REM ********U-PRINT MODULE******
N 3200 FOR CYCLES=1 TO LINES
Z 3210 READ MESP,X,Y,MESSAGE$
G 3220 IF MESP>=MESL AND MESP<=MESH THEN POSITION X,Y:? MESSAGE$
O 3230 NEXT CYCLES
F 3240 RETURN
A 3250 REM ****BLANK MODULE************
Z 3260 ? "{ESC CTRL <}":POKE 710,0:POKE 712,15
Z 3270 FOR Y=1 TO 6
N 3280 POSITION 0,Y:? XBLANK$;
Q 3290 NEXT Y
Z 3300 ? :FOR Y=1 TO 80:? " ";:NEXT Y:FOR Y=1 TO 40:? "{CTRL N}";:NEXT Y
P 3310 FOR Y=11 TO 22
M 3320 POSITION 0,Y:? XBLANK$;
X 3330 NEXT Y
I 3340 RETURN
F 3350 REM *****PLACE BLOCKS ON SCREEN*****
Y 3360 REM ********TOP HALF************
P 3370 LINES=10:MESL=21:MESH=30:GOSUB 3200
U 3380 RETURN
H 3390 DATA 21,3,1,◪CTRL L◪
E 3400 DATA 22,22,1,◪CTRL C◪
N 3410 DATA 23,2,2,◪ {17 CTRL N} ◪
I 3420 DATA 24,21,2,◪ {16 CTRL N} ◪
Q 3430 DATA 25,2,3,◪{CTRL B} LINE NUMBER {CTRL V}1 {CTRL V}◪
C 3440 DATA 26,21,3,◪{CTRL B} LINE CODE {CTRL V} {CTRL V}◪
C 3450 DATA 27,2,4,◪ {17 CTRL M} ◪
S 3460 DATA 28,21,4,◪ {16 CTRL M} ◪
F 3470 DATA 29,3,6,◪CTRL T◪
Q 3480 DATA 30,21,6,◪CTRL E = ERASE ◪
A 3490 REM ******BOTTOM HALF*****
W 3500 LINES=18:MESL=31:MESH=48:GOSUB 3200
D 3510 RETURN
L 3520 DATA 31,23,12,◪CTRL P ◪
S 3530 DATA 32,23,15,◪{15 CTRL M}◪
D 3540 DATA 33,23,13,◪{15 CTRL N}◪
K 3550 DATA 34,22,14,◪{CTRL B} PRINTER {CTRL B} OFF {CTRL V}◪
V 3560 DATA 35,2,12,◪ CTRL M ◪
U 3570 DATA 36,2,13,◪ {9 CTRL N}◪
A 3580 DATA 37,2,14,◪{CTRL B} MENU{CTRL B}     {CTRL B}◪
I 3590 DATA 38,2,15,◪{CTRL B}{18 CTRL M}{CTRL V}◪
J 3600 DATA 39,2,16,◪{CTRL B} 1 PRINT DOCUMENT {CTRL V}◪
R 3610 DATA 40,2,17,◪{CTRL B} 2 LOAD TEXT FILE {CTRL V}◪
```

Continued

TYPE-IN LISTINGS

ELECTRONIC TYPEWRITER *Continued*

ATARI 800/800XL/130XE

```
A 3620 DATA 41,2,18,◩{CTRL B} 3 SAVE TEXT FILE {CTRL V}◩
A 3630 DATA 42,2,19,◩{CTRL B} 4 LOAD TEMPLATE {CTRL V}◩
O 3640 DATA 43,2,20,◩{CTRL B} 5 SAVE TEMPLATE {CTRL V}◩
Z 3650 DATA 44,2,21,◩{CTRL B} 6 EXIT PROGRAM {CTRL V}◩
A 3660 DATA 45,2,22,◩ {18 CTRL M} ◩
Y 3670 DATA 46,23,18,◩{CTRL B}{14 CTRL M}{CTRL V}◩
R 3680 DATA 47,23,19,◩{CTRL B} {CTRL V}◩
N 3690 DATA 48,23,20,◩{CTRL B}{14 CTRL N}{CTRL V}◩
C 3700 REM ********FUNCT MODULE******
A 3710 GET #6,CHAR
H 3720 IF CHAR=3 THEN CHAR=6:GOTO 3790
R 3730 IF CHAR=13 THEN CHAR=2:GOTO 3790
K 3740 IF CHAR=12 THEN CHAR=3:GOTO 3790
U 3750 IF CHAR=5 THEN CHAR=4:GOTO 3790
R 3760 IF CHAR=20 THEN CHAR=5:GOTO 3790
F 3770 IF CHAR=16 THEN CHAR=1:GOTO 3790
O 3780 GOTO 3710
L 3790 RETURN
S 3800 REM *********TO PRINTER********
R 3810 TRAP 4150
T 3820 CLOSE #4:OPEN #4,8,0,"P:"
J 3830 IF LEN(OUT$)>0 THEN IF OUT$(LEN(OUT$))=" " THEN OUT$(LEN(OUT$))="":GOTO 3830
X 3840 WORK$=LCODE$(L,L)
P 3850 IF WORK$="◩ ◩" THEN 4100
L 3860 IF WORK$="◩B◩" OR WORK$="◩b◩" THEN CHAR=1:GOTO 3900
D 3870 IF WORK$="◩C◩" OR WORK$="◩c◩" THEN CHAR=2:GOTO 3900
V 3880 IF WORK$="◩D◩" OR WORK$="◩d◩" THEN CHAR=3:GOTO 3900
N 3890 IF WORK$="◩E◩" OR WORK$="◩e◩" THEN CHAR=4:GOTO 3900
O 3900 ON CHAR GOTO 3920,3950,4000,4050
Y 3910 GOTO 4150
C 3920 REM ***HANDLE 'B'***
Q 3930 ? #4:IF LSPACE=2 THEN ? #4
C 3940 GOTO 4160
E 3950 REM ***HANDLE 'C'***
C 3960 P=LEN(OUT$):P=(MAXLEN-P)/2
Q 3970 WORK$=OUT$:OUT$=BLANK$
R 3980 OUT$(P)=WORK$
L 3990 GOTO 4100
M 4000 REM ***HANDLE 'D'***
N 4010 WORK$=OUT$:INDENT=ASC(INDENT$(L,L))
S 4020 OUT$=BLANK$:OUT$(41)=BLANK$
S 4030 OUT$(INDENT+1)=WORK$
U 4040 GOTO 4100
A 4050 REM ***HANDLE 'E'***
M 4060 P=LEN(OUT$):P=MAXLEN-P+1
H 4070 WORK$=OUT$
Y 4080 OUT$=BLANK$:OUT$(41)=BLANK$
Q 4090 OUT$(P)=WORK$
W 4100 REM ***PRINT**
S 4110 WORK$=OUT$:OUT$=BLANK$:OUT$(LMAR+1)=WORK$
M 4120 ? #4;OUT$
V 4130 IF LSPACE=2 THEN ? #4
T 4140 GOTO 4160
E 4150 MESP=6:GOSUB 4180:FOR WAIT=1 TO 100:NEXT WAIT:L=55
M 4160 TRAP 32767:CLOSE #4:RETURN
Y 4170 REM *********MPRINT MODULE******
S 4180 RESTORE 4240+((MESP-1)*10)
Y 4190 READ X,Y,MESSAGE$
X 4200 POSITION X,Y:? "◩ ◩"
J 4210 POSITION X,Y:? MESSAGE$
R 4220 IF RING THEN GOSUB 3170
F 4230 RETURN
I 4240 DATA 25,19,PICK FUNCTION
E 4250 DATA 25,19,LOADING FILE
D 4260 DATA 25,19,SAVING FILE
M 4270 DATA 25,19,PRINTING TEXT
Q 4280 DATA 25,19,PRINTING FILE
A 4290 DATA 25,19,{ESC CTRL 2}PRINT ERROR
G 4300 DATA 25,19,{ESC CTRL 2}SAVE ERROR
U 4310 DATA 25,19,{ESC CTRL 2}LOAD ERROR
O 4320 DATA 25,19,SET LIMITS
G 4330 DATA 25,19,QUIT?(Y/N)
A 4340 DATA 25,19,SESSION OVER
A 4350 DATA 25,19,{ESC ESC}{ESC CTRL +}SELECT
A 4360 DATA 25,19,{ESC CTRL 2}**INVALID**
N 4370 DATA 25,19,◩ ◩
M 4380 DATA 25,19,ERASE...
P 4390 DATA 25,19,{ESC CTRL 2}ERASING...
M 4400 DATA 25,19,{ESC ESC}{ESC CTRL 2}ENTER LINE#
H 4410 DATA 25,19,D◩ISK/◩T◩APE?◩
Q 4420 DATA 25,19,{ESC ESC}{ESC CTRL -}FILENAME?
C 4430 DATA 25,19,CORRECT?(Y/N)
N 4440 DATA 25,19,ENTER CODE{ESC ESC}{ESC CTRL -}
O 4450 DATA 25,19,{ESC ESC}{ESC CTRL -}ENTER TEXT
O 4460 DATA 25,19,TEXT?(Y/N)
G 4470 DATA 25,19,LCODES?(Y/N)
T 4480 DATA 25,19,{ESC ESC}{ESC CTRL -}ENTER VALUE
P 4490 DATA 25,19,{ESC CTRL 2}OUT OF RANGE
X 4500 DATA 25,19,PLEASE WAIT..
N 4510 DATA 25,19,PRINT TEXT
D 4520 DATA 25,19,LOAD TEXT
T 4530 DATA 25,19,SAVE TEXT
A 4540 DATA 25,19,LOAD TEMPLATE
I 4550 DATA 25,19,SAVE TEMPLATE
M 4560 REM ******SETTINGS MODULE******
U 4570 MESP=9:RING=1:GOSUB 4180
E 4580 RESTORE 4760
D 4590 FOR L=1 TO 3
I 4600 READ DEFAULT,LO,HI,MESSAGE$
P 4610 POSITION 25,19:? "◩ ◩":POSITION 25,19:? "ENTER ";LO;"-";HI
O 4620 POSITION 12,8:? MESSAGE$
D 4630 POSITION 30,8:? DEFAULT
S 4640 POSITION 12,9:? "INPUT NEW SETTING{ESC ESC}{ESC CTRL *}"
O 4650 POSITION 30,9:GOSUB 330
P 4660 IF A=-1 THEN A=DEFAULT:? A
C 4670 IF A<LO OR A>HI THEN 4620
H 4680 SETTING(L)=A
W 4690 POSITION 12,8:? "                "
J 4700 POSITION 12,9:? "                "
N 4710 NEXT L
N 4720 LMAR=SETTING(1):RMAR=SETTING(2):LSPACE=SETTING(3)
X 4730 MAXLEN=80-(LMAR+RMAR):MAX=MAXLEN
T 4740 MESP=1:RING=1:GOSUB 4180
Y 4750 RETURN
B 4760 DATA 5,0,39, LEFT MARGIN
Z 4770 DATA 5,0,39,RIGHT MARGIN
V 4780 DATA 2,1,2,LINE SPACING
Z 4790 REM ***FORMAT INDENT***
M 4800 POSITION 34,3:? "◩ ◩"
N 4810 POSITION 34,3:? LCODE$(LNUM,LNUM);:IF LCODE$(LNUM,LNUM)<>"◩D◩" AND LCODE$(LNUM,LNUM)<>"◩d◩" THEN RETURN
T 4820 WORK$=STR$(ASC(INDENT$(LNUM,LNUM))):FOR P=1 TO LEN(WORK$):L=ASC(WORK$(P,P)):L=L+128:WORK$(P,P)=CHR$(L)
U 4830 NEXT P:? WORK$:RETURN
N 4840 IF LEN(WORK$)=0 THEN 4870
T 4850 MESP=27:GOSUB 4180
Q 4860 IF WORK$>" " THEN IF WORK$(LEN(WORK$))=" " THEN WORK$(LEN(WORK$))="":GOTO 4860
N 4870 RETURN
```

HCM

ELECTRONIC TYPEWRITER

COMMODORE 64

```
N 100 REM ***************************
N 110 REM * ELECTRONIC TYPEWRITER *
Z 120 REM ***************************
M 130 REM COPYRIGHT 1985
X 140 REM EMERALD VALLEY PUBLISHING CO
F 150 REM BY RANDY THOMPSON
N 160 REM HOME COMPUTER MAGAZINE
V 170 REM VERSION 5.5.1
P 180 REM C-64 BASIC
M 190 REM
G 200 REM DISPLAY TITLE SCREEN
Q 210 POKE 53281,0:POKE 53280,0:POKE 53272,23:POKE 657,128:POKE 650,128
O 220 POKE 646,15:FOR I=54272 TO 54296:READ S:POKE I,S:NEXT
M 230 T$="{SHIFT E}LECTRONIC {SHIFT T}YPEWRITER":PRINT"{SHIFT CLR}{11 CRSRDOWN}"TAB(9);
U 240 FOR T=1 TO LEN(T$):PRINT MID$(T$,T,1);:GOSUB900:FOR J=1 TO RND(1)*200+35
G 250 NEXT:NEXT:GOSUB880:X=7:Y=24:GOSUB1320:PRINT"{SHIFT P}RESS {CTRL RVSON}{SHIFT R}{SHIFT E}{SHIFT T}{SHIFT U}{SHIFT R}{SHIFT N}{CTRL RVSOFF} TO CONTINUE.";
T 260 POKE 198,0
S 270 GET K$:IF K$<>CHR$(13) THEN270
U 280 REM INITIALIZE PROGRAM
W 290 PRINT"{SHIFT CLR}{SHIFT I}NITIALIZING..."
B 300 FOR I=1 TO 80:BL$=BL$+" ":NEXT
J 310 RM=5:REM RIGHT MARGIN
X 320 LM=5:REM LEFT MARGIN
H 330 SP=2:REM LINE SPACING
A 340 PR=0:REM PRINTER STATUS 0/1=OFF/ON
K 350 MX=54:DIM TX$(MX),TP$(MX):LN=1:M=3
V 360 GOSUB1510:GOSUB2490:GOSUB790
O 370 REM MAIN CONTROL LOOP
Z 380 ON M GOSUB390,450,570,740,1340:GOTO 380
E 390 REM F1 - LINE
H 400 S$=F1$:T$="{SHIFT E}NTER A 1-"+MID$(STR$(MX),2)+" "
```

Continued

TYPE-IN LISTINGS

```
AEGGYAHNDKSEZIWHNUKLLLXHGKABLNFJXAWRNCPLWWNNPOZYEDXRARWBDDAMSNFHPTBLAM
ZGKH
```

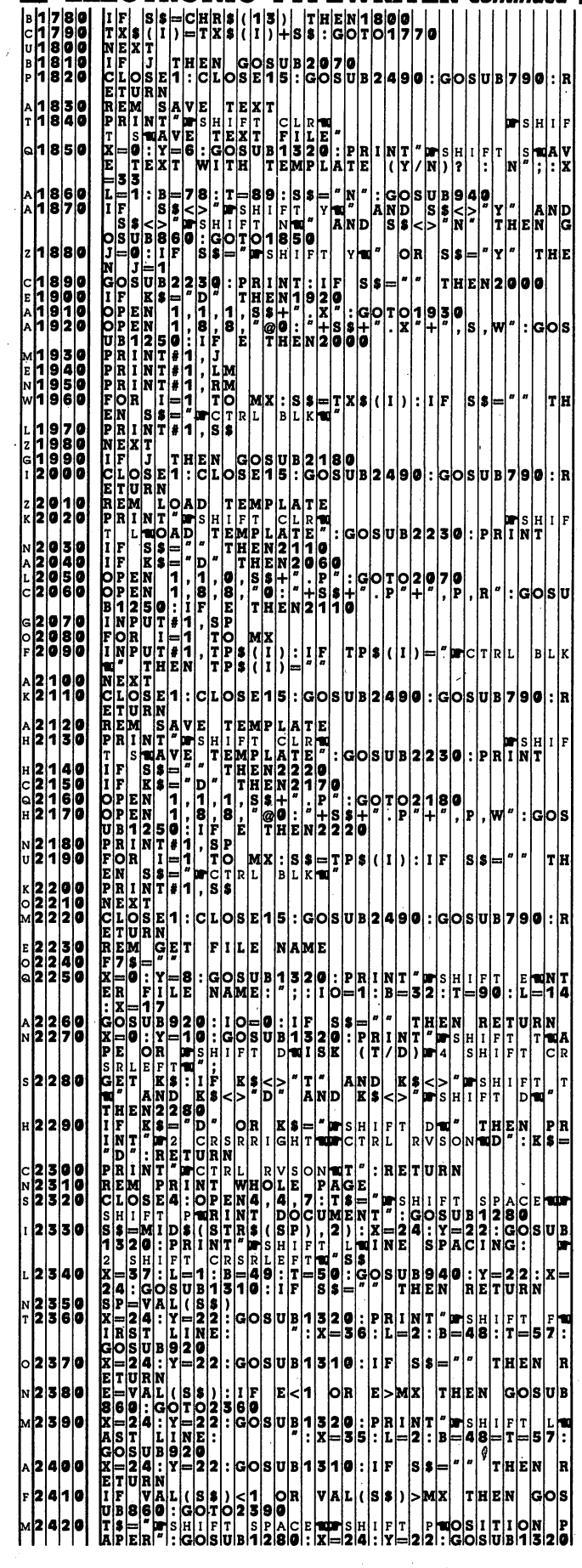
```
410 X=13:Y=2:L=2:B=48:T=57:GOSUB940:IF
420 S$==" SEN":GOSUB860:GOTO410
430 F THEN VAL(F1$)>MX
    IF VAL(F1$)<>LN THEN LN=VAL(F1$):GOS
440 UB790RN
450 REM ----- CODE -----
460 S$=LEFT$(TP$(LN),1)
470 X=34:Y=2:L=LEN(TP$(LN))-1:IF S$="
    "THEN X=35:Y=2:GOSUB1320:PRINT"
480 ";:RETURN
490 IF S$="" AND K<>13 THEN RETURN
500 S$=TP$(LN):IF LEN(TP$(LN))>1 THEN S$=MID
    $(TP$(LN),3)
510 M=2:X=36:Y=2:L=2:B=48:T=57:GOSUB940
    :IF S$="0" THEN S$=TP$(LN):RM=LEN(S$)
520 T=T+1:IF MID$(S$,RM,1)="-" THEN S$=LEFT$(S$,RM-1):GOTO
    510
530 T=T+1:S$=S$+MID$(TP$(LN),2):IF RIGHT$(S$,1)="-" THEN
    540
540 S$=LEFT$(S$,LEN(S$)-1):GOTO530
550 TP$(LN)=S$:RETURN
560 REM ----- TEXT -----
570 REM ----- FIND LN -----
580 S$=S$+" ":LM=LEN(S$):IF LM=1 THEN RETURN
590 L=-1:FOR T=1 TO LM:C$=MID$(S$,T,1):IF C$=" " AND L=-1 THEN
    600
600 NEXT:IF L=-1 THEN RETURN
610 IF C$="-" THEN L=T:NEXT:GOTO600
620 IF L=-1 THEN L=T
630 IF L=1 THEN RETURN
640 IF MID$(S$,L-1,1)="-" THEN L=L-1:GOTO630
650 S$=LEFT$(S$,L-1):RETURN
660 REM ----- ADD CHAR -----
670 IF LEN(S$)<MX THEN S$=S$+C$:RETURN
680 FOR T=1 TO LEN(S$)-1:S$=LEFT$(S$,T)+MID$(S$,T+2):NEXT:S$=S$+C$:RETURN
690 REM ----- INSERT -----
700 IF LEN(S$)=MX THEN RETURN
710 FOR T=LEN(S$)+1 TO L STEP-1:S$=MID$(S$,T-1)+S$:NEXT:S$=LEFT$(S$,L-1)+C$+S$:RETURN
720 REM ----- PRINT -----
730 PRINT#4,LEFT$(LN$,LM);:RETURN
740 REM ----- MENU -----
750 S$=" ":POKE 32,PEEK(232)OR128
760 X=5:Y=1:B=49:L=1:THEN T$=" SHIFT S ELECT
    IF T$=" " THEN RETURN
770 IF T$=":" THEN 1160
780 ON VAL(T$)GOSUB230,240,250,260,270,280,290,300,310,320,330,340,350,360,370,380,390,400,410,420
790 REM ----- SAVE -----
800 PRINT#4,LEFT$(LN$,LM);:RETURN
810 REM ----- LOAD -----
820 POKE781,9:SYS59903:X=0:Y=8:GOSUB1370:POKE781,3:SYS59903:X=0:Y=8:GOSUB1370:POKE212,0:PRINT"
    ";:RETURN
830 POKE212,0:PRINT"
    ";:POKE226,0:POKE226,PEEK(226)AND127
840 POKE226,0:POKE226,PEEK(226)OR128:POKE226,PEEK(226)AND127
850 RETURN
860 REM ----- ERROR -----
870 FOR I=1 TO 2:POKE 54283,33:FOR J=1 TO 75:NEXT:POKE 54283,32:NEXT:RETURN
880 REM ----- DING -----
890 POKE 54276,17:POKE 54276,16:RETURN
900 REM ----- WACK -----
910 POKE 54290,129:FOR I=1 TO 5:POKE 54290,128:RETURN
920 REM ----- INPUT ROUTINE -----
930 S$="":B1320:S=1024+X+Y*40:POKE 213,L+198,0:POKE S,32:POKE 204,0:GOSUB1370:POKE 204,1:POKE S,X:GOSUB1320:IF S$="" THEN 930
940 IF K=13 THEN1150
950 IF K<>20 AND K<>157 AND K>132 AND K<>133 AND K<>134 AND K<>135 AND K<>136 AND K<>137 THEN 970
960 IF LEN(S$)=0 THEN 930
970 IF K=20 AND LEN(S$)=1 THEN S$="":GOTO1000
980 IF K=20 THEN S$=LEFT$(S$,LEN(S$)-1):GOTO1000
990 IF K=157 AND LEN(S$)=1 THEN S$="":GOTO1000
1000 IF K=157 AND LEN(S$)>1 THEN S$=LEFT$(S$,LEN(S$)-1):GOTO1000
1010 IF K=133 OR K=134 OR K=135 OR K=136 OR K=137 THEN 1140
1020 IF LEN(S$)=80 THEN 930
1030 IF K=32 OR K=160 THEN S$=S$+" ":GOTO1000
1040 IF K=34 THEN 930
1050 IF K=44 OR K=58 OR K=59 OR K=61 OR K=63 OR K=64 THEN 1000
1060 IF K<32 OR K>90 THEN 930
1070 S$=S$+CHR$(K):GOTO1000
1080 REM ----- CURSOR -----
1090 IF I=0 THEN I=1:RETURN
1100 X=X+1:IF X=L-5 AND I=0 THEN GOSUB8
1110 S$=LEFT$(S$,X-1)+MID$(S$,X+1)
1120 IF LEN(S$)=80 THEN S$=LEFT$(S$,79)
1130 PRINT K$;:POKE 216,0
```

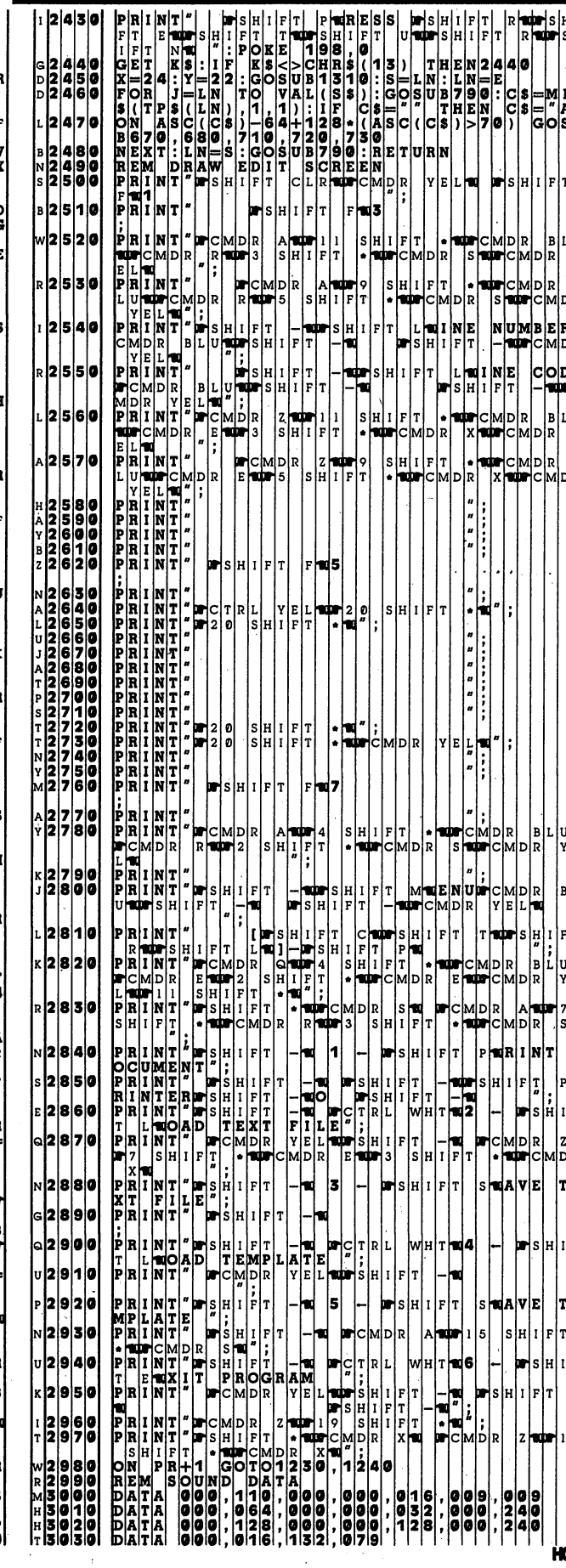
```
M1150 POKE 204,1:POKE S+X,PEEK(S+X) AND 1
2 7:X=X+I
M1160 RETURN
W1170 REM ----- INPUT ONE CHARACTER -----
W1180 POKE 204,0:POKE 207,0:GET K$:IF K$=
F 1190 "" THEN K=ASC(K$):IF K>192 AND K<258 THEN K
  =K-128
V1200 POKE212,0:RETURN
K1210 REM ----- SET/RESET ON PR+1 GOTO1230, 1240
Z1230 POKE4:POKE 1737,6:POKE 1738,6:RETU
  RN:CLOSE4:POKE 1737,7:POKE 1738,6:RETU
N1240 RN:OPEN4,4,7:POKE 1737,14:POKE
  1738,32:RETURN
D1250 REM ----- ERROR -----
F1260 OPEN15,8,15:INPUT#15,E,E$:IF E THE
  N E$:GOSUB1300
E1270 RETURN
A1280 REM ----- DIY -----
N1290 PRINT"
  ";:GOSUB1320:PRINT"
  ";:GOSUB860:FOR
O1300 I=1TO200:NEXT
  ";:RETURN
H1310 GET"
  ";:IF I=0 THEN 1310
  ";:RETURN
N1320 REM ----- CURSOR AT X,Y -----
R1330 POKE781,Y:POKE782,X:POKE783,0:SY
  S65520:REM DATA
A1340 REM ----- SHIFT ENTER/RASE ALL DATA? ":GO
K1350 SUB1450:IF I=0 THEN RETURN
N1360 IF S$="" THEN X=
  24:Y=22:GOSUB1310:RETURN
L1370 X=24:Y=22:GOSUB1320:PRINT" SHIFT S
  PACE SHIFT ENTER/RASING DATA ";:CLOSE4
G1380 FOR I=1TO MX:TX$(I)="":NEXT:TP$(I)=
  EXT:GOSUB1510:GOSUB2490:LN=1:GOTO779
S1390 REM ----- PROGRAM
U1400 RT$=" SHIFT ENTER/RASE PROGRAM? ":GOSUB1
  280:GOSUB1450
V1410 IF S$="" THEN X=
  24:Y=22:GOSUB1310:RETURN
H1420 PRINTTAB(T-1)" SHIFT ENTER/RASING DATA...";:FOR
  T=1TO 900:NEXT:RETURN
X1430 FOR J=0TO RND(1)*200+50:NEXT:NEXT:
  FOR I=1TO 900:POKE 657,0:POKE 650,0
A1440 NEXT:RETURN
F1450 REM ----- GET A YES OR NO RESPONSE
O1460 IF A$="" THEN 1460
M1470 IF A$="Y" OR A$="N" THEN X=24:Y=22:GOSUB1320:PRINT" SHIFT ENTER/RASING YOU SURE? ":G
  OSUB1280:GOTO1450
N1480 IF A$="Y" THEN A=1:RETURN
Y1490 IF A$="N" THEN A=0:RETURN
D1500 GOTO1460
A1510 REM ----- GET/MARGINS/SPACING
X1520 IF A$="" THEN 1520
E1530 IF A$="" THEN A=7:RETURN
N1540 IF A$="" THEN A=3:RETURN
P1550 IF A$="" THEN A=14:RETURN
H1560 IF A$="" THEN A=2:RETURN
L1570 IF A$="" THEN A=1:RETURN
F1580 IF A$="" THEN A=0:RETURN
L1590 IF A$="" THEN A=15:RETURN
S1600 IF A$="" THEN A=13:RETURN
V1610 IF A$="" THEN A=8:RETURN
J1620 IF A$="" THEN A=4:RETURN
A1630 IF A$="" THEN A=5:RETURN
K1640 IF A$="" THEN A=6:RETURN
I1650 IF A$="" THEN A=9:RETURN
O1660 IF A$="" THEN A=10:RETURN
Q1670 IF A$="" THEN A=11:RETURN
Q1680 IF A$="" THEN A=12:RETURN
T1690 IF A$="" THEN A=0:RETURN
L1700 IF A$="" THEN A=1:RETURN
F1710 IF A$="" THEN A=2:RETURN
S1720 IF A$="" THEN A=3:RETURN
D1730 IF A$="" THEN A=4:RETURN
I1740 IF A$="" THEN A=5:RETURN
N1750 IF A$="" THEN A=6:RETURN
A1760 IF A$="" THEN A=7:RETURN
```

Continued

ELECTRONIC TYPEWRITER *Continued*

COMMODORE 64





TYPE-IN LISTINGS

HCM

ELECTRONIC TYPEWRITER

IBM PC/PCjr, TANDY 1000

TYPE-IN LISTINGS

```
100 '*****************************
110 '*    ELECTRONIC TYPEWRITER  *
120 '*                           *
130 '*  COPYRIGHT 1985 BY       *
140 '*  BYRAN K. THOMPSON       *
150 '*  AND THE HCM STAFF       *
160 '*                           *
170 '*  HOMEMADE COMPUTER MAGAZINE *
180 '*  VERSION 5.5 CARTRIDGE BASIC *
190 '*  IBM PCjr W/ BASICA       *
200 '*  IBM PC W/ BASICA GW BASIC *
210 '*  TANDY 1000 W/ GW BASIC    *
220 '*****************************
230 SCREEN 0,0:COLOR 7,0:CLS:WIDTH 40:KEY OFF:DEFINT A-Z:GOSUB 1300
240 COLOR ,,0:RESTORE:DEFSTR C:BLUE="1":GREEN="2":CYAN="3":RED="4":MAGENTA="5":BROWN="6":BLACK="0":WHITE="7":GOSUB 1140:DEFINT C:COLOR WHITE,BLACK
250 KEY 1,"LIST"+CHR$(13):KEY 2,"RUN"+CHR$(13):KEY 3,"LOCATE ":KEY 4,"COLOR ":KEY 5,"PRINT ":KEY 6,"NEXT ":KEY 7,"GOSUB ":KEY 8,"GOTO ":KEY 9,"SCREEN 0,0:WIDTH 40:CLS:LIST 1300-":KEY 10,"SAVE ":KEY 11,"LOAD ":KEY 12,"RUN 1140:"
260 FOR I=1 TO 10:KEY(I) ON:NEXT I:ON KEY(1) GOSUB 1080,1090,1100,1110,1120,1130,1140,1150,1160,1170
270 FOR J=1 TO 200*200+35:NEXT J:
280 GOSUB 1250:CLS
290 MAX.LINE=54:LITE=SUITE:MAX.LINE=MAX.LINE-LEN(TEMPLATE$(MAX.LINE))-1:TEMPLATE$(MAX.LINE)=TEMPLATE$(MAX.LINE)+SPACE$(MAX.LINE):MAX.CHAR$=CHR$(MAX.LINE+32):MAX.CHAR=ASC(MAX.CHAR$)
300 DIM LINE$(MAX.LINE),TEMPLATE$(MAX.LINE):FOR I=1 TO MAX.LINE:LINE$(I)=CHR$(I+32):TEMPLATE$(I)=LINE$(I):NEXT I:LINE$(0)=""
310 LOCATE 5:LEFTM=2:PRINTER.STATUS=0:LOCATE 5:PRINT "SET MARGINS AND LINE SPACING":LOCATE 7:INPUT "LEFT MARGIN";LEFTM:LEFTM=LEFTM+1:LOCATE 8:INPUT "RIGHT MARGIN";RIGHTM:RIGHTM=RIGHTM+1:LOCATE 9:INPUT "LINES PER INCH";LPI:IF LPI=6 THEN LPI$="6" ELSE LPI$="8"
320 WIDTH RIGHTM-LEFTM:LINE=1:PAGE=1:ROW=1:COL=LEFTM:CLS:LOCATE ROW,COL:PRINT "PAGE";PAGE;"LINE";LINE;"COLUMN";COL;:LOCATE ROW,COL:PRINT "PAGE";PAGE;"LINE";LINE;"COLUMN";COL;:LOCATE ROW,COL
330 ON ERROR GOTO 810:ON KEY(12) GOSUB 1180:KEY(12) ON:ON KEY(13) GOSUB 1200:KEY(13) ON:ON KEY(14) GOSUB 1220:KEY(14) ON:ON KEY(15) GOSUB 1240:KEY(15) ON:ON KEY(16) GOSUB 1260:KEY(16) ON
340 LOCATE ROW,COL:A$=INKEY$:IF A$="" THEN 340
350 IF A$=CHR$(13) THEN PRINT#LPT1,CHR$(10);:ROW=ROW+1:COL=LEFTM:LOCATE ROW,COL:GOTO 340
360 IF A$=CHR$(8) THEN COL=COL-1:IF COL<LEFTM THEN ROW=ROW-1:COL=RIGHTM-1:LOCATE ROW,COL:GOTO 340
370 IF A$=CHR$(9) THEN COL=COL+5:IF COL>RIGHTM-1 THEN ROW=ROW+1:COL=LEFTM:LOCATE ROW,COL:GOTO 340
380 IF A$=CHR$(27) THEN GOSUB 1190:GOTO 340
390 IF A$="" THEN 340
400 IF LEFT$(A$,1)=CHR$(27) THEN 340
410 PRINT#LPT1,A$;:COL=COL+1:LOCATE ROW,COL:GOTO 340
420 '
430 '
440 '
450 '
460 '
470 '
480 '
490 '
500 '
510 '
520 '
530 '
540 '
550 '
560 '
570 '
580 '
590 '
600 '
610 '
620 '
630 '
640 '
650 '
660 '
670 '
680 '
690 '
700 '
710 '
720 '
730 '
740 '
750 '
760 '
770 '
780 '
790 '
800 '
810 '
820 '
830 '
840 '
850 '
860 '
870 '
880 '
890 '
900 '
910 '
920 '
930 '
940 '
950 '
960 '
970 '
980 '
990 '
1000 '
1010 '
1020 '
1030 '
1040 '
1050 '
1060 '
1070 '
1080 '
1090 '
1100 '
1110 '
1120 '
1130 '
1140 '
1150 '
1160 '
1170 '
1180 '
1190 '
1200 '
1210 '
1220 '
1230 '
1240 '
1250 '
1260 '
1270 '
1280 '
1290 '
1300 '
1310 '
1320 '
1330 '
1340 '
1350 '
1360 '
1370 '
```

Continued

ELECTRONIC TYPEWRITER *Continued*

```
N 1380 CODE.ROW = 3:CODE.COL = 63:LNUM.ROW
        = 3:LNUM.COL = 25:TEXT.ROW = 8:TEX
       T.COL = 1:MENU.ROW = 14:MENU.COL =
       27
Q 1390 CHANGE.LINE = 1:LINE.CODE = 2:TEXT
       = 3:MENU = 4:ERASE.DATA = 5
D 1400 KEY 1, CHR$(1)::KEY 2, CHR$(2):KEY
       3, CHR$(3): KEY 4, CHR$(4):
A 1410 KEY 5, CHR$(5):KEY 6, CHR$(6):KEY 7
       , "":KEY 8, "":KEY 9, "":KEY 10, CH
       R$(10)
F 1420 RETURN
E 1430 'MENU CONTROL CODE
F 1440 S$ = LAST.MCHOICE$:COLOR YELLOW, BL
       ACK
M 1450 COLOR BLUE:ROW =MENU.ROW:COL =MENU.
       COL:LENGTH =1:LOCATE ROW, COL:PRINT
        S$;
A 1460 MIN.CHAR$ = "1":MAX.CHAR$ = "6"
V 1470 GOSUB 1640:LAST.MCHOICE$ = S$:IF MO
       DE <> MENU THEN RETURN
Z 1480 IF S$ = "" THEN MESSAGE$ = "ENTER A
        NUMBER BETWEEN 1-6 ":GOSUB 1120:
                 GOTO 1440
K 1490 ON VAL(S$) GOSUB 2430,2730,2850,303
       0,3110,3190:GOTO 1440
P 1500 'DISPLAY A SINGLE-SIDED BOX
E 1510 ULC = 218:URC = 191:LLC = 192:LRC =
        217:H = 196:V = 179:GOSUB 1540:RET
       URN
C 1520 'DISPLAY A DOUBLE-SIDED BOX
H 1530 ULC = 201:URC = 187:LLC = 200:LRC =
        188:H = 205:V = 186:GOSUB 1540:RET
       URN
G 1540 ' DISPLAYS A BOX: CALL SINGLE OR DO
       UBLE-SIDED DISPLAYS FIRST !
O 1550 'FOR BEST RESULTS, COLOR BOX BEFORE
        CALL
L 1560 LOCATE ROW, COL, 1
N 1570 PRINT CHR$(ULC);STRING$(WDTH - 2, H
       );CHR$(URC);
A 1580 LOCATE ROW + HEIGHT - 1,COL:PRINT C
       HR$(LLC);STRING$(WDTH - 2,H);CHR$(L
       RC);
W 1590 FOR I = ROW + 1 TO ROW + HEIGHT -2
Q 1600 LOCATE I,COL:PRINT CHR$(V);
L 1610 LOCATE I,COL + WDTH -1:PRINT CHR$(V
       );
S 1620 NEXT
M 1630 RETURN
K 1640 'INPUT ROUTINE
T 1650 'ROW, COL IS WERE THE CURSOR WILL B
       E,LOCATE & PRINT S$ BEFORE CALL
S 1660 SPOS = 1:INSERT = 0
B 1670 LOCATE ROW, COL + SPOS - 1,1
R 1680 X$ = INKEY$:IF X$ = "" THEN GOTO 16
       80
K 1690 IF ((MIN.CHAR$ <= X$) AND (MAX.CHAR
       $ >= X$)) THEN GOSUB 1880:GOTO 1670
N 1700 'CHECK FOR VALID CHARACTER, AND THE
       N BRANCH ON CURSOR CONTROL KEYS
O 1710 IF X$ = CHR$(0) + CHR$(77) THEN GOS
       UB 1990:GOTO 1670 'CURSOR RIGHT
T 1720 IF X$ = CHR$(0) + CHR$(75) THEN GOS
       UB 2020:GOTO 1670 'CURSOR LEFT
N 1730 IF X$ = CHR$(0) + CHR$(82) THEN GOS
       UB 2050:GOTO 1670 'TOGGLE INS MODE
G 1740 IF X$ = CHR$(18) THEN GOSUB 2050:GO
       TO 1670 'TOGGLE INS MODE
X 1750 IF X$ = CHR$(8) THEN GOSUB 2080:GOT
       O 1670 'BACKSPACE
V 1760 IF X$ = CHR$(0) + CHR$(83) THEN GOS
       UB 2110:GOTO 1670 'DELETE A CHAR
I 1770 IF X$ = CHR$(127) THEN GOSUB 2110:G
       OTO 1670 'DELETE A CHAR
G 1780 IF X$ = CHR$(0) + CHR$(79) THEN SPO
       S=LEN(S$):IF SPOS < LENGTH THEN GOS
       UB 1990:GOTO 1670 ELSE GOTO 1670
V 1790 IF X$ = CHR$(0) + CHR$(71) THEN SPO
       S = 1:GOTO 1670 'HOME CURSOR
Z 1800 IF X$ = CHR$(1) THEN MODE = CHANGE.
       LINE:RETURN
X 1810 IF X$ = CHR$(2) THEN MODE = LINE.CO
       DE:RETURN     'CATCH NEW MODE
Y 1820 IF X$ = CHR$(3) THEN MODE = TEXT:RE
       TURN
M 1830 IF X$ = CHR$(4) THEN MODE = MENU:RE
       TURN
N 1840 IF X$ = CHR$(16) THEN GOSUB 1210:GO
       TO 1670 'TOGGLE PRINT MODE
G 1850 IF X$ = CHR$(10) THEN MODE = ERASE.
       DATA:RETURN
C 1860 IF X$ = CHR$(13) THEN RETURN
                   'ENTER AND RETURN
D 1870 GOTO 1670
P 1880 'PLACE A CHARACTER. IF IN INSERT MO
       DE "SCROLL" STRING RIGHT
N 1890 CHARS = LEN(S$):IF ( INSERT = 0 ) T
       HEN GOTO 1940
A 1900 IF ( CHARS >= LENGTH ) THEN RETURN
       ELSE IF (CHARS = LENGTH - 8) THEN B
       EEP
F 1910 S$ = LEFT$(S$, SPOS - 1) + X$ + RIG
       HT$(S$,CHARS - SPOS + 1):LOCATE ROW
       ,COL:     PRINT S$;
H 1920 SPOS = SPOS + 1:IF (SPOS = LENGTH +
        1) THEN SPOS = SPOS - 1:
                 LOCATE ROW, COL+SPOS - 1
J 1930 RETURN
C 1940 IF ( CHARS = LENGTH ) AND (SPOS = L
       ENGTH + 1) THEN RETURN ELSE PRINT X
       $;
U 1950 IF CHARS = LENGTH - 8 THEN BEEP
H 1960 IF (SPOS < CHARS + 1 ) THEN S$ = LE
       FT$( S$, SPOS -1) + X$ + RIGHT$(S$,
        CHARS-SPOS):ELSE S$ = S$ + X$
W 1970 SPOS = SPOS + 1:IF (SPOS = LENGTH +
        1) THEN SPOS = SPOS - 1:LOCATE ROW
       , COL    + SPOS - 1
I 1980 RETURN
O 1990 'CURSOR RIGHT
O 2000 IF (SPOS < LEN(S$) + 1) AND (SPOS <
        LENGTH) THEN SPOS = SPOS + 1:LOCAT
       E ROW, COL + SPOS - 1,1
R 2010 RETURN
K 2020 'CURSOR LEFT
W 2030 IF ( SPOS > 1 ) THEN SPOS = SPOS -
       1:LOCATE ROW, COL + SPOS - 1, 1
U 2040 RETURN
Y 2050 'TOGGLE INSERT MODE
Y 2060 IF ( INSERT = 1 ) THEN INSERT = 0 E
       LSE INSERT = 1
P 2070 RETURN
A 2080 'BACKSPACE
X 2090 CHARS = LEN(S$):IF (( CHARS > 0 ) A
       ND (SPOS <= CHARS+1)) AND (SPOS > 1
       ) THEN S$ = LEFT$(S$, SPOS - 2) + R
       IGHT$(S$, CHARS - SPOS + 1):LOCATE
       ROW, COL:PRINT S$;" ";: SPOS = SPOS
        - 1: LOCATE ROW,COL + SPOS - 2
H 2100 RETURN
M 2110 'DELETE A CHARACTER
M 2120 CHARS = LEN(S$):IF (( CHARS > 0 ) A
       ND (SPOS <= CHARS)) THEN S$ = LEFT$
       (S$,    SPOS - 1) + RIGHT$(S$, CH
       ARS - SPOS):LOCATE ROW, COL:PRINT S
       $;" ";:                LOCATE ROW,COL
       + SPOS - 1:IF (SPOS > CHARS + 1 ) T
       HEN SPOS = CHARS + 1
E 2130 RETURN
Q 2140 'TRAP FILE NAME ERROR
E 2150 GOSUB 2170:RESUME 2160
N 2160 RETURN 550
U 2170 'DISPLAY FILE ERROR MESSAGE
D 2180 COLOR GREY, BLUE
C 2190 CLS:WIDTH 40:ROW = 7:COL = 5:WDTH =
        30:HEIGHT = 5:GOSUB 1520:COL = CO
       L +4
O 2200 COLOR BLACK, MAGENTA:LOCATE ROW + 2
       , COL,0:PRINT " INVALID FILE NAME
        ";
S 2210 FOR J = 1 TO 4000:NEXT:LOCATE ,,1:C
       OLOR WHITE, BLACK
O 2220 RETURN
N 2230 'GET FILE NAME
N 2240 LAST.MCHOICE$ = ""        'CLEAR LAS
       T MENU CHOICE
X 2250 CLS:WIDTH 40:ROW = 5:COL = 5:WDTH =
        30:HEIGHT = 5:COLOR WHITE:GOSUB 15
       20:COLOR RED:LOCATE ROW + 2,(40 - L
       EN(PROMPT$))/2:PRINT PROMPT$;
Z 2260 PROW = 13:PCOL = 3:ROW = PROW - 2:C
       OL = 1:WDTH = 39:HEIGHT = 5:COLOR W
       HITE:GOSUB 1500:COLOR WHITE:LOCATE
       PROW,PCOL:PRINT"FILE NAME: ";
J 2270 ROW = PROW:COL = PCOL + 12
Y 2280 MIN.CHAR$ = "!":MAX.CHAR$ = "Z":COL
       OR YELLOW
C 2290 DEF SEG = &H40:STATUS = PEEK(&H17):
       POKE (&H17), STATUS OR 64 'SET CAPS
T 2300 S$ = "":LENGTH = 20:GOSUB 1640:F$ =
        S$:POKE (&H17), STATUS
Y 2310 RETURN
P 2320 'ERASE ALL DATA
G 2330 MODE=TEXT:MESSAGE$ = " ERASE ALL DA
       TA ? ":COLOR BLACK,HWHITE:GOSUB 114
       0:GOSUB 2370
S 2340 IF (YES <> 1) THEN RETURN
N 2350 FOR I = 1 TO MAX.LINE:TEXT$(I) = ""
       :TEMPLATE$(I) = "":NEXT
C 2360 COLOR WHITE, BLACK:RETURN 310
O 2370 'GET A YES/NO RESPONSE
P 2380 LOCATE MESSAGE.ROW, MESSAGE.COL:PRI
       NT" ARE YOU SURE (Y/N) ? ";
H 2390 X$ = INKEY$:IF X$ = "" THEN GOTO 23
       90
A 2400 IF ((X$ = "y") OR (X$ = "Y")) THEN
       YES = 1 ELSE YES = 0
B 2410 PRINT X$;:FOR J = 1 TO 50:NEXT:GOSU
       B 1160
E 2420 RETURN
S 2430 'PRINT DOCUMENT
V 2440 MESSAGE$ = "PRINT ENTIRE DOCUMENT"
S 2450 COLOR BLACK,WHITE:GOSUB 1140
N 2460 S$ = MID$(STR$(SPACING),2):COLOR WH
       ITE, BLACK
N 2470 LOCATE MESSAGE.ROW, MESSAGE.COL:PRI
       NT"LINE SPACING: ";S$
Q 2480 ROW = MESSAGE.ROW:COL = MESSAGE.COL
        + 14:LENGTH = 1
```

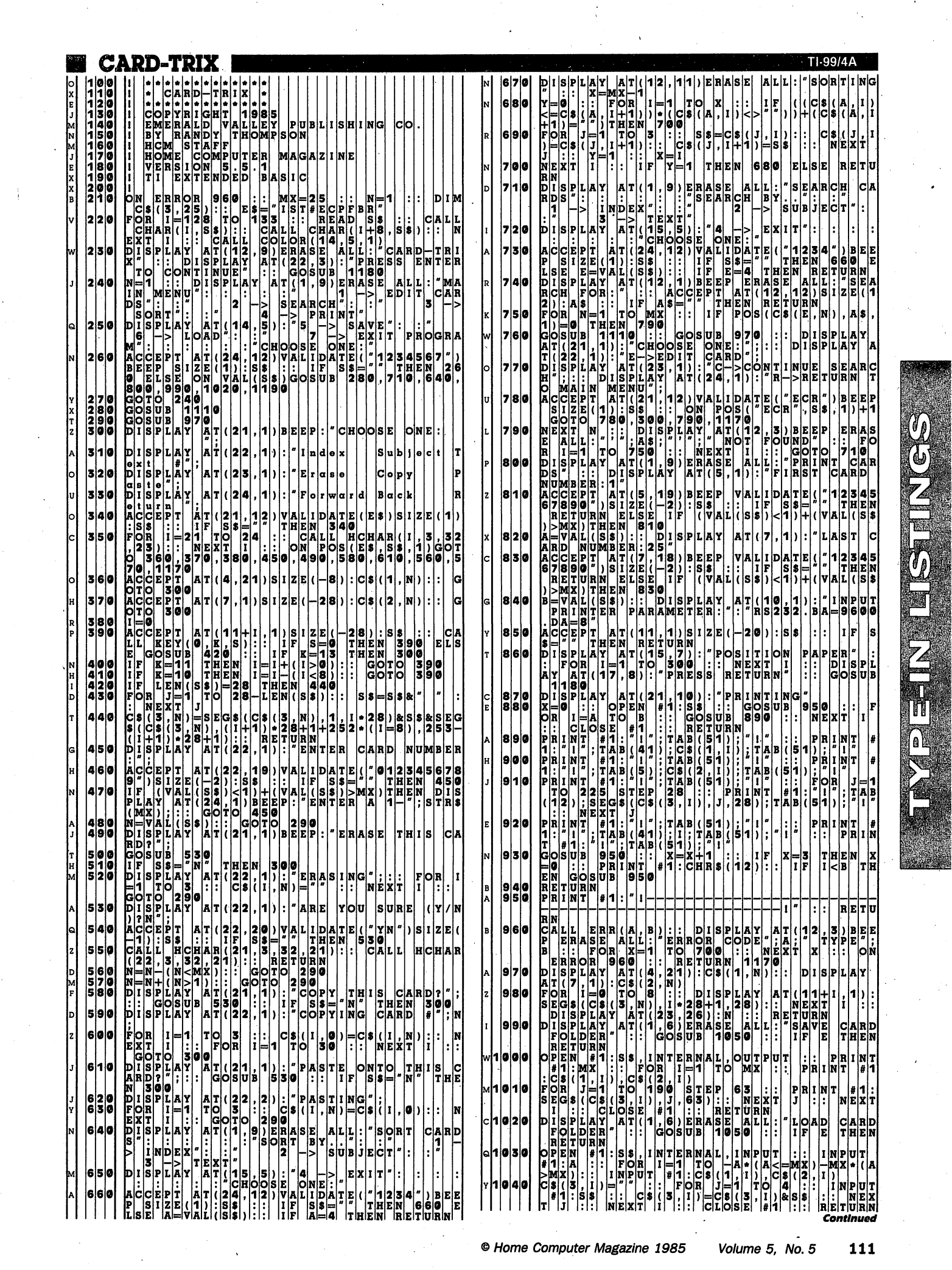
Continued

ELECTRONIC TYPEWRITER *Continued*

IBM PC/PCjr, TANDY 1000

```
DU 2490  MIN.CHAR$ = "1":MAX.CHAR$ = "9"
DU 2500  GOSUB 1640: IF S$="" THEN GOSUB 1160: RETURN
60: RET"URN: ELSE SPACING = VAL(S$)
UV 2510  S$ = "": GOSUB 1160
UV 2520  COLINT = MIN.COL: LOCATE ROW, COL: P
RINT"MESSAGE$ = "";
COLINT = MIN.COL: LOCATE ROW, COL: P
P 2530   RINT"FIRST LINE: ";: MIN.CHAR$ = "0": MAX.
COLCHAR$ = "9": LENGTH = 2
CHARINT = VAL(S$): GOSUB 1160: RETURN
DE 2540  PRINT"MESSAGE$ = "";: GOSUB 1640: IF S$="" THEN GOSUB 1160: RETURN
DE 2550  IF (E < 1) OR (E > MAX.
EH 2560  LINE = E: GOSUB 1190: GOTO 2520
C 2570   S$ = "": GOSUB 1160: COLINT = LOCATE ROW, COL: P
CG 2580  RINT"LAST LINE: ";: MIN.CHAR$ = "0": MAX.
COLCHARINT = VAL(S$): GOSUB 1640: IF S$="" THEN RETURN
O 2590   PRINT"ENTER BLACKPAPER MESSAGE: ";: GOSUB 1160: COLOR 0,
C 2600   IF (VAL(S$) < 1) OR (VAL(S$) > MAX.
CJ 2610  LINE = VAL(S$): GOSUB 1190: GOTO 2520
JB 2620  IF LINE = 0 THEN RETURN
M 2630   COLOR 0,0: CLS: LOCATE 15,1: PRINT"BLACKPAPER MESSAGE"
S 2640   LOCATE ROW, COL: PRINT"ENTER MESSAGE: ";: GOSUB 1160: COLOR
CONT"PRINT CHR$(17);: GOSUB 1160: COLOR 0,0: CLS: LOCATE ROW, COL: P
RESUME NEXT: CHR$(217);" TO
TR 2650  CURSOR = CUR.LINE: CURSOR = CURSOR.LINE + LINE
TR 2660  SOR = CURSOR.LINE: TO VAL(S$): GOSUB 3
570 E.CODE$ = MID$(TEMPLATE$(CUR.LIN
J 2670   E),1,1)+LIFE.CODE$+"THEN GOSUB 740: ELSE
Z 2680   IF LIFE.CODE$ = "" THEN GOSUB 740: ELSE
SE 760: GOSUB 770: ELSE IF LINE.CODE$ = "THEN" THE
UB GOSUB 760
SI 2690  FOR I = 1 TO SPACING: LPRINT: NEXT I
SI 2700  CURR.LINE = CURR.LINE + 1
SI 2710  NEXTLINE: CUR.LINE = S: GOSUB 3570
SI 2720  RETURN
NU 2730  'LOADPTS = "LOAD TEXT FILE": GOSUB 22
UF 2740  PROMPT$ = "ERROR 21 GOTO 140": ON ERROR
50: PEEK(30) + 256 * PEEK(31): POKE 30, X
OP 2750  INPUT"#1, JTM, FSTM, RIGHTM
JA 2760  FOR I = 1 TO MAX.LINE
FO 2770  INPUT #1, S$: IF S$ = EMPTY.LINE$ THEN
EQ 2790  WHILE (QT > 0): QT = INSTR(QT
Q 2790   $,CHR$(34)): MID$(S$,
QW 2800  'REPLACE QUOTATION MARK
WE 2810  S.T$(I) = S$
TE 2820  EXT$(J) = CLOSE: CURR.LINE = 1
IG 2830  FOR J = 1 TO 3050: RETURN
G 2840   'SAVE WITH TEMPLATE (Y/N)
OS 2850  SAVE TEXT
CL 2860  FILE$: GOSUB 3570: RETURN
R 2870   'SAVE TEXT F
LO 2880  ILE$: GOSUB 3570: RETURN
Y 2890   'LOAD TEXT F
TS 2900  ILE$: GOSUB 3570: RETURN
RO 2910  'ENTER ROW
R 2910   ?X$ = "": INKEY$: IF X$ = "" THEN GOTO 29
Z 2920   IF (X$ = "Y") OR (X$ = "y")) THEN
IJ 2930  GOSUB 2230: ON ERROR GOTO 2140
EO 2940  PRINT#1, FSTM, RIGHTM
CN 2950  FOR I = 1 TO MAX.LINE: S$ = TEXT$(I)
NX 2970  IF S$ = EMPTY.LINE$ THEN
J 2980   TQ = (QT > 0): QT = INSTR(QT,
H 2990   S$,CHR$(34)): 'REPLACE QUOTATION MARK
WE 3000  WITH NON-VOLATILE CHARACTERS
AG 3010  WRITE#1, S$: NEXT: CLOSE
RG 3020  IF SUB 3130: RETURN
GN 3030  GOSUB 3260: GOTO 3570: RETURN
NC 3040  'LOAD TEMPLATE: GOSUB 223
Z 3050   PROMPT$ = "ERROR GOTO 2140": ON ERROR
QI 3060  OPEN"#1, 1, STSTRING$(80,223)
QT 3070  INPUT #1, STS$: IF S$ = EMPTY.LINE$ THEN
TD 3080  FINTEMPLATE$(I) = S$: NEXT: CLOSE
NK 3100  GOSUB 3260: GOTO 3570: RETURN
PA 3110  'SAVE TEMPLATE: GOSUB 223
PR 3120  PROMPT$ = "ERROR GOTO 2140": ON ERROR
EI 3130  OPEN"#1, 1, STRING$(80,223)
EI 3140  FOR I = 1 TO MAX.LINE
MJ 3150  PRINT#1, TEMPLATE$(I): IF S$ = "" THEN S
MJ 3160  EMPTY.LINE$
A 3170   WRITE #1, S$: NEXT: CLOSE
M 3180   GOSUB 3570: RETURN
G 3190   'EXIT PROGRAM
E 3200   MESSAGE$ = "EXIT PROGRAM ?": GOSUB 1
120: GOSUB 2370
P 3210   IF (YES = 0) THEN RETURN
D 3220   CLS: COLOR MAGENTA, GREEN: T$ = "BYE
END
C 3230   'FOR T = 1 TO LEN(T$): PRINT MID$((T$
), T, 1);: GOSUB 1080: NEXT: RETURN
I 3240   FOR J = 1 TO RND * 200 + 35: NEXT J:
NEXT T: GOSUB 1100
R 3250   END
G 3260   'DISPLAY EDIT SCREEN
F 3270   COLOR , BLACK: CLS: WIDTH 80: ROW = LNU
M.ROW: COL = LNUM.COL - 18: WDTH
= 17: HEIGHT = 3: COLOR WHITE: GOSUB 1
520
F 3280   COLOR "RED": LOCATE ROW - 1, COL + 4: P
RINT"F1";"BLUE: LOCATE ROW + 1, COL + 2
P 3290   COLOR "BLUE": LOCATE ROW + 1, COL + 2
: PRINT"LINE NUMBER: ";
O 3300   COL = LNUM.COL + 2: WDTH = 7: COLOR W
HITE: GOSUB 1520: LOCATE ROW, COL: PRINT
'INT CHR$(203);: LOCATE ROW + 2, COL: PR
INT CHR$(202)
I 3310   'COLOR WHITE: ROW = CODE.ROW - 1: COL
A 3320   = CODE.COL - 15: WDTH = 14: GOSUB 152
0
G 3330   COLOR "RED": LOCATE ROW - 1, COL + 4: P
RINT"F2";"BLUE: LOCATE ROW + 1, COL + 2
L 3340   COLOR "WHITE: COL = CODE.COL - 2: WDTH
A 3350   = 8: GOSUB 1520: LOCATE ROW, COL: PRI
NT CHR$(203);: LOCATE ROW + 2, COL: P
RINT CHR$(202);
P 3360   COLOR "RED": LOCATE TEXT.ROW - 3, TEXT.
X 3370   COLOR + 4: PRINT"F3";
COLOR + 4: YELLOW
Z 3380   COLORTEXT.ROW - 2, TEXT.COL: PRINT
W 3390   STRING$(80,223);: LOCATE TEXT.ROW +
2, TEXT.COL: PRINT STRING$(80,223)
C 3400   'COLOR WHITE: ROW = MENU.ROW - 1: COL =
R 3410   MENU.COL - 20: HEIGHT = 3: WDTH = 18
: GOSUB 1520: COL = COL + 17: WDTH = 8
N 3420   ROW = MENU.ROW + 1: COL = MENU.COL -
20: WDTH = 25: HEIGHT = 8: GOSUB 1520
Y 3430   : COLOR 4
L COLATE ROW + 1, COL: PRINT "1) PRINT
DOCUMENT";
U 3440   LOCATE ROW + 2, COL: PRINT "2) LOAD
LOCATE FILE";
E 3450   TEXTATE ROW + 3, COL: PRINT "3) SAVE
TEXTATE FILE";
R 3460   TEXTATE ROW + 4, COL: PRINT "4) LOAD
TEMPLATE";
L 3470   TEMPLATE ROW + 5, COL: PRINT "5) SAVE
TEMPLATE";
T 3480   LOCATE ROW + 6, COL: PRINT "6) EXIT
PROGRAM";
T 3490   PLOCATE MENU.ROW - 1, MENU.COL - 20:
PRINT CHR$(201);: LOCATE MENU.ROW +
1, MENU.COL - 20: PRINT CHR$(204);: L
OCATE MENU.ROW + 1, MENU.COL - 3: PR
INT CHR$(202);
O 3500   LOCATE MENU.ROW - 1, MENU.COL - 3: P
RINT CHR$(203);: LOCATE MENU.ROW + 1
, MENU.COL + 4: PRINT CHR$(185);
E 3510   COL = 15: PRINT"MENU";
S 3520   COLOR - 2, MENU.ROW - 2, MENU
M 3530   .COLOR - 15: PRINT"F4";: STATUS.ROW - 1: C
OL = 2: WDTH = 30: HEIGHT =
T 3540   3: GOSUB 1520: STATUS.ROW - 2, PS
X 3550   ROW = MESSAGE.ROW - 1: COL = MESSAGE
.COL - 2: WDTH = MESSAGE.LENGTH + 3:
GOSUB 1520
I 3560   RETURN
RZ 3570  'UPDATE SCREEN
Z 3580   COLOR 1: PRINT: LOCATE .CURR.LINE;
G 3590   LOCATE.CODE.COL: PRINT"
LOCATE;: LOCATE.CODE.ROW, COL: PRINT
TEMPLATE$(CURR.LINE);
X 3600   COLOR WHITE, BLACK: LOCATE TEXT.ROW,
TEXT.COL: PRINT STRING$(LINE.LENGTH,
F 3610   COLOR GREEN: LOCATE TEXT.ROW, LEFTM+
V 3620   1: PRINT"INSERT";: IF (PRINT"TER.STATUS = "OFF") THEN COL
OR WHITE, BLACK: S$ = ": ELSE COLOR BLACK, WHITE
: S$ = "INTER
ON
G 3630   LOCATE PSTATUS.ROW, PSTATUS.COL: PRI
NT S$;: COLOR WHITE, BLACK
V 3640   RETURN
```

HCM



Continued

CARD-TRIX *Continued*

TI-99/4A

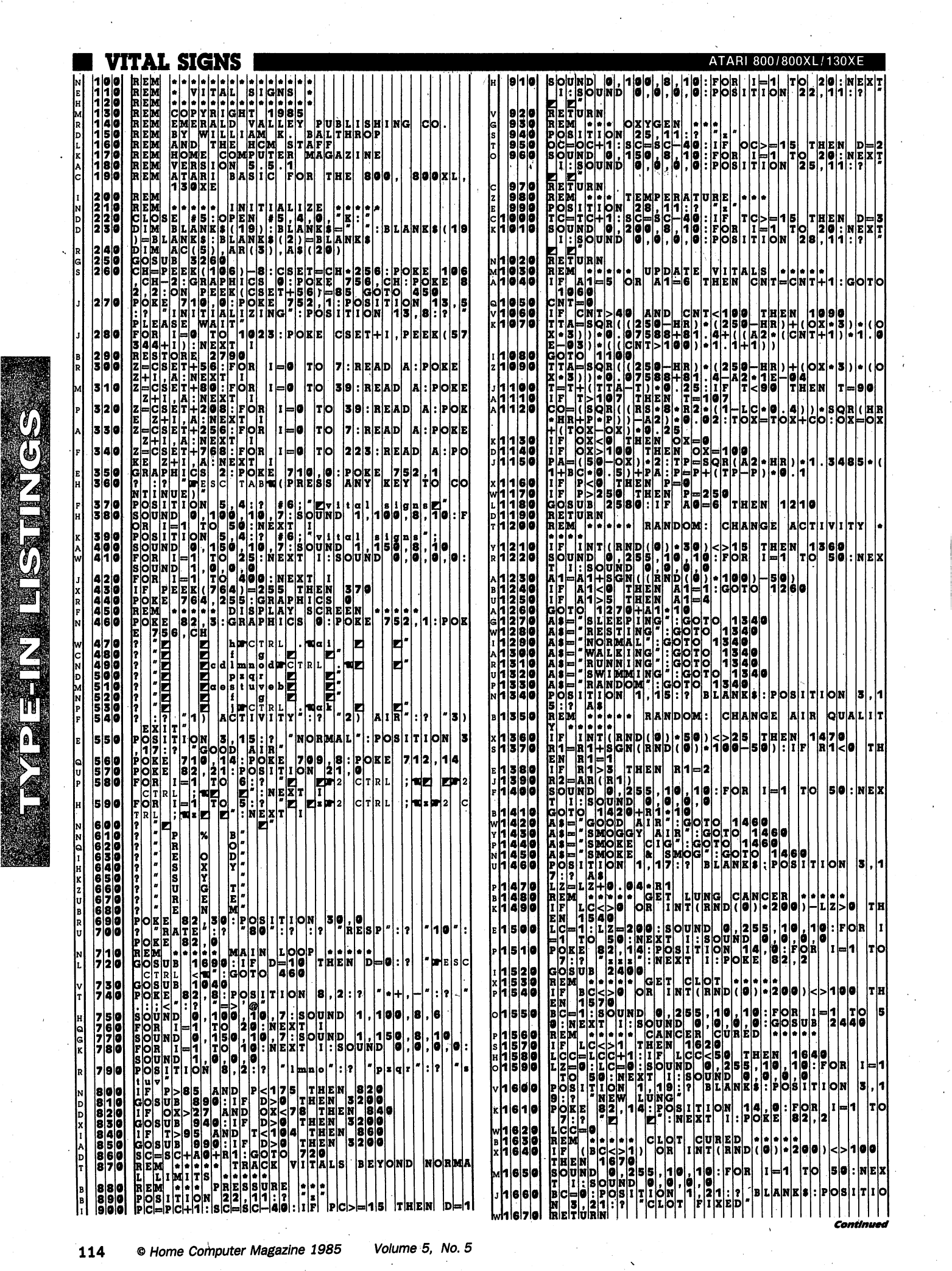
```
100 ::  CALL HCHAR(X,Y+1,32-8*(I=3),B-Y-1)
110 ::  CALL HCHAR(X+1,Y,32-8*(I=3),A-X-1)
120 ::  CALL VCHAR(X+1,Y+1,32-8*(I=3),B-Y-1)
130 ::  CALL HCHAR(X+1,Y+1,32-8*(I=3),A-X-1)
140 ::  CALL KEY(0,K,S)::IF S=0 THEN 140
150 ::  IF K<49 OR K>51 THEN 140
160 ::  CALL SOUND(100,330-110*(K=51),0)
170 ::  ON K-48 GOSUB 180,190,200
180 ::  CALL HCHAR(X,Y,32)::X=X-1::GOTO 210
190 ::  CALL HCHAR(X,Y,32)::Y=Y+1::GOTO 210
200 ::  CALL HCHAR(X,Y,32)::X=X+1
210 ::  CALL HCHAR(X,Y,42)::GOTO 140
```

VITAL SIGNS

APPLE // Family

```
100 REM
110 REM
120 REM
130 REM
140 REM
150 REM
160 REM
170 REM
180 REM
190 REM
200 TEXT : HOME : VTAB 10: PRINT "
210 PRINT "
220 PRINT "
230 PRINT "
240 PRINT "
250 X = 20: Y = 10: A = 1: B = 1
260 VTAB 10: HTAB 1: PRINT "
270 IF A$ = "" THEN A$ = "
280 IF A$ = "" THEN A$ = "
290 IF A$ = "" THEN A$ = "
300 IF A$ = "" THEN A$ = "
310 IF A$ = "" THEN A$ = "
320 IF A$ = "" THEN A$ = "
330 IF A$ = "" THEN A$ = "
340 IF A$ = "" THEN A$ = "
350 IF A$ = "" THEN A$ = "
360 IF A$ = "" THEN A$ = "
370 IF A$ = "" THEN A$ = "
380 IF A$ = "" THEN A$ = "
390 IF A$ = "" THEN A$ = "
400 IF A$ = "" THEN A$ = "
410 IF A$ = "" THEN A$ = "
420 IF A$ = "" THEN A$ = "
430 IF A$ = "" THEN A$ = "
440 IF A$ = "" THEN A$ = "
450 IF A$ = "" THEN A$ = "
460 IF A$ = "" THEN A$ = "
470 IF A$ = "" THEN A$ = "
480 IF A$ = "" THEN A$ = "
490 IF A$ = "" THEN A$ = "
500 IF A$ = "" THEN A$ = "
510 IF A$ = "" THEN A$ = "
520 IF A$ = "" THEN A$ = "
530 IF A$ = "" THEN A$ = "
540 IF A$ = "" THEN A$ = "
550 IF A$ = "" THEN A$ = "
560 IF A$ = "" THEN A$ = "
570 IF A$ = "" THEN A$ = "
580 IF A$ = "" THEN A$ = "
590 IF A$ = "" THEN A$ = "
600 IF A$ = "" THEN A$ = "
610 IF A$ = "" THEN A$ = "
620 IF A$ = "" THEN A$ = "
630 IF A$ = "" THEN A$ = "
640 IF A$ = "" THEN A$ = "
650 IF A$ = "" THEN A$ = "
660 IF A$ = "" THEN A$ = "
670 IF A$ = "" THEN A$ = "
680 IF A$ = "" THEN A$ = "
690 IF A$ = "" THEN A$ = "
700 IF A$ = "" THEN A$ = "
710 IF A$ = "" THEN A$ = "
720 IF A$ = "" THEN A$ = "
```

Continued



Continued

```
K 1680 REM ***** KEY IN & PROCESS CHANGE *****
C 1690 IF PEEK(764)=255 OR PEEK(764)=39 OR PEEK(764)=60 THEN 1780
J 1700 GET #5,K:SOUND 0,95,10,10:FOR I=1 TO 10:NEXT I:SOUND 0,0,0,0
J 1710 IF K>=49 AND K<=51 THEN 1940
O 1720 IF K=69 THEN 1790
A 1730 IF K=83 THEN 1810
A 1740 IF K=68 THEN 1830
D 1750 IF K=88 THEN 1850
D 1760 IF K=65 THEN 1870
A 1770 IF K=70 THEN 1900
Y 1780 RETURN
Y 1790 IF RS<>30 THEN RS=RS+1:GOSUB 2510
B 1800 RETURN
M 1810 IF HR<>0 THEN HR=HR-1:GOSUB 2490
J 1820 RETURN
Z 1830 IF HR<>250 THEN HR=HR+1:GOSUB 2490
J 1840 RETURN
M 1850 IF RS<>0 THEN RS=RS-1:GOSUB 2510
B 1860 RETURN
T 1870 IF HR>4 THEN HR=HR-5:GOSUB 2490:GOTO 1890
T 1880 HR=0:GOSUB 2490
A 1890 RETURN
N 1900 IF HR<246 THEN HR=HR+5:GOSUB 2490:GOTO 1920
M 1910 HR=250:GOSUB 2490
Y 1920 RETURN
W 1930 REM ***** CHANGE ACTIVITY *****
H 1940 GOSUB 2540:ON K-48 GOTO 1950,2040,2100
J 1950 POSITION 2,8:? "◩ ◩":POKE 82,3
P 1960 POSITION 3,15:? "A) SLEEPING":? "B) RESTING":? "C) NORMAL":? "D) WALKING"
L 1970 ? "E) RUNNING":? "F) SWIMMING":? "G) RANDOM":POKE 82,2
U 1980 GOSUB 2750:IF K<65 OR K>71 THEN 1980
X 1990 A0=K-65:A1=A0:IF A0>=6 THEN A1=2
J 2000 A2=AC(A1)
G 2010 GOSUB 2540:GOSUB 2240:POSITION 2,8:? "  "
O 2020 RETURN
V 2030 REM ***** CHANGE AIR *****
N 2040 POSITION 2,9:? "◩ ◩":POKE 82,3
K 2050 POSITION 3,15:? "A) GOOD AIR":? "B) SMOGGY AIR":? "C) SMOKE CIG":? "D) SMOKE & SMOG":POKE 82,2
G 2060 GOSUB 2750:IF K<65 OR K>68 THEN 2060
E 2070 R1=K-65:R2=AR(R1)
A 2080 GOSUB 2540:GOSUB 2240:POSITION 2,9:? "  ":RETURN
P 2090 REM ***** END OF GAME *****
A 2100 POSITION 2,10:? "◩ ◩"
B 2110 POSITION 1,15:? "ARE YOU SURE? (Y/N)"
E 2120 GOSUB 2750:IF K=89 THEN 2150
D 2130 IF K<>78 THEN 2120
A 2140 POSITION 2,10:? "   ":GOSUB 2240:RETURN
A 2150 ? "ESC CTRL <"
I 2160 POKE 756,224:POSITION 10,10:? "FINAL SCORE:";SC:POSITION 10,12:? "PLAY AGAIN? (Y/N)";
N 2170 GOSUB 2750:IF K<>89 THEN 2210
D 2180 GOSUB 3260:IF D=0 THEN 2200
P 2190 POKE 106,PEEK(740):CLR :GOTO 220
K 2200 D=10:RETURN
D 2210 IF K<>78 THEN 2170
N 2220 ? "ESC CTRL <":POSITION 12,12:? "THANKS FOR PLAYING!":FOR I=255 TO 0 STEP -1:SOUND 0,I,10,10:NEXT I:GRAPHICS 0:END
Y 2230 REM ***** UPDATE CONDITIONS *****
Q 2240 GOTO 2250+A1*10
A 2250 A$="SLEEPING":GOTO 2320
U 2260 A$="RESTING":GOTO 2320
Q 2270 A$="NORMAL":GOTO 2320
F 2280 A$="WALKING":GOTO 2320
S 2290 A$="RUNNING":GOTO 2320
S 2300 A$="SWIMMING":GOTO 2320
P 2310 A$="RANDOM":GOTO 2320
Y 2320 POSITION 1,15:? BLANK$:POSITION 3,15:? A$
D 2330 GOTO 2340+R1*10
E 2340 A$="GOOD AIR":GOTO 2380
A 2350 A$="SMOGGY AIR":GOTO 2380
B 2360 A$="SMOKE CIG":GOTO 2380
L 2370 A$="SMOKE & SMOG":GOTO 2380
M 2380 POSITION 1,17:? BLANK$:POSITION 3,17:? A$
B 2390 A2=AC(A1):GOSUB 2400:GOSUB 2440:RETURN
N 2400 IF LC<>1 THEN 2430
G 2410 POSITION 1,19:? BLANK$
U 2420 A$="LUNG CANCER":POSITION 3,19:? A$
V 2430 RETURN
F 2440 IF BC<>1 THEN 2470
X 2450 POSITION 1,21:? BLANK$
V 2460 A$="BLOOD CLOT":POSITION 3,21:? A$
X 2470 RETURN
N 2480 REM ***** DISPLAY HRATE AND RESP *****
H 2490 A$=STR$(HR):A$(LEN(A$)+1)="  "
K 2500 POSITION 30,1:? A$:RETURN
D 2510 A$=STR$(RS):A$(LEN(A$)+1)="  "
R 2520 POSITION 30,4:? A$:RETURN
Z 2530 REM ***** CLEAR MESSAGE AREA *****
E 2540 FOR Z=0 TO 6
Y 2550 POSITION 1,15+Z:? BLANK$
A 2560 NEXT Z:RETURN
M 2570 REM ***** DISPLAY GRAPH *****
O 2580 PB=INT(P-75):IF PB<0 THEN PB=0
R 2590 IF PB>99 THEN PB=99
U 2600 POSITION 22,10-INT(PB*0.1)-1:? "◩ ◩";
H 2610 POSITION 22,10-INT(PB*0.1):A=INT(118+(PB-INT(PB*0.1)*10)*0.445):A=A+42*(A=118):? CHR$(A);
N 2620 IF PB>9 THEN POSITION 22,10-INT(PB*0.1)+1:? "z";
T 2630 OB=INT(OX-25)*2:IF OB<0 THEN OB=0
P 2640 IF OB>99 THEN OB=99
N 2650 POSITION 25,10-INT(OB*0.1)-1:? "◩ ◩";
W 2660 POSITION 25,10-INT(OB*0.1):A=INT(118+(OB-INT(OB*0.1)*10)*0.445):A=A+42*(A=118):? CHR$(A);
D 2670 IF OB>9 THEN POSITION 25,10-INT(OB*0.1)+1:? "z";
S 2680 T1=INT((T-94)*11.11):IF T1<0 THEN T1=0
C 2690 IF T1>99 THEN T1=99
A 2700 POSITION 28,10-INT(T1*0.1)-1:? "◩ ◩";
B 2710 POSITION 28,10-INT(T1*0.1):A=INT(118+(T1-INT(T1*0.1)*10)*0.445):A=A+42*(A=118):? CHR$(A);
C 2720 IF T1>9 THEN POSITION 28,10-INT(T1*0.1)+1:? "z";
W 2730 RETURN
D 2740 REM ***** SINGLE KEY IN *****
Z 2750 GET #5,K
T 2760 SOUND 0,100,10,10:FOR I=1 TO 15:NEXT I:SOUND 0,0,0,0
Y 2770 RETURN
I 2780 REM ***** GR CHAR DATA *****
S 2790 DATA 85,85,20,20,21,5,1,1
W 2800 DATA 40,40,40,170,170,40,40,40
T 2810 DATA 0,0,0,0,128,160,32,21
F 2820 DATA 0,0,0,0,0,0,0,1
N 2830 DATA 20,20,20,22,86,90,88,104
A 2840 DATA 40,40,37,21,21,21,1,0
T 2850 DATA 85,85,85,85,85,85,85,85
Z 2860 DATA 65,85,85,84,85,85,85,85
R 2870 DATA 104,40,160,160,160,64,64,64
F 2880 DATA 0,0,1,85,85,0,0,2
U 2890 DATA 85,85,82,66,10,40,160,128
R 2900 DATA 64,80,84,21,5,64,64,64
I 2910 DATA 0,2,8,162,136,34,8,2
K 2920 DATA 64,16,68,17,69,16,64,0
H 2930 DATA 0,1,4,81,68,17,4,1
Y 2940 DATA 128,32,136,34,138,32,128,0
Y 2950 DATA 0,0,0,170,170,0,0,0
L 2960 DATA 0,0,0,85,85,0,0,0
A 2970 DATA 40,40,40,40,40,40,40,40
T 2980 DATA 20,20,20,20,20,20,20,20
J 2990 DATA 0,0,40,170,162,160,168,40
C 3000 DATA 0,0,20,84,69,5,5,20
D 3010 DATA 40,42,42,10,2,0,0,0
F 3020 DATA 20,20,84,80,64,0,0,0
P 3030 DATA 40,40,41,170,169,41,57,133
G 3040 DATA 0,0,85,170,90,85,85,85
Z 3050 DATA 0,0,1,65,149,85,85,85
X 3060 DATA 20,84,80,66,74,104,168,160
E 3070 DATA 85,85,85,85,85,85,21,21
Y 3080 DATA 85,86,86,86,85,85,85,85
A 3090 DATA 160,144,144,148,84,84,84,84
S 3100 DATA 21,21,21,85,85,41,40,40
B 3110 DATA 85,85,85,85,85,85,0,0
H 3120 DATA 85,85,85,85,85,85,85,0
L 3130 DATA 84,84,84,85,85,85,85,20
U 3140 DATA 255,255,255,255,255,255,85,85
R 3150 DATA 255,255,255,255,85,85,85,85
G 3160 DATA 255,255,85,85,85,85,85,85
N 3170 DATA 85,85,85,85,85,85,85,85
J 3180 DATA 255,255,255,255,255,255,255,85
R 3190 REM ***** END OF GAME MESSAGES *****
Q 3200 POKE 82,2:POKE 756,224:? "ESC CTRL <":ON D GOTO 3210,3220,3230
Z 3210 ? "BAD BLOOD PRESSURE":GOTO 3240
D 3220 ? "YOU PASSED OUT FROM EITHER TOO MUCH,":? "OR NOT ENOUGH,":? "OXYGEN IN YOUR BLOOD.":GOTO 2160
N 3230 ? "YOUR BODY CAN'T HANDLE":? "EXTREME TEMPERATURES FOR":? "VERY LONG."
C 3240 ? :? "YOU WILL NEED AN EXTENDED STAY IN THE LOCAL HOSPITAL.":GOTO 2160
W 3250 REM ***** INIT VARIABLES FOR START *****
T 3260 RESTORE 3300
L 3270 READ BC,LC,P,OX,TOX,T,HR,RS,A0,A1,A2,R1,R2,SC,OD
V 3280 FOR I=0 TO 5:READ Z:AC(I)=Z:NEXT I
V 3290 FOR I=0 TO 3:READ Z:AR(I)=Z:NEXT I
M 3300 DATA 0,0,125,50,50,98.6,80,10,2,2,107.4,0,1,0,50
N 3310 DATA 69.5,91.7,107.414,121.6,152,171.9,1,.85,.7,.5
Q 3320 RETURN
```

HCM

TYPE-IN LISTINGS

```
N  100 REM *****************
E  110 REM * VITAL SIGNS *
H  120 REM *****************
M  130 REM COPYRIGHT 1985
X  140 REM EMERALD VALLEY PUBLISHING CO
D  150 REM BY WILLIAM K. BALTHROP
L  160 REM AND THE HCM STAFF
K  170 REM HOME COMPUTER MAGAZINE
A  180 REM VERSION 5.5.1
B  190 REM COMMODORE 64 BASIC
M  200 REM
H  210 FORZ=54272TO54296:POKEZ,0:NEXT:POKE54295,4:POKE53281,2:POKE53280,2
G  220 PRINT"{SHIFT CLR}{CTRL WHT}{7 CRSR DOWN}{14 CRSR RIGHT}VITAL SIGNS"
R  230 PRINT"{3 CRSR DOWN}{10 CRSR RIGHT}PLEASE WAIT A MOMENT"
Z  240 GOSUB2270:PRINT"{11 CRSR DOWN}{8 CRSR RIGHT}PRESS {CTRL RVS ON}RETURN{CTRL RVS OFF} TO START"
U  250 GOSUB2140:IFK$<>CHR$(13)THEN250
Y  260 DIMAC(5),AC$(6),AR(3),AR$(3):A0=1
Q  270 CLR:A0=2:RESTORE:GOSUB2240
Z  280 PRINT"{SHIFT CLR}":BL$="{CTRL WHT}{CMDR CYN}"
G  290 POKE53281,1:POKE53280,1
Y  300 READQ1,Q2,Q3:IFQ1=-1THEN320
V  310 POKEQ1,Q2:POKEQ1+CL,Q3:GOTO300
W  320 D=0:PRINTHM$"{CMDR CYN}{3 CRSR RIGHT}1) ACTIVITY":PRINT"{3 CRSR RIGHT}2) AIR":PRINT"{3 CRSR RIGHT}3) EXIT"
N  330 READQ1:IFQ1=-1THEN350
E  340 POKEQ1,107:POKEQ1+CL,15:POKEQ1+1,107:POKEQ1+1+CL,15:GOTO330
D  350 Q1=1045:FORZ=0TO8:POKEQ1+Z,121:POKEQ1+Z+CL,11:NEXTZ
V  360 FORQ1=1085TO1525STEP40:POKEQ1,117:POKEQ1+CL,11:NEXTQ1
F  370 Q1=1565:FORZ=0TO8:POKEQ1+Z,120:POKEQ1+Z+CL,11:NEXTZ
S  380 FORQ1=1093TO1533STEP40:POKEQ1,118:POKEQ1+CL,11:NEXTQ1:Q1=1127
C  390 POKEQ1,100:POKEQ1+CL,2:POKEQ1+1,100:POKEQ1+1+CL,2
A  400 POKEQ1+3,100:POKEQ1+3+CL,2:POKEQ1+4,100:POKEQ1+4+CL,2
F  410 Q1=Q1+40:IFQ1<1500THEN390
N  420 Q1=1326
L  430 POKEQ1,107:POKEQ1+CL,2:POKEQ1+3,107:POKEQ1+3+CL,2
T  440 POKEQ1+6,107:POKEQ1+6+CL,2:Q1=Q1+40:IFQ1<1500THEN430
Y  450 READQ1,Q2:IFQ1=-1THEN470
H  460 POKEQ1,Q2:POKEQ1+CL,11:GOTO450
N  470 PRINT"{HOME}{CRSR DOWN}";TAB(31);"RATE:":PRINTTAB(32);"80"
K  480 PRINT"{CRSR DOWN}";TAB(31);"RESP:":PRINTTAB(32);"10"
Z  490 GOSUB1810
N  500 GOSUB1280:IFD=10THEN270
X  510 GOSUB750:W=2:GOSUB1510
B  520 M1=100:M2=15:GOSUB2640
U  530 W=10:GOSUB1510
B  540 M1=200:M2=12:GOSUB2640
B  550 IFP>80ANDP<170THEN570
C  560 GOSUB620:IFD>0THEN2160
Q  570 IFOX>25ANDOX<75THEN590
H  580 GOSUB670:IFD>0THEN2160
P  590 IFT>94ANDT<103THEN610
F  600 GOSUB710:IFD>0THEN2160
S  610 SC=SC+A0+R1:GOTO500
E  620 POKE1526,107:POKE1526+CL,2
A  630 PC=PC+1:SC=SC-40:IFPC<15THEN650
A  640 D=1
V  650 GOSUB2660
X  660 POKE1526,99:POKE1526+CL,12:RETURN
X  670 POKE1529,107:POKE1529+CL,2:OC=OC+1:SC=SC-40:IFOC<15THEN690
R  680 D=2
P  690 GOSUB2660
W  700 POKE1529,99:POKE1529+CL,12:RETURN
C  710 POKE1532,107:POKE1532+CL,2:TC=TC+1:SC=SC-40:IFTC<15THEN730
B  720 D=3
Q  730 GOSUB2660
S  740 POKE1532,99:POKE1532+CL,12:RETURN
A  750 IFA1=5ORA1=6THENCNT=CNT+1:GOTO770
T  760 CNT=0
A  770 IFCNT>40ANDCNT<100 THEN800
N  780 F=SQR((250-HR)↑2+(OX*3)↑2)*.0759+81.3+((A2*(CNT+1)*.001)*((CNT>100)*1.1+1))
I  790 GOTO810
Y  800 F=SQR((250-HR)↑2+(OX*3)↑2)*.07589+81.3-A2*.0001
D  810 T=T+(F-T)*.25:IFT>=90THEN830
O  820 T=90
J  830 IFT<=107THEN850
T  840 T=107
K  850 CO=(SQR((RS*8*R2*(1-LC*.4))*SQR(HR↑2+P↑2))-A2)*.02
A  860 TX=TX+CO:OX=OX+(TX-OX)*.25:IFOX>=0THEN880
F  870 OX=0
E  880 IFOX<=100THEN900
W  890 OX=100
X  900 PA=(50-OX)*2
F  910 TP=SQR(A2*HR)*1.3485*(1+BC*.5)+PA
X  920 P=P+(TP-P)*.1:IFP<0THENP=0
P  930 IFP>250THENP=250
F  940 GOSUB1930:IFA0=6THEN960
S  950 RETURN
F  960 IFINT(RND(1)*30)<>15THEN1020
U  970 GOSUB2680
B  980 A1=A1+SGN(RND(1)*100-50):IFA1>=0THEN1000
F  990 A1=1:GOTO1010
Z 1000 IFA1>5THENA1=4
G 1010 Y=17:X=0:A$=BL$:GOSUB2130:X=3:A$=AC$(A1):GOSUB2130
Q 1020 IFINT(RND(1)*50)<>25THEN1080
A 1030 R1=R1+SGN(RND(1)*100-50):IFR1<0THENR1=1
H 1040 IFR1>3THENR1=2
O 1050 R2=AR(R1):Y=19:X=0:A$=BL$:GOSUB2130:X=3:A$=AR$(R1)
N 1060 GOSUB2680
S 1070 GOSUB2130
A 1080 LZ=LZ+.04*R1
Q 1090 IF(LC<>0)OR(RND(1)*(200-LZ)>0)THEN1140
M 1100 GOSUB2660
W 1110 LC=1:FORZ=1195TO1315STEP40:POKEZ,107:POKEZ+CL,2:NEXTZ
Z 1120 LZ=200:FORZ=1196TO1316STEP40:POKEZ,107:POKEZ+CL,2:NEXTZ
C 1130 GOSUB1840
I 1140 IF(BC<>0)OR(INT(RND(1)*200)<>100)THEN1170
V 1150 GOSUB2660
T 1160 BC=1:GOSUB1870
W 1170 IFLC<>1THEN1240
W 1180 L2=L2+1:IFL2<50THEN1250
W 1190 LZ=0
T 1200 GOSUB2660
E 1210 GOSUB2660
V 1220 FORZ=1195TO1315STEP40:POKEZ,99:POKEZ+CL,12:POKEZ+1,99:POKEZ+1+CL,12:NEXTZ
H 1230 Y=21:X=3:GOSUB2130
S 1240 L2=0
A 1250 IF(BC<>1)OR(INT(RND(1)*200)<>100)THENRETURN
D 1260 GOSUB2660
W 1270 BC=0:X=0:Y=23:A$=BL$:GOSUB2130:X=3:A$="CLOT FIXED":GOSUB2130:RETURN
W 1280 GETK$:IFK$<>""THENK=ASC(K$):POKE198,0:GOTO1300
Y 1290 RETURN
W 1300 GOSUB2680
J 1310 IFK>48ANDK<52THEN1540
C 1320 IFK=69THEN1390
A 1330 IFK=83THEN1410
S 1340 IFK=68THEN1430
N 1350 IFK=88THEN1450
V 1360 IFK=65THEN1470
G 1370 IFK=70THEN1490
W 1380 RETURN
M 1390 IFRS=30THENRETURN
X 1400 RS=RS+1:GOSUB1910:RETURN
X 1410 IFHR=0THENRETURN
I 1420 HR=HR-1:GOSUB1900:RETURN
M 1430 IFHR=250THENRETURN
I 1440 HR=HR+1:GOSUB1900:RETURN
A 1450 IFRS=0THENRETURN
J 1460 RS=RS-1:GOSUB1910:RETURN
X 1470 IFHR>4THENHR=HR-5:GOSUB1900:RETURN
K 1480 HR=0:GOSUB1900:RETURN
X 1490 IFHR<246THENHR=HR+5:GOSUB1900:RETURN
T 1500 HR=200:GOSUB1900:RETURN
Y 1510 POKE1231,73:POKE1231+CL,W:POKE1232,74:POKE1232+CL,W
N 1520 POKE1271,75:POKE1271+CL,W:POKE1272,76:POKE1272+CL,W
F 1530 POKE1311,78:POKE1311+CL,W:POKE1312,78:POKE1312+CL,W:RETURN
C 1540 GOSUB1920:ONK-48GOTO1550,1620,1670
E 1550 PRINTHM$;TAB(2);"{CMDR PUR}{CTRL RVS ON}{CTRL RVS OFF}":X=3:FORY=17TO23
Y 1560 A$=CHR$(48+Y)+")    "+AC$(Y-17):GOSUB2130:NEXTY
O 1570 GOSUB2140:IFK$<"A"ORK$>"G"THEN1570
G 1580 A0=ASC(K$)-65:A1=A0:IFA0<6ANDA0>=0THEN1600
M 1590 A1=2
I 1600 A2=AC(A1):GOSUB1920:GOSUB1810
A 1610 PRINTHM$;TAB(2);"{CTRL WHT}{CTRL RVS ON}{CTRL RVS OFF}{CMDR CYN}":RETURN
J 1620 PRINTHM$"{CRSR DOWN}{CMDR CYN}";TAB(2);"{CTRL RVS ON}{CTRL RVS OFF}":X=3
H 1630 FORY=17TO20:A$=CHR$(48+Y)+")    "+AR$(Y-17):GOSUB2130:NEXTY
D 1640 GOSUB2140:IFK$<"A"ORK$>"D"THEN1640
Q 1650 K=ASC(K$):R1=K-65:R2=AR(R1)
D 1660 GOSUB1920:GOSUB1810:PRINTHM$;"{CTRL WHT}{CRSR DOWN}";TAB(2);"{CTRL RVS ON}{CTRL RVS OFF}{CMDR CYN}":RETURN
T 1670 PRINTHM$"{2 CRSR DOWN}{CMDR CYN}";TAB(2);"{CTRL RVS ON}{CTRL RVS OFF}":X=1:Y=17:A$="EXIT NOW? (Y/N)"
A 1680 GOSUB2130:GOSUB2140:IFK$="Y"THEN1710
U 1690 IF K$<>"N" THEN1680
```

Continued

VITAL SIGNS *Continued*

COMMODORE 64

```
K 1700  PRINTHM$"■2 CRSRDOWN■ CTRL WHT■";:TA
        B(2);"■ CTRL RVSON■ CTRL RVSOFF■CM
        DR CYN■":GOSUB1810:RETURN
A 1710  PRINT"■ SHIFT CLR■CMDR PUR■5 CRSRD
        OWN■5 CRSRRIGHT■";:RETURN
O 1720  PRINT"■FINAL SCORE: ";SC
N 1730  PRINT"■6 CRSRDOWN■12 CRSRRIGHT■PLA
        Y AGAIN (Y/N)"
N 1740  GOSUB2140
O 1750  IFK$<>"Y"THEN1790
D 1760  GOSUB2130:IFD=0THEN1780
K 1770  D=10:PRINT"■SHIFT CLR■":GOTO270
W 1780  D=1:RETURN
C 1790  IFK$<>"N"THEN1740
R 1800  POKE53272,21:END
N 1810  X=1:Y=17:A$=BL$:GOSUB2130:X=3:A$=AC
        $(A1):GOSUB2130:Y=19:X=0:A$=BL$:GOSUB2130:X=3:A$=AR
        $(R1):GOSUB2130
A 1830  A2=AC(A1):GOSUB1840:GOSUB1870:RETUR
        N
Q 1840  IFLC<>1THEN1860
Y 1850  Y=21-CANCER:GOSUB2130
V 1860  RETURN
W 1870  IFBC<>1THEN1890
T 1880  Y=23-CLOT":GOSUB2130
W 1890  RETURN
H 1900  X=32:Y=2:A$=STR$(HR)+" ":GOSUB2130:
        X=32:Y=5:A$=STR$(RS)+" ":GOSUB2130:
        RETURN
H 1920  PRINTHM$;"■2 CRSRDOWN■":FORZ=1TO8:P
        RINTBL$:NEXTZ:RETURN
A 1930  PF=PF+75:IFPF<0THENPF=100
D 1940  PF=PF-100
H 1950  Q1=INT((100-PF)*.1+1):Q2=1024+22+40
        *Q1
K 1960  Q3=INT(99+INT(PF-INT(PF*.1)*10)*.89
        )
C 1970  POKEQ2,99:POKEQ2+CL,12:POKEQ2+40,Q3
        :POKEQ2+40+CL,2:IFPF<10THEN1990
K 1980  POKEQ2+80,107:POKEQ2+80+CL,2
K 1990  POKEQ2-25)*2:IFOF<0THENOF=0
J 2000  IFOF>100THENOF=100
T 2010  Q1=INT((100-OF)*.1+1):Q2=1024+25+40
        *Q1
I 2020  Q3=INT(99+INT(OF-INT(OF*.1)*10)*.89
        )
M 2030  POKEQ2,99:POKEQ2+CL,12:POKEQ2+40,Q3
        :POKEQ2+40+CL,2
A 2040  IFOF<10THEN2060
J 2050  POKEQ2+80,107:POKEQ2+80+CL,2
M 2060  T1=INT((T-94)*11.11):IFT1<0THENT1=0
W 2070  IFT1>100THENT1=100
V 2080  Q1=INT((100-T1)*.1+1):Q2=1024+28+40
        *Q1
K 2090  Q3=INT(99+(T1-INT(T1*.1-.01)*10)*.8
        9)
S 2100  POKEQ2,99:POKEQ2+CL,12:POKEQ2+40,Q3
        :POKEQ2+40+CL,2
S 2110  IFT1<10THENRETURN
V 2120  POKEQ2+80,107:POKEQ2+80+CL,2:RETURN
T 2130  DW$=■HOME■":FORZ=1TOY:DW$=DW$+"■CR
        SRDOWN■":NEXTZ:PRINTDW$;TAB(X);A$:R
        ETURN
V 2140  GETK$:IFK$=""THEN2140
V 2150  K=ASC(K$):RETURN
V 2160  PRINT"■SHIFT CLR■CMDR CYN■":ONDGOT
        O2170,2180,2200
U 2170  PRINT"■3 CRSRDOWN■11 CRSRRIGHT■BAD
        PRINT■3 CRSRDOWN■5 CRSRRIGHT■YOU
K 2180  PRINT"■3 CRSRDOWN■5 CRSRRIGHT■YOUR
        BLOOD.":GOTO2220
K 2200  PRINT"■3 CRSRDOWN■5 CRSRRIGHT■YOU
        PASSED OUT FROM EITHER TOO■ HIGH
        PRINT"■5 CRSRRIGHT■MUCH, OR NOT ENO
        UGH OXYGEN.":PRINT"■5 CRSRRIGHT■Y
        OUR BLOOD.":GOTO1720
T 2220  PRINT"■3 CRSRDOWN■5 CRSRRIGHT■YOUR
        BODY CAN'T HANDLE EXTREME■ HEAT■ TEMPERATURES FOR
        VERY LONG."
K 2240  PRINT"■2 CRSRDOWN■5 CRSRRIGHT■YOU
        WILL NEED AN EXTENDED STAY■ IN THE LOCAL HOS
        PITAL.":GOTO1720
Q 2240  READAC(A),AC(B),AC(C),AC(D),AC(E),
        AC(F),AC(G),AC(H),AC(I),AC(J),AC(K),
        AC(L),AC(M),AC(N),AR(0),AR(1),AR(2),
        AC$(0),AC$(1),AC$(2),AC$(3),AC$(4),
```

```
R 2260  READAC$(5),AC$(6),AR$(0),AR$(1),AR$
        (2),AR$(3),CL,HM$:RETURN
U 2270  POKE56334,(PEEK(56334)AND254):POKE1
        ,(PEEK(1)AND251)
R 2280  FORZ=0TO2047:POKE12288+Z,PEEK(53248
        +Z):NEXTZ
Y 2290  POKE1,PEEK(1)OR4:POKE56334,PEEK(563
        34)OR1
W 2300  FORZ=1TO228:READQ1$:NEXTZ
V 2310  READQ=1:IFQ1=-1THEN2330
F 2320  FORZ=0TO7:READQ2:POKE12288+Q1*8+Z,Q
        2:NEXTZ:GOTO2310
L 2330  POKE53272,29:RETURN
R 2340  DATA 0,0,125,50,50,98,6,80,10,2,107,
        4,1,69,5,91,7,107,414,121,6,152,171
        ,9,1
M 2350  DATA .85,.7,.5,0,50,SLEEPING,RESTING
        ,NORMAL,WALKING,RUNNING,SWIMMING,RA
        NDOM
N 2360  DATA GOOD AIR,SMOGGY AIR,SMOKE CIG.,
        SMOKE & SMOG,54272,"■HOME■13 CRSRD
        OWN■"
K 2370  DATA 107,90,14,107,92,10,1110,87,1
        4,1111,91,14,1111,94,10,1113,88,10,
        1150
P 2380  DATA 87,14,1153,87,10,1189,92,14,11
        90,83,14,1191,81,14
S 2390  DATA 1192,80,10,1193,120,10,1194,90,
        14,1229,94,14,1230,97,14,1231,73,10
N 2400  DATA 1232,74,10,1233,95,14,1234,91,1
        4,1269,92,10,1271,75,10,1272,76,10,
J 2410  DATA 1274,90,10,1309,94,10,1310,120,
        10,1311,77,10,1312,78,10,1313,120,1
        0
K 2420  DATA 1314,91,10,1350,85,14,1351,86,1
        4,1352,79,10,1389,87,14,1390,120,14
D 2430  DATA 1391,92,87,10,1429,87,14,1430,90,1
        4,1431,92,10,1432,87,10,1469,98,14
O 2440  DATA 1470,91,14,1471,94,10,1472,120,
        10,-1,-1,-1
V 2450  DATA 1187,1195,1227,1235,1267,1275,1
        307,1315,-1
E 2460  DATA 1606,16,1609,37,1612,2,1646,18,
        1652,15,1686,5,1689,15,1692,4,1726,
        19
I 2470  DATA 1729,24,1732,25,1766,19,1769,25
        ,1806,21,1809,7,1812,20,1846,18,1849
        ,5
G 2480  DATA 1852,5,1886,5,1889,14,1892,13,1
        932,16,-1,-1
J 2490  DATA 1973,80,81,233,84,216,89,90,90,74
        ,1600,32,96,63,112,135,56,68
F 2500  DATA 1675,74,78,66,1130,128,128,128,4,7
N 2510  DATA 77,64,72,46,41,40,40,40,40,78,1
        ,1,1,1,93,62,20,20,20,20
S 2520  DATA 79,28,28,28,30,14,14,15,7,80,3,
        7,31,60,120,240,224,224
W 2530  DATA 81,0,128,192,192,224,240,240,11
        2,83,7,7,3,3,1,0,0,0
Y 2540  DATA 85,0,0,0,1,3,7,7,7,86,56,120,24
        0,224,192,128,0,0
W 2550  DATA 87,7,7,7,7,7,7,7,7,88,255,255,2
        55,7,7,7,0,0,0,0,0,0,0,0,0,0,0,0,0
T 2560  DATA 90,3,0,0,0,1,6,1,25,46,83,91,200,55
        ,200,0,83,36,0,25,6,1
N 2570  DATA 92,0,0,0,128,96,152,4,212,94,19
        ,236,95,3,202,36,152,96,128
F 2580  DATA 95,3,31,255,254,240,16,0,0,97,2
        24,252,255,191,7,0,0,0
K 2590  DATA 98,7,7,7,0,0,0,0,0,99,0,0,0,0,0
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
L 2600  DATA 101,0,0,255,0,0,0,0,255,255,102,0,0
        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
O 2610  DATA 103,0,0,255,255,255,255,255,104
        0,0,0,255,255,255,255,255,255,255
L 2620  DATA 105,0,0,255,255,255,255,255,255,255
        ,106,0,0,255,255,255,255,255,255,255
D 2630  DATA 107,255,255,255,255,255,255,255
        ,255,255,-1,-1
F 2640  POKE54296,M2:POKE54284,4:POKE54285,
        0:POKE54280,195:POKE54279,195
X 2650  POKE54283,17:FORZ=1TOM1:NEXT:POKE54
        283,16:RETURN
R 2660  POKE54296,15:POKE54273,159:POKE5427
        7,9:POKE54278,4:POKE54276,17
F 2670  POKE54276,16:RETURN
H 2680  POKE54296,15:POKE54273,40:POKE54277
        ,9:POKE54278,16:POKE54276,17
N 2690  POKE54276,16:RETURN
```

HCM

VITAL SIGNS

IBM PC/PCjr, TANDY 1000

```
U  100  ' ********************
L  110  ' *  VITAL SIGNS  *
L  120  ' ********************
E  130  ' COPYRIGHT 1985
A  140  ' EMERALD VALLEY PUBLISHING CO.
O  150  ' BY WILLIAM K. STAFF
Q  160  ' AND THE HCM STAFF
Z  170  ' HOME COMPUTER MAGAZINE
R  180  ' VERSION 5.1 either
E  190  ' DOS 2.1 and with
C  200  ' IBM PCjr with CARTRIDGE BASIC or
U  210  ' IBM PC with BASICA &
R  220  ' COLOR/GRAPHICS ADAPTER &
E  230  ' COLOR MONITOR or with GW BASIC
S  240  ' TANDY 1000 PC with BASIC
S  250  ' BLACK = 0:GREEN=1:RED=2:BROWN=3
G  260  DIM PS(80),OX(80),TP(80),EX(33),
        CON(33),LUN$(4),ACTIVITY$(6),A
L  270  DIM AIR.QUALITY(4),OF$(ACTIVITY.LEVEL(6)
D  280  SCREEN 0:KEY OFF:WIDTH 40:COLOR ,2:
G  290  CLS:COLOR 15,4,12
```

Continued

TYPE-IN LISTINGS

```
300 LOCATE 4,7:PRINT STRING$(26,"  "):LOCATE 6,7:PRINT STRING$(26,"  "):LOCATE 5,7:PRINT "  Welcome To  Vital Signs  ":LOCATE 15,10,0:PRINT "PRESS";:LOCATE 15,2:LOCATE ,,2:PRINT "TO START";CHR$(17);CHR$(217);
330 K$ = INKEY$:IF K$ <> CHR$(13) THEN 330
340 SUB 1570:SCREEN 1:COLOR BLACK,0:CLS:LOCATE ,,0
950 GOSUB 1240:GOSUB 1320:GOSUB 990:GOSUB 1240:GOSUB 1440
410 IF PRESSURE > 170 THEN PRESSURE = 170:LOCATE 22,5:PRINT "PRESSURE TOO HIGH!":SOUND 500,5:IF AIR.OPTION$="A" THEN AIR.OPTION=AIR.OPTION-32
420 IF OXYGEN > 75 THEN OXYGEN = 75:LOCATE 22,5:PRINT "OXYGEN TOO HIGH!":SOUND 500,5:IF AIR.OPTION$="A" THEN AIR.OPTION=AIR.OPTION-32
430 IF TEMP > 103 THEN TEMP = 103:LOCATE 22,5:PRINT "TEMPERATURE TOO HIGH!":SOUND 500,5:IF AIR.OPTION$="A" THEN AIR.OPTION=AIR.OPTION-32
440 RETURN
450 'ADJUST VITALS
460 IF SWIMMING) OR (ACTIVITY = 1 THEN SWEAT.COUNT = 0 ELSE SWEAT.COUNT = SWEAT.COUNT + 1
470 TEMP = 37 + (.3 * OXYGEN) ^ 2 / 379 + .90 - SWEAT.COUNT / 200
480 IF (SWEAT.COUNT) OR (100 < TEMP) SWEAT.COUNT = SWEAT.COUNT - 1
IF SWEAT.COUNT < 0 THEN SWEAT.COUNT = 0
ACTIVITY.LEVEL = (SWEAT.COUNT + 1) * ((SWEAT.COUNT / 200) + 1.1) - ACTIVITY.LEVEL
TEMP = TEMP + (TEMP.TMP - TEMP) / 4
490 IF TEMP = 100 THEN TEMP = 90 ELSE IF TEMP > 107 THEN TEMP = 107
500 OX.TMP = OX.TMP + (SQR((RESP * AIR.QUALITY * .4)) * (1 - LUNG.FLAG * .4)) ^ 2 + PRESSURE ^ 2)
QUALITY = 8 * RATE.HEART / 50 + (TP - PRESSURE) / 2
ACTIVITY.LEVEL = 1 + BLOOD.FLAG * .5
510 OXYGEN = OXYGEN + (OX.TMP - OXYGEN) / 6
IF OXYGEN < 0 THEN OXYGEN = 0 ELSE IF OXYGEN = 100 THEN OXYGEN = 100
520 PULSE = SQR(ACTIVITY.LEVEL * HEART.RATE * 3.485 * (1 + BLOOD.FLAG * .5)) * 1.3
PRESSURE = PRESSURE + (TP - PRESSURE) / 8
530 IF PRESSURE < 0 THEN PRESSURE = 0 ELSE IF PRESSURE > 250 THEN PRESSURE = 250
540 IF (ACTIVITY = 0) AND (RND * 100 < 2) THEN ACTIVITY = 1
550 IF ACTIVITY = 1 THEN ACTIVITY = 2 ELSE IF ACTIVITY = 2 THEN ACTIVITY = INT(RND * 3) + 1
560 IF (ACTIVITY <> RANDOM) THEN 590
570 IF (INT(RND * 30) <> 15) THEN 630
580 ACTIVITY = INT(RND * 3) + 1
590 IF (ACTIVITY = ACTIVITY) THEN 610
600 ACTIVITY = ACTIVITY + SGN(RND * 100 - 50):SOUND 250,5
610 IF ACTIVITY < 1 THEN ACTIVITY = 1 ELSE IF ACTIVITY > 3 THEN ACTIVITY = 3
620 GOSUB 1240:ACTIVITY.LEVEL = ACTIVITY
630 IF INT(RND * 50) <> 25 THEN 670
640 AIR.OPTION = AIR.OPTION + SGN(RND * 100 - 50):SOUND 500,5
650 IF AIR.OPTION < 0 THEN AIR.OPTION = 0 ELSE IF AIR.OPTION > 3 THEN AIR.OPTION = 3
660 AIR.QUALITY = AIR.QUALITY + (AIR.OPTION - 2) / 10
GOSUB 1240
670 LUNG.DAMAGE = LUNG.DAMAGE + .04 * ABS(AIR.QUALITY - 1)
680 IF LUNG.FLAG <> 0 OR ((RND * 200) - LUNG.DAMAGE) < 0 THEN 700
690 LUNG.FLAG = 1:LUNG.DAMAGE = 200
700 IF <> 0 OR INT(RND * 200) <> BLOOD.FLAG THEN 720
710 BLOOD.FLAG = 1:GOSUB 1240
720 IF LUNG.FLAG = 1 THEN 790
730 LUNG.COUNTER = LUNG.COUNTER + 1:IF LUNG.COUNTER = 50 THEN 800
740 LUNG.DAMAGE = LUNG.DAMAGE - 0
750 GET (83,36)-(97,68),LUNG
760 LOCATE 22,2:PRINT "NEW LUNG"
770 FOR D=1 TO 10:PUT (83,36),LUNG,PRESET:NEXT
780 LOCATE 22,2:PRINT "  "
790 LUNG.COUNTER = 0
800 IF <> 1 OR INT(RND * 840) <> BLOOD.FLAG THEN 810
810 BLOOD.FLAG = 0
820 COLOR 15:LOCATE 23,2:PRINT "BLOOD CLOT CURED"
830 NEXT:LOCATE 23,2:COLOR ,1:PRINT STRING$(16," ");
840 RETURN
850 'CHANGE AIR OPTION
860 GOSUB 1220
870 LOCATE 16,2:PRINT "CHANGE AIR QUALITY"
880 FOR D = 0 TO 3:LOCATE .AIR TO SMOKE.SMOG:LOCATE ,,0:PRINT CHR$(D+ASC("A"));:NEXT
AIR.OPTION$ = INKEY$:WHILE AIR.OPTION$ = "":AIR.OPTION$ = INKEY$:WEND:IF ASC(AIR.OPTION$)-65 > 3 THEN 880
890 AIR.OPTION = ASC(AIR.OPTION$)-65:IF AIR.OPTION<0 THEN AIR.OPTION=0
ON>4 THEN AIR.OPTION=AIR.OPTION-32
900 GOSUB 1220:GOSUB 1240:GOSUB 990:GOSUB 1240
910 'QUALITY = AIR.QUALITY
920 RETURN
930 'UPDATE RESPIRATION AND HEART RATE
940 ON SCREEN
950 LOCATE 5,32:PRINT "RATE: ";MID$(STR$(HEART.RATE),2)" ";:LOCATE 7,32:PRINT "RESP:";RESP;
960 RETURN
970 'DISPLAY MAIN MENU
980 LOCATE 15,2:PRINT "1) ACTIVITY":PRINT "2) AIR":PRINT "3) EXIT":RETURN
990 'GET NEW COMMAND, IF ANY, AND BRANCH
1000 K$="":WHILE K$="":K$=INKEY$:GOSUB 370:WEND:P=ASC(K$):IF P<49 OR P>51 THEN 1000 ELSE P=INT(P*.5+.6):ON P GOSUB 1020,1040,1050:GOTO 1010
1010 HEART.CHANGE = 1:GOTO 1080
1020 HEART.CHANGE = -1:GOTO 1080
1030 HEART.CHANGE = 5:GOTO 1080
1040 HEART.CHANGE = -5:GOTO 1080
1050 RESP.CHANGE = 1:GOTO 1100
1060 RESP.CHANGE = -1:GOTO 1100
1070 HEART.RATE = HEART.RATE + HEART.CHANGE
NGE:IF HEART.RATE > 200 THEN HEART.RATE = 200:ELSE IF HEART.RATE < 0 THEN HEART.RATE = 0
HEART.RATE = 0
1090 GOSUB 950:RETURN
1100 RESP = RESP + RESP.CHANGE:IF RESP < 0 THEN RESP = 0:ELSE IF RESP > 30 THEN RESP = 30
THEN RESP = 0
950:RETURN
1110 'GOSUB CHANGE IN ACTIVITY
1120 GOSUB 1220
1130 LOCATE 16,2:PRINT "CHANGE ACTIVITY"
1140 :
1150 FOR D=SLEEPING TO RANDOM:LOCATE ,,18+D,2:PRINT CHR$(D+ASC("A"));")";ACTIVITY$(D):NEXT
1160 K$=INKEY$:WHILE K$="":K$=INKEY$:WEND:IF K$<"a" OR K$>"g" THEN 1160 ELSE IF K$="g" THEN 1170 ELSE ACTIVITY.OPTION = ASC(K$)-65:IF ACTIVITY.OPTION>7 THEN ACTIVITY.OPTION=ACTIVITY.OPTION-32
1170 ACTIVITY.OPTION = ASC(K$)-65:IF ACTIVITY.OPTION<0 THEN ACTIVITY.OPTION=RANDOM THEN ACTIVITY = 2 ELSE ACTIVITY = ACTIVITY.OPTION
1190 ACTIVITY.LEVEL = ACTIVITY.LEVEL(ACTIVITY)
1200 GOSUB 1220:GOSUB 1240:GOSUB 990
1210 RETURN
1220 LOCATE 21,5:PRINT ACTIVITY$(ACTIVITY);:RETURN
1230 'CONDITIONS
1240 LOCATE 20,5:PRINT AIR.OPTION$(AIR.OPTION);
1250 IF LUNG.FLAG = 1 THEN LOCATE 22,5:PRINT "LUNG CANCER";:IF BLOOD.FLAG = 1 THEN LOCATE 23,5:PRINT "BLOOD CLOT";
1260 RETURN
1270 'INITIALIZE BAR GRAPH
1280 'LINE (160,0)-(248,199),GREEN,B:PAINT (205,100),BROWN,GREEN
1290 ROW=16:COL=23:SOUND 1
1300 LINE (176,8)-(184,120),BLACK,BF
1310 COL=26:S$="%  OXYGEN":GOSUB 1550:LINE (198,126)-(209,193),1,B:GET (199,127)-(208,192),OXY
1320 LINE (200,8)-(208,120),BLACK,BF
1330 LINE (224,8)-(232,120),BLACK,BF
1340 'GET (83,36)-(97,68),LUNG
1350 'FOR D=1 TO 10:PUT (83,36),LUNG,PRESET:NEXT
```

Continued

VITAL SIGNS *Continued*

IBM PC/PCjr, TANDY 1000

```
N 1360 COL=29:S$="BODY TMP":GOSUB 1550:LIN
  E(22,126)-(233,193),1,B:GET (223,1
  27)-(232,192),TP
X 1370 FOR D=8 TO 120 STEP 8:LINE (162, D)
  -STEP(14,0),RED:STEP(8,0)-STEP(16,0),RED:LINE
C 1380 LINE(8,0)-STEP(8,0)-STEP(16,0),RED
  SLINE
S 1390 NEXT
G 1400 NEXT
E 1410 PRINT SURE.OLD = 0:OXYGEN.OLD = 0:TEM
  P.OLD = 0:GOSUB 1440
F 1420 RETURN
A 1430 'DISPLAY BAR GRAPH
O 1440 NEW.ROW = INT((PRESSURE - 80)* 112/
  90)+1:LAST.ROW = INT((PRESSURE.OLD-
  80)*112/90)+1
J 1450 COL = 177:GOSUB 1510:PRESSURE.OLD =
  PRESSURE
Q 1460 NEW.ROW=INT((OXYGEN-25)* 112/50)+1:
  LAST.ROW=INT((OXYGEN.OLD-25)*112/50
  )+1
D 1470 COL = 201:GOSUB 1510:OXYGEN.OLD = O
  XYGEN
E 1480 NEW.ROW = INT((TEMP-94)*112/9)+1:LA
  ST.ROW = INT((TEMP.OLD-94)*112/9)+1
N 1490 COL = 225:GOSUB 1510:TEMP.OLD = TEM
  P
S 1500 RETURN
Y 1510 IF NEW.ROW > G.ROW THEN NEW.ROW=G.R
  OW ELSE IF NEW.ROW < 0 THEN NEW.ROW
  =0
  IF LAST.ROW>G.ROW THEN LAST.ROW=G.R
  OW ELSE IF LAST.ROW<0 THEN LAST.ROW
  =0
H 1530 IF (NEW.ROW>LAST.ROW) THEN FOR I=LA
  ST.ROW TO NEW.ROW:LINE (COL,G.ROW-I
  +8)-STEP(6,0),BLACK:NEXT ELSE FOR I=LO
  AST.ROW TO NEW.ROW STEP -1:LINE (CO
  L,G.ROW-I+8)-STEP(6,0),BLACK:NEXT
K 1540 RETURN
V 1550 FOR D = 1 TO LEN(S$)):LOCATE ROW + D
  ,COL:PRINT MID$(S$,D,1);:NEXT:RETU
  RN
F 1560 'SET SYSTEM VARIABLES
W 1570 OX.TMP = 50:BLOOD.FLAG = 0:LUNG.FLA
  G = 0:SCORE = 0:G.ROW = 112
B 1580 SLEEPING = 0:RESTING = 1:NORMAL = 2
Y 1590 :WALMMING = 3:RUNNING = 4
N 1600 SDONE = 0:LUNG.COUNTER = 0:LUNG.DAMA
  GE = 0
V 1610 ACTIVITY.LEVEL = 107.4:ACTIVITY.OPT
  ION = NORMAL:ACTIVITY = NORMAL:SMOG
  G0OD.AIR = 0:SMOGGY = 1:SMOKE.CIG =
S 1620 2:SMOKE.SMOG = 3:AIR.QUALITY=1
A 1630 PRESS=SURE.PROBLEMS = 1:OXYGEN.PROBLE
  MS = 2:TEMP.PROBLEMS = 3
V 1640 PRESSURE = 125:OXYGEN = 50:TEMP = 9
  8.6
E 1650 RESP = 10:HEART.RATE = 80:PC = 0:OC
  = 0
K 1660 RESTORE 1710:FOR D = SLEEPING TO RA
  NDOM:READ ACTIVITY$(D):NEXT
S 1670 FOR D = 0 TO 3:READ GOOD.AIR$(D):NEXT
  FOR D = OPTIMUM TO SWIMMING:READ A
L 1680 FOR D = SLEEPING TO RANDOM:READ ACTI
  VITY$(D):NEXT
M 1690 FOR D = GOOD.AIR TO SMOKE.SMOG:READ
  QUALITY$(D):NEXT
W 1700 RETURN
O 1710 DATA Sleeping, Resting, Normal, Walkin
  g, Running, Swimming, Good, Air, Smo
  gGy, Smoke, Cig., Cig. & Smog, 69.5,
  91.7, 107.4, 121.6, 152.2, 171.9, 1,
  .7, .5
```

```
K 1720 DRAW "BM50,40C3D5GD5GD7FDG2D3FD2FD9
  R4U8R7FHLHU8L4BM48,46D5GL2HLHD3NU8R
  6U4L3HU2ER6U4L3GLHD4F2DD55,49
  DR3"
F 1730 DRAW "BM51,39C0R3C3U3HU3HU3HU3HU2EU
  R2NU10R3E2DFUD3GD3GD3GD3GD3RC
  NDGLGL3CG3NUE3FUD3FD3GD2GD2GD2HU2LHLHLH3L2G
  DRAW "BM50,79C0R3C3D3HD3GD2GD2GD2HU2LHLHLH3L2G
H 1780 DRAW "BM50,79C0R3C3D3HD3GD2GD2GD2HU2LHLHLH3L2G
  DRAW "BM31,59C3E2R3ER3UC0U3C3L3GL3HL
K 1790 DRAW "BM84,48R5F2D2D3C3R5FRFNR2GNL5D
  CDGD3C3R3F2D2DGR2FD4GF2R$(177)+CHR$(27)+CHR$(198)
B 1800 DRAW "BM31,59C3E2R3ER3UC0U3C3L3GL3HL
  DRAW "BM31,59C3E2R3ER3UC0U3C3L3GL2GH2U3HU2
  DRAW "BM31,59C3E2R3ER3UC0U3C3L3GL2GH2U3HU2E2H
A 1810 PAINT(47,47),1,3:PAINT(54,50),1,3:PA
  INT(53,55),2,3:PAINT(64,40),2,3:PAINT(64,58),2,3
  PAINT(47,60)-(72,77),EX
L 1820 'RETURN
M 1830 'RETURN
O 1840 'RETURN END DUE TO PROBLEMS
H 1850 'CLS:LOCATE 25,1:PRINT"PROBLEMS THEN
  'FOR I=1 TO 1000:NEXT:LOCATE 25,1:
  PRINT"PROBLEMS THEN
K 1860 'CLS:LOCATE 25,1:PRINT"PROBLEMS THEN
Q 1870 'CLS:LOCATE 25,1:PRINT"PROBLEMS THEN
S 1880 'CLS:LOCATE 25,1:PRINT"PROBLEMS THEN
C 1890 'GOTO 1950
A 1900 'GAME
A 1910 'GOSUB 1220
I 1920 'PRINT"ARE YOU SURE? (Y/N)":A$=INKEY$:WEND:IF A$="N" OR A$="n" THEN GOSUB 1220:GOTO 1930 ELSE IF A$<>"Y" AND A$<>"y" THEN 1920
Y 1930 'CLS:LOCATE 25,1:PRINT"PLAY AGAIN?
  (Y/N)":A$=INKEY$:WEND:IF A$="Y" OR A$="y" THEN RUN ELSE IF A$="N" OR A$="n" THEN SYSTEM ELSE 1930
T 1940 'CLS:LOCATE 25,1:PRINT"PRINTING SCRE
B 1950 'COPY
  'PRINT:PRINT:PRINT"PLAY AGAIN?
  (Y/N)":WHILE INPUT$(1):IF A$="Y" OR A$="y" THEN RUN ELSE END
```

HCM

VITAL SIGNS

TI-99/4A

```
P 100 REM ***********
Q 110 REM * VITAL SIGNS *
P 120 REM ***********
M 130 REM COPYRIGHT 1985
H 140 REM EMERALD VALLEY PUBLISHING CO.
A 150 REM BY WILLIAM K. BALTHROP
R 160 REM HOME COMPUTER MAGAZINE
G 170 REM VERSION 5.5.1
T 180 REM TI BASIC OR
E 190 REM TI EXTENDED BASIC
M 200 REM
A 210 CALL CLEAR
W 220 PRINT TAB(8);"VITAL SIGNS":::::::
  PRESS ENTER TO START
N 230 GOSUB 4000
H 240 RANDOMIZE
T 250 DIM AC(5),AC$(6),AR(3)
F 260 CALL CLEAR
H 270 GOSUB 4240
W 280 GOSUB 5000
L 290 CALL BORDER(16)
S 300 FOR D=1 TO 96
V 310 READ A$
```

```
D 320 CALL CHAR(Z,A$)
X 330 NEXT Z
A 340 CALL CLEAR
J 350 FOR Z=9 TO 14
A 360 READ C$
V 370 CALL COLOR(Z,A,B)
G 380 NEXT Z
O 390 FOR Z=1 TO 15
T 400 READ A$
H 410 PRINT TAB(3);A$
Z 420 NEXT Z
Y 430 ::::::::::::::::
P 440 ::::::::::::::::
U 450 CALL VCHAR(4,3,128,4)
Q 460 CALL CHAR(4,3,128,4)
L 470 CALL VCHAR(4,11,128,4)
J 480 CALL VCHAR(4,12,128,4)
S 490 FOR Z=1 TO 12
U 500 CALL HCHAR(Z,17,128,9)
U 510 NEXT Z
B 520 FOR Z=19 TO 23
D 530 CALL VCHAR(2,Z,137,10)
E 540 NEXT Z
```

Continued

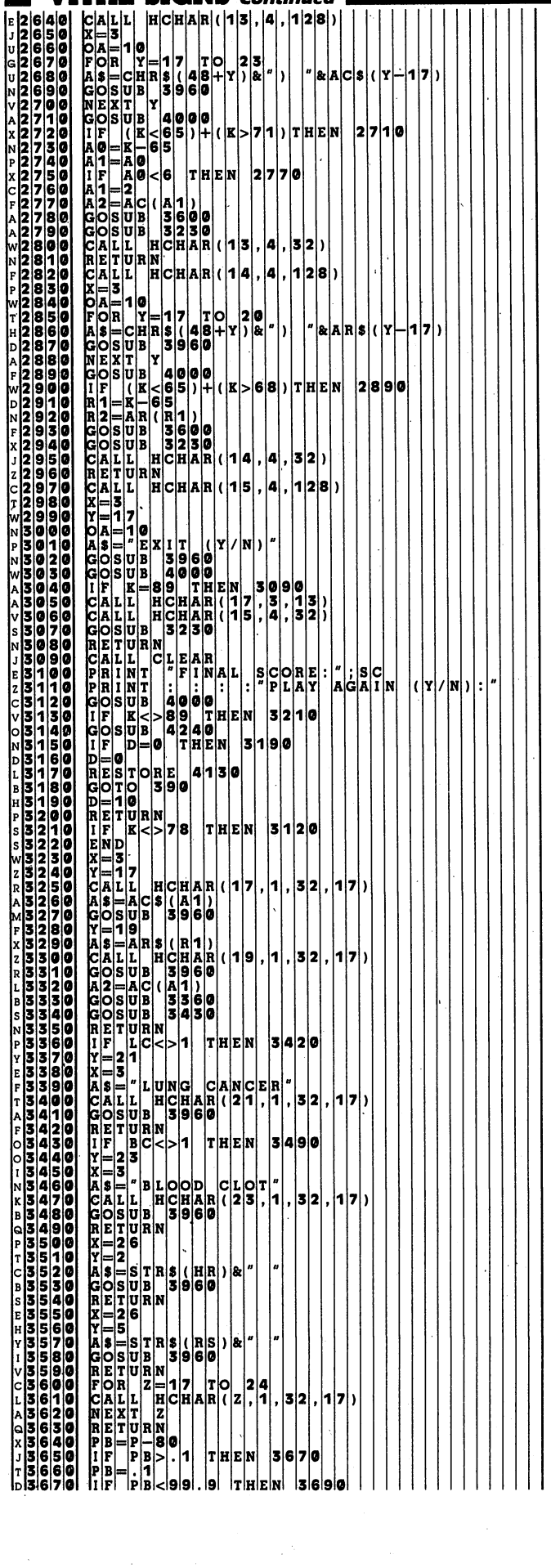
TYPE-IN LISTINGS

TYPE-IN LISTINGS

```
 550 CALL VCHAR(1,21,128,12)
 560 CALL VCHAR(7,18,136,5)
 570 CALL VCHAR(7,21,136,5)
 580 CALL VCHAR(7,24,136,5)
 590 Y=13
 600 X=18::PRESSURE:
 610 GOSUB 3920
 620 X=21::PB
 630 GOSUB 3920
 640 A$="OXYGEN":
 650 X=24::PB
 660 GOSUB 3920
 670 A$="BODY TEMP":
 680 GOSUB 3920
 690 Y=16::PRATE::"
 700 A$="":
 710 GOSUB 3960
 720 Y=2::PB
 730 GOSUB 3960
 740 A$="80":
 750 GOSUB 3960
 760 Y=4::PRESP:"
 770 A$="":
 780 GOSUB 3960
 790 Y=5::PB
 800 A$="10":
 810 GOSUB 3960
 820 GOSUB 3230
 830 220 GOSUB 3230
 840 IF >10 THEN 880
 850 RESTORE 4130
 860 CALL CLEAR
 870 GOSUB 3900
 880 GOSUB 1290
 890 CALL COLOR(9,7,16)
 900 CALL SOUND(100,110,0)
 910 CALL SOUND(30,110,30)
 920 CALL SOUND(9,10,16)
 930 CALL SOUND(-200,110,5)
 940 IF (PB>80)*(P<180)THEN 970
 950 GOSUB 1050
 960 IF PB>0 THEN 4150
 970 IF OX>25)*(OX<75)THEN 1000
 980 GOSUB 1130
 990 IF T>94)*(T<103)THEN 1030
1000 GOSUB 1210
1010 IF PB>0 THEN 4150
1020 SC=SC+A0+R1
1030 GOSUB 8210
1040 CALL HCHAR(12,18,136)
1050 CC=CC+40
1060 IF R<15 THEN 1100
1070 CC=CC+40
1080 IF R<15 THEN 1100
1090 CALL SOUND(10,-3,0)
1100 CALL HCHAR(12,18,128)
1110 RETURN
1120 CALL HCHAR(12,21,136)
1130 CC=CC+40
1140 IF R<15 THEN 1180
1150 CALL SOUND(10,-3,0)
1160 CALL HCHAR(12,21,128)
1170 RETURN
1180 CALL HCHAR(12,24,136)
1190 CC=CC+40
1200 IF R<15 THEN 1260
1210 CALL SOUND(10,-3,0)
1220 CALL HCHAR(12,24,128)
1230 RETURN (A1=5)+(A1=6)THEN 1320
1240 CNT=0
1250 GOTO 1330
1260 CNT=CNT+1
1270 IF (CNT>40)*(CNT<100)THEN 1360
1280 TTA=(250-HR)^2+(OX*3)^2)*.0758
     8
1290 T=81.1+((A2*(CNT+1))*.001)*((CNT
     *1.1+1))
1300 GOTO 1370
1310 TTA=SQR((250-HR)^2+(OX*3)^2)*.0758
     8
1320 T=81.1+T>=107 THEN 1420
1330 IF T<=107 THEN 1470
1340 OXF=OX+(T-100)*.25
1350 IF OXF<100 THEN 1490
1360 PA=(150-OX)*2
1370 PP=(SQR(A2*HR)*1.3485*(1+BC*.5)+PA
     P)/2
1380 IF P<>0 THEN 1540
1390 IF P<=250 THEN 1560
1400 PP=250
1410 GOSUB 3640
1420 IF SUB=6 THEN 1590
1430 RETURN
1440 IF INT(RND*30)<>15 THEN 1720
1450 CALL SOUND(100,440,0)
1460 A1=INT(RND*100-50)
1470 IF A1>=0 THEN 1650
1480 IF A1<=5 THEN 1670
1490 A1=4
1500 X=17
1510 A$=SUB$(A1)
1520 CALL HCHAR(17,1,32,17)
1530 GOSUB 1740::IF R1>=0 THEN 1760
1540 IF R1<=3 THEN 1780
1550 R2=AR(R1)
1560 A$=AR$(R1)
1570 CALL HCHAR(19,1,32,17)
1580 CALL SOUND(100,440,0)
1590 GOSUB 3960
1600 CALL HCHAR(4,11,136,4)
1610 CALL HCHAR(4,12,136,4)
1620 GOSUB 3360
1630 IF <>0)+(INT(RND*200)<>100)THEN
     1910
1640 CALL SOUND(300,-1,0)
1650 CALL HCHAR(4,11,136,4)
1660 CALL HCHAR(4,12,136,4)
1670 GOSUB 3360
1680 IF <>0)+(INT(RND*200)<>100)THEN
     1910
1690 CALL SOUND(300,-1,0)
1700 GOSUB 3430
1710 IF <>1 THEN 2100
1720 IF <>1+1 THEN 2110
1730 IF <50 THEN 2110
1740 CALL SOUND(300,-3,0)
1750 CALL HCHAR(21,1,32,17)
1760 A$="NEW LUNG"
1770 CALL HCHAR(4,11,128,4)
1780 CALL HCHAR(4,12,128,4)
1790 Y=23
1800 X=23
1810 A$="CLOT FIXED"
1820 CALL HCHAR(23,1,32,17)
1830 GOSUB 3960
1840 CALL KEY(0,K,S)
1850 IF S=0 THEN 2230
1860 CALL SOUND(-10,880,0)
1870 IF (K<52)*(K>48)THEN 2620
1880 IF K=69 THEN 2320
1890 IF K=83 THEN 2360
1900 IF K=68 THEN 2400
1910 IF K=70 THEN 2440
1920 IF K=65 THEN 2480
1930 RESTORE 4130
1940 GOSUB 3960
1950 IF R<15 THEN 2350
1960 GOSUB 1550
1970 IF R<15 THEN 2390
1980 GOSUB 1550
1990 IF R<15 THEN 2430
2000 GOSUB 1550
2010 IF R<15 THEN 2470
2020 GOSUB 1550
2030 IF R<15 THEN 2510
2040 GOSUB 1550
2050 IF R<15 THEN 2550
2060 GOSUB 1550
2070 IF R<15 THEN 2590
2080 GOSUB 1550
2090 IF R<15 THEN 2630
2100 GOSUB 3600
2110 GOTO 2640,2820,2970
```

Continued

VITAL SIGNS *Continued*



TYPE-IN LISTINGS

```
100 REM ********************
110 REM *                  *
120 REM *  NANOPROCESSOR   *
130 REM *                  *
140 REM *  COPYRIGHT 1985  *
150 REM *  BY ROGER WAGNER *
160 REM *                  *
170 REM *  HCM COMPUTER    *
180 REM *  CONSULTANTS     *
190 REM *                  *
200 REM *  PUBLISHED BY    *
210 REM *  HOME APPLE      *
220 REM *  MAGAZINE        *
230 REM *                  *
240 REM *  SOFTWARE FOR    *
250 REM *  THE APPLE ][    *
260 REM ********************
270 REM
280 REM
290 REM
300 REM
310 REM
320 REM
330 REM
340 REM
350 REM
360 REM
370 REM
380 REM
390 REM
400 REM
410 REM
420 REM
430 REM
440 REM
450 REM
460 REM
470 REM
480 REM
490 REM
500 REM
510 REM
520 REM
530 REM
540 REM
550 REM
560 REM
570 REM
580 REM
590 REM
600 REM
610 REM
620 REM
630 REM
640 REM
650 REM
660 REM
670 REM
```

```
680 REM
690 REM
700 REM
710 REM
720 REM
730 REM
740 REM
750 REM
760 REM
770 REM
780 REM
790 REM
800 REM
810 REM
820 REM
830 REM
840 REM
850 REM
860 REM
870 REM
880 REM
890 REM
900 REM
910 REM
920 REM
930 REM
940 REM
950 REM
960 REM
970 REM
980 REM
990 REM
1000 REM
1010 REM
1020 REM
1030 REM
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 REM
1100 REM
1110 REM
```

```
10 :
20 :
30 :
40 :
50 :
60 :
70 :
80 :
90 :
100 :
110 :
120 :
130 :
140 :
150 :
160 :
170 :
180 :
190 :
200 :
210 :
220 :
230 :
240 :
250 :
260 :
270 :
280 :
290 :
300 :
310 :
320 :
330 :
340 :
350 :
360 :
370 :
380 :
390 :
400 :
410 :
420 :
430 :
440 :
450 :
460 :
470 :
480 :
490 :
500 :
510 :
520 :
530 :
540 :
550 :
560 :
570 :
580 :
590 :
600 :
610 :
620 :
630 :
640 :
650 :
660 :
670 :
```

```
680 :
690 :
700 :
710 :
720 :
730 :
740 :
750 :
760 :
770 :
780 :
790 :
800 :
810 :
820 :
830 :
840 :
850 :
860 :
870 :
880 :
890 :
900 :
910 :
920 :
930 :
940 :
950 :
960 :
970 :
980 :
990 :
1000 :
1010 :
1020 :
1030 :
1040 :
1050 :
1060 :
1070 :
1080 :
1090 :
1100 :
1110 :
```

Continued

```
110 VTAB 15: HTAB 36: PRINT "/#"
120 VTAB 17: HTAB 36: PRINT "&"
130 VTAB 19: HTAB 36: PRINT "("
140 RETURN
150 REM CLOTT
160 REM HPLOTT
170 REM VPLOTT
180 REM HPLOTAB
190 REM BUSY
200 REM POWER
210 REM DATA
220 REM SWITCH
230 REM DIRECT
240 REM INDICATE
250 REM INDICATE
260 REM INDICATE
270 REM INDICATE
280 REM INDICATE
290 REM LOAD
300 REM END
310 REM TEXT
320 REM HOME
330 REM POWER
340 REM
350 REM
360 REM
370 REM
380 REM
390 REM
400 REM
410 REM
420 REM
430 REM
440 REM
450 REM
460 REM
470 REM
480 REM
490 REM
500 REM
510 REM
520 REM
530 REM
540 REM
550 REM
560 REM
570 REM
580 REM
590 REM
600 REM
610 REM
620 REM
630 REM
640 REM
650 REM
660 REM
670 REM
680 REM
690 REM
700 REM
710 REM
720 REM
730 REM
740 REM
750 REM
760 REM
770 REM
780 REM
790 REM
800 REM
810 REM
820 REM
830 REM
840 REM
850 REM
860 REM
870 REM
880 REM
890 REM
900 REM
910 REM
920 REM
930 REM
940 REM
950 REM
960 REM
970 REM
980 REM
990 REM
1000 REM
1010 REM
1020 REM
1030 REM
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 REM
1100 REM
1110 REM
1120 REM
1130 REM
1140 REM
1150 REM
1160 REM
1170 REM
1180 REM
1190 REM
1200 REM
1210 REM
1220 REM
1230 REM
1240 REM
1250 REM
1260 REM
1270 REM
1280 REM
1290 REM
1300 REM
1310 REM
1320 REM
1330 REM
1340 REM
1350 REM
1360 REM
1370 REM
1380 REM
1390 REM
1400 REM
1410 REM
1420 REM
1430 REM
1440 REM
1450 REM
1460 REM
1470 REM
1480 REM
1490 REM
1500 REM
1510 REM
1520 REM
1530 REM
1540 REM
1550 REM
1560 REM
1570 REM
1580 REM
1590 REM
1600 REM
1610 REM
1620 REM
1630 REM
1640 REM
1650 REM
1660 REM
1670 REM
1680 REM
1690 REM
1700 REM
1710 REM
1720 REM
1730 REM
1740 REM
1750 REM
1760 REM
1770 REM
1780 REM
1790 REM
1800 REM
1810 REM
1820 REM
1830 REM
1840 REM
1850 REM
1860 REM
1870 REM
1880 REM
1890 REM
1900 REM
1910 REM
1920 REM
1930 REM
1940 REM
1950 REM
1960 REM
1970 REM
1980 REM
1990 REM
2000 REM
2010 REM
2020 REM
2030 REM
2040 REM
2050 REM
2060 REM
2070 REM
2080 REM
2090 REM
2100 REM
2110 REM
2120 REM
2130 REM
2140 REM
2150 REM
2160 REM
2170 REM
2180 REM
2190 REM
2200 REM
2210 REM
2220 REM
2230 REM
2240 REM
2250 REM
2260 REM
2270 REM
2280 REM
2290 REM
```

Continued

TYPE-IN LISTINGS

TYPE-IN LISTINGS

```
D 23000  FOR I=0 TO 3: READ D: POKE 768+I,D: NEXT: DATA 169,0,133,6,169,3,133,7,96
Z 23110  NEXT S(1): GOSUB 3010: GOSUB 2350,2360,2370,2380,2390,2400,2410,2420,2430,2440,2450,2460,2470,2480,2490,2500,2510,2520,2530,2540,2550,2560,2570,2580,2590,2600,2610,2620,2630,2640,2650,2660,2670,2680,2690,2700,2710,2720,2730,2740,2750,2760,2770,2780,2790,2800,2810,2820,2830,2840,2850,2860,2870,2880
I 23220  GOSUB 3010: GOSUB 1480
S 23330  RETURN
R 23400  M(0)=S(1): GOSUB 3010: GOSUB 1480
S 23550  ALX=M(0): GOSUB 3010: GOSUB 1480
M 23660  ALX=M(0): GOSUB 3010: GOSUB 1480
T 23770  GOSUB 2440: ADX=ADX(AD): NB=NB: IF AD<>0 THEN GOSUB 2440
B 23880  IF AD<>0 THEN GOSUB 2440
C 23990  IF AD<>0 THEN GOSUB 2440
A 24000  IF AD<>0 THEN GOSUB 2440
U 24110  IF AD<>0 THEN GOSUB 2440
Z 24220  POKE 768,68: POKE 769,8: CALL 768: RETURN
D 24330  POKE 768,68: POKE 769,8: CALL 768: RETURN
Z 24440  FOR I=0 TO 8: M(I)=S(I): NEXT: GOSUB 3010: RETURN
V 24550  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
S 24660  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
A 24770  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
Y 24880  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
Q 24990  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
S 25100  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
A 25210  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
N 25320  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
Y 25430  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
B 25540  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
Q 25650  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
E 25760  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
G 25870  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
J 25980  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
A 26090  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
V 26200  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
E 26310  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
N 26420  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
F 26530  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
L 26640  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
C 26750  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
A 26860  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
Z 26970  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
L 27080  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
Q 27190  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
N 27300  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
U 27410  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
L 27530  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
A 27640  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
J 27760  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
W 27870  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
L 27980  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
X 28090  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
P 28200  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
N 28310  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
A 28420  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
O 28530  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
U 28640  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
B 28750  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
V 28860  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
Y 28970  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
V 29080  FOR I=0 TO S(1): A(I)=S(I): NEXT: RETURN
L 28900  REM  SPLAY DATA SUBS
A 29000  REM  SUBS 2920,2930,2940,2950,2960,2970,2980,2990,3000,3010
S 29100  REM  SUBS 3020,3030,3040,3050,3060,3070,3080,3090,3100,3110
P 29200  REM  SUBS 3120,3130,3140,3150,3160,3170,3180,3190,3200,3210
E 29300  REM  SUBS 3220,3230,3240,3250,3300,3310,3320,3330,3340,3350
J 29400  REM  SUBS 3360,3370,3380,3390,3400,3410,3420,3430,3440,3450
R 29500  REM  SUBS 3460
E 29700  REM  MISCELLANEOUS SUBS
G 29800  REM  SUBS 2820,2830,2840,2850,2860,2870,2880,2890
X 29900  REM  SUBS 2820,2830,2840,2850,2860,2870,2880,2890
I 30000  REM  MISCELLANEOUS SUBS
Z 30100  REM  SUBS 3010,3020,3030,3040,3050,3060,3070,3080,3090,3100,3110,3120,3130,3140,3150,3160,3170,3180,3190,3200,3210,3220,3230,3240,3250,3300,3310,3320,3330,3340,3350,3360,3370,3380,3390,3400,3410,3420,3430,3440,3450,3460
K 30200  REM  SUB 3020
Y 30300  REM  SUB 3030
N 30400  REM  SUB 3040
A 30500  REM  SUB 3050
L 30600  REM  SUB 3060
J 30700  REM  SUB 3070
G 30800  REM  SUB 3080
P 30900  REM  SUB 3090
B 31000  REM  SUB 3100
D 31110  REM  SUB 3110
R 31220  REM  SUB 3120
N 31330  REM  SUB 3130
C 31400  REM  SUB 3140
R 31500  REM  SUB 3150
F 31600  REM  SUB 3160
Y 31700  REM  SUB 3170
W 31800  REM  SUB 3180
F 31900  REM  SUB 3190
Z 32000  REM  SUB 3200
W 32100  REM  SUB 3210
B 32200  REM  SUB 3220
W 32400  REM  SUB 3240
O 32500  REM  SUB 3250
N 32600  REM  SUB 3260
S 32700  REM  SUB 3270
F 32800  REM  SUB 3280
O 32900  REM  SUB 3290
J 33000  REM  SUB 3300
T 33110  REM  SUB 3310
Y 33220  REM  SUB 3320
G 33330  REM  SUB 3330
M 33400  REM  SUB 3340
R 33500  REM  SUB 3350
D 33600  REM  SUB 3360
B 33700  REM  SUB 3370
U 33800  REM  SUB 3380
A 33900  REM  SUB 3390
N 34000  REM  SUB 3400
E 34100  REM  SUB 3410
Y 34200  REM  SUB 3420
T 34300  REM  SUB 3430
W 34400  REM  SUB 3440
B 34500  REM  SUB 3450
Y 34600  REM  SUB 3460
```

NANOPROCESSOR


```
100 REM ********************
110 REM *  NANOPROCESSOR   *
120 REM ********************
130 REM  COPYRIGHT 1985
140 REM  PUBLISHING CO.
150 REM  BY ROGER VALLEY
160 REM  AND DOT WOOD
170 REM  HOME COMPUTER MAGAZINE
180 REM  VERSION 5.1
190 REM  ATARI BASIC FOR THE 800, 800XL,
    130XE
200 REM
210 REM  ADDR(256),PCS(8),PPC$(8)
220 REM  BIN$(68),BIN(16)
230 REM  A$(4),NT(31)
240 REM  MEM$(4),PMEM$(4)
250 REM  SW$(4),LON$(2),RSY(5),
260 REM  RSX$(20):DPRO
270 REM  #2,4,PRO
280 REM
290 REM
300 REM
310 REM
320 REM
330 REM
340 REM
350 GOSUB 1450
360 IF PC=0 THEN 350
370 GOSUB 1450
380 IF PC=0 THEN 370
390 IF PC=0 THEN 350
400 REM
410 FOR I=1+1 TO 8:GOSUB 2310:PRINT #2,
    PC;:NEXT I:PRINT #2:RETURN
420 REM
430 REM
440 REM
450 REM
460 REM
470 REM
480 REM
490 REM
500 REM
510 REM
520 REM
530 REM
540 REM
550 REM
560 REM
570 REM
580 REM
590 REM
600 REM
610 REM
620 REM
630 REM
640 REM
650 REM
660 REM
670 REM
680 REM
690 REM
700 REM
710 REM
720 REM
730 REM
740 REM
750 REM
760 REM
770 REM
780 REM
790 FOR I=1 TO 9:READ A:BIN(I)=A:NEXT I
800 FOR I=1 TO 64 STEP 4:READ A$:BIN$(I
    ,I+4)=A$:NEXT I
810 FOR I=1 TO 5
820 READ A:RSX(I)=A:READ A:RSY(I)=A
830 NEXT I
840 FOR I=1 TO 256
850 NEXT I
860 DATA 128,64,32,16,8,4,2,1,0
870 DATA 0,1,3,7,15,31,63,127,255
880 FOR I=0 TO 31:READ A:NT(I)=A:NEXT I
890 RETURN
900 DATA 0,0,0,0,0,0,0,0,0001,0010,0011,
    0100,0101,0110,0111,1000,1001,1010,
    1011,1100,1101,1110,1111
910 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,
    14,15
920 DATA 15,14,13,12,11,10,9,8,7,6,5,4,
    3,2,1,0
930 DATA 255,254,253,252,251,250,249,
    248,247,246,245,244,243,242,241,240
940 DATA 240,224,208,192,176,160,144,
    128,112,96,80,64,48,32,16,0
950 REM PLOT A POINT IN PMBASE
960 REM XPOS=C/8
970 REM YPOS=INT(C/8)*1000
980 REM BIT=INT(128/(2^(C-8*XPOS)))
990 REM POKE PMBASE+YPOS*4+2,PEEK(PMBASE
    +YPOS*4+2)+BIT
1000 REM OR IN CLEAR BIT
1010 FOR I=1 TO LOC STEP 2
1020 POKE PINC,PEEK(PINC)+A
1030 NEXT PINC
1040 RETURN
1050 REM CONVERT ANDPRINT ADDR
1060 PINC=INT((PC/16)*4+1
1070 PC$=BIN$(A,A+3)
1080 PC=(PC-16*INT(PC/16))*4+1
1090 PC$=BIN$(A,A+3)
1100 YPOS=5:XPOS=6
1110 IF PC$(INC,INC)="1" THEN LON=1
1120 GOSUB 950
1130 XPOS=XPOS+4
1140 INC
1150 RETURN
1160 REM CONVERT ?TO BINARY AND DISPLAY
1170 REM RSW GOTO 1210,1220,1230,1240,125
1180 A=PC-(INT(PC/16)*16):GOTO 1260
1190 A=INT(PC/16):GOTO 1260
1200 A=ADREG:GOTO 1260
1210 A=BREG
1220 MEM$=BIN$(A*4+1)
1230 YPOS=INC:XPOS=15
1240 FOR INC=1 TO 4
1250 IF MEM$(INC,INC)="1" THEN LON=1
1260 NEXT INC
1270 RETURN
1280 REM CONVERT SW TO BINARY AND DISPLA
1290 SW$=BIN$(SW*4+1)
1300 YPOS=17:XPOS=15
1310 FOR INC=1 TO 4
1320 IF SW$(INC,INC)="1" THEN LON$="
     CTRL X"
1330 NEXT INC
1340 PRINT LON$;
1350 RETURN
1360 REM KEY INPUT
1370 IF PEEK(764)<>255 THEN GET #2,B:GOT
     O 1390
1380 B=0
1390 IF PEEK(53279)=3 AND POWER=1 THEN G
     OSUB 3610:GOSUB 3410
1400 IF PEEK(53279)=5 AND POWER=1 THEN G
     OSUB 3410
1410 RETURN
1420 XPOS=XPOS+3
1430 NEXT INC
1440 RETURN
1450 REM KEY IN
1460 IF PEEK(764)<>255 THEN GET #2,B:GOT
     O 1470
1470 IF PEEK(53279)=3 AND POWER=1 THEN G
     OSUB 3610:GOSUB 3410
1480 IF PEEK(53279)=5 AND POWER=1 THEN G
     OSUB 3410
1490 RETURN
1500 FOR I=1 TO 4:READ A:POKE 1536+I,A:
     NEXT I
1510 FOR I=1 TO 4:READ A:POKE 1540+I,A:
     NEXT I
1520 REM
1530 REM
1540 GOTO 1490
1550 DATA 1,2,3,4,<,>,P,E,B,I,R,L,H
1560 IF RSW=1 THEN GOTO 1690
1610 IF RSW=5 THEN GOTO 1640
1620 A=1
1630 A=1
1640 GOTO 1690
1650 A=2
1660 A=2
1670 A=4
1680 A=4
1690 RETURN
1700 REM
1710 REM
1720 REM
1730 REM
1740 REM
1750 REM
1760 REM
1770 REM
1780 REM
1790 REM
1800 REM
1810 REM
1820 REM
1830 REM
1840 REM
1850 REM
1860 REM
1870 REM
1880 REM
1890 REM
1900 REM
1910 REM
1920 REM
1930 REM
1940 REM
1950 REM
1960 REM
1970 REM
1980 REM
1990 REM
2000 REM
2010 REM
2020 REM
2030 REM
2040 REM
2050 REM
2060 REM
2070 REM
2080 REM
2090 REM
2100 REM
2110 REM
2120 REM
2130 REM
2140 REM
2150 REM
2160 REM
2170 REM
2180 REM
2190 REM
2200 REM
2210 REM
2220 REM
2230 REM
2240 REM
2250 REM
2260 REM
2270 REM
2280 REM
2290 REM
2300 REM
2310 IF RSW=1 THEN GOSUB 1190:GOSUB 1050
2320 IF RSW=5 THEN GOSUB 1190:GOSUB 1050
2330 IF RSW=9 THEN GOSUB 1190:GOSUB 1050
2340 IF RSW=13 THEN GOSUB 1190:GOSUB 105
     0
2350 RETURN
2360 REM
2370 REM
2380 REM
2390 REM
2400 REM
2410 REM
2420 REM
2430 REM
2440 REM
2450 REM
2460 REM
2470 REM
2480 REM
2490 REM
2500 REM
2510 REM
2520 REM
2530 REM
2540 REM
2550 REM
2560 REM
2570 REM
2580 REM
2590 REM
2600 REM
2610 REM
2620 REM
2630 REM
2640 REM
2650 REM
2660 REM
2670 REM
2680 REM
2690 REM
2700 REM
2710 REM
2720 REM
2730 REM
2740 REM
2750 REM
2760 REM
2770 REM
2780 REM
```

Continued

NANOPROCESSOR *Continued*

ATARI 800/800XL/130XE

```
10 POSITION RSX(RSW),RSY(RSW):PRINT "
RSW=RSW+A
POSITION RSX(RSW),RSY(RSW):PRINT RS
C$(RSW,RSW)
IF POWER=0 THEN GOTO 1690
GOSUB 1190
KEY=0
RETURN
REM WORK THE BIT SWITCHES
A$=BIN$(SW*4+1)
ON KEY GOTO 1740,1760,1780,1800
A=-8:IF A$(1,1)="0" THEN A=-A
GOTO 1810
A=-4:IF A$(2,2)="0" THEN A=-A
GOTO 1810
A=-2:IF A$(3,3)="0" THEN A=-A
GOTO 1810
A=-1:IF A$(4,4)="0" THEN A=-A
KEY=0:SW=SW+A
GOSUB 1360
IF PRSW<3 THEN GOSUB 1190
RETURN
REM TURN ON POWER, LIGHT UP
IF POWER=0 THEN 1930
PC=0:GOSUB 1050:A=0:GOSUB 1260
POWER=0:PRINT "■"
FOR INC=PMBAS+6 TO PMBAS+8:POKE INC
,0:NEXT INC:ADDR(254)=0
GOTO 1990
IF KEY=0 THEN 1970
IPC=INT(RND(0)*252):ADDR(PC+1)=INT(R
ND(0)*16):AREG=INT(RND(0)
*16):GOSUB 1190
PRINT "■CTRL U■"
FOR INC=PMBAS-54 TO PMBAS-52
POKE INC,POWER*32
NEXT INC
KEY=0
RETURN
REM ZERO PC
POSITION 33,9:PRINT "■CTRL U■"
PC=0
POSITION 33,9:RETURN
REM INCREMENT PC
POSITION 33,12:PRINT "■CTRL U■"
PC=PC+1
IF PC=256 THEN PC=0
POSITION 33,12:RETURN
REM CLR RUN FLAG
RF=0:GOTO 2170
REM SET RUN FLAG
RF=1
POSITION 33,18-RF*3:PRINT "■CTRL U■"
FOR INC=PMBAS-66 TO PMBAS-64
POKE INC,RF*32
NEXT INC
POSITION 33,18-RF*3:RETURN
REM LOAD FROM SWITCH
POSITION 33,21:PRINT "■CTRL U■"
ON RSW GOTO 2250,2260,2270,2270,227
0
PC=INT(PC/16)*16+SW:GOTO 2300
PC=SW*16+16*(PC/16-INT(PC/16)):GOTO
2300
ADDR(PC+1)=SW
IF PC<252 THEN 2300
ON PC-252 GOSUB 2560,2590,2600
POSITION 33,21:RETURN
REM ADD AREG TO BREG
AREG=AREG+BREG
CFLAG=0:IF AREG>15 THEN CFLAG=1:ARE
G=AREG-16
RETURN
REM LOAD AREG IMMEDIATE
GOSUB 3820
AREG=ADDR(PC+1)
ZFLAG=0:IF AREG=0 THEN ZFLAG=1
RETURN
REM LOAD AREG FROM LOCATION
GOSUB 3820
A=ADDR(PC+1)
GOSUB 3820
A=ADDR(PC+1)*16+A
AREG=ADDR(A+1)
GOTO 2380
REM STORE AREG IN LOCATION
GOSUB 3820
A=ADDR(PC+1)
GOSUB 3820
ADDR(A+1)=AREG
IF A<253 THEN 2550
GOSUB 2560,2590,2600
RETURN
IF ADDR(254)>0 THEN PV=16:GOTO 2580
PV=0
FOR INC=PMBAS+6 TO PMBAS+8:POKE INC
,PV:NEXT INC:RETURN
SOUND 0,NT(ADDR(255)),10,15:GOTO 26
10
SOUND 0,NT(ADDR(256)+16),10,15
FOR DELAY=1 TO 50:NEXT DELAY:SOUND
0,0,0:RETURN
REM TABLE
BREG=AREG
RETURN
REM TBA
BREG=AREG
RETURN
REM ROTATE AREG RIGHT
A=0:IF CFLAG=1 THEN A=8
CFLAG=0:IF INT(AREG/2)<>AREG/2 THEN
CFLAG=1
AREG=INT(AREG/2)+A
RETURN
REM ROTATE AREG LEFT
A=0:IF CFLAG=1 THEN A=1
CFLAG=0:IF AREG>7 THEN CFLAG=1:AREG
=AREG-8
REM AND BREG
GOSUB 3190
FOR INC=1 TO 4
IF AREG$(INC,INC)="1" AND BREG$(INC
,INC)="1" THEN AREG=AREG+BIN(INC+4)
NEXT INC
ZFLAG=0:IF AREG=0 THEN ZFLAG=1
RETURN
REM OR BREG
GOSUB 3190
FOR INC=1 TO 4
IF AREG$(INC,INC)="1" OR BREG$(INC,
INC)="1" THEN AREG=AREG+BIN(INC+4)
NEXT INC
GOTO 2840
REM XOR BREG
GOSUB 3190
FOR INC=1 TO 4
IF AREG$(INC,INC)<>BREG$(INC,INC) T
HEN AREG=AREG+BIN(INC+4)
NEXT INC
GOTO 2840
REM BRANCH IF ZERO
IF ZFLAG=1 THEN GOTO 3120
PC=PC+2:RETURN
REM BRANCH NOT ZERO
IF ZFLAG=0 THEN GOTO 3120
PC=PC+2:RETURN
REM BRANCH IF CARRY
IF CFLAG=1 THEN GOTO 3120
PC=PC+2:RETURN
REM BRANCH IF NOT CARRY
IF CFLAG=0 THEN GOTO 3120
PC=PC+2:RETURN
REM PUSH AREG$,BREG$
AREG$=BIN$(AREG*4+1)
BREG$=BIN$(BREG*4+1)
RETURN
REM END THE PROGRAM
KEY=0
IF POWER=1 THEN RETURN
GOSUB 3610
REM SAVE TO DISK OR TAPE
MD=0
PRINT "SAVE TO"
GOSUB 3740:IF A<>ASC("C") THEN PRINT
"CUE TAPE
IF A<>ASC("D") THEN 3290
PRINT "SAVE FILE:
D1:■4ESC CTRL +■";:INPUT F$:OPEN
#3,8,0,F$
FOR INC=1 TO 256
PUT #3,ADDR(INC)
NEXT INC
CLOSE #3
GOSUB 3650
REM LOAD FROM DISK OR TAPE
MD=1
PRINT "LOAD FROM"
GOSUB 3740:IF A<>ASC("C") THEN PRINT
"CUE TAPE
AND PRESS RET
URN":OPEN #3,4,0,"C:":GOTO 3480
IF A<>ASC("D") THEN 3420
PRINT "LOAD FILE:
D1:■4ESC CTRL +■";:INPUT F$:OPEN
#3,4,0,F$
FOR INC=1 TO 256
GET #3,A:ADDR(INC)=A
NEXT INC
CLOSE #3
GOSUB 3650
```

Continued

NANOPROCESSOR *Continued*

ATARI 800/800XL/130XE

```
3530 RETURN
3540 REM  DISK  OR  TAPE  ERROR  HANDLING
3550 GOSUB 3610
3560 PRINT :PRINT "I/O ERROR ";PEEK(195)
3570 PRINT "PRESS ANY KEY TO CONT
3580 CLOSE #3 :THEN 3290
3590 IF TO 3420
3600 REM CLOSE THE NANOPROCESSOR
3610 POKE 5327 7,0
3620 GRAPHICS 0
3630 RETURN
3640 REM OPEN THE NANOPROCESSOR
3650 GRAPHICS 0
3660 GOSUB 6400
3670 GOSUB 5100
```

```
3690 POKE 53277,3
3700 POWER=1-POWER:KEY=0:GOSUB 1860
3710 GOSUB 1360
3720 TRAP 32768
3730 RUN
3740 REM SAVE LOAD SCREEN
3750 TRAP 3540
3760 POSITION 4,4:PRINT "◢D◣isk"
3770 POSITION 4,5:PRINT "◢C◣asette"
3780 POSITION 4,6:PRINT "◢E◣xit"
3790 GET #2,A
3800 IF A=ASC("E") THEN POP :GOSUB 3650
3810 RETURN
3820 REM INCREMENT PC, ROLL > 255
3830 PC=PC+1
3840 IF PC>255 THEN PC=PC-256
3850 RETURN
```

HCM

NANOPROCESSOR

COMMODORE 64

```
100 REM ********************
110 REM *  NANOPROCESSOR  *
120 REM ********************
130 REM COPYRIGHT 1985
140 REM EMERALD VALLEY PUBLISHING CO.
150 REM BY DAVID ROGER WOOD
160 REM AND PETER HCM STAFF
170 REM HOME COMPUTER MAGAZINE
180 REM VERSION 5.5.1
190 REM C-64 5.1 BASIC
200 DIM AD(255),D$(15):DIM NT%(1,31):Z=54272:POKE Z+24,79:RO$=CHR$(147)
210 JN$=CHR$(17)+CHR$(17):POKE 53280,6:POKE 53281,12:POKE 646,11
220 OS=PEEK(44)*256:SW=2
230 GOSUB 2146:GOSUB 5000,5100,5200,5300,5400
240 ON SW+1 GOSUB 5000,5100,5200,5300,5400
250 OS=PEEK(44)*256
260 ON-(KEY=69)-2*(KEY=80)-3*(KEY=60)-3*(KEY=44)-3*(KEY=62)GOTO 2930,280,29
    0
270 ON-(KEY=46)-(KEY=49)-(KEY=50)-(KEY=51)-(KEY=52)GOTO 290:GOTO 250
280 GOSUB 3800:GOSUB 5500:GOTO 340
290 IF(KEY<48)OR(KEY>59)THEN 310
300 GOSUB 3610
310 ON-(KEY=44)-(KEY=60)-2*(KEY=46)-2*(KEY=62)GOSUB 1470,1500
320 ON-(KEY>48)AND(KEY<53)GOSUB 1420:GOTO 2
    040
330 REM MAIN LOOP
340 GOSUB 5600:IF KEY<>80 THEN 370
350 POKE 56101,11:POKE 1869,121:GOSUB 1020
    :POKE 52302
360 REM  BRANCH TO ROUTINE
370 ON-(KEY=66)-2*(KEY=44)-2*(KEY=60)-3*(KEY=46)-3*(KEY=62)GOSUB 1050,1380
    ,1400,-(KEY<53))-2*(KEY=73
380 ON-3*(KEY=82)GOSUB 1420,1080,1210,-(KEY=76)GOSUB 1240:ONRFGOTO 400:GO
    TO 340
390 ON-(KEY=72)-2*(KEY=80)GOTO 420,440:GOTO 450
400 GOSUB 3000
410 ON-(KEY=72)-2*(KEY=80)GOTO 420,440:GOTO 450
420 POKE 1820,81:POKE Z+1820,13:POKE 1821,87+Z,11:RF=0:GOTO 340
430 POKE Z+1820,11:RF=0:GOTO 340
440 POKE 1709+Z,11:RF=0:GOTO 350
450 ON-(CA)+1 GOSUB 1630,1690,1710,1740,1820,1840,1860,1930,2000,2050
    ,2090,2120,2190,2230,2270
460 IF AD(CA)<10 GOSUB 4800
470 ON-(CA)-9 GOSUB 2120,2190,2230,2270,2310,2350
480 ON SW+1 GOSUB 5000,5100,5200,5300,5400:IFRF=0THEN430
490 GOTO 4000
500 TEMP=16*(CA/16-INT(CA/16)):RETURN
510 TEMP=INT(CA/16):RETURN
520 TEMP=AD(CA):RETURN
530 TEMP=AR:RETURN
540 TEMP=BRT:RETURN
550 REM SUB 2990
560 ON-(KEY=66)-(KEY=80)-2*(KEY=60)-2*(KEY=44)-2*(KEY=46)GOTO 620,630,6
    40,650,660,670,680
570 ON-(KEY=49)-(KEY=50)-(KEY=51)-(KEY=52)GOTO 630
580 ON-(KEY=73)-(KEY=82)-(KEY=76)GOTO 620,640,720,610
590 ON-(KEY=133)-2*(KEY=134)GOTO 640,720
600 ON-(KEY=133)-2*(KEY=134)GOTO 640,720
    :GOTO 560
610 REM SUB 3010
620 RUN":SA=1:PRINTRO$;TAB(12)"SAVE F
    ILE":FL$:IFFL$=""THEN840
630 GOSUB 3800:IF DV=8THEN SA=8:FL$=CHR$(34)+FL$+",S,W":GOSUB 2
    960:IF DV=8 THEN GOSUB 3800
640 FOR IT=0TO255
```

```
700 PRINT#1,AD(IT)
710 CLOSE1:CLOSE15:GOTO840
720 IF SA=0:PRINTRO$;TAB(12)"LOAD F
    ILE";FL$:IFFL$=""THEN840
730 INPUT"FILENAME";FL$:IFFL$=""THEN840
740 OPEN1,8,8,FL$+",S,R"
750 DV=8:THENSA=8:FL$=FL$+",S,R"
760 OPEN1,DV,SA,FL$:IF DV=8 THEN GOSUB2
    960:IT=0:THEN720
770 FOR IT=0TO255:INPUT#1,AD(IT)
780 INPUT#1:CLOSE1:CLOSE15:GOTO 840
790 PRINT"■CTRL RVSON■D■CTRL RVSOFF■APE
    ■CTRL RVSOFF■ISK?
800 GET DV$:IF DV$<>"T" AND DV$<>"D" TH
    ENGET DV$:IF DV$<>"T" AND DV$<>"D" TH
    EN810
810 PRINT DV$:IF DV$="T" THEN DV=1:RETU
    RN
820 DV=8:OPEN15,8,15:RETURN
830 PRINT"OS:";GOSUB2540:POKE1508,32:POKE
    1468,32:POKE1469,32:POKE1470,32
840 POKE1510,32:ONSW+1GOSUB500,510,520,
    530,540:GOSUB880:GOSUB950:GOSUB960
850 CLOSE15:CLOSE2:POKE198,0:POKE631,13:
    POKE632,13:POKE633,13:POKE634,13:END
860 REM CONVERT TO BINARY
870 REM $=D$(TEMP) TO SCREEN
880 REM WRITE TO 1517 TO 1529 STEP 4:CH=(IT-1
    )/4:IFMID$(PL$,CH,1)="1"THEN920
890 POKE54272+IT,11:GOTO930
900 POKE54272+IT,2
910 NEXT:RETURN
920 POKE54272+IT,1
930 NEXT:RETURN
940 REM ADDRESS
950 AD$=D$(INT(CA/16)):AD$=AD$+D$(CA-16
    *INT(CA/16)) ADDRESS TO SCREEN
960 RFOR IT=13TO01334STEP3:CH=(IT-1310)/
    3:IFMID$(AD$,CH,1)="1"THEN990
970 POKE54272+IT,11:GOTO1000
980 POKE54272+IT,2
990 NEXT:RETURN OF Z+1334STEP3:POKEIT,11
1000 REM LIGHT Z+1334TOZ+1517STEP4
1010 REM TEST Z+1517TOZ+1529STEP4
1020 IFSW>2THEN1150
1030 IFSW<2 THEN1110
1040 IFTEMP=AD(CA):GOTO1140
1050 IFSW=0 THEN1130
1060 GOSUB5100
1070 GOSUB5200
1080 GOSUB5300
1090 GOSUB5400
1100 POKE15808,87:POKEZ+15808,11:RETURN
1110 IFCA<256 THEN1190
1120 CA=CA-256
1130 RETURN
1140 REM RUN
1150 POKE17009,87:POKEZ+17009,13:POKE17
    010,81:POKEZ+17010,11:GOSUB2440:RF=1:
    RETURN
1160 POKE17009,87:POKEZ+17009,11:RETURN
1170 REM LOAD
1180 POKE1940,81:POKEZ+1940,13:CT=0:TEMP
    =0:PLS=":FORIT=1809TO1797STEP-4:TEMP=
    TEMP+(PEEK(IT)):IFM=120THENM=121:GOTO127
    0
1190 M=120
1200 PLS=CHR$(M-72)+PLS:TEMP=TEMP+((M-12
    0)*2^CT):CT=CT+1:NEXTIT
1210 ON SW+1GOSUB1290,1300,1320,1320,1320,13
    20,1340
1220 CA=(INT(CA/16)*16+TEMP):GOTO1310
1230 CA=CA-8:GOSUB950:RETURN
1240 GOSUB900
```

Continued

TYPE-IN LISTINGS

NANOPROCESSOR *Continued*


```
K 1340 IF CA<253 THEN1360
L 1350 ON CA-252 GOSUB1790,3040,3060
G 1360 RETURN
O 1370 REM ROTATE TO LEFT
H 1380 GOSUB1470:GOTO1410
N 1390 REM ROTATE TO RIGHT
Z 1400 GOSUB1500
U 1410 ON SW+1 GOSUB500,510,520,530,540:GO
   SUB8880:RETURN
L 1420 KEY=KEY-48:IFPEEK(KEY*4+17793)=121TH
   EN1440
K 1430 POKEKEY*4+17793,121:GOTO1450
L 1440 POKEKEY*4+17793,120
G 1450 GOSUB3010:RETURN
B 1460 ON SW+1 GOSUB500,510,520,530,540:GO
   SUB8880:GOSUB950:RETURN
L 1470 IF SW=0 THEN1490
X 1480 SW=SW-1:GOSUB3010:GOSUB1530
O 1490 RETURN
D 1500 IF SW=4 THEN1520
P 1510 SW=SW+1:GOSUB3010:GOSUB1530
K 1520 RETURN
X 1530 ON SW+1 GOSUB1540,1550,1570,1590,16
   10:RETURN
V 1540 POKE1468,32:POKE1508,64:POKEZ+1468,
   13:POKEZ+1508,11:RETURN
J 1550 POKE1508,32:POKE1469,32:POKE1468,77
J 1560 POKEZ+1508,13:POKEZ+1469,13:POKEZ+1
   468,11:RETURN
P 1570 POKE1468,32:POKE1470,32:POKE1469,66
P 1580 POKEZ+1468,13:POKEZ+1470,13:POKEZ+1
   469,11:RETURN
I 1590 POKE1469,32:POKE1510,32:POKE1470,78
E 1600 POKEZ+1469,13:POKEZ+1510,13:POKEZ+1
   470,11:RETURN
N 1610 POKE1470,32:POKE1510,64:POKEZ+1470,
   13:POKEZ+1510,11:RETURN
C 1620 REM ADD
A 1630 AR=BR+AR
A 1640 IFAR<16THEN1660
T 1650 AR=AR-16:CF=1:GOTO1670
H 1660 CF=0
I 1670 GOTO2380
X 1680 REM IMMEDIATE
S 1690 CA=CA+1:GOSUB1170:AR=AD(CA):GOTO238
   0
I 1700 REM LDA FROM MEMORY
N 1710 CA=CA+1:GOSUB1170:TEMP=AD(CA):CA=CA
   +1:GOSUB1170:TEMP=TEMP+16*AD(CA)
U 1720 AR=AD(TEMP):GOTO2380
L 1730 REM STA
L 1740 CA=CA+1:GOSUB1170:TEMP=AD(CA):CA=CA
   +1:GOSUB1170:TEMP=TEMP+16*AD(CA)
J 1750 AD(TEMP)=AR
J 1760 IF TEMP<253 THEN1780
I 1770 ON TEMP-252 GOSUB1790,3040,3060
M 1780 CA=CA+1:GOSUB1170:RETURN
O 1790 IF AD(253)=0 THEN PRINT"■HOME■2 CR
   SRDOWN■ ■CMDR CYN■■SHIFT Q■":RETURN
U 1800 PRINT"■HOME■2 CRSRDOWN■ ■CTRL RED
   ■SHIFT Q■":RETURN
N 1810 REM TAB
V 1820 BR=AR:GOTO2380
D 1830 REM TBA
D 1840 AR=BR:GOTO2380
R 1850 REM RRC
T 1860 TEMP=0:IFCF=0THEN1880
V 1870 TEMP=8
Y 1880 IF INT(AR/2)=AR/2 THEN1900
B 1890 CF=1:GOTO1910
M 1900 CF=0
S 1910 AR=INT(AR/2)+TEMP:GOTO2380
T 1920 REM RLC
R 1930 TEMP=0:IFCF=0THEN1950
G 1940 TEMP=1
P 1950 IF AR<8 THEN1970
Y 1960 CF=1:AR=AR*2-16+TEMP:GOTO1980
T 1970 CF=0:AR=AR*2+TEMP
M 1980 GOTO2380
Y 1990 REM AND
Z 2000 GOSUB2370:FORIT=1TO4
R 2010 ON-((MID$(AR$,IT,1)="1"))GOTO2020:GOTO2030
O 2020 AR=AR+(2↑(4-IT))
N 2030 NEXT IT:GOTO2380
R 2040 REM OR
A 2050 GOSUB2370
X 2060 FOR IT=1 TO 4
T 2070 ON-((MID$(AR$,IT,1)="1"))OR(MID$(BR$
   ,IT,1)="1"))GOTO2080:GOTO2090
O 2080 AR=AR+(2↑(4-IT))
N 2090 NEXT IT
Z 2100 GOTO2380
V 2110 REM XOR
J 2120 GOSUB2370
Z 2130 FOR IT=1 TO 4
A 2140 ON-((MID$(AR$,IT,1)="1"))<>(MID$(BR$
   ,IT,1)="1"))GOTO2020:GOTO2030
K 2150 AR=AR+(2↑(4-IT))
N 2160 NEXT IT
G 2170 GOTO2380
H 2180 REM BZ
W 2190 IF ZF=0 THEN2210
X 2200 GOTO2350
F 2210 CA=CA+3:GOSUB1170:RETURN
I 2220 REM BNZ
M 2230 IF ZF=1 THEN2250
P 2240 GOTO2350
N 2250 CA=CA+3:GOSUB1170:RETURN
K 2260 REM BCS
A 2270 IF CF=0 THEN2290
O 2280 GOTO2350
B 2290 CA=CA+3:GOSUB1170:RETURN
P 2300 REM BCC
P 2310 IF CF=1 THEN2330
E 2320 GOTO2350
K 2330 CA=CA+3:GOSUB1170:RETURN
Q 2340 REM JMP
U 2350 CA=CA+1:GOSUB1170:TEMP=AD(CA):CA=CA
   +1:GOSUB1170:TEMP=TEMP+16*AD(CA)
E 2360 CA=TEMP:RETURN
O 2370 AR$=D$(AR):BR$=D$(BR):AR=0:CF=0:RET
   URN
Y 2380 IF AR=0 THEN2410
N 2390 ZF=0
B 2400 GOTO2420
H 2410 ZF=1
E 2420 CA=CA+1:GOSUB1170:RETURN
O 2430 REM DELAY
G 2440 FORD=1TO300:NEXTD:RETURN
C 2450 REM RANDOMIZE REGISTERS
R 2460 AR=INT(RND(0)*15)+1
B 2470 BR=INT(RND(0)*15)+1
Q 2480 CA=INT(RND(0)*255)+1
H 2490 AD(CA)=INT(RND(0)*15)+1
R 2500 RETURN
N 2510 REM INITIALIZATION
Z 2520 FOR IT=0 TO 15:READ D$(IT):NEXT IT
L 2530 FOR IT=0 TO 1:FOR JT=0 TO 31:READ N
   T%(IT,JT):NEXT:NEXT
R 2540 POKE53265,PEEK(53265)AND239
Y 2550 PRINTRO$;JN$;"■SHIFT Q■"TAB(12)"N
   ANOPRO OCESSOR
A 2560 PRINT■2 CRSRDOWN■■ 128 64 32
   16 8 4 2 1"
M 2570 PRINTTAB(7)CHR$(167);:FORI=1TO24:PR
   INTCHR$(163);:NEXT:PRINTCHR$(165)
K 2580 PRINTTAB(7)CHR$(167);:FORI=1TO8:PRI
   NT"CHR$(113)";:NEXT:PRINTCHR$(1
   65)
J 2590 PRINTTAB(7)CHR$(167)TAB(32)CHR$(165
   )
L 2600 PRINTTAB(8);:FORI=1TO24:PRINTCHR$(1
   63);:NEXT:PRINT"H"TAB(5)"M"TAB(7)"A";
O 2610 PRINTTAB(3)"H"TAB(5)"M"TAB(7)"A";
W 2620 PRINTTAB(12)" 8 4 2 1
   ■SHIFT W■"
T 2630 PRINTTAB(5)CHR$(98);
P 2640 PRINTTAB(11)CHR$(167);:FORI=1TO16:P
   RINTCHR$(163);:NEXT:PRINTCHR$(165);
H 2650 PRINTSPC(5)"B"CHR$(155)"EGIN"CHR$(1
   51)
Y 2660 PRINTTAB(3)"L"TAB(5)CHR$(113)TAB(7)
   "BRANCH";
G 2670 PRINTTAB(11)CHR$(167);:FORI=1TO4:PR
   INT" "CHR$(113)";:NEXT
H 2680 PRINTCHR$(165):PRINTTAB(4)"<"TAB(6)
   ">";
Z 2690 PRINTTAB(11)CHR$(167)TAB(28)CHR$(16
   5)SPC(7)CHR$(119)
C 2700 PRINTTAB(12);:FORI=1TO16:PRINTCHR$(
   163);:NEXT:PRINTSPC(6)"I"CHR$(155)"
   NC"
R 2710 PRINT:PRINTCHR$(151)TAB(36)CHR$(119
   )
J 2720 PRINTTAB(5)CHR$(113);
F 2730 PRINTTAB(11)CHR$(167);:FORI=1TO16:P
   RINTCHR$(163);:NEXT:PRINTCHR$(165);
Z 2740 PRINTSPC(5)"R"CHR$(155)"UN"CHR$(151
   )
J 2750 PRINT" BUSY";
X 2760 PRINTTAB(11)CHR$(167);:FORI=1TO4:PR
   INT" CHR$(49)";:NEXT:PRINTCHR$(
   165)
O 2770 PRINTTAB(11)CHR$(167);:FORI=1TO4:PR
   INT" CHR$(185)";:NEXT
L 2780 PRINTCHR$(165)SPC(7)CHR$(119)
A 2790 PRINTTAB(5)CHR$(113);
W 2800 PRINTTAB(11)CHR$(167);:FORI=1TO4:PR
   INT" CHR$(48)";:NEXT
N 2810 PRINT" CHR$(165);:PRINTSPC(5)"H"CHR$(
   155)"ALT"CHR$(151)
M 2820 PRINTTAB(5)CHR$(185)TAB(11)CHR$(167
   )TAB(28)CHR$(165)
M 2830 PRINT" POWER";
P 2840 PRINTTAB(12);:FORI=1TO16:PRINTCHR$(
   163);:NEXT:PRINTTAB(36)CHR$(119)
R 2850 PRINTTAB(34)"L"CHR$(155)"OAD"CHR$(1
   51)
B 2860 POKE53265,PEEK(53265)OR16:RETURN
U 2870 DATA 0,0001,0010,0011,0100,0101,
   0110,0111,1000,1001,1010,1011,1100,
   1101
F 2880 DATA 1110,1111
U 2890 DATA 7,8,8,9,9,10,11,11,12,13,14,14
   ,15,16,17,18,19,21,22,23,25,26,28,2
   9,31
A 2900 DATA 33,35,37,39,42,44,47
I 2910 DATA 233,97,225,104,247,143,48,218,
   143,78,24,239,210,195,195,209,239,3
   1,96
C 2920 DATA 181,30,156,49,223,165,135,134,
   162,223,62,193,107
P 2930 PRINTRO$;
W 2940 END
```

Continued

NANOPROCESSOR *Continued*

COMMODORE 64

```
J 2950  REM ERROR HANDLING
A 2960  INPUT #15,E,E$
F 2970  IF E THEN PRINT "***  "E$"  ***":FOR I=
        1 TO 1000:NEXT:CLOSE 15:CLOSE 1:CLOSE 15
T 2980  RETURN
L 2990  POKE198,0:WAIT197,64:WAIT197,64,64
Q 3000  GET KEY$:KEY=ASC(KEY$+CHR$(0)):RETUR
N
D 3010  FOR I=0TO23:POKEZ+I,0:NEXT:POKEZ,240
        :POKEZ+1,33:POKEZ+5,8:POKEZ+22,104
C 3020  POKEZ+23,1:POKEZ+4,129:POKEZ+24,79:
F       POKEI=0TO25:NEXT:POKEZ+4,128
P       POKEZ+13,240:RETURN
L 3040  POKEZ+24,15:POKEZ+8,NT%(0,AD(254)):
        POKEZ+11,33
C 3050  FOR IT=1 TO 300:NEXT IT:POKEZ+11,0
        :POKEZ+24,79:RETURN
J 3060  POKEZ+24,15:POKE Z+8,NT%(0,AD(255)
        +16):POKEZ+11,33:FOR IT=1 TO 300:NEXT I
I 3070  T:POKEZ+11,0:POKEZ+24,79:RETURN
```

HCM

NANOPROCESSOR

IBM PC/PCjr, TANDY 1000

```
A 100   '  ********** NANOPROCESSOR **********
F 110   '  *                                *
Y 120   '  *  COPYRIGHT 1985 BY RALPH VALLEY  *
X 130   '  *  EMERALD VALLEY PUBLISHING CO.  *
A 140   '  *  BY RODGER HOOTS AFF            *
M 150   '  *  AND HOMEBREW COMPUTER MAGAZINE *
Q 160   '  *  VERSION 2.1 FOR IBM PC/jr      *
Z 170   '  *  DOS 2.1 AND WITH 128K          *
L 180   '  *  IBM COLOR/GRAPHICS ADAPTER     *
E 190   '  *  COLOR/GRAPHICS MONITOR         *
L 200   '  *  AND COMPATIBLES                *
N 210   '  *  CORFINTY AF:SCREEN 1:COLOR 1,0:CLS:SW=
G 220   DEFINT A-Z:DIM D%(255),D$(15),P1(40),P2(40),P
C 230   DIM M(40),P4(40),P5(40),UP(40),DN(40),ONL
B 240   DIM T(40),OFFL(40),NT$(31)
Q 250   FOR I=1 TO 10:KEY I,"":NEXT:GOSUB
K 260   FOR I=1 TO LEN(K$):PRINT ASC(MID$(K
L 270   NEXT:DOT(40),LDOT(40),SC(4):GOSUB 2
K 280   GOSUB 2480:ON SW + 1 GOSUB 550,560,
C 290   80,590:IF K$="" THEN 310
B 300   INKEY$:IF K$<"," OR K$>"z" THEN 310
A 310   IF INSTR("AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz",K$):
C 320   IF INSTR("ABCDEFGHIJKLMNOPQRSTUVWXYZ",K$)THEN K$=CHR$(ASC(K$)+32)
F 330   IF K$="h" THEN 2740
L 340   IF K$="l" THEN 2780
B 350   IF K$="p" THEN 2820
P 360   IF K$="q" THEN CLS:END ELSE 310
G 370   GOSUB 930:GOTO 420
B 380   IF K$="t" THEN 390 ELSE 1530
N 390   GOSUB 1530,1580,1580,1580,14
X 400   GOTO 1420,1400,1400,1420,14
T 410   GOTO 1420,1400,1400,1420,12
R 420   IF K$="r" THEN 1070 ELSE 1200
M 470   IF K$="m" THEN 480 ELSE 510
K 480   IF K$="k" THEN 490 ELSE 520
I 490   IF K$="i" THEN 1070 ELSE 1200
V 500   IF K$="v" THEN 1070 ELSE 1200
R 510   IF K$="r" THEN 1070 ELSE 1200
O 520   IF K$="o" THEN 530 ELSE 540
C 530   IF K$="c" THEN 550 ELSE 560
D 540   IF K$="d" THEN 570 ELSE 580
E 550   IF K$="e" THEN 590 ELSE 600
K 560   IF K$="k" THEN 610 ELSE 620
C 570   IF K$="c" THEN 630 ELSE 640
B 580   IF K$="b" THEN 650 ELSE 660
A 590   IF K$="a" THEN 670 ELSE 680
O 600   IF K$="o" THEN 690 ELSE 700
O 700   GOTO 770
U 710   ON ERROR GOTO 3300
K 720   IF FL$="" THEN 770
S 730   OPEN FL$ FOR INPUT AS 1:FOR IT = 0
M 740   INPUT #1,AD(IT+1),AD(IT+2),A
        D(IT+3),AD(IT+4),AD(IT+5),AD(IT+6),
E 750   NEXT IT
F 760   ERROR 0:LOCATE 1,1,0:SCREEN
A 770   GOTO 0:CLS:P1=0:P2=0:GOSUB 27
        50
AG 780  GOSUB 550,560,570,580,590
AF 790  GOSUB 860:GOSUB 2100:STEP 40:IF SC(((IT
        -1)/40)+1)=0 THEN PUT (0,150),UP,PSE
WE 810  NEXT:PUT (25,160),UP,PSET:PUT (25,145),O
GY 840  THIS IS THE END OF THE CONTROL LOOP
X 850   'CONVERT TO BINARY NUMBER
XD 860  'CLS:DIM WRITE TO SCREEN
NA 870  FOR I=0 TO 211 STEP 40
CJ 880  IF M(IT/40)=1 THEN PUT (0,I),DN,PSET
HP 890  IF M(IT/40)=1 THEN PUT (C
VQ 900  *PUT TO ON,PSET
BH 930  'SET ADDRESS
HE 940  'ADDRESS TO SCREEN
DT 950  'SCREEN ADDRESS
FR 960  FOR IT=0 TO 254 STEP 30
Q 990   IT,32),ONLIT,PSET:GOTO 1000
VN 1000 'BUTTON
NA 1010 'BUTTON
HT 1030 SW + 1 GOSUB 550,560,570,580,590
TY 1040 GOSUB 860:GOSUB 2100:STEP 40:IF SC(((IT
        -1)/40)+1)=0 THEN PUT (0,150),UP,PSE
YX 1050 'CONVERT TO BINARY NUMBER
XC 1060 FOR IT=0 TO 254 STEP 30
CZ 1080 'SCREEN ADDRESS
ZJ 1090 'SCREEN TO ADDRESS
RR 1100 'BUTTON
RK 1110 'BUTTON
KB 1120 'BUTTON
BC 1130 'BUTTON
CH 1140 'BUTTON
HK 1150 'BUTTON
KE 1160 'BUTTON
EC 1170 'BUTTON
CE 1180 'BUTTON
EF 1190 'BUTTON
FH 1200 'BUTTON
HJ 1210 'BUTTON
YN 1220 'BUTTON
NF 1230 'BUTTON
FG 1240 'BUTTON
GF 1250 'BUTTON
MD 1260 'BUTTON
DM 1270 'BUTTON
AF 1280 'BUTTON
FA 1290 'BUTTON
ZI 1310 'BUTTON
IA 1320 'BUTTON
QI 1330 'BUTTON
IY 1340 'BUTTON
OU 1350 'BUTTON
```

Continued

TYPE-IN LISTINGS

NANOPROCESSOR *Continued*

IBM PC/PCjr, TANDY 1000

```
A 1360  IF CA=254 THEN PLAY NT$(AD%(254)):G
   1380  OTO 1380
V 1370  PLAY NT$(AD%(255)+16)
W 1380  RETURN
J 1390  'ROTATE TO LEFT
V 1400  GOSUB 1530:GOTO 1430
M 1410  'OSTATE RIGHT
L 1420  OSUB 1580
T 1430  ONOSUBW 1+86 GOSUB 550,560,570,580,590
T 1440  OSUBC CC):RETURN
M 1450  IF SC(C)=1 THEN 1490
F 1460  OTO(C)=1:PUT (C*40+51,150),UP,PSET
O 1470  OTO(C)=0:PUT (C*40+50,150),DN,PSET
S 1480  RETURN
R 1490  RONOSUBW +1 GOSUB 550,560,570,580,590
I 1500  OSUBW +86:GOSUB 930:RETURN
X 1510  IF SW =0 THEN 1570
R 1520  SW =SW
T 1530  $=K$:K$="":GOSUB 1630
   1540  $=$ THEN 1570
I 1550  OTO 1530
Z 1560  ETURN
A 1570  IF SW=4 THEN 1620
Y 1580  SW+1:GOSUB 1630
D 1590  $=K$:K$=INKEY$:IF K$="" OR K
   1600  $=$ THEN 1620
J 1610  ETURN
N 1620  ET SW+1 GOSUB 1650,1660,1670,1680,1
   1630  0
N 1640  ETURN
C 1650  PUT (25,95),P1,PSET:RETURN
W 1660  PUT (25,95),P2,PSET:RETURN
I 1670  PUT (25,95),P3,PSET:RETURN
M 1680  PUT (25,95),P4,PSET:RETURN
I 1690  PUT (25,95),P5,PSET:RETURN
K 1700  'ADD ROUTINE
   1710  AR=AR+B:IF AR < 16 THEN CF=0:GOTO
   1720  440
J 1720  AR=AR-16:CF=1:GOTO 2440
A 1730  'LDA IMMEDIATE
U 1740  CA=CA+1:GOSUB 1170
V 1750  AR=AD%(CA):GOTO 2440
Z 1760  'LDA FROM MEMORY
J 1770  CA=CA+1:GOSUB 1170
N 1780  TEMP=AD%(CA):CA=CA+1
A 1790  GOSUB 1170
L 1800  TEMP=TEMP+16*AD%(CA):AR=AD%(TEMP):G
   1810  OTO 2440
C 1820  'STORE "A" REGISTER IN MEMORY
N 1830  CA=CA+1:GOSUB 1170
M 1840  TEMP=AD%((CA)):CA=CA+1:GOSUB 1170
   1850  TEMP=TEMP+16*AD%(CA):AD%(TEMP)=AR:C
E 1850  F TEMP<253 THEN 1880 ELSE IF TEMP>
   1860  253 THEN 1880 ELSE IF AD%(253)>0 THE
   1870  N PUT (2,6),ONFLIT,PSET:GOTO 1880
   1880  SE PUT (2,6),OFFLIT,PSET:GOTO 1880
A 1860  IF TEMP=254 THEN PLAY NT$(AD%(254))
   1880  :GOTO 1880
S 1870  PLAY NT$(AD%(255)+16)
L 1880  RETURN
E 1890  'TAB
T 1900  'BR=AR:GOTO 2440
T 1910  'TBA
F 1920  'BR=AR:GOTO 2440
N 1930  'RC
K 1940  TEMP=0:IF CF=0 THEN 1960
B 1950  TEMP=8
J 1960  IF INT(AR/2)=AR/2 THEN 1980
A 1970  F=1:GOTO 2440
N 1980  F=0:AR=INT(AR/2)+TEMP:GOTO 2440
H 1990  'RLC
   2000  EMP=0:IF CF=0 THEN 2010 ELSE TEMP=
Z 2010  F AR < 8 THEN 2030
G 2020  F=1:AR=AR*2-16+TEMP:GOTO 2040
X 2030  F=0:AR=AR*2+TEMP
T 2040  OTO 2440
N 2050  'AND
O 2060  OSUB 2430:FOR IT = 1 TO 4
L 2070  J1$=MID$(AR$,IT,1):J2$=MID$(BR$,IT,
   1)
G 2080  F J1$="1" AND J2$="1" THEN AR=AR+(
   2^(4-IT))
G 2090  EXT IT:GOTO 2440
V 2100  'OR
O 2110  OSUB 2430:FOR IT = 1 TO 4
N 2120  J1$=MID$(AR$,IT,1):J2$=MID$(BR$,IT,
   1)
I 2130  F J1$="1" OR J2$="1" THEN AR=AR+(2
   ^(4-IT))
B 2140  EXT IT
P 2150  OTOR 2440
N 2160  'XOR
Z 2170  OSUB 2430
E 2180  OR IT = 1 TO 4
W 2190  J1$=MID$(AR$,IT,1):J2$=MID$(BR$,IT,
   1)
D 2200  F J1$ <> J2$ THEN AR=AR+(2^(4-IT))
Z 2210  EXT IT
F 2220  OTO 2440
Z 2230  'BRANCH ON ZERO
T 2240  IF ZF=0 THEN 2260
C 2250  GOTO 2400
K 2260  CA=CA+3:GOSUB 1170:RETURN
K 2270  'BRANCH ON NOT ZERO
F 2280  IF ZF=1 THEN 2300
A 2290  GOTO 2400
T 2300  CA=CA+3:GOSUB 1170
Y 2310  RETURN
Q 2320  'BC
A 2330  IF CF=0 THEN 2350
K 2340  GOTO 2400
Y 2350  CA=CA+3:GOSUB 1170:RETURN
X 2360  'RC
N 2370  IF CF=1 THEN 2350
A 2380  GOTO 2400
O 2390  'JUMP
M 2400  'CA=CA+1:GOSUB 1170
R 2410  TEMP=CA+ADD%((CA)):CA=CA+1:GOSUB 1170
D 2420  TEMP=TEMP+16*AD%((CA)):CA=TEMP:RETURN
F 2430  AR$=D$(AR):BR$=D$(BR):AR=0:CF=0:RET
   2440  URN
E 2440  IF AR=0 THEN 2460
Q 2450  GOTO 2470
Z 2460  F=1
W 2470  CA=CA+1:GOSUB 1170:RETURN
J 2480  'RANDOMIZE TIMER
C 2490  AR=INT(RND*15)+1:BR=INT(RND*15)+1
L 2500  CA=INT(RND*252)+1
C 2510  CA=INT(RND*15)+1:RETURN
D 2520  'CLS
P 2530  'PRINT"Nano Processor"
M 2540  RD$(IT)):NEXT IT:FOR IT=0 TO 31:READ
   2550  N=N+$:IF N=0 THEN GOTO 2580 ELSE RE
E 2550  AD GS(N*20,20):DRAW GS
N 2560  I 9/8:PAINT (N*20,21):IF N<10 THEN
   2600  ,9/8:PAINT (N*20,20),4,,
H 2570  PAINT (N*20,21):IF N<10 THEN 2600
Z 2580  IF N=10 THEN CIRCLE (N*20,20),3,,,,
   2590  9/8
E 2590  IF N=11 THEN CIRCLE (N*20,20),3,2,,
   2600  ,9/8:PAINT (N*20,20),2
J 2600  O 26800,26900,27000,27100,27200,2730
   2610  0,IFLS: N<12 THEN GOTO 2550
J 2610  CLS: GOTO 2750
V 2620  'PRINT(N*20-8,12)-(N*20+8,28),P1:RETU
I 2630  RNET (N*20-8,12)-(N*20+8,28),P2:RETU
   2640  RNET (N*20-8,12)-(N*20+8,28),P3:RETU
A 2640  RNET (N*20-8,12)-(N*20+8,28),P4:RETU
K 2650  RNET (N*20-8,12)-(N*20+8,28),P5:RETU
M 2660  RNET (N*20-8,12)-(N*20+8,28),DN:RETU
G 2670  RNET (N*20-8,12)-(N*20+8,28),UP:RETU
M 2680  RNET (N*20-8,12)-(N*20+8,28),BDOT:RE
K 2690  RNET (N*20-8,12)-(N*20+8,28),LDOT:RE
   2700  RNET (N*20-8,12)-(N*20+8,28),OFFLIT:
S 2700  'RETURN
F 2710  'GET (N*20-8,12)-(N*20+8,28),ONLIT:R
   2720  'RETURN
F 2720  'GET (N*20-8,12)-(N*20+8,28),ONLIT:R
   2730  'GET (N*20-8,12)-(N*20+8,28),ONLIT:R
A 2730  'GET RUN
   2740  'KEY CLS:END
I 2740  LOCATE 1,15:PRINT"Nano Processor"
A 2750  LOCATE 4,1:PRINT"Output"
R 2760  'LINE (35,24)-(285,60),1,B
X 2770  'LINE (38,27)-(282,56),1,B:LOCATE 2,
U 2780  'PRINT"128 64 32 16 8 4 2 1
T 2790  LOCATE 7,17:PRINT"Address"
Y 2800  'LINE (76,88)-(247,115),1,B
X 2810  'LINE (73,85)-(250,115),1,B
F 2820  LOCATE 10,13
E 2830  'PRINT"8 4 2 1"
A 2840  LOCATE 13,3:PRINT"Busy"
F 2850  'LINE (73,130)-(250,188),1,B
L 2860  'LINE (76,133)-(247,185),1,B
Z 2870  LOCATE 17:PRINT"Switches"
N 2880  LOCATE 22,13:PRINT"0 0 0 0 0 0 0 0
D 2890  LOCATE 18,13:PRINT"1 1 1 1 1 1 1 1
N 2900  LOCATE 18,3:PRINT"Busy"
J 2910  LOCATE 23,2:PRINT"Power"
B 2920  LOCATE 14,1:PRINT"PLOT"
Q 2930  LOCATE 12,1:PRINT"HOME"
W 2940  LOCATE 11,5:PRINT"MEMORY"
W 2950  LOCATE 11,34:PRINT"Begin"
P 2960  LOCATE 14,34:PRINT"Incnlt"
Y 2970  LOCATE 17,34:PRINT"RHalo"
X 2980  LOCATE 20,34:PRINT"gcnltd"
C 2990  LOCATE 23,34:PRINT"
N 3000  PAINT (39,25),,1
O 3010  PAINT (74,86),,1
F 3020  PAINT (75,132),,1
F 3030  FOR IT=64 TO 180:PUT STEP (24:PUT (270,I
A 3040  T),,B DOT:NEXT IT:PUT STEP (25,95),P3,PSET
   3050  OR DN,(2,6),,PSET:NEXT IT,PUT STEP (40:PUT (IT,15
Q 3040  F=90 TO 210:PUT STEP (270,I
W 3050  ),,PSET:NEXT IT,PUT STEP (40:PUT (IT,94
H 3060  FOR F=90 TO 210:PUT STEP (270,I
```

Continued

NANOPROCESSOR *Continued*

IBM PC/PCjr, TANDY 1000

```
F 3070 FOR IT=44 TO 260 STEP 30:PUT (IT,32),OFFLIT,PSET:NEXT IT
Y 3080 PUT(25,120),OFFLIT,PSET:PUT(25,160),DN,PSET:PUT(25,145),OFFLIT,PSET:PUT(2,6),OFFLIT,PSET:AD%(253)=0:RETURN
A 3090 DATA "0000","0001","0010","0011","0100","0101","0110","0111","1000","1001","1010","1011","1100","1101","1110","1111"
N 3100 DATA "O1L2B","O2L2C","O2L2C#","O2L2D","O2L2D#","O2L2E","O2L2F","O2L2F#","O2L2G","O2L2G#","O2L2A","O2L2A#","O2L2B","O3L2C","O3L2C#","O3L2D"
N 3110 DATA "O3L2D#","O3L2E","O3L2F","O3L2F#","O3L2G","O3L2G#","O3L2A","O3L2A#","O3L2B","O4L2C","O4L2C#","O4L2D","O4L2D#","O4L2E","O4L2F","O4L2F#","O4L2G"
K 3120 DATA "BL3L6UR7","BH3H4DF5","BU3U7RD7"
S 3130 DATA "BE3E5DG5","BR3R6UL7"
L 3140 DATA "UL3D2RD3R5U3RU2L6","BD2R3U2LU3L5D3LD2R4"
X 3150 DATA "BL3D4R7U8L7D5R","BL2D3R5U6L5D5"
Y 3160 LOCATE 5,1:PRINT "INPUT FILE NAME:";:ROW=5:COL=18:MAXLEN=8:SELECT$="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%&(){}'`-_":IN$="":GOSUB 3190:IF IN$="" THEN RETURN ELSE FL$=IN$
B 3170 LOCATE 7,1:PRINT "WHICH DRIVE? A":ROW=7:COL=14:MAXLEN=1:SELECT$="AaBb":IN$="A":GOSUB 3190:IF IN$="" THEN FL$="A:"+FL$:RETURN ELSE FL$=IN$+":"+FL$:RETURN
F 3180 ' INPUT SUBROUTINE *****
X 3190 PT=1
P 3200 LOCATE ROW,COL,0:PRINT IN$;SPACE$(MAXLEN-LEN(IN$));:LOCATE ROW,COL+(PT-1),1:K$="":WHILE K$="":K$=INKEY$:WEND
C 3210 IF K$=CHR$(13) THEN RETURN
X 3220 IF INSTR(SELECT$,K$) THEN IN$=LEFT$(IN$,PT-1)+K$+MID$(IN$,PT+1):PT=PT+1:IF PT>MAXLEN THEN PT=MAXLEN:GOTO 3200
K 3230 IF K$=CHR$(8) AND PT>1 THEN IN$=LEFT$(IN$,PT-2)+MID$(IN$,PT):PT=PT-1:GOTO 3200
Z 3240 IF K$=CHR$(0)+CHR$(83) THEN IN$=LEFT$(IN$,PT-1)+MID$(IN$,PT+1):GOTO 3200
J 3250 IF K$=CHR$(0)+CHR$(82) AND LEN(IN$)<MAXLEN THEN IN$=LEFT$(IN$,PT-1)+" "+MID$(IN$,PT):GOTO 3200
H 3260 IF K$=CHR$(0)+CHR$(77) AND LEN(IN$)>=PT THEN PT=PT+1:IF PT>MAXLEN THEN PT=MAXLEN:BEEP:GOTO 3200
I 3270 IF K$=CHR$(0)+CHR$(75) AND LEN(IN$)>1 THEN PT=PT-1:IF PT<1 THEN PT=1:GOTO 3200
N 3280 GOTO 3200
M 3290 DIM ERCD(14),ERM$(14):RESTORE 3350:FOR I=1 TO 14:READ ERCD(I):READ ERM$(I):NEXT:RETURN
E 3300 ' ERROR ROUTINES *****
U 3310 CLOSE:LOCATE 24,1,0:R=ERR:L=ERL:FOR Z=1 TO 14:IF ERCD(Z)=R THEN 3330
O 3320 NEXT:PRINT "ERROR #";R;" IN LINE #";L;:GOTO 3340
H 3330 PRINT ERM$(Z);"-- #";R;
N 3340 SOUND 110,20:FOR TD=1 TO 4000:NEXT:LOCATE 24,1:PRINT SPACE$(39);:IF MD1=1 THEN RESUME 660 ELSE RESUME 710
N 3350 DATA 64,BAD FILE NAME,69,COMMUNICATIONS BUFFER OVERFLOW,25,DEVICE FAULT,57,DEVICE I/O ERROR,24,DEVICE TIMEOUT,68,DEVICE UNAVAILABLE,61,DISKETTE IS FULL,72,DISK MEDIA ERROR,71,DISK NOT READY,70,THIS DISK IS WRITE PROTECTED
O 3360 DATA 53,FILE IS NOT ON THE DISK,14,DATA STORAGE AREA FULL--START NEW FILE,67,TOO MANY FILES ON THIS DISK,52,BAD FILE NUMBER OR NAME
```

HCM

NANOPROCESSOR

TI-99/4A

```
P 100 REM *****************
Q 110 REM * NANOPROCESSOR *
P 120 REM *****************
M 130 REM COPYRIGHT 1985
H 140 REM EMERALD VALLEY PUBLISHING CO.
A 150 REM BY ROGER WOOD
R 160 REM HOME COMPUTER MAGAZINE
G 170 REM VERSION 5.5.1
T 180 REM TI BASIC OR EXTENDED BASIC
A 190 DIM AD(255)
N 200 DIM D$(15)
O 210 DIM TN(31)
C 220 GOSUB 3940
Y 230 SW=2
N 240 CALL SCREEN(4)
E 250 GOSUB 3880
F 260 ON SW+1 GOSUB 710,730,750,770,790
X 270 CALL KEY(0,K,S)
N 280 IF S<1 THEN 270
F 290 C=POS("EP<,>.1234",CHR$(K),1)
K 300 IF C=0 THEN 270
W 310 IF C=1 THEN 4270
A 320 IF C<>2 THEN 390
F 330 CALL SOUND(1,-5,0)
M 340 CALL HCHAR(20,5,128)
U 350 CALL HCHAR(21,5,92)
O 360 GOSUB 1260
Y 370 GOSUB 1360
L 380 GOTO 430
B 390 IF C<7 THEN 410
R 400 CALL SOUND(1,-5,0)
K 410 ON C-2 GOSUB 2300,2300,2380,2380,2190,2190,2190,2190
J 420 GOTO 260
Q 430 GOSUB 810
X 440 IF C>1 THEN 490
P 450 CALL HCHAR(20,5,40)
N 460 CALL HCHAR(21,5,91)
G 470 GOSUB 1480
G 480 GOTO 250
B 490 ON C-1 GOSUB 1570,2130,2130,2150,2150,2190,2190,2190,2190,1640,1820,1880
R 500 IF RF=1 THEN 510 ELSE 430
A 510 CALL KEY(0,K,S)
Y 520 IF S=0 THEN 650
T 530 IF K<>72 THEN 610
M 540 CALL SOUND(1,-5,0)
D 550 CALL HCHAR(19,29,94)
N 560 CALL HCHAR(17,5,40)
U 570 GOSUB 3850
B 580 CALL HCHAR(19,29,93)
V 590 RF=0
E 600 GOTO 430
Z 610 IF K<>80 THEN 650
E 620 CALL HCHAR(17,5,40)
U 630 RF=0
A 640 GOTO 450
M 650 ON AD(CA)+1 GOSUB 2720,2790,2830,2910,3040,3060,3080,3170,3270,3330,3390,3450,3500,3550,3600,3650
D 660 ON SW+1 GOSUB 710,730,750,770,790
A 670 GOSUB 1260
G 680 GOSUB 1360
B 690 IF RF=0 THEN 560
A 700 GOTO 510
M 710 TEMP=16*(CA/16-INT(CA/16))
J 720 RETURN
J 730 TEMP=INT(CA/16)
X 740 RETURN
J 750 TEMP=AD(CA)
D 760 RETURN
M 770 TEMP=AR
B 780 RETURN
R 790 TEMP=BR
O 800 RETURN
B 810 CALL KEY(0,K,S)
Z 820 IF S<1 THEN 810
R 830 C=POS("PB<,>.1234IRL",CHR$(K),1)
X 840 IF (K=6)+(K=12)THEN 890
A 850 IF C=0 THEN 810
X 860 IF (C>2)*(C<7)THEN 880
O 870 CALL SOUND(1,-5,0)
C 880 RETURN
U 890 CALL CLEAR
X 900 CALL COLOR(13,2,4)
H 910 IF K=6 THEN 1020
C 920 PRINT TAB(8);"SAVE FILE": : : :"DEVICE AND/OR FILE NAME:"
S 930 INPUT "":FL$
J 940 IF FL$="" THEN 1110
C 950 OPEN #1:FL$,INTERNAL,OUTPUT,FIXED 64
T 960 FOR IT=0 TO 245 STEP 7
L 970 PRINT #1:AD(IT);AD(IT+1);AD(IT+2);AD(IT+3);AD(IT+4);AD(IT+5);AD(IT+6)
F 980 NEXT IT
K 990 PRINT #1:AD(252);AD(253);AD(254);AD(255)
N1000 CLOSE #1
U1010 GOTO 1110
J1020 PRINT TAB(8);"LOAD FILE": : :"DEVICE AND/OR FILE NAME:"
W1030 INPUT "":FL$
P1040 IF FL$="" THEN 1110
D1050 OPEN #1:FL$,INTERNAL,INPUT ,FIXED 64
D1060 FOR IT=0 TO 245 STEP 7
M1070 INPUT #1:AD(IT),AD(IT+1),AD(IT+2),AD(IT+3),AD(IT+4),AD(IT+5),AD(IT+6)
N1080 NEXT IT
C1090 INPUT #1:AD(252),AD(253),AD(254),AD(255)
W1100 CLOSE #1
Q1110 CALL CLEAR
S1120 CALL COLOR(13,2,2)
K1130 GOSUB 4090
```

Continued

```
110 CALL HCHAR(13,4,32)
115 CALL HCHAR(12,5,32)
116 CALL HCHAR(13,6,32)
117 ON SW+1 GOSUB 710,730,750,770,790
118 GOSUB 2460
119 GOSUB 1260
120 CALL HCHAR(20,5,128)
121 CALL HCHAR(21,5,92)
122 GOSUB 1360
123 CALL COLOR(13,7,4)
124 CALL SCREEN(4)
130 GOTO 810
140 PL$=DT$(TEMP)
150 FOR I=10 TO 22 STEP 4
160 CH=INT(I/4)-1
170 IF SEG$(PL$,CH,1)="1" THEN 1320
180 CH2=128
190 CALL HCHAR(12,I,CH2)
200 NEXT I
210 RETURN
220 ADD$=ADD$&DT$(INT(CA/16))
230 ADD$=ADD$&DT$(CA-16*INT(CA/16))
240 REM WRITE ADDRESS TO SCREEN
250 FOR I=6 TO 27 STEP 3
260 CH=INT(I/3)-1
270 IF SEG$(AD$,CH,1)="1" THEN 1440
280 CH2=0
290 GOTO 1450
300 CH2=128
310 CALL HCHAR(6,I,CH2)
320 NEXT I
330 RETURN
340 FOR I=10 TO 22 STEP 4
350 CALL HCHAR(12,I,40)
360 NEXT I
370 FOR I=6 TO 27 STEP 3
380 CALL HCHAR(6,I,40)
390 NEXT I
400 CALL HCHAR(2,3,40)
410 AD(253)=0
420 RETURN
430 CALL HCHAR(10,29,94)
440 CA=0
450 GOSUB 1360
460 GOSUB 710,730,750,770,790
470 GOSUB 1260
480 CALL HCHAR(10,29,93)
490 RETURN
500 CALL HCHAR(13,29,94)
510 CA=CA+1
520 GOSUB 1790
530 IF SW<2 THEN 1760
540 IF SW<2 THEN 1710
550 TEMP=AD(CA)
560 GOTO 1750
570 IF SW=0 THEN 1740
580 GOSUB 730
590 GOSUB 710
600 GOSUB 1260
610 GOSUB 1360
620 CALL HCHAR(13,29,93)
630 RETURN
640 IF CA<256 THEN 1810
650 CA=CA-256
660 RETURN
670 CALL HCHAR(16,29,94)
680 CALL HCHAR(17,5,128)
690 RF=1
700 GOSUB 3850
710 CALL HCHAR(16,29,93)
720 RETURN
730 CALL HCHAR(22,29,94)
740 CT=0
750 TEMP=0
760 PL$=""
770 FOR I=22 TO 10 STEP -4
780 CALL HCHAR(19,I,GHR$(M-43)&PL$)
790 TEMP=TEMP+((M-91)*2^CT)
800 CT=CT+1
810 NEXT I
820 ON SW+1 GOSUB 2010,2030,2070,2070,2
070
830 CALL HCHAR(22,29,93)
840 RETURN
850 CA=INT((CA/16)*16+TEMP
860 GOTO 2040
870 CA=(CA-(INT((CA/16)*16)))+(TEMP*16)
880 GOSUB 1260
890 GOSUB 1360
900 RETURN
910 ADD(CA)=TEMP
920 IF SW>2 THEN 2100
930 GOSUB 1270
940 IF CA<255 THEN 2120
950 ON CA-252 GOSUB 2620,2680,2700
960 RETURN
970 GOTO 2300
980 GOSUB 2360
990 ON SW+1 GOSUB 710,730,750,770,790
1000 GOSUB 1260
1010 CALL HCHAR(19,C*4+6,M)
```

```
1020 IF M=91 THEN 2240
1030 CALL HCHAR(19,C*4+6,91)
1040 GOTO 2250
1050 CALL HCHAR(19,C*4+6,92)
1060 RETURN
1070 ON SW+1 GOSUB 710,730,750,770,790
1080 GOSUB 1260
1090 GOSUB 1360
1100 RETURN
1110 IF SW=0 THEN 2370
1120 SW=SW-1
1130 CALL SOUND(1,-5,0)
1140 GOSUB 2460
1150 CALL KEY(0,K,S)
1160 IF S>=0 THEN 2370
1170 GOTO 2300
1180 RETURN
1190 IF SW=4 THEN 2450
1200 SW=SW+1
1210 CALL SOUND(1,-5,0)
1220 GOSUB 2460
1230 CALL KEY(0,K,S)
1240 IF S>=0 THEN 2450
1250 GOTO 2380
1260 RETURN
1270 ON SW+1 GOSUB 2480,2500,2530,2570,2
600
1280 RETURN
1290 CALL HCHAR(13,4,123)
1300 RETURN
1310 CALL HCHAR(13,4,41)
1320 CALL HCHAR(12,5,32)
1330 RETURN
1340 CALL HCHAR(13,4,32)
1350 CALL HCHAR(12,5,124)
1360 CALL HCHAR(13,6,32)
1370 RETURN
1380 CALL HCHAR(12,5,32)
1390 CALL HCHAR(13,6,42)
1400 RETURN
1410 CALL HCHAR(13,6,125)
1420 RETURN
1430 IF AD(253)<1 THEN 2650
1440 CH2=128
1450 GOTO 2660
1460 CH2=40
1470 CALL HCHAR(2,3,CH2)
1480 RETURN
1490 CALL SOUND(500,TN(AD(254)),0)
1500 CALL SOUND(500,TN(AD(255)+16),0)
1510 RETURN
1520 AR=BR+AR
1530 IF AR<16 THEN 2770
1540 AR=AR-16
1550 CF=1
1560 GOTO 2780
1570 CF=0
1580 GOTO 3780
1590 CA=CA+1
1600 GOSUB 1790
1610 GOSUB TEMP+16*AD(CA)
1620 AR=AD(CA)
1630 GOTO 3780
1640 CA=CA+1
1650 GOSUB 1790
1660 TEMP=AD(CA)
1670 CA=CA+1
1680 GOSUB 1790
1690 TEMP=TEMP+16*AD(CA)
1700 AR=AD(TEMP)
1710 GOTO 3780
1720 CA=CA+1
1730 GOSUB 1790
1740 TEMP=AD(CA)
1750 CA=CA+1
1760 GOSUB 1790
1770 TEMP=TEMP+16*AD(CA)
1780 AD(TEMP)=AR
1790 AD(TEMP)=AR
1800 IF TEMP<253 THEN 3010
1810 ON TEMP-252 GOSUB 2620,2680,2700
1820 CA=CA+1
1830 GOSUB 1790
1840 RETURN
1850 BR=AR
1860 GOTO 3780
1870 GOTO 3780
1880 IF TEMP=0 THEN 3110
1890 IF TEMP=8 THEN 3110
1900 IF AR=INT(AR/2)=AR/2 THEN 3140
1910 CF=1
1920 GOTO 3150
1930 CAR=INT(AR/2)+TEMP
1940 GOTO 3780
1950 TEMP=0
1960 IF TEMP=1 THEN 3200
1970 TEMP=1
1980 IF AR<8 THEN 3240
1990 AR=1
2000 AR=AR*2-16+TEMP
2010 GOTO 3260
2020 CF=0
2030 CAR=AR*2+TEMP
2040 GOTO 3780
2050 GOSUB 3730
```

Continued

NANOPROCESSOR *Continued*

TI-99/4A

```
330 FOR I=1 TO 4
340 IF (ATH=INT(ATH)) THEN ATH=ATH-1 :: GOTO 340
350 NEXT I
360 FOR I=1 TO 4
370 IF (ATH=INT(ATH)) THEN ATH=ATH-1 :: GOTO 370
380 NEXT I
390 FOR I=1 TO 4
400 IF (ATH=INT(ATH)) THEN ATH=ATH-1 :: GOTO 400
410 NEXT I
420 FOR I=1 TO 4
430 IF (ATH=INT(ATH)) THEN ATH=ATH-1 :: GOTO 430
440 NEXT I
450 IF ATH=0 THEN 470
460 CALL SOUND(-300,110*2^(ATH-1),0)
470 RETURN
480 !
490 !
500 !
510 !
520 !
530 !
540 !
550 !
560 !
570 !
580 !
590 !
600 !
610 !
620 !
630 !
640 !
650 !
660 !
670 !
680 !
690 !
700 !
710 !
720 !
730 !
740 !
750 !
760 !
770 !
780 !
790 !
800 !
810 !
820 !
830 !
840 !
850 !
860 !
870 !
880 !
890 !
```

```
900 !
910 !
920 !
930 !
940 !
950 !
960 !
970 !
980 !
990 !
1000 !
1010 !
1020 !
1030 !
1040 !
1050 !
1060 !
1070 !
1080 !
1090 !
1100 !
1110 !
1120 !
1130 !
1140 !
1150 !
1160 !
1170 !
1180 !
1190 !
1200 !
1210 !
1220 !
1230 !
1240 !
1250 !
1260 !
1270 !
1280 !
```

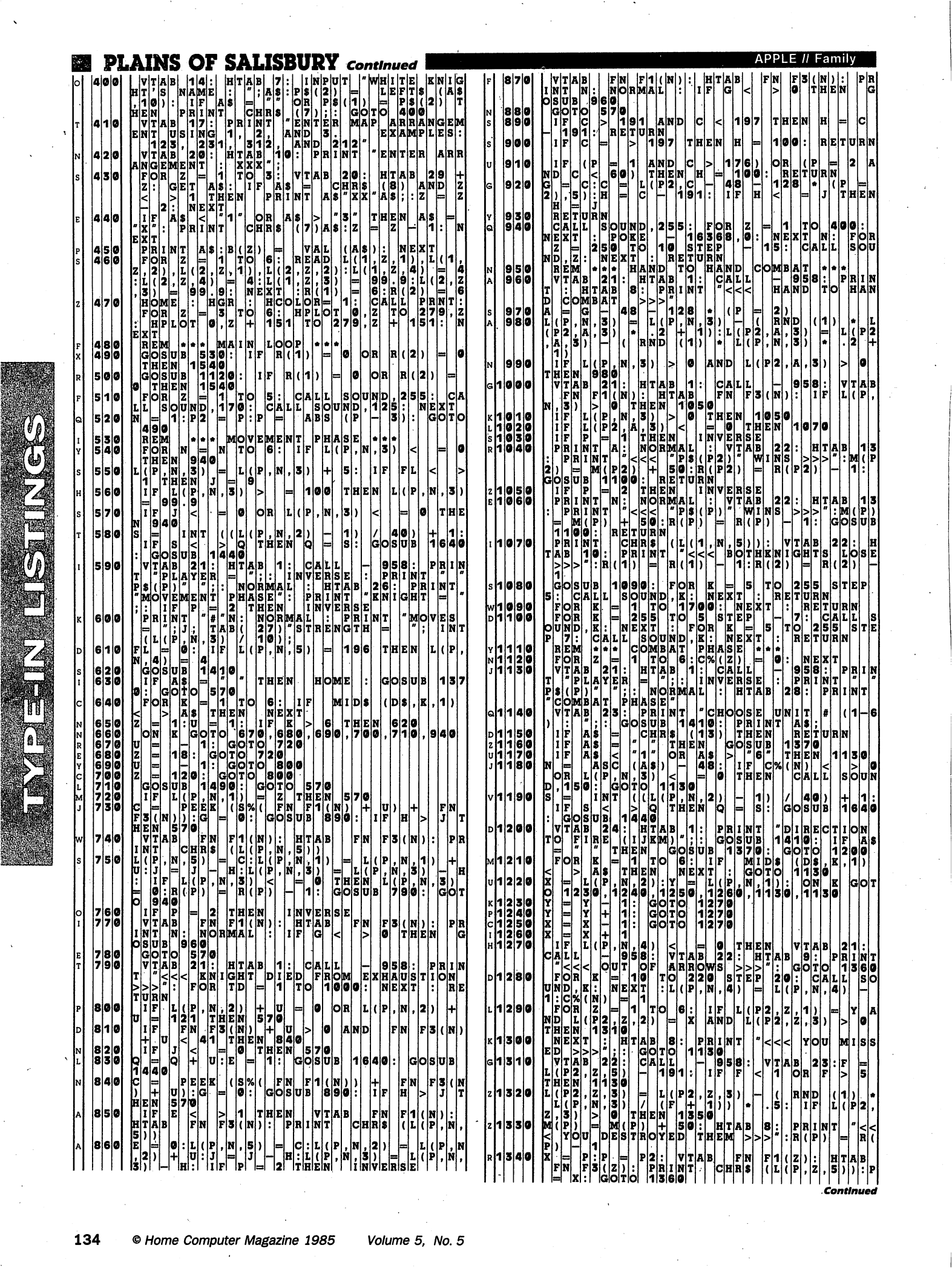

PLAINS OF SALISBURY

APPLE *II* Family

```
100 REM ***********************
110 REM *                     *
120 REM *  PLAINS OF SALISBURY *
130 REM *                     *
140 REM *  COPYRIGHT 1985 BY  *
150 REM *  RALLY WITH COMPUTER *
160 REM *                     *
170 REM *  HOME SOFTWARE INC. *
180 REM *                     *
190 REM ***********************
200 REM
210 REM
220 IF PEEK (104) = 64 THEN 240
230 PRINT CHR$ (4);"RUN MASTER.CREATE"
240 PRINT CHR$ (4);"RUN LOADER"
250 END
260 REM
270 REM
280 REM
290 REM
300 REM
310 REM
320 REM
330 REM
340 REM
350 REM
360 REM
370 REM
380 REM
390 REM
```

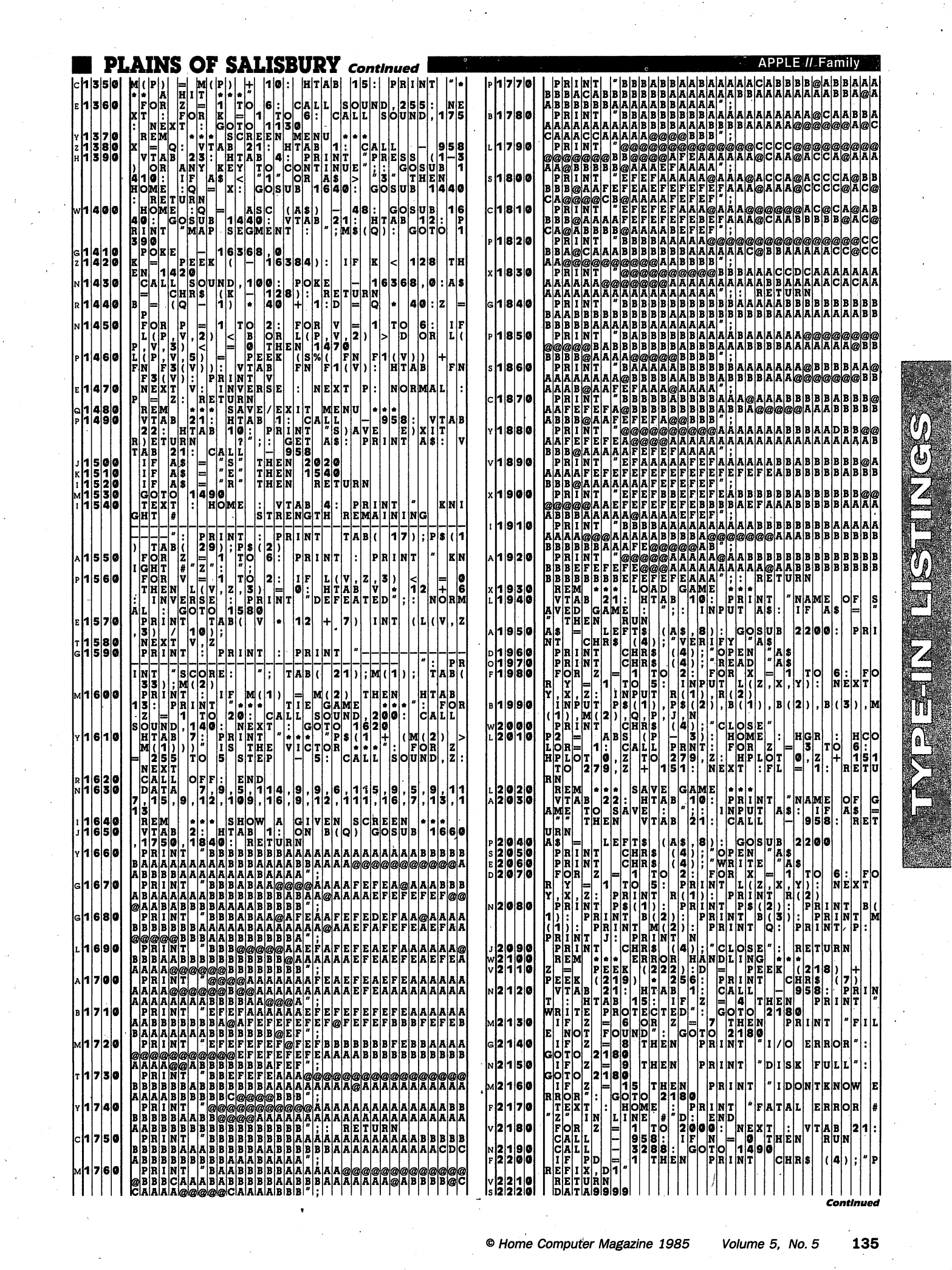
Continued

PLAINS OF SALISBURY *Continued*



Continued

PLAINS OF SALISBURY *Continued*



Continued

PLAINS OF SALISBURY *Continued*

APPLE // Family

```
B 2230  DATA76,106,8,76,112,8,32,76,231,160,32,138,202,208,253,44,48,192,170,136,208,246,96,160,0,36,50,48,2,160,127,132,6,169,7,133,7,165,8,10
L 2240  DATA10,10,170,164,36,24,165,40,133,38,165,41,105,24,133,39,24,165,39,105,4,133,39,189,142,8,69,6,145,38,232,198,7,16,237,96,133,69,134,70
N 2250  DATA132,71,201,160,144,11,201,207,176,7,41,31,133,8,32,23,8,165,69,166,70,164,71,76,240,253,160,76,169,8,208,4,160,240,169,253,174,9,191,224
M 2260  DATA76,208,12,174,15,191,208,7,140,48,190,141,49,190,96,132,54,133,55,76,234,3,213,170,213,170,213,170,213,170,16,0,2,0,4,0,32,0,8,28
N 2270  DATA42,8,62,73,8,0,0,8,28,62,127,50,50,50,0,107,107,127,127,62,62,62,213,215,213,213,213,245,213,213,170,175,170,170,170,250,170,170
O 2280  DATA9999
U 2290  DATA8,12,8,8,8,8,28,0,28,34,32,24,4,2,62,0,62,32,16,24,32,34,28,0,16,24,20,18,62,16,16,0,62,2,30,32,32,34,28,0,56,4,2,30,34,34,28,0,0,0
```

HCM

PLAINS OF SALISBURY

ATARI 800/800XL/130XE

```
B  100 REM ******************
C  110 REM *   THE PLAINS    *
Q  120 REM *  OF SALISBURY   *
K  130 REM ******************
H  140 REM COPYRIGHT 1985
G  150 REM EMERALD VALLEY PUBLISHING CO.
O  160 REM BY WILLIAM K. BALTHROP
K  170 REM HOME COMPUTER MAGAZINE
A  180 REM VERSION 5.5.1
R  190 REM ATARI BASIC
O  200 REM ATARI 800, 800XL, 130XE
J  210 REM
L  220 OPEN #1,4,0,"K:"
T  230 GRAPHICS 0:CLR :POSITION 15,11:PRINT "THE PLAINS":POSITION 14,12:PRINT "OF SALISBURY":POSITION 8,22
Z  240 PRINT "PRESS ◩RETURN◩ TO CONTINUE":GET #1,A
I  250 DIM N$(21),L(12,5),R(2),M(2),LN$(60),MAP$(3),CL$(1),E$(1),A$(10),Q$(1200),S$(1200),MV$(20),FA(6),DEV$(14)
A  260 DIM SO(3)
L  270 CL$=CHR$(125):E$=CHR$(155):N$(1)=" ":N$(20)=N$:N$(2)=N$
E  280 PRINT CL$;:POSITION 1,2:PRINT "LOAD AN OLD FILE <Y/N>?";:GET #1,A:IF A=78 THEN 310
N  290 IF A<>89 THEN 280
C  300 GOSUB 2020:PRINT CL$:POSITION 1,12:PRINT "STAND BY PLEASE...":GOTO 410
S  310 IF A<>78 THEN 280
D  320 POSITION 1,5:PRINT "ENTER RED KNIGHT'S NAME: ";:ML=10:Y=5:X=25:GOSUB 2210:N$(1)=A$:POSITION 1,7
Q  330 PRINT "ENTER BLUE KNIGHT'S NAME:";:Y=7:X=25:GOSUB 2210:N$(11)=A$:N$(21)=" "
Z  340 PRINT :PRINT :PRINT "ENTER MAP ARRANGEMENT: e.g. 123, 321:";:PRINT :PRINT "YOUR ARRANGEMENT:";
F  350 FOR Z=1 TO 3
L  360 GET #1,A:IF A<49 OR A>51 THEN 360
Z  370 MAP$(Z)=CHR$(A):PRINT CHR$(A);:NEXT Z;PRINT :PRINT :PRINT "STAND BY PLEASE..."
X  380 FOR Z=1 TO 3:SO(Z)=VAL(MAP$(Z,Z))-Z:NEXT Z
T  390 RESTORE 2780:FOR Z=1 TO 12:READ A,B:L(Z,1)=A:L(Z,2)=B:L(Z,4)=4:L(Z,3)=99.9:NEXT Z:R(1)=6:R(2)=6
B  400 P=1:P2=2:Q=1:M(1)=0:M(2)=0
D  410 GOSUB 2360:GOSUB 510:GOSUB 2320:GOSUB 2110
A  420 REM
X  430 REM MAIN CONTROL LOOP
W  440 GOSUB 590:IF R(1)=0 OR R(2)=0 THEN 1780
F  450 GOSUB 1290:IF R(1)=0 OR R(2)=0 THEN 1780
G  460 N=1:FOR Z=0 TO 30000 STEP 232:SOUND 1,Z,10,14:NEXT Z:SOUND 1,0,0,0
N  470 IF P=1 THEN P=2:P2=1:GOTO 440
W  480 P=1:P2=2:GOTO 440
P  490 REM
D  500 REM SET UP GRAPHICS
U  510 RESTORE 2460
A  520 GRAPHICS 1:POSITION 1,23:PRINT "STAND BY PLEASE...";:FOR Z=0 TO 4:READ COL,BRI:SETCOLOR Z,COL,BRI:NEXT Z
Q  530 A=PEEK(106)-8:B=A*256:POKE 204,A:POKE 206,224
N  540 FOR Z=0 TO 1023:POKE B+Z,PEEK(57344+Z):NEXT Z
M  550 FOR Z=1 TO 7:READ ADD:FOR ZZ=0 TO 7:READ GC:POKE B+ADD+ZZ,GC:NEXT ZZ:NEXT Z
N  560 POKE 756,A:RETURN
Q  570 REM
N  580 REM MOVEMENT PHASE
Q  590 FOR N=N TO 6:KN=N+(P-1)*6:IF L(KN,3)<=0 THEN 1020
E  600 SOUND 1,50,12,15:GOSUB 2170:SOUND 1,0,0,0:IF FL=0 THEN NM=9
B  610 L(KN,3)=L(KN,3)+5:IF L(KN,3)>99.9 THEN L(KN,3)=99.9
R  620 IF NM=0 THEN 1020
N  630 S=INT(L(KN,1)/20+1):IF S<>Q THEN Q=S:GOSUB 2320:GOSUB 2110
H  640 IF (N=1 AND NM=9) OR FL=1 THEN FL=0
T  650 GOSUB 1230
S  660 IF L(KN,5)=5 THEN L(KN,4)=4
F  670 POKE 764,255:GET #1,A:IF A>64 AND A<68 THEN GOSUB 1180:GOSUB 1230:GOTO 670
M  680 IF A>96 AND A<100 THEN A=A-32:GOSUB 1180:GOSUB 1230:GOTO 670
P  690 IF A=27 THEN GOSUB 1600:GOTO 670
G  700 IF A=42 THEN LIM=60:DIR=1:GOTO 770
J  710 IF A=43 THEN LIM=0:DIR=-1:GOTO 770
I  720 IF A=61 THEN LIM=19:DIR=1:GOTO 900
A  730 IF A=45 THEN LIM=0:DIR=-1:GOTO 900
E  740 IF A=27 THEN RETURN
A  750 IF A=155 THEN 1020
P  760 GOTO 670
K  770 IF L(KN,1)+DIR<0 OR L(KN,1)+DIR>59 THEN 670
Z  780 IF L(KN,1)+DIR>=(Q-1)*20 AND L(KN,1)+DIR<=(Q-1)*20+19 THEN 800
J  790 Q=Q+DIR:E=1:GOSUB 2320:GOSUB 2110
P  800 I=L(KN,2)*60+L(KN,1)+DIR+1+20*SO(Q):C=VAL(Q$(I,I))
I  810 G=0:L1=L(KN,1)+DIR:L2=L(KN,2):GOSUB 950:IF H>NM THEN 620
N  820 IF E=0 THEN POSITION L(KN,1)-(Q-1)*20,L(KN,2):GOSUB 1010
R  830 IF L(KN,3)-H>0 OR L(KN,3)<=0 THEN 870
S  840 PRINT CL$:POSITION 1,22:PRINT "KNIGHT DIED FROM EXHAUSTION..."
P  850 GOSUB 1010:SOUND 0,255,10,15:GOSUB 2170:SOUND 0,0,0,0:R(P)=R(P)-1:L(KN,3)=0
E  860 GOSUB 1230:GOTO 1020
D  870 E=0:L(KN,5)=C:NM=NM-H:L(KN,3)=L(KN,3)-H:L(KN,1)=L(KN,1)+DIR
M  880 POSITION L(KN,1)-(Q-1)*20,L(KN,2):PRINT #6;CHR$(16+N+(128*(P-1)));:IF G=1 THEN GOSUB 1050
I  890 GOTO 620
V  900 IF L(KN,2)=LIM THEN 670
E  910 I=(L(KN,2)+DIR)*60+L(KN,1)+1+20*SO(Q):C=VAL(Q$(I,I)):G=0:L1=L(KN,1):L2=L(KN,2)+DIR:GOSUB 950:IF H>NM THEN 670
U  920 POSITION L(KN,1)-(Q-1)*20,L(KN,2):GOSUB 1010:L(KN,5)=C:L(KN,2)=L(KN,2)+DIR:NM=NM-H
F  930 L(KN,3)=L(KN,3)-H:POSITION L(KN,1)-(Q-1)*20,L(KN,2):PRINT #6;CHR$(16+N+(P-1)*128);:IF G=1 THEN GOSUB 1050
U  940 GOTO 620
A  950 FOR Z=1 TO 12:IF L(Z,1)=L1 AND L(Z,2)=L2 AND L(Z,3)>0 THEN 980
N  960 NEXT Z:G=0:IF C<>6 THEN H=C:RETURN
A  970 H=100:RETURN
W  980 IF (Z<7 AND P=1) OR (Z>6 AND P=2) THEN H=100:RETURN
S  990 G=1:H=C:GT=Z:IF H<NM THEN NM=H
F 1000 RETURN
Y 1010 ON L(KN,5) GOSUB 2380,2390,2400,2410,2420,2430:PRINT #6;CHR$(A);:RETURN
I 1020 NEXT N:SOUND 1,100,14,15:GOSUB 2170:SOUND 1,120,12,15:GOSUB 2170:SOUND 1,0,0,0:RETURN
Y 1030 REM
G 1040 REM HAND-TO-HAND COMBAT
N 1050 PRINT CL$;:POSITION 1,22:PRINT "HAND-TO-HAND COMBAT"
I 1060 SOUND 1,120,14,15:GOSUB 2180:SOUND 1,110,14,15:GOSUB 2180:SOUND 1,0,0,0
A 1070 B=INT(RND(0)*L(KN,3)*0.2)+1:C=INT(RND(0)*L(GT,3)*0.2)+1:L(KN,3)=L(KN,3)-C:L(GT,3)=L(GT,3)-B
B 1080 IF L(KN,3)>0 AND L(GT,3)>0 THEN 1060
V 1090 PRINT CL$;:POSITION 1,22:IF L(KN,3)<=0 AND L(GT,3)<=0 THEN L(KN,3)=0:L(GT,3)=0:GOTO 1140
W 1100 IF L(GT,3)<=0 THEN L(GT,3)=0:GOTO 1130
K 1110 POSITION L(GT,1)-(Q-1)*20,L(GT,2):PRINT #6;CHR$(16+(GT-(6*(P2-1))+(128*(P2-1))));:R(P)=R(P)-1
```

Continued

PLAINS OF SALISBURY *Continued*

ATARI 800/800XL/130XE

Continued

TYPE-IN LISTINGS

PLAINS OF SALISBURY *Continued*

ATARI 800/800XL/130XE

```
A 2360  X=0:RESTORE 2560:FOR Z=0 TO 19:READ
        LN$:FOR ZZ=1 TO 60:I=Z*60+ZZ:Q$(I,
        I)=LN$(ZZ,ZZ):X=X+1
Q 2370  ON VAL(Q$(I,I)) GOSUB 2380,2390,2400
        0,2410,2420,2430:S$(X,X)=CHR$(A):NE
        XT ZZ:NEXT Z:RETURN
R 2380  A=6:RETURN
A 2390  A=33:RETURN
C 2400  A=35:RETURN
F 2410  A=165:RETURN
G 2420  A=8:RETURN
S 2430  A=137:RETURN
A 2440  REM
T 2450  REM  GRAPHICS CHARACTER DATA
N 2460  DATA 12,8,15,7,9,10,8,7,1,10
X 2470  DATA 48,85,34,85,136,85,34,85,136
Z 2480  DATA 8,0,32,16,2,4,64,128,0
A 2490  DATA 24,0,16,56,84,186,84,146,16
X 2500  DATA 40,0,24,60,126,255,102,102,102
T 2510  DATA 64,0,195,195,255,255,126,102,1
        02
Q 2520  DATA 72,255,255,255,255,255,255,255
        ,255
K 2530  DATA 0,0,0,0,0,0,0,0,0,0
Z 2540  REM
D 2550  REM  SCREEN TERRAIN DATA
J 2560  DATA 3333333333333333333333333333
        3333333333333333333333333333
A 2570  DATA 2222222222222222222222222234
W 2580  DATA 3333222222222333333333222241
        1111111111111111111111111111
U 2590  DATA 3322231116661232232332224132
        2211
```

```
X 2600  DATA 3332322166666611223333333333
        1332224433322222333112222222213333
A 2610  DATA 3222211226566111222222222222
        1333333222222112233332333213333
E 2620  DATA 3331113626266662111133333344111
        1114433333333111333322311111111
D 2630  DATA 4311111111333626266622231111111111134
Q 2640  DATA 3333336626266626111332221221444
        4412222333333223333553311666662223
U 2650  DATA 2222226622222226122222221221334
        4312222222233322111331266666662
Y 2660  DATA 2222226666666666612222662122111
        1111222223333331333316666662
R 2670  DATA 6662226666666666612266622122222
        2213411422666633333316666666
B 2680  DATA 6666616662222111122666624142243
        4313414666661666666113331266666666
G 2690  DATA 6666162222222266666624111111
        1111666666616666666113331112666666
S 2700  DATA 6666333222222311226666662241422222
        2211222266666666311133112232366
D 2710  DATA 2333333222222333133333322221233333
        3212222222333333333333333332
N 2720  DATA 3333322222211113333333331233444
        3212223333333333111133333333331333
V 2730  DATA 3333222111233334133333111111
        1111141333333333111333311113
J 2740  DATA 1111111111123333333331111111134454
        4222311111111133332221233333333333
T 2750  DATA 3222223333333333333221233333333333
        3322333333333333333221233333333333
V 2760  REM
D 2770  REM  INITIAL KNIGHT'S POSITIONS
D 2780  DATA 4,5,3,5,3,4,7,18,5,18,3,18,55,
        6,56,6,57,6,53,15,54,15,55,15
```

HCM

PLAINS OF SALISBURY

COMMODORE 64

```
RWG 100  REM  ********** PLAINS ********** 
ANC 110  REM  *          THE PLAINS      *
ICV 120  REM  *            OF SALISBURY  *
RKZ 130  REM  ********** 1985 ********** 
NIV 140  REM  COPYRIGHT (C) 1985
WRU 150  REM  EMERALD VALLEY PUBLISHING CO.
CL  160  REM  BY WILLIAM K. STAFFORD
HZ  170  REM  AND THE HCM STAFF
IXD 180  REM  HOME COMPUTER MAGAZINE
PDZ 190  REM  VERSION 5.1
DAC 200  REM  COMMODORE-64
    210  REM
NV  220  GOTO 5110
W   230  PRINT CHR$(147):POKE 53280,13:POKE 532
        81,15
    240  K=VAL(MID$(MAP$(MP(SC),I),J,1)):L=
        VAL(MID$(MAP$(MP(SC),I),J+1,1))
R   250  FOR M=1 TO L:PRINT GR$(K);:NEXT:NE
        XT:FOR I=1 TO 2:FOR J=1 TO 6
U   260  K=PL(I,J,1):IF INT(K)/1000=SCAND PL
        (I,J,3)>0 THEN GOSUB 280
CL  270  NEXT:K=NESC+1000)+1024:PL(I,J,5)=PEEK(
        K%+34816):PL(I,J,6)=PEEK(K%+54272)A
        ND 15
IXD 290  POKE EK%+34816,176+J
PDZ 300  POKE EK%+54272,CO(I):RETURN
    310  GOSUB 2580
    320  POKE 211,0:POKE 214,18:PRINT:PRINT BL$
        NAM$(TE)SP$(2)
P   330  PRINT MOVEMENT PHASE/UNIT:"KN";" SHI
        FT CRSR RLEFT ":PRINT "MOVES
        LEFT:"PL(TE,KN,2);
    340  PRINT CHR$(157)SP$(2)
DA  350  PRINT "SPTRENGTH :";INT(PL(TE,KN,3))+
        .005);"SP$(1):GOSUB 2580:RETURN
C   360  PRINT "  SHIFT CLR 10 CRSR DOWN
        PLEASE WAIT WHILE I"
K   370  PRINT "  CRSR DOWN  ASSEMBLE
        YOUR TROOPS."
J   380  POKE 56,128:CLR:FI=56334:POKE FI,PEEK
        (FI)AND 254:POKE 1,PEEK(1)AND 251
Z   390  FOR I=0 TO 2047:POKE 32768+I,PEEK(53248
        +I):NEXT
HKG 400  POKE 1,PEEK(1)OR 4:POKE FI,PEEK(FI)OR 1
KG  410  PRINT CHR$(147):READ J:IF J=-1 THEN 430
G   420  FOR I=0 TO 7:READ K:POKE 32768+J*8+I,K:N
        EXT:GOTO 410
O   430  POKE 53272,49:POKE 657,128:POKE 648,14
P   440  DIM MA$(3,8),GR$(6),PL(2,6,6):FOR I=1
        TO 3:FOR J=1 TO 8:READ MA$(I,J):NEXT:NEX
        T
J   450  GR$(1)=CHR$(158)+"#":GR$(2)=CHR$(30
        )+"$":GR$(4)=CHR$(5)+"&":BL$=CHR$(1
        44)
D   460  GR$(3)=CHR$(30)+"%":GR$(5)=CHR$(151
        )+"":GR$(6)=CHR$(31)+"":CO(1)=2
G   470  CO(2)=6:CL$=CHR$(147):DN$=CHR$(17)+
        CHR$(17):CR$=CHR$(13):TR(2)=9:TR(4)
        =4
S   480  FOR I=1 TO 40:SP$(1)=SP$(1)+CHR$(32):N
        EXT:SP$(2)=LEFT$(SP$(1),12):FL$="RA
        J"
F   490  FOR I=1 TO 2:FOR J=1 TO 6:READ L:PL(I,J,1)
        =L:PL(I,J,3)=9.9:NEXT:WI(I)=6:NEXT
```

```
N   500  GOTO 5660
A   510  PRINT CHR$(147):POKE 53280,13:POKE 532
        81,15
H   520  POKE 646,0:POKE 214,11:PRINT:PRINT TAB
        (8)"THE PLAINS OF SALISBURY"
N   530  POKE 214,22:PRINT:PRINT TAB(8)"PRESS
        RETURN TO CONTINUE"
N   540  GOSUB 2540:IF PEEK<>13 GOTO 540
M   550  GOTO 3600
A   560  PRINT CHR$(147):POKE 214,11:PRINT:PRI
        NT TAB(8)"LOAD OLD GAME <Y/N>?"
O   570  GOSUB 2540:ON-(PEEK=89)GOTO 2320
W   580  PRINT CL$:INPUT"NAME OF RED KNIGHTS:
        ";NA$(1)
    590  INPUT"NAME OF BLUE KNIGHTS:";NA$(2)
V   600  FOR I=1 TO 2:NA$(I)=LEFT$(NA$(I),8):NE
        XT
G   610  PRINT DN$NA$(1)" YOU ARE RED. YOU GO
        FIRST. "CR$DN$NA$(2)" YOU ARE BLUE"
W   620  PRINT"ENTER MAP ARRANGEMENT USING 1
        2 AND 3, E.G. 231 OR 312 ETC.":PR
        INT
H   630  INPUT"MAP:";MP$
HZ  640  IF LEN(MP$)<>3 THEN PRINT"ENTER THREE
        NUMBERS, SHIFT CRSR UP":GOTO 630
S   650  FOR I=1 TO 3:J$=MID$(MP$,I,1):MP(I)=VA
        L(J$):IF J$<CHR$(49)OR J$>CHR$(51)GOT
        O 620
Z   660  NEXT:SC=0:OK=1:FOR I=1 TO 6:PL(1,I,4)=
        4:PL(2,I,4)=4:NEXT
M   670  GOSUB 2300
W   680  IF(WI(1)=0)OR(WI(2)=0)THEN 2280
W   690  IFTE=1THENTE=2:HO=1:GOTO 710
EA  700  TE=1:HO=2
T   710  FOR J=1 TO 6:PL(TE,J,2)=TR(2):NEXT:KN=
        OK
F   720  REM
FC  730  ON-(KN=7)GOTO 1650:IFPL(TE,KN,3)<=0T
        HENKN=KN+1:GOTO 730
D   740  IFZZ=0ORPL(TE,KN,2)=9THENPL(TE,KN,3
        )=PL(TE,KN,3)+.5
TU  750  ZZ=0
U   760  IFPL(TE,KN,3)>9.9THENPL(TE,KN,3)=9.
        9
D   770  IFPL(TE,KN,5)=39THENPL(TE,KN,4)=4
P   780  IFPL(TE,KN,2)=<0ORPL(TE,KN,3)=<0THE
        NKN=KN+1:GOTO 730
O   790  IFSC<>INT(PL(TE,KN,1)/1000):GOSUB 230
        0
W   800  IFPL(TE,KN,2)=<0ORPL(TE,KN,3)=<0THE
        NKN=KN+1:GOTO 730
T   810  GOSUB 3100:GOSUB 2540:ON-(KE=133)-(KE=
        134)-(KE=135)GOTO 840
C   820  I=-(KE=17)-2*(KE=29)-3*(KE=145)-4*(
        KE=157)-5*(KE=13)-6*(KE=136)
H   830  ON-(I>0ANDI<5)-2*(I=5)-3*(I=6)GOTO 9
        80,880,2030:GOTO 810
E   840  IFSC=KE-132GOSUB 2300:GOTO 790
ET  850  SC=KE-132:GOSUB 2300:PRINT" HOME 19
        CRSR DOWN"
F   860  FOR Z=1 TO 4:PRINT"
                                    ";:NEXT
G   870  PRINT" 3 SHIFT CRSR UP CRSR RIGHT PR
        ESS RETURN TO CONTINUE...";:GOSUB 25
        40:GOTO 790
```

Continued

```
C  880 KN=KN+1:GOTO730
V  890 ON-(PL(TE,6,2)=0)GOTO1650:PL(TE,KN,2)=PL(TE,KN,2)-(PL(TE,KN,5)-34)
B  900 PL(TE,KN,3)=PL(TE,KN,3)-(PL(TE,KN,5)-34)*.1
D  910 IFPL(TE,KN,3)>0THEN960
S  920 J=(PL(TE,KN,1)-SC*1000)+1:WI(TE)=WI(TE)-1:PL(TE,KN,1)=0:PL(TE,KN,3)=0
A  930 POKE211,0:POKE214,19:PRINT"KNIGHT DIED OF EXHAUSTION!!";:FORTD=1TO1500:NEXT
Y  940 POKE211,0:POKE214,19:PRINT"                    ";
C  950 POKEJ+1023+34816,PL(TE,KN,5):POKEJ+55295,PL(TE,KN,6)
Y  960 IFWI(TE)=0GOTO2280
O  970 GOTO760
S  980 VL=0:K=0:J=(PL(TE,KN,1)-SC*1000)+1:ONIGOSUB1300,1320,1300,1320
P  990 IFVL=0GOTO800
H 1000 L=(J+(I=4))/40:ON-(L<>INT(L))GOTO1020
D 1010 ON-(((I=2)AND((SC=1)OR(SC=2)))OR((I=4)AND((SC=2)OR(SC=3))))GOTO1100
B 1020 I=-40*(I=1)-(I=2)+40*(I=3)+(I=4)
U 1030 IF(PEEK(I+J+1023+34816)=40)OR((PEEK(I+J+55295)AND15)=CO(TE))GOTO800
T 1040 IFPEEK(J+I+1023+34816)>176GOTO1340
B 1050 ON-((PEEK(J+I+1023+34816)-34)>(PL(TE,KN,2)))GOTO800
R 1060 POKEJ+1023+34816,PL(TE,KN,5):POKEJ+55295,PL(TE,KN,6)
I 1070 PL(TE,KN,5)=PEEK(J+I+1023+34816):PL(TE,KN,6)=(PEEK(J+I+55295)AND15)
B 1080 PL(TE,KN,1)=1000*SC+I+J-1:POKEJ+I+1023+34816,KN+176:POKEJ+I+55295,CO(TE)
M 1090 GOTO890
P 1100 IFPL(TE,KN,5)-34>PL(TE,KN,2)THEN800
N 1110 N=0:ON-((I=2)AND((SC=1)OR(SC=2)))GOTO1210
J 1120 FORI=1TO6:IFPL(TE,I,1)=PL(TE,KN,1)-1000+39THENN=1
U 1130 NEXT:ON-(N=1)GOTO800
L 1140 SC=SC-1:PL(TE,KN,1)=PL(TE,KN,1)-1000+39
L 1150 PL(TE,KN,2)=PL(TE,KN,2)-PL(TE,KN,5)+34
M 1160 FORI=1TO6:IFPL(HO,I,1)=PL(TE,KN,1)THENN=I
U 1170 NEXT:ON-(N=0)GOTO1180:GOSUB1390
Y 1180 PL(TE,KN,3)=PL(TE,KN,3)-(PL(TE,KN,5)-34)*.1:GOSUB230:FORI=1TO1200:NEXT
V 1190 IFPL(TE,KN,3)<=0 THENPL(TE,KN,3)=0
L 1200 ON N+1GOTO800,880,880
Z 1210 FORI=1TO6:IFPL(TE,I,1)=PL(TE,KN,1)+1000-39THENN=1
E 1220 NEXT:ON-(N=1)GOTO800
T 1230 SC=SC+1:PL(TE,KN,1)=PL(TE,KN,1)+1000-39
X 1240 PL(TE,KN,2)=PL(TE,KN,2)-PL(TE,KN,5)+34
Y 1250 FORI=1TO6:IFPL(HO,I,1)=PL(TE,KN,1)THENN=I
W 1260 NEXT:ON-(N=0)GOTO1270:GOSUB1390
U 1270 PL(TE,KN,3)=PL(TE,KN,3)-(PL(TE,KN,5)-34)*.1:GOSUB230:FORI=1TO1200:NEXT
D 1280 IFPL(TE,KN,3)<=0THENPL(TE,KN,3)=0
W 1290 ON N+1GOTO800,880,880
M 1300 IF(((J>680)AND(J<721))AND(I=1))OR(((J>0)AND(J<41))AND(I=3))THENRETURN
T 1310 VL=1:RETURN
P 1320 L=(J+(I=4))/40:IFL=INT(L)THENIF(I=2ANDSC=3)OR(I=4ANDSC=1)THENRETURN
F 1330 VL=1:RETURN
S 1340 N=PEEK(1023+34816+I+J)-176
R 1350 IFPL(TE,KN,2)<PL(HO,N,5)-34GOTO800
C 1360 POKEJ+1023+34816,PL(TE,KN,5):POKEJ+55295,PL(TE,KN,6)
G 1370 GOSUB1380:ONNGOTO1080,880
S 1380 PL(TE,KN,5)=PL(HO,N,5):PL(TE,KN,6)=PL(HO,N,6)
G 1390 PL(TE,KN,2)=0:POKE214,19:PRINT:PRINT"HAND TO HAND COMBAT"
P 1400 HA=PL(TE,KN,3)*RND(0)*.1+1:HB=PL(HO,N,3)*RND(0)*.1+1
X 1410 PL(TE,KN,3)=PL(TE,KN,3)-HB:PL(HO,N,3)=PL(HO,N,3)-HA
K 1420 IF(PL(TE,KN,3)<=0)AND(PL(HO,N,3)<=0)THEN1590
G 1430 IFPL(TE,KN,3)<=0THEN1530
P 1440 IFPL(HO,N,3)<=0THEN1470
Q 1450 POKE54272,0:POKE54273,2:POKE54277,20:POKE54278,70:POKE54296,15
W 1460 POKE54276,129:FORTD=1TO100:NEXT:POKE54276,128:GOTO1400
P 1470 PL(HO,N,3)=0:WI(HO)=WI(HO)-1
Q 1480 POKE214,19:PRINT:PRINTNAME$(TE)" WINS HAND TO HAND"
A 1490 FORM=1TO1200:NEXT:S(TE)=S(TE)+50
N 1500 POKEJ+I+1023+34816,KN+176:POKEJ+I+55295,CO(TE)
M 1510 IFWI(HO)=0GOTO2280
C 1520 N=1:RETURN
P 1530 PL(TE,KN,3)=0:WI(TE)=WI(TE)-1
H 1540 POKE214,19:PRINT:PRINTNAME$(HO)" WINS HAND TO HAND"
J 1550 FORM=1TO1200:NEXT:S(HO)=S(HO)+50
O 1560 POKEJ+I+1023+34816,N+176:POKEJ+I+55295,CO(HO)
O 1570 IFWI(TE)=0GOTO2280
H 1580 N=2:RETURN
S 1590 PL(TE,KN,3)=0:PL(HO,N,3)=0:WI(1)=WI(1)-1:WI(2)=WI(2)-1
G 1600 POKE214,19:PRINT:PRINT"BOTH LOSE HAND TO HAND"
V 1610 FORM=1TO1200:NEXT
D 1620 POKEJ+I+1023+34816,PL(HO,N,5):POKEJ+I+55295,PL(HO,N,6)
H 1630 IF(WI(1)=0)OR(WI(2)=0)THENGOTO2280
A 1640 N=2:RETURN
U 1650 FORI=1TO6:AR(I)=0:NEXT
R 1660 KN=0:TX$(1)="COMBAT PHASE      ":TX$(2)="CHOOSE UNIT   <1-6>:   ":TX$(3)=""
H 1670 GOSUB2580:POKE211,0:POKE214,18:PRINT:PRINTBL$NAM$(TE)SP$(2)
Q 1680 PRINTTX$(1)SP$(2):PRINTTX$(2)SP$(1):IFKN<>0THENRETURN
I 1690 GOSUB2540:Z$=""
J 1700 ON-(KE=13)GOTO680
C 1710 IF-(KE=133)-(KE=134)-(KE=135)THENSC=KE-132:GOSUB230:KN=0:GOTO1670
M 1720 KN=-(KE=49)-2*(KE=50)-3*(KE=51)-4*(KE=52)-5*(KE=53)-6*(KE=54)
K 1730 IF(KN<1)OR(KN>6)OR(PL(TE,KN,3)<=0)GOTO1690
Z 1740 IF(AR(KN)=1)OR(PL(TE,KN,3)<=0)GOTO1690
K 1750 PRINT KN;"SHIFT CRSRUP2 SHIFT CRSRLEFT";
D 1760 IFPL(TE,KN,4)<1THENPOKE214,22:PRINT:PRINT"OUT OF ARROWS"SP$(2);:GOTO1690
H 1770 I=INT(PL(TE,KN,1)/1000):IFSC<>ITHENSC=I:GOSUB230:GOSUB1670
L 1780 POKE214,22:PRINT:PRINT"USE CURSOR TO AIM AND FIRE":GOSUB2580
N 1790 GOSUB2540:I=-(KE=17)-2*(KE=29)-3*(KE=145)-4*(KE=157)-5*(KE=13)
V 1800 POKE214,22:PRINT:PRINTSP$(1);
R 1810 ON-((I>0)AND(I<5))-2*(I=5)GOTO1820,1690:GOTO1780
N 1820 VL=0:J=(PL(TE,KN,1)-SC*1000)+1:L=(J+(I=4))/40:ONIGOSUB1300,1320,1300,1320
O 1830 N=PEEK(1023+34816+I+J)-176:IFVL=0GOTO2010
I 1840 ON-(L<>INT(L))GOTO1860
G 1850 ON-(((I=2)AND((SC=1)OR(SC=2)))OR((I=4)AND((SC=2)OR(SC=3))))GOTO1950
H 1860 I=-40*(I=1)-(I=2)+40*(I=3)+(I=4)
M 1870 N=PEEK(J+I+1023+34816)-176
N 1880 ON((N>0)AND((PEEK(I+J+55295)AND15)=CO(HO)))+2GOTO1890,2010
O 1890 M=INT(RND(0)*PL(TE,KN,3)):PL(HO,N,3)=PL(HO,N,3)-M:IFPL(HO,N,3)>0GOTO1990
A 1900 IFSC<>INT(PL(HO,N,1)/1000)GOTO1930
G 1910 L=((PL(HO,N,1)/1000)-INT(PL(HO,N,1)/1000))*1000
Y 1920 POKEJ+I+1023+34816,PL(HO,N,5):POKEJ+I+55295,PL(HO,N,6)
P 1930 S(TE)=S(TE)+40
L 1940 Z$="AND KILL":PL(HO,N,3)=0:WI(HO)=WI(HO)-1:ONWI(HO)+1GOTO2280:GOTO1990
K 1950 X=0:M=-1039:IF((I=2)AND((SC=1)OR(SC=2)))THENM=961
M 1960 FORY=1TO6:IFPL(HO,Y,1)=PL(TE,KN,1)+MTHENX=1:N=Y
Q 1970 NEXT:IFX=0GOTO2010
Z 1980 GOTO1890
C 1990 POKE214,22:PRINT:PRINT"A HIT "Z$SP$(2);:PL(TE,KN,4)=PL(TE,KN,4)-1:AR(KN)=1
C 2000 S(TE)=S(TE)+10:GOTO1690
J 2010 POKE214,22:PRINT:PRINT"MISSED"SP$(2);:PL(TE,KN,4)=PL(TE,KN,4)-1:AR(KN)=1
H 2020 GOTO1690
V 2030 PRINTCL$"1. SAVE, 2. EXIT, 3. RETURN TO GAME"
M 2040 GOSUB2540:ONKE-48GOTO2050,2250:GOSUB230:GOTO770
F 2050 SA=1:PRINTCL$
Q 2060 INPUT"TAPE OR DISK < T OR D >";KE$
N 2070 FL$="":INPUT"FILENAME";FL$:IF FL$="" THEN GOSUB230:GOTO770
Q 2080 DV=1:IFKE$="D"THENSA=2:FL$="@:"+FL$+",S,W":DV=8:OPEN15,8,15
W 2090 OPEN1,DV,SA,FL$::IF DV=8 THEN GOSUB2610:IF E THEN CLOSE 1:GOTO2050
B 2100 PRINT#1,NA$(1)
L 2110 PRINT#1,NA$(2)
P 2120 PRINT#1,MP(1)
N 2130 PRINT#1,MP(2)
D 2140 PRINT#1,MP(3)
A 2150 PRINT#1,TE
Z 2160 PRINT#1,HO
I 2170 PRINT#1,KN
A 2180 PRINT#1,SC
W 2190 PRINT#1,S(1)
X 2200 PRINT#1,S(2)
F 2210 FORI=1TO2:FORJ=1TO6:FORK=1TO6
U 2220 PRINT#1,PL(I,J,K)
V 2230 NEXT:NEXT:NEXT:CLOSE1:CLOSE 15
W 2240 GOSUB230:GOTO800
```

Continued

PLAINS OF SALISBURY *Continued*

COMMODORE 64

```
10 PRINT"■SHIFT CLR■ARE YOU SURE YOU W
20 ANT TO QUIT (Y/N)"
30 GET K$:IF K$<>"Y" AND K$<>"N" THEN2
60
40 IF K$="N" THEN GOSUB230:GOTO770 ;NA
50 PRINT"■SHIFT CLR■":PRINT" ";
60 FOR Z=1TO6:PRINTZ;TAB(8);:INPUT"";PL(1,Z,
70 )):NEXT:PRINT
80 PRINT"SCORE";:PRINT
90 PRINT:PRINT" ";S(1):PRINTN
100 PRINT:PRINT" ";S(2):PRINTN
110 PRINT:PRINT"INPUT FILE NAME"
120 INPUT"";K$:OPEN1,8,15,"S=";K$+",S,
130 R":IFDV=8THEN15
140 OPEN1,DV,SA,FL$:IF DV=8 THEN GOSUB2
150 10:IF#1,NA$(1)
160 INPUT#1,NA$(2)
170 INPUT#1,MP(1)
180 INPUT#1,MP(2)
190 INPUT#1,MP(3)
200 INPUT#1,TE
210 INPUT#1,HO
220 INPUT#1,KN
230 INPUT#1,SC
240 INPUT#1,S(1)
250 INPUT#1,S(2)
260 FORI=1TO2:FORJ=1TO6:FORK=1TO6
270 INPUT#1,PL(I,J,K)
280 NEXT:NEXT:CLOSE15
290 GOSUB230:Z=1:GOTO730
300 HOME:TE
310 PRINTCL$DN$DN$DN$NAME$(HO)"
320 PRINS IT ALL!!!!!"
330 END
340 POKE653,0:POKE198,0:WAIT197,64:WAIT
197,64,64:GETKE$
350 KE=ASC(KE$+CHR$(0)):POKE198,0
360 KE=KE%5427%2:PO%5427%2:PO%5427%,
370 20:POKE54278,70:PORTD=1TO50:NEXT:POKE
54276,128
380 FORQ=217TO242:IFPEEK(Q)<128THENPOKE
Q,PEEK(Q)+128
390 NEXT:RETURN
400 ERROR GOTO184 0
410 OF:SCREEN:COLOR 1,0:LOCA
420 TE 11,10:PRINT"T H E"
430 PRINTAB(8);"O P E N"
440 Z=0:FOR Z=1TO6:(Z)=0:NEXT
450 72=(236+Z*3+PRE
460 EXT:LOCATE 23,1:PRINT"
470 CE CONTINUE"
480 ON ERROR GOTO 0:CLS:COLOR 6,5)
490 YS,XE(2),ALIVE(2),SROAD(10),SGR(10),
500 CO),ESTREE(10),SUILD(10),SFOR(10),S
510 WATERR(10),SCN%(120,18),MP(3),SR4(10)
520 DIMSR5(10),SR2(10),SB1(10),SB2(10),
530 3(10),SB4(10),SB5(10),SB6(10),
540 LINE(4,4)-(36,75),3,BF:DRAW"C0;
550 )-(17,17),0,BF:DRAW"BM16,10C2NF2BL2
560 2C3NF4BL2C2F2":GET(10,10)-(17,17),
570 NF4BD2C2F2":GET(10,10)-(17,17),
580 DRAW"BM11,20C1FDBG2D2FBR2UHBEBRUEBDD
590 3RRGD":GET(10,20)-(17,27),SGRASS%(
600 RAW"BM13,30C1NR2GLNR5GNR7R6D,NR7RE%R5D
610 FNDRND"BM":GET(10,30)-(17,37),SDRED%D
620 RAW"BM10,44C2END4RND2RND2FND4R4
630 ND4FL3DC0ND2LND2":GET(10,40)-(17,4
640 7),SBUILDING
650 DRAW"BM11,51C2ND6RD3C1ND3R3D3RC2U6
660 LD2L2SF"D2C0ND2":GET(10,60)-(17,67),
670 :DRAW"BM11,61C1RBF2BRRBD2BR(17,67),
680 1LHC0HLC1UC0U":GET(10,60)-(17,67),
690 SWATER
```

```
700 DATA37,60,126,219,189,126,219,153,2
710 4
720 DATA38,24,60,126,255,126,82,114,114
730 DATA39,195,255,255,255,255,231,231,
740 231
750 DATA40,255,255,223,255,255,251,255,
760 255
770 DATA63,0,0,0,0,0,0,0,0,0,0,1
780 DATA39,34,25,38,25,9,31,33,2,136,27,36,27,34,2
790 133,23,31,25,31,22,19,21,36,22,31,25,31,23
800 DATA31,25,32,21,31,11,32,16,52,11,13,7,23,34,23,42
810 4,32,22,13,11,12,36,35,16,12,11,12,22,32,26,39
820 DATA31,25,31,22,31,11,24,62,21,16,22,13,31,22,33,24,33
830 123,33,23,31,25,31,11,21,62,21,62,21,64,11,23,33,25
840 31,25
850 DATA31,21,31,21,31,21,32,13,22,16,22,16,22,16,41,14,3
860 92,13,43,23,22,33,23,62,29,22,11,13,22,33,13,4
870 DATA31,92,36,92,41,42,13,31,72,33,12,96,33,26,33,25,3
880 21,62,66,22,23,64,24,66,33,26,32,53,22,11,12,36,62,21,66
890 43,72,11,12,23,36,52,26,13,16,1
900 DATA69,61,12,23,36,33,71,22,23,36,65,53,16,13,12,33,33,63,2
910 41,91,13,12,32,21,13,99,31
920 DATA22,31,21,29,22,11,24,32,23,35,15,32,21,32,19,17,2
930 43,93,14,12,16,21,17,69,69,69,64
940 DATA36,14,61,12,73,38,22,71,13,24,52,11,12,73,38
950 32,74,15,14,12,13,32,26,33,12,88
960 DATA18,18,31,22,51,14,12,31,18,18,69,69,69,69,64
970 DATA36,35,21,39,13,22,33,38,36,35,21,39,32,23,33,83
980 62,83,32,33,26,36,24,32,32,38
990 DATA38,22,32,23,41,21,32,11,41,22,11,26,34,36,29,2
1000 92,11,14,22,24,22,33,12,52,5
1010 DATA37,21,14,23,11,23,21,32,11,24,32,23,35,15,32,21,32,11,91,72
1020 41,12,11,62,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1030 DATA22,13,42,44,12,21,12,42,21,11,12,14,12,82,66,83,22
1040 44,11,14,22,11,22,11,35,22,36,8
1050 DATA47,34,23,61,12,51,11,23,33,2
1060 11,14,32,36,46,6
1070 DATA47,32,25,41,19,11,14,22,33,64,66,38,26,41,11,13,12
1080 34,12,31,11,27,38,38,27,11,32,45,21,11,27,38
1090 DATA41,81,83,12,25,14,13,18,18,69,69,69,69,64
1100 DATA43,63,52,13,32,29,37,22,33,26,36,24,33,22,33,83
1110 62,83,32,33,52,29,37,22,33,26,36,24,33,22,33,38
1120 DATA43,82,33,23,11,26,19,12,23,43,63,92,11,34,11,12,41
1130 12,13,53,36,28,31,24,11,52,11,28
1140 DATA41,12,11,62,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1150 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1160 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1170 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1180 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1190 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1200 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1210 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1220 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1230 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1240 DATA43,11,22,16,21,11,12,43,22,13,71,91,11,21,41,22,81,89,18,3
1250 DATA36,3,0,48,0,12,0,192,0
```

HCM

PLAINS OF SALISBURY

IBM PC/PCjr, TANDY 1000

```
10 '
20 '
30 '
40 '
50 '
60 '
70 '
80 '
90 '
100 '
110 '
120 '
130 '
140 '
150 '
160 '
170 '
180 '
190 '
200 '
210 '
220 '
230 '
240 '
250 '
260 '
270 '
280 '
290 '
300 '
310 '
```

```
320 M1$="BM22,12ENDBR6NR2NL3":M2$="BM21
330 DRAW"XM3$;XM4$;:M5$="BM2
340 XM1$;XM2$;XM3$;XM4$;XM5$;X
350 CLS:LOCATE 20,1:PRINT
360 GOSUB 1310:IF A$="Y" OR A$="y" THEN
370 CLS:LOCATE 8,1:PRINT:PRINT
380 MAP$="":PRINT:PRINT" ARRA
390 GOSUB 1310:IF A$<"1"+OR A$>"3" THEN
400 MPA(Z)=(VAL(A$)-1)*40
410 NEXT Z:N=0:P=1:P1=1:P2=2:Q=1:SCORE(1)=0:
420 SCORE(2)=0:PRESTO%RE 1650:FOR Z=1 TO
430 6:READ KNIGHT(1,2),KNIGHT(1,Z,2),
440 KNIGHT(2,4),KNIGHT(2,Z,4):KNIGHT(1,Z
450 1,Z,4)=99:KNIGHT(2,Z,3)=99:NEXT
460 IVE(1):6:ALIVE(2)=6
470 CLS:LOCATE 12,1:PRINT" INITIALIZING
480 .....
490 GOSUB 1520:FOR Z=1 TO 6:KNIGHT(1,Z,)
500 5)=SCN(KNIGHT(1,Z,1),KNIGHT(2,Z,1),KN
510 :KNIGHT(2,Z,2)):NEXT
520 IGHT(2,Z,2)):KNIGHT(2,Z,1),KN
530 GOSUB 1320:GOSUB 13700
```

Continued

PLAINS OF SALISBURY *Continued*

```
K  450 ON ERROR GOTO 1840:GOSUB 480:IF ALIVE(P)=0 OR ALIVE(P2)=0 THEN 1220
Y  460 GOSUB 910:IF ALIVE(P)=0 OR ALIVE(P2)=0 THEN 1220
A  470 SWAP P,P2:FOR A=2 TO .15 STEP .05:FOR Z=1 TO 5:SOUND Z*200,A:NEXT:SOUND 20000,.5:NEXT:GOTO 450
E  480 GOSUB 830:FOR N=N TO 6:IF KNIGHT(P,N,3)<=0 THEN 820
C  490 IF FL=0 THEN MLEFT=9
X  500 KNIGHT(P,N,3)=KNIGHT(P,N,3)+5:IF KNIGHT(P,N,3)>99.9 THEN KNIGHT(P,N,3)=99.9
E  510 IF MLEFT=0 THEN SOUND 330,4:SOUND 440,8:FOR TD=1 TO 300:NEXT:GOTO 820
K  520 S=INT((KNIGHT(P,N,1)-1)/40)+1:IF Q<>S THEN Q=S:GOSUB 1320:GOSUB 1370:GOSUB 830 ELSE IF (N=1 AND MLEFT=9) OR FL=1 THEN FL=0:GOSUB 830 ELSE GOSUB 840
Q  530 IF KNIGHT(P,N,5)=5 THEN KNIGHT(P,N,4)=4
O  540 GOSUB 1310:IF A$=CHR$(13) THEN 820 ELSE A$=RIGHT$(A$,1):K=INSTR("HKMPABCD"+CHR$(27),A$):IF K=0 THEN 540 ELSE ON K GOSUB 550,570,580,560,810,810,810,1090,1090:GOTO 510
B  550 MV=-1:LIM=1:GOTO 590
F  560 MV=1:LIM=18:GOTO 590
J  570 MV=-1:LIM=1:GOTO 660
J  580 MV=1:LIM=120:GOTO 660
R  590 IF KNIGHT(P,N,2)=LIM THEN RETURN
D  600 FOR Z=1 TO 6:IF KNIGHT(P,N,1)=KNIGHT(P,Z,1) AND KNIGHT(P,N,2)+MV=KNIGHT(P,Z,2) AND KNIGHT(P,Z,3)>0 THEN RETURN
M  610 NEXT
G  620 C=SCN(KNIGHT(P,N,1)+MPA(Q),KNIGHT(P,N,2)+MV):IF C=6 OR C>MLEFT THEN RETURN ELSE MLEFT=MLEFT-C:X=KNIGHT(P,N,1):Y=KNIGHT(P,N,2):Z=KNIGHT(P,N,5):GOSUB 750:KNIGHT(P,N,2)=Y+MV:GOSUB 730:KNIGHT(P,N,3)=KNIGHT(P,N,3)-C
W  630 IF KNIGHT(P,N,3)<=0 THEN ALIVE(P)=ALIVE(P)-1:Z=C:X=KNIGHT(P,N,1):Y=KNIGHT(P,N,2):GOSUB 750:LOCATE 24,1:PRINT "KNIGHT DIED FROM EXHAUSTION...";:FOR TD=1 TO 1000:NEXT:GOSUB 830:LOCATE 24,1:PRINT SPACE$(39);:MLEFT=0:RETURN
K  640 KNIGHT(P,N,5)=C:FOR Z=1 TO 6:IF KNIGHT(P2,Z,1)=KNIGHT(P,N,1) AND KNIGHT(P2,Z,2)=KNIGHT(P,N,2) AND KNIGHT(P2,Z,3)>0 THEN MLEFT=0:GOSUB 850:RETURN
C  650 NEXT:RETURN
I  660 IF KNIGHT(P,N,1)=LIM THEN RETURN
A  670 FOR Z=1 TO 6:IF KNIGHT(P,N,1)+MV=KNIGHT(P,Z,1) AND KNIGHT(P,N,2)=KNIGHT(P,Z,2) AND KNIGHT(P,Z,3)>0 THEN RETURN
P  680 NEXT
T  690 IF KNIGHT(P,N,1)=LIM THEN 540 ELSE C=SCN(KNIGHT(P,N,1)+MV+MPA(Q),KNIGHT(P,N,2)):IF C=6 OR C>MLEFT THEN RETURN ELSE MLEFT=MLEFT-C:X=KNIGHT(P,N,1):Y=KNIGHT(P,N,2):Z=KNIGHT(P,N,5):GOSUB 750:KNIGHT(P,N,1)=X+MV:GOSUB 730:KNIGHT(P,N,3)=KNIGHT(P,N,3)-C
C  700 IF KNIGHT(P,N,3)<=0 THEN ALIVE(P)=ALIVE(P)-1:Z=C:X=KNIGHT(P,N,1):Y=KNIGHT(P,N,2):GOSUB 750:LOCATE 24,1:PRINT "KNIGHT DIED FROM EXHAUSTION...";:FOR TD=1 TO 1000:NEXT:GOSUB 830:LOCATE 24,1:PRINT SPACE$(39);:MLEFT=0:RETURN
S  710 KNIGHT(P,N,5)=C:FOR Z=1 TO 6:IF KNIGHT(P2,Z,1)=KNIGHT(P,N,1) AND KNIGHT(P2,Z,2)=KNIGHT(P,N,2) AND KNIGHT(P2,Z,3)>0 THEN MLEFT=0:GOSUB 850:RETURN
W  720 NEXT:RETURN
S  730 S=INT((KNIGHT(P,N,1)-1)/40+1):IF S<>Q THEN Q=S:GOSUB 1320:GOSUB 1370:GOSUB 830
M  740 ON (P-1)*6+N GOSUB 1400,1410,1420,1430,1440,1450,1460,1470,1480,1490,1500,1510:RETURN
S  750 ON Z GOTO 760,770,780,790,800
M  760 PUT ((X-1)*8-((Q-1)*320),(Y-1)*8),SROAD,PSET:RETURN
B  770 PUT ((X-1)*8-((Q-1)*320),(Y-1)*8),SGRASS,PSET:RETURN
G  780 PUT ((X-1)*8-((Q-1)*320),(Y-1)*8),STREE,PSET:RETURN
D  790 PUT ((X-1)*8-((Q-1)*320),(Y-1)*8),SBUILD,PSET:RETURN
M  800 PUT ((X-1)*8-((Q-1)*320),(Y-1)*8),SFORT,PSET:RETURN
W  810 TQ=Q:Q=K-4:GOSUB 1320:GOSUB 1370:LOCATE 22,1:PRINT "PRESS [RETURN] TO GO BACK":GOSUB 1310:Q=TQ:GOSUB 1320:GOSUB 1370:ON SMD GOSUB 830,1080:RETURN
L  820 NEXT:N=1:FOR Z=110 TO 600 STEP 5:SOUND Z,.15:NEXT:RETURN
G  830 SMD=1:LOCATE 20,1:PRINT NAM$(P):PRINT "MOVEMENT PHASE -- MOVE UNIT:";N:PRINT "MOVES LEFT:";MLEFT:PRINT "STRENGTH:";INT(KNIGHT(P,N,3)*.1);:RETURN
U  840 LOCATE 21,29:PRINT N:LOCATE 22,12:PRINT MLEFT:LOCATE 23,10:PRINT INT(KNIGHT(P,N,3)*.1):RETURN
Q  850 LOCATE 24,1:PRINT "HAND-TO-HAND COMBAT";:SOUND 440,3:SOUND 660,8:ATT=RND*KNIGHT(P,N,3)*.3+1:CATT=RND*KNIGHT(P2,Z,3)*.3+1:KNIGHT(P,N,3)=KNIGHT(P,N,3)-CATT:KNIGHT(P2,Z,3)=KNIGHT(P2,Z,3)-ATT:IF KNIGHT(P,N,3)>0 AND KNIGHT(P2,Z,3)>0 THEN 850
A  860 IF KNIGHT(P2,Z,3)<=0 AND KNIGHT(P,N,3)>0 THEN ALIVE(P2)=ALIVE(P2)-1:LOCATE 24,1:PRINT SPACE$(39);:LOCATE 24,1:PRINT NAM$(P);" WINS!";:KNIGHT(P2,Z,3)=0:GOSUB 740:SCORE(P)=SCORE(P)+50:LOCATE 24,1:PRINT SPACE$(39);:RETURN
I  870 IF KNIGHT(P,N,3)<0 AND KNIGHT(P2,Z,3)<=0 THEN 900
A  880 ALIVE(P)=ALIVE(P)-1:LOCATE 24,1:PRINT SPACE$(39);:LOCATE 24,1:PRINT NAM$(P2);" WINS!";
Q  890 KNIGHT(P,N,3)=0:SWAP P,P2:TN=N:N=Z:GOSUB 740:SCORE(P)=SCORE(P)+50:SWAP P,P2:N=TN:FOR TD=1 TO 1000:NEXT:LOCATE 24,1:PRINT SPACE$(39);:RETURN
Z  900 ALIVE(1)=ALIVE(1)-1:ALIVE(2)=ALIVE(2)-1:LOCATE 24,1:PRINT SPACE$(39);:LOCATE 24,1:PRINT "BOTH LOSE!";:KNIGHT(P,N,3)=0:KNIGHT(P2,Z,3)=0:Z=KNIGHT(P2,Z,5):X=KNIGHT(P,N,1):Y=KNIGHT(P,N,2):GOSUB 750:RETURN
O  910 SMD=2:C$="":FOR Z=1 TO 6:IF KNIGHT(P,Z,3)<=0 OR KNIGHT(P,Z,4)=0 THEN C$=C$+CHR$(48+Z)
U  920 NEXT:IF LEN(C$)=6 THEN N=1:RETURN ELSE GOSUB 1080
X  930 IF LEN(C$)=6 THEN 1070 ELSE LOCATE 23,1:PRINT SPACE$(78);
P  940 GOSUB 1310:IF A$=CHR$(13) OR A$=CHR$(27) THEN 1070 ELSE N=INSTR("123456ABCD",A$):IF N=0 THEN 940 ELSE IF N=10 THEN GOSUB 1090:GOTO 940 ELSE IF N>6 THEN K=N-2:GOSUB 810:GOTO 940
E  950 S=INT((KNIGHT(P,N,1)-1)/40)+1:IF Q<>S THEN Q=S:GOSUB 1320:GOSUB 1370:GOSUB 1080
K  960 A=INSTR(C$,A$):IF A>0 THEN 940 ELSE C$=C$+A$:LOCATE 23,1:PRINT "SELECT A DIRECTION TO FIRE AN ARROW:"
V  970 GOSUB 1310:IF A$=CHR$(27) OR A$=CHR$(13) THEN 930 ELSE A$=RIGHT$(A$,1):K=INSTR("HKMPABCD",A$):IF K=0 THEN 970 ELSE X=KNIGHT(P,N,1):Y=KNIGHT(P,N,2):ON K GOSUB 980,990,1000,1010,810,810,810,1090:LOCATE 24,1:GOTO 930
M  980 Y=Y-1:GOTO 1020
V  990 X=X-1:GOTO 1020
G 1000 X=X+1:GOTO 1020
R 1010 Y=Y+1
C 1020 FOR Z=1200 TO 400 STEP -10:SOUND Z,.15:NEXT:KNIGHT(P,N,4)=KNIGHT(P,N,4)-1
L 1030 FOR Z=1 TO 6:IF KNIGHT(P2,Z,1)<>X OR KNIGHT(P2,Z,2)<>Y OR KNIGHT(P2,Z,3)<=0 THEN NEXT:RETURN
X 1040 ATT=RND*(KNIGHT(P,N,3)/(SCN(X,Y)+1))*.5:KNIGHT(P2,Z,3)=KNIGHT(P2,Z,3)-ATT:IF KNIGHT(P2,Z,3)>0 THEN 1060
N 1050 KNIGHT(P2,Z,3)=0:LOCATE 24,1:PRINT "YOU DESTROYED THEM";:ALIVE(P2)=ALIVE(P2)-1:FOR TD=1 TO 1000:NEXT:Z=SCN(X,Y):GOSUB 750:LOCATE 23,1:PRINT SPACE$(60);:SCORE(P)=SCORE(P)+50:RETURN
A 1060 LOCATE 24,1:PRINT "YOU HIT THEM";:FOR TD=1 TO 1000:NEXT:LOCATE 24,1:PRINT SPACE$(18);:SCORE(P)=SCORE(P)+10:RETURN
J 1070 N=1:LOCATE 20,1:PRINT SPACE$(199);:RETURN
W 1080 LOCATE 21,1:PRINT SPACE$(120);:LOCATE 21,1:PRINT "COMBAT PHASE":PRINT "SELECT A UNIT (1 TO 6):":RETURN
J 1090 LOCATE 24,1:PRINT SPACE$(39);:LOCATE 24,1:PRINT "S)AVE   E)XIT   R)ETURN";:SOUND 880,5
L 1100 GOSUB 1310:A=INSTR("SsEeRr",A$):IF A=0 THEN 1100 ELSE A=INT(A*.5+.6):LM=1:ON A GOTO 1110,1220,1250
I 1110 F$="":LOCATE 24,1:PRINT SPACE$(39);:LOCATE 24,1:PRINT "FILE NAME:";
X 1120 B$="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"+CHR$(8)+CHR$(13)
N 1130 FOR Z=1 TO 8
J 1140 GOSUB 1310:A=INSTR(B$,A$):IF A=0 THEN 1140 ELSE IF A$=CHR$(8) THEN 1160 ELSE IF A$=CHR$(13) THEN LOCATE 24,1:PRINT SPACE$(39);:IF F$="" THEN RETURN ELSE ON LM GOTO 1190,1270
V 1150 F$=F$+A$:LOCATE 24,10+Z:PRINT A$;:GOTO 1170
N 1160 IF Z>1 THEN F$=LEFT$(F$,LEN(F$)-1):Z=Z-1:LOCATE 24,10+Z:PRINT " ";:GOTO 1140 ELSE 1140
W 1170 IF Z=8 THEN 1140
H 1180 NEXT
```

Continued

TYPE-IN LISTINGS

PLAINS OF SALISBURY *Continued*

```
N 1190 ON ERROR GOTO 1260:OPEN F$ FOR OUTPUT AS 1
O 1200 FOR A=1 TO 2:FOR Z=1 TO 6:WRITE #1,KNIGHT(A,Z,1);KNIGHT(A,Z,2);KNIGHT(A,Z,3);KNIGHT(A,Z,4);KNIGHT(A,Z,5):NEXT:NEXT:WRITE #1,NAM$(1);NAM$(2);SCORE(1);SCORE(2);ALIVE(1);ALIVE(2);P;P2;N;SMD;MLEFT;Q;MAP$;MPA(1);MPA(2);MPA(3)
Q 1210 CLOSE:FL=1:GOSUB 1320:GOSUB 1370:ON SMD GOSUB 830,1080:RETURN
L 1220 CLS:PRINT "FINAL BATTLE REPORT:":PRINT:PRINT "REMAINING UNIT STRENGTH FOR":PRINT:PRINT NAM$(1);TAB(20);NAM$(2):FOR Z=1 TO 6:PRINT INT(KNIGHT(1,Z,3)*.1);TAB(20);INT(KNIGHT(2,Z,3)*.1):NEXT:PRINT:PRINT "FINAL SCORE:":PRINT NAM$(1);" ";SCORE(1)
Z 1230 PRINT NAM$(2);" ";SCORE(2):PRINT:PRINT "PLAY AGAIN (Y/N)?"
H 1240 GOSUB 1310:IF A$="N" OR A$="n" THEN END ELSE IF A$<>"Y" AND A$<>"y" THEN 1240 ELSE GOTO 350
K 1250 LOCATE 24,1:PRINT SPACE$(39);:RETURN
N 1260 RESUME 1110
W 1270 ON ERROR GOTO 1300:OPEN F$ FOR INPUT AS 1
D 1280 FOR A=1 TO 2:FOR Z=1 TO 6:INPUT #1,KNIGHT(A,Z,1),KNIGHT(A,Z,2),KNIGHT(A,Z,3),KNIGHT(A,Z,4),KNIGHT(A,Z,5):NEXT:NEXT:INPUT #1,NAM$(1),NAM$(2),SCORE(1),SCORE(2),ALIVE(1),ALIVE(2),P,P2,N,SMD,MLEFT,Q,MAP$,MPA(1),MPA(2),MPA(3)
K 1290 CLOSE:FL=1:RETURN
S 1300 RESUME 1110
A 1310 DEF SEG=0:POKE 1050,PEEK(1052):A$="":WHILE A$="":A$=INKEY$:WEND:SOUND 800,.1:DEF SEG=0:POKE 1050,PEEK(1052):RETURN
T 1320 ON ERROR GOTO 1360:CLS:ON VAL(MID$(MAP$,Q,1)) GOTO 1330,1340,1350
N 1330 BLOAD "MAP1.PL":ON ERROR GOTO 1840:RETURN
N 1340 BLOAD "MAP2.PL":ON ERROR GOTO 1840:RETURN
N 1350 BLOAD "MAP3.PL":ON ERROR GOTO 1840:RETURN
E 1360 GOSUB 1530:RESUME 1320
X 1370 FOR ZZ=1 TO 2:FOR Z=1 TO 6:IF INT((KNIGHT(ZZ,Z,1)-1)/40)+1=Q AND KNIGHT(ZZ,Z,3)>0 THEN GOSUB 1390
W 1380 NEXT:NEXT:RETURN
P 1390 ON (ZZ-1)*6+Z GOTO 1400,1410,1420,1430,1440,1450,1460,1470,1480,1490,1500,1510
A 1400 PUT ((KNIGHT(1,1,1)-1)*8-(Q-1)*320,(KNIGHT(1,1,2)-1)*8),SR1,PSET:RETURN
Q 1410 PUT ((KNIGHT(1,2,1)-1)*8-(Q-1)*320,(KNIGHT(1,2,2)-1)*8),SR2,PSET:RETURN
Q 1420 PUT ((KNIGHT(1,3,1)-1)*8-(Q-1)*320,(KNIGHT(1,3,2)-1)*8),SR3,PSET:RETURN
A 1430 PUT ((KNIGHT(1,4,1)-1)*8-(Q-1)*320,(KNIGHT(1,4,2)-1)*8),SR4,PSET:RETURN
G 1440 PUT ((KNIGHT(1,5,1)-1)*8-(Q-1)*320,(KNIGHT(1,5,2)-1)*8),SR5,PSET:RETURN
C 1450 PUT ((KNIGHT(1,6,1)-1)*8-(Q-1)*320,(KNIGHT(1,6,2)-1)*8),SR6,PSET:RETURN
S 1460 PUT ((KNIGHT(2,1,1)-1)*8-(Q-1)*320,(KNIGHT(2,1,2)-1)*8),SB1,PSET:RETURN
E 1470 PUT ((KNIGHT(2,2,1)-1)*8-(Q-1)*320,(KNIGHT(2,2,2)-1)*8),SB2,PSET:RETURN
A 1480 PUT ((KNIGHT(2,3,1)-1)*8-(Q-1)*320,(KNIGHT(2,3,2)-1)*8),SB3,PSET:RETURN
G 1490 PUT ((KNIGHT(2,4,1)-1)*8-(Q-1)*320,(KNIGHT(2,4,2)-1)*8),SB4,PSET:RETURN
J 1500 PUT ((KNIGHT(2,5,1)-1)*8-(Q-1)*320,(KNIGHT(2,5,2)-1)*8),SB5,PSET:RETURN
T 1510 PUT ((KNIGHT(2,6,1)-1)*8-(Q-1)*320,(KNIGHT(2,6,2)-1)*8),SB6,PSET:RETURN
R 1520 RESTORE 1660:FOR Y=1 TO 18:READ A$:FOR X=1 TO 120:SCN(X,Y)=VAL(MID$(A$,X,1)):NEXT:NEXT:RETURN
F 1530 CLS:LOCATE 20,1:PRINT "PLACE A DATA DISK IN DRIVE A:":PRINT "PRESS ENTER TO SAVE SCREENS!":GOSUB 1310
E 1540 CLS:XO=0:FOR Y=1 TO 18:FOR X=1 TO 40:GOSUB 1580:NEXT:NEXT:A$="MAP1.PL":GOSUB 1570
K 1550 CLS:XO=40:FOR Y=1 TO 18:FOR X=41 TO 80:GOSUB 1580:NEXT:NEXT:A$="MAP2.PL":GOSUB 1570
P 1560 CLS:XO=80:FOR Y=1 TO 18:FOR X=81 TO 120:GOSUB 1580:NEXT:NEXT:A$="MAP3.PL":GOSUB 1570:LOCATE 22,1:PRINT "ALL THREE MAPS ARE NOW SAVED":PRINT:PRINT "PRESS SPACE BAR TO CONTINUE":GOSUB 1310:RETURN
I 1570 DEF SEG=&HB800:BSAVE A$,0,&H4000:RETURN
Y 1580 XP=(X-1-XO)*8:YP=(Y-1)*8:ON SCN(X,Y) GOTO 1590,1600,1610,1620,1630,1640
N 1590 PUT (XP,YP),SROAD,PSET:RETURN
F 1600 PUT (XP,YP),SGRASS,PSET:RETURN
O 1610 PUT (XP,YP),STREE,PSET:RETURN
Z 1620 PUT (XP,YP),SBUILD,PSET:RETURN
V 1630 PUT (XP,YP),SFORT,PSET:RETURN
O 1640 PUT (XP,YP),SWATER,PSET:RETURN
Y 1650 DATA 9,7,114,5,9,9,115,6,5,9,117,9,8,15,109,12,8,16,111,12,6,16,113,13
P 1660 DATA 333333322222222222222223333322222222333333333322222222222222223333333333223333333333333333332222233333333322333333333
T 1670 DATA 2222332221111111111122333322222222232222223333332222222222454233333333332223222222333333333322222222333333322223322222222
G 1680 DATA 333232211112222666621222333232222223333332233333222222111111111111113334222323333233333322232222222111111111111132333333
O 1690 DATA 3332322122222666666611122323333222233333332233322222221233331442222111114222233332333222333322222221333333122222111113333
S 1700 DATA 3332322126622666656622122223333332222223333233223222224233331233222333242233333332222233333322222221333322122222222213333
H 1710 DATA 222222212266266662662211111333223333322222222233222222223321223333333222222322222233323333222111111133222312266622212222
Y 1720 DATA 333111112266226662662222221333223333333332333332222222222222142233222222222223333333323333222122233333233312266666213333
W 1730 DATA 333122222266626662666222221111113333333332223333222221111112144222442222111133333333233211111222333332333122666662113333
V 1740 DATA 11112222222662666266622222222221111113111111111111111111444411111111111111661111111111111112222222333225331122666662111111
L 1750 DATA 2222222226662222222222222222223333221112266222222142212442122222133333122262222222222222222222222222223333122222666662222
V 1760 DATA 666622222266666666662222222223333333212666662222122212442124442133333122666626666222226662222223323333331222226666666666
Y 1770 DATA 666666616666633366663322222222333333316666662222122214444124142111143122226666666666666666662333333233333122222226666666
I 1780 DATA 666666616663333333663322222111111111116666666622212221111112414212333312222666666663366666233333323333333111111122666666
C 1790 DATA 6666662222333333333322222111233333326666636622214223333312414212333312222366666633332662223333332222233322222122226666
K 1800 DATA 336666622211111111111111111333333233333333322222111111111111111114433214222333333333222222222233333333222222222111222223
J 1810 DATA 322222222212222222222222223333334111133333222224133222224414422111111111223333333211111112223333333333333322266111112 3
I 1820 DATA 11111111111222222222222223333333223311111111111111333222445422222222222211111111111111222212233333333333333336666666111
Y 1830 DATA 222222222222222222222222333333333333333322222222332222242422222222222222222222222222222221223333333333333333666666222
F 1840 ER=ERR:EL=ERL:RESTORE 1890:FOR Z=1 TO 11:READ A,A$:IF A=ER THEN 1860
B 1850 NEXT:LOCATE 23,1:PRINT SPACE$(79);:LOCATE 23,1:PRINT "ERROR #";ER;" DISCOVERED":PRINT "IN LINE";EL;:GOTO 1870
Q 1860 LOCATE 23,1:PRINT SPACE$(79);:LOCATE 23,1:PRINT A$
N 1870 FOR TD=1 TO 2000:NEXT:RESUME 1880
Y 1880 CLEAR:GOTO 270
X 1890 DATA 64,BAD FILE NAME,53,FILE NOT ON DISK,57,DEVICE I/O ERROR,61,DISK IS TOO FULL,67,BAD FILE NAME,70,THIS DISK WRITE PROTECTED,71,DISK NOT READY,72,PROBLEM WITH DISKETTE,75,PATH/FILE ACCESS ERROR,76,PATH NOT FOUND,5,ILLEGAL FUNCTION CALL
```

HCM

PLAINS OF SALISBURY

```
100 REM ********** PLAINS OF SALISBURY **********
110 REM *                                        *
120 REM *  THE PLAINS OF SALISBURY  1985         *
130 REM *                                        *
140 REM *  COPYRIGHT VALLEY PUBLISHING CO.       *
150 REM *  BY WILLIAM BALL                       *
160 REM *  HOME COMPUTER MAGAZINE                *
170 REM *  VERSION 5.5.1  EXTENDED BASIC         *
180 REM *  TI CLEAR                              *
190 REM ****************************************
200 CALL CLEAR :: CALL SCREEN(2)
210 RANDOMIZE :: DIM P$(2),M$(4),R(2),M(2)
220 DIM A(1,2,3,4,5,6,8,32,1,1,2,0,0,0)
230 CALL CHAR(91,"FF8181818181FF00")
240 CALL CHAR(93,"FF8181818181FF00")
250 CALL CHAR(123,"FF8181818181FF00")
260 CALL CHAR(125,"FF8181818181FF00")
270 CALL CHAR(126,"FF8181818181FF00")
280 CALL CHAR(94,"FF8181818181FF00")
290 CALL CHAR(96,"FF8181818181FF00")
300 CALL CHAR(124,"FF8181818181FF00")
310 CALL CHAR(64,"FF8181818181FF00")
320 CALL CHAR(35,"FF8181818181FF00")
330 CALL CHAR(36,"FF8181818181FF00")
340 CALL CHAR(37,"FF8181818181FF00")
350 CALL CHAR(38,"FF8181818181FF00")
360 CALL CHAR(42,"FF8181818181FF00")
370 CALL CHAR(43,"FF8181818181FF00")
380 CALL CHAR(45,"FF8181818181FF00")
390 CALL CHAR(47,"FF8181818181FF00")
400 CALL CHAR(60,"FF8181818181FF00")
410 CALL CHAR(61,"FF8181818181FF00")
420 CALL CHAR(62,"FF8181818181FF00")
430 CALL CHAR(63,"FF8181818181FF00")
440 CALL CHAR(92,"FF8181818181FF00")
450 CALL CHAR(95,"FF8181818181FF00")
460 CALL CHAR(33,"FF8181818181FF00")
470 CALL CHAR(34,"FF8181818181FF00")
480 CALL CHAR(39,"FF8181818181FF00")
490 CALL CHAR(40,"FF8181818181FF00")
500 CALL CHAR(41,"FF8181818181FF00")
510 CALL CHAR(44,"FF8181818181FF00")
520 CALL CHAR(58,"FF8181818181FF00")
530 CALL CHAR(59,"FF8181818181FF00")
540 CALL CHAR(91,"FF8181818181FF00")
550 CALL CHAR(93,"FF8181818181FF00")
560 CALL CHAR(123,"FF8181818181FF00")
570 CALL CHAR(125,"FF8181818181FF00")
580 CALL CHAR(126,"FF8181818181FF00")
590 CALL CHAR(94,"FF8181818181FF00")
600 CALL CHAR(96,"FF8181818181FF00")
610 CALL CHAR(124,"FF8181818181FF00")
620 CALL CHAR(64,"FF8181818181FF00")
630 CALL CHAR(35,"FF8181818181FF00")
640 CALL CHAR(36,"FF8181818181FF00")
650 CALL CHAR(37,"FF8181818181FF00")
660 CALL CHAR(38,"FF8181818181FF00")
670 CALL CHAR(42,"FF8181818181FF00")
680 CALL CHAR(43,"FF8181818181FF00")
690 CALL CHAR(45,"FF8181818181FF00")
700 CALL CHAR(47,"FF8181818181FF00")
710 CALL CHAR(60,"FF8181818181FF00")
720 CALL CHAR(61,"FF8181818181FF00")
730 CALL CHAR(62,"FF8181818181FF00")
740 CALL CHAR(63,"FF8181818181FF00")
750 CALL CHAR(92,"FF8181818181FF00")
760 CALL CHAR(95,"FF8181818181FF00")
770 CALL CHAR(33,"FF8181818181FF00")
780 CALL CHAR(34,"FF8181818181FF00")
790 CALL CHAR(39,"FF8181818181FF00")
800 CALL CHAR(40,"FF8181818181FF00")
810 CALL CHAR(41,"FF8181818181FF00")
820 CALL CHAR(44,"FF8181818181FF00")
830 CALL CHAR(58,"FF8181818181FF00")
840 CALL CHAR(59,"FF8181818181FF00")
850 CALL CHAR(91,"FF8181818181FF00")
860 CALL CHAR(93,"FF8181818181FF00")
870 CALL CHAR(123,"FF8181818181FF00")
880 CALL CHAR(125,"FF8181818181FF00")
890 CALL CHAR(126,"FF8181818181FF00")
900 CALL CHAR(94,"FF8181818181FF00")
910 CALL CHAR(96,"FF8181818181FF00")
920 CALL CHAR(124,"FF8181818181FF00")
930 CALL CHAR(64,"FF8181818181FF00")
940 CALL CHAR(35,"FF8181818181FF00")
950 CALL CHAR(36,"FF8181818181FF00")
960 CALL CHAR(37,"FF8181818181FF00")
970 CALL CHAR(38,"FF8181818181FF00")
980 CALL CHAR(42,"FF8181818181FF00")
990 CALL CHAR(43,"FF8181818181FF00")
1000 CALL CHAR(45,"FF8181818181FF00")
1010 CALL CHAR(47,"FF8181818181FF00")
1020 CALL CHAR(60,"FF8181818181FF00")
1030 CALL CHAR(61,"FF8181818181FF00")
1040 CALL CHAR(62,"FF8181818181FF00")
1050 CALL CHAR(63,"FF8181818181FF00")
1060 CALL CHAR(92,"FF8181818181FF00")
1070 CALL CHAR(95,"FF8181818181FF00")
1080 CALL CHAR(33,"FF8181818181FF00")
1090 CALL CHAR(34,"FF8181818181FF00")
1100 CALL CHAR(39,"FF8181818181FF00")
1110 CALL CHAR(40,"FF8181818181FF00")
1120 CALL CHAR(41,"FF8181818181FF00")
1130 CALL CHAR(44,"FF8181818181FF00")
1140 CALL CHAR(58,"FF8181818181FF00")
1150 CALL CHAR(59,"FF8181818181FF00")
1160 CALL HCHAR(22,14,J+48)
1170 CALL HCHAR(23,12,INT(L(P,N,A3)/10+4
     8))
1180 IF L(P,N,A5)<>113 THEN 1200
1190 L(P,N,A4)=A4
1200 OSUB 3860
1210 IF <128 THEN 1250
1220 OSUB 3310
1230 CALL HCHAR(21,25,C1)
1240 CALL COLOR(10,30,
1250 KEY$(K),CHR$(K),A1)
1260 IF A0 THEN 1200
1270 A1
1280 GOTO 1300,1320,1340,1360,2040,
1290 K
1300 A1
1310 A0 1400
1320 A0 1400
1330 A0 1400
1340 A1 1580
1350 A0 1580
1360 A0 1580
1370 OUB 4020
1380 OUB 1030
1390 OUB (P,N,A1)=Z THEN 1030
1400 L(P,N,A1)=L(P,N,A3)-H
1410 (P,N,A3)=L(P,N,A3)-H
1420 (P,N,A5)=C
1430 >A0 THEN 1540
1440 (P,N,A3)=L(P,N,A3)+U
1450 (P,N,A3)=L(P,N,A3)-H
1460 (P,N,A3)=L(P,N,A3)+U=A0)+(L(P,N,A2)+U=85
1470 (L(P,N,A2)=A0)THEN 1570
1480 (F$(L)="(Y/N)?":F$(L)="
     ")
1490 (F$(L)="USING 1 2
     ")
1500 (L(P,N,A3)=L(P,N,A3)-H
1510 (P,N,A3)=L(P,N,A3)+U=A0)+(L(P,N,A2)+U=85
1520 (L(P,N,A2)=A0)THEN 1570
1530 (F$(L)="(Y/N)?":F$(L)="
     ")
1540 (F(N)+U>A2)*(F3(N)+U<31)THEN 16
     60
1550 IF J>A0 THEN 1620
1560 O+U
1570 A1 1030
1580 (P,N,A3)=L(P,N,A3)-H
1590 (P,N,A2)=L(P,N,A2)+U
1600 A1 3340
1610 (P,N,A5)=C
1620 (F1(N),F3(N)+U,C)
1630 A0
1640 IF J>A1 THEN 1720
1650 (F1(N),F3(N),L(P,N,A5)))
1660 A0
1670 (P,N,A5)=C
1680 IF J>A1 THEN 1720
1690 (P,N,A3)>0 THEN 1780
1700 (F1(N),F3(N),L(P,N,5)))
1710 O+U
1720 A0 1030
1730 (P,N,A3)=L(P,N,A3)-H
1740 (P,N,A2)=L(P,N,A2)+U
1750 (F1(N),F3(N),C1)
1760 A1 1030
1770 (P,N,A3)>0 THEN 1780
1780 (F1(N),F3(N),L(P,N,5)))
1790 O+U
1800 A0 1030
1810 (F1(N),F3(N),C1)
1820 A0 1030
1830 (E$,CHR$(C),A1)
1840 A6 THEN 1890
1850 A6 THEN 1880
1860 IF 100
1870 RN A2 THEN 1950
1880 (C<136)+(C>143)THEN 2020
1890 (A2,C-135,A5)
1900 A0 1840
1910 A0 1990
1920 (C<128)+(C>135)THEN 2020
1930 (A1,C-127,A5)
1940 A0 1840
1950 J THEN 2010
1960 H
1970 RN
1980 RN
1990 XT
2000 SOUND(100,440,A0)
2010 SOUND(250,660,A0)
2020 "HAND TO HAND COMBAT"
2030 A4
2040 A0 395
2050 (P=A1)*A8
2060 -1 27+(P=A1)*A8
2070 SOUND(100,392,A0)
2080 SOUND(150,440,A0)
2090 D *L(P,N,A3)*.2+A1
2100 D *L(P2,A,A3)*.2+A1
2110 L(P,2,N,A3)=L(P,N,A3)-D
2120 L(P,2,A,A3)=L(P2,A,A3)-B
2130 (L(P2,N,A3)>A0)*((L(P2,A,A3)>A0)TH
     EN 2190
2140 CALL HCHAR(24,A1,I,I))
```

Continued

TYPE-IN LISTINGS

PLAINS OF SALISBURY *Continued*


```
O 2200 IF L(P,N,A3)>A0 THEN 2310
M 2210 IF L(P2,A,A3)<=A0 THEN 2410
K 2220 CALL HCHAR(F1(N),F3(N),G)
Z 2230 A$=P$(P2)&" WINS"
X 2240 M(P2)=M(P2)+50
J 2250 Z=A4
P 2260 GOSUB 3950
U 2270 GOSUB 3990
X 2280 CALL HCHAR(24,A1,I,I)
Z 2290 R(P2)=R(P2)-A1
B 2300 RETURN
R 2310 CALL HCHAR(F1(N),F3(N),119+P*A8+N)
D 2320 A$=P$(P)&" WINS"
Y 2330 L(P,N,A5)=L(P2,A,A5)
W 2340 M(P)=M(P)+50
W 2350 Z=A4
C 2360 GOSUB 3950
Z 2370 GOSUB 3990
S 2380 CALL HCHAR(24,A1,I,I)
M 2390 R(P)=R(P)-A1
I 2400 RETURN
V 2410 A$="BOTH LOSE"
I 2420 Z=A4
C 2430 CALL HCHAR(F1(N),F3(N),L(P2,A,A5))
L 2440 GOSUB 3950
W 2450 GOSUB 3990
Y 2460 R(A1)=R(A1)-A1
J 2470 R(A2)=R(A2)-A1
O 2480 RETURN
D 2490 C$=""
N 2500 IF LEN(C$)<6 THEN 2520
K 2510 RETURN
O 2520 CALL HCHAR(20,A1,I,160)
V 2530 M$(A2)="COMBAT PHASE"
A 2540 M$(A3)="CHOOSE UNIT (1-6):"
A 2550 M$(A4)=""
J 2560 GOSUB 3900
Z 2570 CALL HCHAR(23,A1,I,64)
U 2580 GOSUB 3860
I 2590 IF K<>13 THEN 2610
Q 2600 RETURN
D 2610 IF K<128 THEN 2640
H 2620 GOSUB 3310
X 2630 GOTO 2580
G 2640 IF (K<49)+(K>54)THEN 2580
I 2650 N=K-48
J 2660 B=POS(C$,CHR$(N),A1)
O 2670 IF (B=A0)*(L(P,N,A3)>A0)THEN 2700
S 2680 CALL SOUND(300,110,A0)
N 2690 GOTO 2580
P 2700 S=INT((L(P,N,A2)-A1)/28)+A1
M 2710 IF S=Q THEN 2740
G 2720 Q=S
S 2730 GOSUB 3340
O 2740 A$="DIRECTION: ESDX"
R 2750 Z=A3
I 2760 GOSUB 3950
L 2770 GOSUB 3860
N 2780 K=POS(D$,CHR$(K),A1)
Y 2790 IF K=A0 THEN 2810
W 2800 ON K GOTO 2840,2860,2890,2910,2570,
       2570
L 2810 IF K<128 THEN 2770
F 2820 GOSUB 3310
U 2830 GOTO 2770
W 2840 Y=L(P,N,A1)-A1
N 2850 GOTO 2870
W 2860 Y=L(P,N,A1)+A1
A 2870 X=L(P,N,A2)
N 2880 GOTO 2930
L 2890 X=L(P,N,A2)-A1
T 2900 GOTO 2920
Q 2910 X=L(P,N,A2)+A1
E 2920 Y=L(P,N,A1)
J 2930 IF L(P,N,A4)>A0 THEN 3010
W 2940 A$="OUT OF ARROWS"
M 2950 Z=A4
A 2960 CALL SOUND(200,110,A0)
D 2970 GOSUB 3950
G 2980 GOSUB 3990
B 2990 CALL HCHAR(24,A1,I,I)
Z 3000 GOTO 2500
L 3010 FOR Z=1200 TO 440 STEP -200
G 3020 CALL SOUND(-100,Z,A0)
B 3030 NEXT Z
D 3040 L(P,N,A4)=L(P,N,A4)-A1
R 3050 C$=C$&CHR$(N)
L 3060 FOR Z=A1 TO A6
R 3070 IF (L(P2,Z,A1)=Y)*(L(P2,Z,A2)=X)*(L
       (P2,Z,A3)>A0)THEN 3100
A 3080 NEXT Z
D 3090 GOTO 2570
L 3100 F=POS(E$,CHR$(L(P2,Z,A5)),A1)-A1
T 3110 IF F<A0 THEN 2500
F 3120 B=RND*(L(P,N,A3)/(F+A1))*.5
R 3130 L(P2,Z,A3)=L(P2,Z,A3)-B
E 3140 IF L(P2,Z,A3)>A0 THEN 3230
M 3150 M(P)=M(P)+50
N 3160 CALL HCHAR(L(P2,Z,A1),L(P2,Z,A2)+A2
       -(Q-A1)*28,L(P2,Z,A5))
P 3170 A$="YOU DESTROYED THEM"
A 3180 Z=A4
C 3190 GOSUB 3950
A 3200 GOSUB 3990
V 3210 CALL HCHAR(24,A1,I,I)
L 3220 GOTO 2500
K 3230 A$="A HIT"
A 3240 M(P)=M(P)+10
A 3250 R(P2)=R(P2)-A1
B 3260 Z=A4
Z 3270 GOSUB 3950
I 3280 GOSUB 3990
F 3290 CALL HCHAR(24,A1,I,I)
A 3300 GOTO 2500
N 3310 IF (K>176)*(K<180)THEN 3330
Y 3320 RETURN
H 3330 Q=K-176
V 3340 GOSUB 3480
V 3350 GOSUB 3380
M 3360 GOSUB 3900
H 3370 RETURN
T 3380 B=(Q-A1)*28+A1
Z 3390 D=Q*28
N 3400 FOR Z=A1 TO A2
O 3410 FOR V=A1 TO A6
Z 3420 IF (L(Z,V,A2)<B)+(L(Z,V,A2)>D)+(L(Z
       ,V,A3)<=A0)THEN 3450
U 3430 CALL GCHAR(L(Z,V,A1),L(Z,V,A2)-(Q-A
       1)*28+A2,L(Z,V,A5))
R 3440 CALL HCHAR(L(Z,V,A1),L(Z,V,A2)+A2-(
       Q-A1)*28,Z*A8+119+V)
D 3450 NEXT V
I 3460 NEXT Z
O 3470 RETURN
B 3480 V=VAL(SEG$(B$,Q,A1))
A 3490 ON V GOTO 3500,3520,3540
Q 3500 RESTORE 3670
P 3510 GOTO 3550
N 3520 RESTORE 3730
J 3530 GOTO 3550
N 3540 RESTORE 3790
A 3550 CALL CLEAR
N 3560 FOR Z=A1 TO 17
O 3570 READ A$
E 3580 PRINT A$
K 3590 NEXT Z
A 3600 PRINT : : : : : :
F 3610 CALL HCHAR(18,A3,120,28)
N 3620 RETURN
V 3630 DATA 35,0080000800200002,97,103854B
       A549210,104,0080000800200002,112,00
       183C7EFF5252 72,120,0070000000007
S 3640 DATA 113,00C3C3FFFFE7E7E7
E 3650 DATA 0010301010101038,0038440408102
       07C,0038440418044438,00081828487C08
       08,007C407804044438,00182040784444 3
       8
W 3660 DATA 11,4,2,4,2,4,2,4,2,4,2,4,2,4,2
       ,4,13,4,2,12,16,4,16,5,2,10,2,8
W 3670 DATA aaaaaa######aaaaa######aa,#
       ##a##hhhhhhhhh#aaaaaa##a###,aa#ahhh
       #xxxxx#haaaaaa###aaa
F 3680 DATA aa#ah###xxxqx#h#aa#####aaa,a
       a#ah####xx#xxhhha##aaa####a,###ah#x
       x#xx#xxxxh###aaa#####
U 3690 DATA aahhh#xx#xx#xxxxhhhhaaaaaaaa,a
       ah###xx#########haa##aaaa,hhh###x
       xxxxxxxx###hhhh#aaah
F 3700 DATA #######xxxaaxxx#####aahhhhhh,x
       xx####xxxaaxxx####aa#h###x,xxxxxhx
       xxxaaaaaa##hhhhhhxxxx
X 3710 DATA xxxxxhxxxxaaaaaa#h##aaaxxxx,x
       xxx###xxxaaaaaahh##aaaaaax,axxx###
       #hhhhhhhhhha##aa#aaaa
X 3720 DATA #######h####aa###aaaaahhhhh,h
       hhhhhhhh####aaaaaaaaaap#aah
F 3730 DATA aaaaa##########aaa######aa,#
       #aaa######pqp#aaa#####a##,a##aaa#
       ##phhhhhhha#p####a#a
U 3740 DATA a###aa###phaahpp#hhhh#####a,a
       a##aa###p#aahp##h######aaaa,######
       ######hpp##pp###a####
W 3750 DATA a#aaaa####p##h####p#####aaa,a
       ##aa####hp#hhhhh#h####aa#a,hhhhhhh
       hhhhhh#p##hhhhhhhhhh
I 3760 DATA #######pph##hhpp#h#p########,x
       xaa####phpph#p##haaaaa###xx,xaaaa##
       ##hpphhhhhaaaa####xx
A 3770 DATA xxxx#####h###aaa#hppp###xxxx,x
       aa#####phhhhhhhhhpp###xxxx,aa#####
       #pha###p##h######aa
Y 3780 DATA aa#######haappppp#h#######aa,h
       hhhhhhhhha##qp###hhhhhhhhh
D 3790 DATA aaaaa#aaaaaaaaaaa###aaaaaaaa,#
       #aaa####aaaaaaa###aaaaa###,a##aa##
       ####aaaaaa####aa###aa
A 3800 DATA aa##a#####hhhhhhhhhh##aaaa,a
       aa######haaaah####h#aaaaa,##a###
       hhhhha###h####hhhhh##
N 3810 DATA a#a###h###aqa#h##xx###haa,a
       #a#hhhh#aaaa#aah##xx####haa,hhhh##
       ##aa###aah##xxxxx#hhh
U 3820 DATA #######aaa##aaaah##xxxxx####,x
       ######aa####aah####xxxxxx,xxx###a
       aa#a###aahhhh#xxxxxx
S 3830 DATA xxx#######a#aaaaa#h######xx,x
       aa#####aaa#####aa#hhhhhhaxx,aa####a
       aaaaa####aa######haaa
B 3840 DATA a###aaaaa#aaa##########haaa,h
       hhh###aa###a######xxx#hhhh
U 3850 DATA 7,6,7,79,9,6,7,81,11,6,9,81,15
       ,6,13,79,17,6,15,79,17,4,15,81
Y 3860 CALL KEY(A0,K,S)
U 3870 IF S=A0 THEN 3860
N 3880 CALL SOUND(50,880,A0)
Y 3890 RETURN
```

Continued

PLAINS OF SALISBURY *Continued*

TI-99/4A

```
100 FOR Z=A0 TO A3
110 A$=MS(Z+A1)
120 GOSUB 3950
130 NEXT Z
140 RETURN
150 FOR V=A0 TO LEN(A$)-A1
160 CALL HCHAR(20+Z,A3+V,ASC(SEG$(A$,V+
    A1,A1)))
170 NEXT V
180 RETURN
190 FOR Z=A1 TO 600
200 NEXT Z
210 RETURN
220 A$="S)AVE E)XIT R)ETURN?"
230 Z=A4
240 GOSUB 3950
250 GOSUB "STHEN 4050
260 IF Z=A0 THEN 4230,4090,4210
270 ON CALL CLEAR
280 PRINT "'S STRENGTH:":::TAB(8)
    ;PS(A1);TAB(16);PS(A2)
290 FOR Z=A1 TO A6
300 PRINT :::Z;")";
310 FOR V=A1 TO A3)>A0 THEN 4160
320 IF L(V,Z,A3)=A0
330 PRINT TAB(V*8);INT(L(V,Z,A3)/10);
340 NEXT V
350 PRINT :::"SCORE:";M(A1);TAB(16);
    M(A2)
360 END
370 CALL HCHAR(24,A1,I,I)
380 RETURN
390 CALL CLEAR
400 INPUT "DEVICE/FILE NAME ":F$
410 IF THEN 4270
420 OPEN #A1:F$,OUTPUT,FIXED 128,INTERN
    AL
430 FOR Z=A1 TO A6
440 PRINT #A1:PS(A1);PS(A2);BS;R(A1);R(
    A2);M(A1);Q;P;J;N
450 NEXT Z
460 CLOSE #A1
470 END
480 CALL HCHAR(24,A1,I,I)
490 RETURN
500 CALL CLEAR
510 INPUT "DEVICE/FILE NAME ":F$
520 IF THEN 4380
530 OPEN #A1:F$,INPUT ,FIXED 128,INTERN
    AL
540 FOR Z=A1 TO A6
550 INPUT #A1:PS(A1),PS(A2),BS,R(A1),R(
    A2),M(A1),Q,P,J,N
560 NEXT Z
570 CLOSE #A1
580 RETURN
```

HCM

SOUND-ON-SOUND

ATARI 800/800XL/130XE

```
100 REM ********** SOUND-ON-SOUND **********
110 REM *  SOUND-ON-SOUND  *
120 REM ********** SOUND-ON-SOUND **********
130 REM COPYRIGHT 1985
140 REM EMERALD VALLEY PUBLISHING CO.
150 REM BY WILLIAM BALTHROP
160 REM HOME COMPUTER MAGAZINE
170 REM VERSION 5.5.1
180 REM ATARI BASIC FOR THE 800, 800XL,
    130XE
190 REM
200 DIM CL$(1),V0$(1001),V1$(1001),V2$(
    100),V3$(1001),KPRESS(26),PITCH(21
    ),SHARP(21),FLAT(21),CHAN(4),FL$(20
    )
210 DIM S0$(1001),S1$(1001),S2$(1001),S
    3$(1001),E$(10),V(10):E$="CHORTLONG
    ":TMPO=240
220 TMP=16.166666:DIM T$(20),BF1$(100)
    :V$(1)=10:V(2)=10:V(3)=10:V(4)=10
230 GOSUB 1680:GOSUB 1690:GOSUB 1700:GO
    SUB 1710
240 CL$=CHR$(125):FOR Z=0 TO 21:READ A,
    B,C,D:FLAT(Z)=C:NEXT Z
250 FOR Z=1 TO 26:READ A:CHAN(Z)=1:NEXT
    Z
260 OPEN #1,4,0,"K:"
270 GRAPHICS 0:PRINT CL$:POSITION 13,2:
    PRINT "SOUND-ON-SOUND":PRINT :PRINT
    :PRINT "COMPOSE MUSIC":PRINT :PRINT
    :PRINT "PLAY MUSIC":PRINT :PRINT
    :PRINT "FILES":PRINT "EXIT PROGRAM"
280 POKE 764,255:GET #1,A:IF A<49 OR A>
    53 THEN 330
290 ON A-48 GOSUB 350,770,1030,1410:GOTO
    310
300 PRINT CL$:POSITION 13,2:PRINT "COMP
    OSE MUSIC":PRINT :PRINT "1) RECORD
    TIME":PRINT :PRINT "2) SET TEMPO":
    PRINT :PRINT "3) MODE":PRINT "4) CL
    EAR TRACK":PRINT :PRINT "5) SET VOI
    CE TYPE":PRINT "ESC) RETURN TO MAIN
    MENU"
310 POKE 764,255:GET #1,A:IF A<49 OR A>
    53 THEN 330
320 IF A=27 AND TH=48 GOSUB 350,770,103
    0,1410:GOTO 310
330 PRINT CL$:POSITION 13,2:PRINT "COMP
    OSE MUSIC":PRINT :PRINT "1) RECORD
    TIME":PRINT :PRINT "2) SET TEMPO":
    PRINT :PRINT "3) MODE":PRINT "4) CL
    EAR TRACK":PRINT :PRINT "5) SET VOI
    CE TYPE":PRINT "ESC) RETURN TO MAIN
    MENU"
340 POKE 764,255:GET #1,A:IF A<48 OR A>
    53 THEN 340
350 PRINT CL$:POSITION 13,2:PRINT "RECO
    RD TIME":PRINT :PRINT "ENTER TRACK
    (0 TO 15)":INPUT FL$:FL$(LEN(FL$)+1
    )=".CE":V(0)=VAL(FL$(1,2)):IF V(0)>
    15 THEN 350
360 OPEN #1,8,0,FL$:POKE 752,1:PRINT CL
    $:POSITION 14,2:PRINT "RECORDING":
    PRINT :PRINT "PLAY MUSIC":PRINT :PR
    INT "ENTER TRACK TO PLAY":PRINT :PR
    INT "ESC) RETURN TO MENU"
370 POKE 764,255:GET #1,A:IF A=27 THEN
    POKE 752,0:CLOSE #1:GOTO 300
380 IF A<48 OR A>57 THEN 370
390 TRACK=A-48:IF TRACK=0 THEN 370
400 POKE 764,255:GET #1,A:IF A<48 OR A>
    57 THEN 400
410 TRACK=TRACK*10+A-48:GOTO 410,860,510
    ,680,720,740,760,770,790,810,830,84
    0,850:GOTO 400
420 POKE 764,255:GET #1,A:IF A<48 OR A>
    57 THEN 420
430 VOICE=A-48:IF VOICE<0 OR VOICE>7 TH
    EN 420
440 PRINT CL$:POSITION 14,2:PRINT "ENTE
    R VOICE TYPE (0 TO 15)"
450 POKE 764,255:GET #1,A:IF A<48 OR A>
    53 THEN 450
460 IF A=48 THEN A=10
470 IF A=49 THEN A=11
480 IF A=50 THEN A=12
490 IF A=51 THEN A=13
500 IF A=52 THEN A=14
510 IF A=53 THEN A=15
520 POKE 764,255:GET #1,A:IF A<48 OR A>
    57 THEN 520
530 VOICE=A-48:IF VOICE<0 OR VOICE>15 T
    HEN 530
540 POKE 752,1:PRINT CL$:POSITION 2,2:P
    RINT "NOTES:":PRINT :PRINT "SUSS"
550 POKE 752,0:PRINT CL$:POSITION 2,2:P
    RINT "TIMER:"
560 POKE 764,255:GET #1,A:IF A<48 OR A>
    57 THEN 560
570 TIMER=A-48:IF TIMER<0 OR TIMER>9 TH
    EN 570
580 POKE 752,1:PRINT CL$:POSITION 8,20:
    PRINT "TIMER=";TIMER
590 POKE 764,255:GET #1,A:IF A=255 THEN
    590
600 IF KEY=27 THEN GOSUB 640:GOTO 560
610 IF KEY=42 THEN GOSUB 660:GOTO 560
620 IF (KEY<112 OR KEY>122) AND KEY<>32
    AND KEY<>129 THEN 570
630 IF KEY<>146 THEN 570
640 IF TIMER<1 THEN TIMER=10
650 RETURN
660 IF TIMER>10 THEN TIMER=1
670 RETURN
680 POKE 752,0:PRINT CL$:POSITION 2,10:
    PRINT "(4) CLEAR TRACK"
690 POKE 752,1:PRINT CL$:POSITION 2,15:
    PRINT "ENTER TRACK TO CLEAR:";
700 INPUT A:IF A<1 OR A>15 THEN 690
710 IF A<47 OR A>48 THEN 690
720 POKE 752,0:PRINT CL$:POSITION 2,12:
    PRINT "(5) SET VOICE TYPE"
730 POKE 752,1:PRINT CL$:POSITION 2,16:
    PRINT "ENTER TRACK (0 TO 15)":INPUT
    FL$:FL$(LEN(FL$)+1,LEN(FL$)+2)=".CE
    ":V(0)=VAL(FL$(1,2)):IF V(0)>15 THE
    N 730
740 POKE 764,255:GET #1,A:IF A<48 OR A>
    57 THEN 740
750 VOICE=A-48:PRINT CL$:POSITION 2,1:P
    RINT "CURRENT VOICE TYPE=";V(VOICE)
760 POKE 752,0:PRINT CL$:POSITION 2,15:
    PRINT "ENTER VOICE TYPE (0 TO 15)":
    INPUT FL$:FL$(LEN(FL$)+1,LEN(FL$)+2
    )=".CE":V(0)=VAL(FL$(1,2)):IF V(0)>
    15 THEN 760
770 DIM J$(70),REST$(70),NOTE$(70),TEMP
    O$(70),MODE$(70),SET$(70),TRACK$(70
    ),PLAY$(70),REC$(70),CLEAR$(70),SUS
    $(70),TIMER$(70),NOTES$(70),VOICE$(
    70),ENTER$(70),TRACK$(70),PLAY$(70)
    ,MUSIC$(70),PRINT$(70),SET$(70),TEM
    PO$(70)
```

Continued

TYPE-IN LISTINGS

```
N  780 PRINT :PRINT "3) SET TRACK":PRINT :PRINT "ESC) RETURN TO MAIN MENU"
B  790 POKE 764,255:GET #1,A:IF (A<49 OR A>51) AND A<>27 THEN 790
N  800 IF A=27 THEN RETURN
B  810 M=2:ON A-48 GOTO 820,860,910
Z  820 POSITION 2,4:PRINT "1) ■PLAY MUSIC■":POSITION 2,18:T=1:PRINT "TIMER:";T;
T  830 POKE 764,255:FOR T=1 TO 1000:POSITION 8,18:PRINT T;"  ";:GOSUB 1560:FOR TD=1 TO TMP:NEXT TD
Y  840 A=PEEK(764):KEY=PEEK(A+64337):IF KEY=27 THEN GOSUB 1450:GOTO 770
N  850 NEXT T:GOSUB 1450:GOTO 770
N  860 POSITION 2,6:PRINT "2) ■SET TEMPO■":POSITION 2,15:PRINT "CURRENT TEMPO=";TMPO:PRINT "ENTER TEMPO:";
Q  870 INPUT FL$:FL$(LEN(FL$)+1,LEN(FL$)+2)=".0":TMPO=VAL(FL$):POSITION 14,16:PRINT TMPO;
N  880 IF TMPO>380 THEN TMPO=380
C  890 IF TMPO<40 THEN TMPO=40
L  900 TMP=10600/TMPO-28:ON M GOTO 350,770
Y  910 POSITION 2,8:PRINT "3) ■SET TRACK■":FOR CH=1 TO 4:ON CH GOSUB 950,970,990,1010:NEXT CH
O  920 POKE 764,255:GET #1,A:IF (A<48 OR A>51) AND A<>27 THEN 920
N  930 IF A=27 THEN 770
A  940 A=A-47:CHAN(A)=ABS(CHAN(A)-1):ON A GOSUB 950,970,990,1010:GOTO 920
T  950 POSITION 5,15:IF CHAN(1)=0 THEN PRINT "0    ■OFF■":RETURN
V  960 PRINT "0      ■ON■  ":RETURN
N  970 POSITION 5,17:IF CHAN(2)=0 THEN PRINT "1    ■OFF■":RETURN
U  980 PRINT "1      ■ON■  ":RETURN
H  990 POSITION 5,19:IF CHAN(3)=0 THEN PRINT "2    ■OFF■":RETURN
B 1000 PRINT "2      ■ON■  ":RETURN
Q 1010 POSITION 5,21:IF CHAN(4)=0 THEN PRINT "3    ■OFF■":RETURN
Q 1020 PRINT "3      ■ON■  ":RETURN
L 1030 PRINT CL$:POSITION 13,2:PRINT "■LOAD/SAVE DATA■":POSITION 2,4:PRINT "1) LOAD DATA FILE":PRINT
X 1040 PRINT "2) SAVE DATA FILE":PRINT :PRINT "ESC) RETURN TO MAIN MENU":POKE 764,255
M 1050 GET #1,A:IF (A<49 OR A>50) AND A<>27 THEN 1050
U 1060 IF A=27 THEN RETURN
X 1070 IF A=50 THEN 1250
O 1080 POSITION 2,4:PRINT "1) ■LOAD DATA FILE■":POSITION 2,12:PRINT "ENTER FILE NAME:";
T 1090 INPUT FL$:IF LEN(FL$)=0 THEN 1030
Y 1100 IF LEN(FL$)>1 THEN IF FL$(1,2)="C:" THEN FL$="C:":GOTO 1140
U 1110 IF LEN(FL$)>1 THEN IF FL$(1,2)="D:" THEN 1310
E 1120 IF LEN(FL$)>2 THEN IF FL$(1,1)="D" AND FL$(3,3)=":" THEN 1310
H 1130 T$=FL$(1,LEN(FL$)):FL$(3,14)=T$:FL$(1,2)="D:"
A 1140 OPEN #3,4,0,FL$
O 1150 V0$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:V0$(Z,Z+99)=BF1$(1,LEN(BF1$))
O 1160 NEXT Z:V1$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:V1$(Z,Z+99)=BF1$(1,LEN(BF1$))
X 1170 NEXT Z:V2$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:V2$(Z,Z+99)=BF1$(1,LEN(BF1$))
A 1180 NEXT Z:V3$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:V3$(Z,Z+99)=BF1$(1,LEN(BF1$))
L 1190 NEXT Z:S0$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:S0$(Z,Z+99)=BF1$(1,LEN(BF1$))
D 1200 NEXT Z:S1$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:S1$(Z,Z+99)=BF1$(1,LEN(BF1$))
S 1210 NEXT Z:S2$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:S2$(Z,Z+99)=BF1$(1,LEN(BF1$))
B 1220 NEXT Z:S3$(1001)=E$:FOR Z=1 TO 901 STEP 100:INPUT #3;BF1$:S3$(Z,Z+99)=BF1$(1,LEN(BF1$))
V 1230 NEXT Z
K 1240 INPUT #3;TMPO:INPUT #3;TMP:CLOSE #3:RETURN
P 1250 POSITION 2,6:PRINT "2) ■SAVE DATA FILE■":POSITION 2,12:PRINT "ENTER FILE NAME:";
U 1260 INPUT FL$:IF LEN(FL$)=0 THEN 1030
F 1270 IF LEN(FL$)>1 THEN IF FL$(1,2)="C:" THEN FL$="C:":GOTO 1310
U 1280 IF LEN(FL$)>1 THEN IF FL$(1,2)="D:" THEN 1310
A 1290 IF LEN(FL$)>2 THEN IF FL$(1,1)="D" AND FL$(3,3)=":" THEN 1310
E 1300 T$=FL$(1,LEN(FL$)):FL$(3,14)=T$:FL$(1,2)="D:"
T 1310 OPEN #3,8,0,FL$
M 1320 V0$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;V0$(Z,Z+99):NEXT Z
J 1330 V1$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;V1$(Z,Z+99):NEXT Z
M 1340 V2$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;V2$(Z,Z+99):NEXT Z
V 1350 V3$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;V3$(Z,Z+99):NEXT Z
A 1360 S0$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;S0$(Z,Z+99):NEXT Z
H 1370 S1$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;S1$(Z,Z+99):NEXT Z
U 1380 S2$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;S2$(Z,Z+99):NEXT Z
N 1390 S3$(1001)="Z":FOR Z=1 TO 901 STEP 100:PRINT #3;S3$(Z,Z+99):NEXT Z
U 1400 PRINT #3;TMPO:PRINT #3;TMP:CLOSE #3:RETURN
U 1410 PRINT CL$:POSITION 2,2:PRINT "ARE YOU SURE YOU WANT TO EXIT?":PRINT "Yes or No?"
Q 1420 POKE 764,255:GET #1,A:IF A<>89 AND A<>78 THEN 1420
Z 1430 IF A=78 THEN RETURN
U 1440 END
Q 1450 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,0,0,0:SOUND 3,0,0,0:RETURN
I 1460 IF KEY>64 AND KEY<91 THEN KEY=SHARP(KPRESS(KEY-64)):GOTO 1510
F 1470 IF KEY>96 AND KEY<123 THEN KEY=PITCH(KPRESS(KEY-96)):GOTO 1510
X 1480 IF KEY>0 AND KEY<27 THEN KEY=FLAT(KPRESS(KEY)):GOTO 1510
P 1490 IF KEY=129 THEN PFLG=ABS(PFLG-1):POSITION 2,22:PRINT "NOTES:      ";SUS$(PFLG*5+1,PFLG*5+5);:POKE 764,255:RETURN
A 1500 KEY=0
V 1510 ON VOICE GOTO 1520,1530,1540,1550
Q 1520 V0$(T,T)=CHR$(KEY):S0$(T,T)=CHR$(PFLG):GOTO 1560
J 1530 V1$(T,T)=CHR$(KEY):S1$(T,T)=CHR$(PFLG):GOTO 1560
W 1540 V2$(T,T)=CHR$(KEY):S2$(T,T)=CHR$(PFLG):GOTO 1560
A 1550 V3$(T,T)=CHR$(KEY):S3$(T,T)=CHR$(PFLG)
C 1560 KEY=ASC(V0$(T,T)):IF KEY=0 OR CHAN(1)=0 THEN SOUND 0,0,0,0:GOTO 1590
Q 1570 IF ASC(S0$(T,T))=0 THEN SOUND 0,0,0,0:FOR TD=1 TO TMP*0.05:NEXT TD
K 1580 SOUND 0,KEY,V(1),8
W 1590 KEY=ASC(V1$(T,T)):IF KEY=0 OR CHAN(2)=0 THEN SOUND 1,0,0,0:GOTO 1620
A 1600 IF ASC(S1$(T,T))=0 THEN SOUND 1,0,0,0:FOR TD=1 TO TMP*0.05:NEXT TD
S 1610 SOUND 1,KEY,V(2),8
S 1620 KEY=ASC(V2$(T,T)):IF KEY=0 OR CHAN(3)=0 THEN SOUND 2,0,0,0:GOTO 1650
N 1630 IF ASC(S2$(T,T))=0 THEN SOUND 2,0,0,0:FOR TD=1 TO TMP*0.05:NEXT TD
N 1640 SOUND 2,KEY,V(3),8
F 1650 KEY=ASC(V3$(T,T)):IF KEY=0 OR CHAN(4)=0 THEN SOUND 3,0,0,0:RETURN
Q 1660 IF ASC(S3$(T,T))=0 THEN SOUND 3,0,0,0:FOR TD=1 TO TMP*0.05:NEXT TD
A 1670 SOUND 3,KEY,V(4),8:RETURN
P 1680 V0$=CHR$(0):S0$=V0$:S0$(1001)=V0$:V0$(1001)=V0$:V0$(2)=V0$:S0$(2)=S0$:RETURN
N 1690 V1$=CHR$(0):S1$=V1$:S1$(1001)=V1$:V1$(1001)=V1$:V1$(2)=V1$:S1$(2)=S1$:RETURN
K 1700 V2$=CHR$(0):S2$=V2$:S2$(1001)=V2$:V2$(1001)=V2$:V2$(2)=V2$:S2$(2)=S2$:RETURN
K 1710 V3$=CHR$(0):S3$=V3$:S3$(1001)=V3$:V3$(1001)=V3$:V3$(2)=V3$:S3$(2)=S3$:RETURN
K 1720 DATA 0,0,0,35,33,37,40,37,42,45,42,47,47,45,50,53,50,57,60,57,64,64,60,68
Z 1730 DATA 72,68,76,81,76,85,91,85,96,96,91,102,108,102,114,121,114,128,128,121,136,144,136,153,162,153,173
S 1740 DATA 182,173,193,193,182,204,217,204,230,243,230,255,255,255,255
A 1750 DATA 14,3,5,12,19,11,10,9,0,8,0,0,1,2,0,0,21,18,13,17,15,4,20,6,16,7
```

HCM

HOME COMPUTER magazine SHOPPING LIST

BACK ISSUES OF HOME COMPUTER MAGAZINE & MEDIA

	U.S. Surface	CANADA	FOREIGN Surface	FOREIGN Air
Magazine	3.95	5.95	5.50	7.50
Media	6.95	7.95	N/A	10.95
Magazine/Media Set	9.90	12.95	13.95	17.50

SEE PAGES 2 & 3 FOR ALL INDIVIDUAL CONTENTS

PCjr Disk Expansion Kit

Add A Second Disk Drive To Your PCjr for **$49.95** PLUS 3.50 SHIP. & HANDL.

The Kit Includes:

- Special Cable To Connect the Second Disk Drive.
- 2 Integrated Circuits.
- A Copy of HCM Vol. 4 No. 4 Containing Instructions.
- Plus An Additional Update Sheet.

—PCjr, Disk Drive, and Power Supply NOT Included—

COMPUTER ACCESSORIES

ITEM NO.	ITEM DESCIPTION	PRICE	SHIPPING 1st one	SHIPPING each add'l
DSK20	20 Blank 5¼" Disks	22.00	3.50	3.50
DSK40	40 Blank 5¼" Disks	42.00	3.50	3.50
DSK60	60 Blank 5¼" Disks	59.40	3.50	3.50
PCJ101	PCjr Disk Expansion Kit	49.95	3.50	—
ODI	ON DISK Revue Vol. 1	14.95	2.00	2.00
	DUST COVERS			
DUS997	10" Monitor	10.95	2.00	.50
DUS72	13" Monior	12.95	2.00	.50
DUS74	MX80 Printer	9.95	2.00	.50
DUS75	Cassette Recorder	4.95	2.00	.50

TEXAS INSTRUMENTS 99/4A CLOSE-OUTS

ITEM NO.	ITEM DESCIPTION	PRICE	SHIPPING 1st one	SHIPPING each add'l
TBK	Best of 99'er	19.95	3.00	3.00
BTPOOL	Best of 99'er—Book & Tape Combo	35.00	ppd./U.S.	—
BPG999	Programmer's Guide	5.00	ppd./U.S.	—
TI0001	99'er Directory	5.00	ppd./U.S.	—
FSN-01	Simon's Saucer	10.95	2.00	2.00
	DUST COVERS			
DUS73	Expansion Box	12.95	2.00	.50
DUS76	TI Console	8.95	2.00	.50
DUS77	Speech Synthesizer	3.95	2.00	.50
DUS78	Disk Box Memory	5.95	2.00	.50
DUS79	32K Memory Exp.	5.95	2.00	.50
DUS80	RS232 Interface	5.95	2.00	.50
DUS81	Disk Controller	5.95	2.00	.50
DUS82	Thermal Printer	8.95	2.00	.50
CBL11	Single Cassette Cable	4.95	1.50	1.50
CBL12	Dual Cassette Cable	4.95	1.50	1.50

Complete the Subscription & Product Order Form and enclose payment or credit card information. Mail To:

Emerald Valley Publishing Co. ● P.O. Box 70288 ● Eugene, OR 97401

Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:

1-800-828-2212 (Minimum $10 Order) In Oregon, Alaska, Hawaii Tel. (503) 485-8796

Defective media gladly exchanged. NO REFUNDS on media. Offer & Prices Subject To Change Without Notice. Satisfaction Guaranteed—or the unfilled portion of your subscription will be refunded, less the cost of any premiums you have received.

BLANK DISK MEDIA OFFER

AS LOW AS 99¢ EACH!

As a service to our readers who prefer to type-in HCM programs, we are able to offer high-quality blank diskettes at very low prices.

	20 DISKS DSK20	40 DISKS DSK40	60 DISKS DSK60
5¼" certified single-sided, double density with reinforced hub ring. Bulk packed with white sleeves and indentification labels.	$22.00	$42.00	$59.40

PLUS $3.50 SHIPPING & HANDLING

BACK ISSUES OF 99'er HOME COMPUTER MAGAZINE & MEDIA

(The Forerunner of HCM)

QTY	MAGAZINE PRICES* U.S.	Canada	Foreign‡	MAG/MEDIA SET PRICES U.S.	Canada	Foreign‡	MEDIA PRICES U.S.	Canada	Foreign‡
3	$7.95	$10.95	$10.95	$15.95	$19.95	$21.95	$10.95	$12.95	$14.95
6	$12.95	$18.95	$18.95	$29.50	$37.95	$39.95	$20.95	$22.95	$25.95
12	$21.95	$29.95	$29.95	$49.95	$59.95	$63.95	$39.95	$44.95	$47.95

‡ Magazines Shipped to Foreign Countries via Surface (Minimum Order of 3)

— Partial Contents —

Issues No. 1-5 OUT OF PRINT

Contents available in book form as "Best of 99'er"—Vol. 1.

See Home Computer Digest or Inquire

THESE 2 MAGAZINES ARE OUT OF PRINT —TAPES AND DISKS ARE STILL AVAILABLE

ISSUE #6 (Partial Contents)
• How To Produce Sound Effects • Debugging a Game Program • How to Start a User's Group • Verbose: A speech Vocabulary Expansion Aid • Color Mapping • Dynamic Manipulation of Screen Character Graphics • The Beginner's Guide to Cassette Operation With the Home Computer • Pre-School Block Letters and Data Compaction • Picking the Ponies in TI BASIC • Battle Star Space Game • 3-D Animation on the Home Computer • Programing Tips • Who is LOGO for? • Tower of Hanoi in TI LOGO • A Review of the TI Lesson-Development Software • An Interview with a Game Designer • Learning Assembly Language with a Magic Crayon • and much, much more.

NOVEMBER 1982 (Partial Contents)
• Chatting with Your Micro: Languages for the Home Computer • A Review of the Smith Corona TP-1 Daisy Wheel Printer • The Micro Jaws Arcade Game • A Knight's Tour in TI BASIC • LOGO Has Style • ASPIC: A Language for Children • A p-System Beginner's Tutorial • An Interview with a p-System Pioneer • A Mini-Memory Screen Dump to the Home Computer Printer • Up Scope!—An exciting Undersea Combat Game • Strategy for Munch Man • A Brief Encounter with a TI Hand Held Computer • 99'er Shopping Bus • A Pocket Battleship • Sub-Programs in Extended BASIC • Arcade & Adventure Game Reviews • and much, much more

DECEMBER 1982 (Partial Contents)
• Tex-Scribe: A Text Editor for the Home Computer • A Christmas Computer Carol • Managing a Mailing List the Futura Way • Parsec: The Arcade Game • Plotting With the Home Computer—Pixel by Pixel • Preventing the Situation—Oh No! Memory Full • A Colorful Tour of TI-Fest: The Home Computer Show • Santa's Workshop: The Making of a Home Computer • The Turtle Arcade: Movies & Video Games in LOGO • Controlling a BASIC Termite • The 99'er Gold Rush—An Arcade/Adventure in the Home • 99'er Digest of News & Happenings in the TI World • Plus Games, Reviews, and much, much more.

JANUARY 1983 (Partial Contents)
• Computer Assisted Instruction for the Handicapped • p-System Basics • Debugging in LOGO • The Dow-4 Gazelle Flight Simulator • Note Whiz and Pitch Master Musical Game Reviews • Learning With the PLATO Computer Library • Strategies for Adventure Gaming • Death Drones • Using the Line By-Line Assembler • Close Encounters of the Simon Kind • Electrical Engineering Education Program • Interview With an Arcade Game Designer • TI Invaders • Programming With Pascal • Cyber-Dice • News and Happenings in the Home Computer World • Arcade Game Reviews • The Thief Adventure Game • Programming Tips • and much, much more.

FEBRUARY 1983 (Partial Contents)
• Texas Instruments at the Winter Consumer Electronics Show • Home Computer Printers on Review • How to Create Math Daisies in LOGO • Vectors in LOGO • ASPIC: A Language for Teachers • The Joys of Adventuring—Part 2 • Programming Pointers with Chuck-A-Luck—Part 4 • Interview With the Voice of Parsec • Why You Need a Printer for Your Home Computer • Lifeline to Titan Space Game • Night Blockade Battleship Game • Tower of Hanoi Pocket Program • Computer Gaming Software Reviews • News of Late Developments in the World of Home Computers • and much, much more.

MARCH 1983 (Partial Contents)
• An Introduction to the TI-99/2 Basic Computer • The Hex-bus and the 4/A Connection • Making Your Own Say and Speli Game • Disabled Children Learn and Grow • Super Cataloger—A Review of a Disk Library Utility Program • TI's New CC-40 Compact Computer • Robots and Their Social Impact • Twenty Questions With Robot Rectord • The Gravity of LOGO • Joystick Jockey—An Overview of Remote Controllers • Parsec Strategy • Converting Extended BASIC to Assembly Language • Matrix Muncher • Mini Memory Disassembler Utility • Pulling the Shade on Sprites • Letters on LOGO • Tiny Tutorials • Games programs, reviews, and much, much more.

APRIL 1983 (Partial Contents)
• Computer Assisted Savings Planning to Build Your Nest Egg • Tex-Cipher Writes and Decodes Secret Messages • Crossbytes—Computer Vocabulary Crossword Puzzle • Cutting Corners On Your Food Budget Using Coupons • Introducing Financial Planning with Multiplan • The Design Philosophy of the Compact Computer • LOGO Takes On the Popular Fifteen Puzzle • Super Language—Programming Sprites in Mini Memory • Colorful Word World—Reading Readiness for Pre-schoolers • Gameware Buffet's A-Maze-ing Boa Alley Game • Giant and Dwarfs Entrapment Game • Game Reviews • Programming Tips • Money Saving Hints • and much, much more.

MAY 1983 (Partial Contents)
• A Consumer's Guide to Word Processing • Word Processing Market Basket • A Generalized Filing Program for VIPs • The Multiplan Medium Balances Your Checkbook and Budget • Activity Accountant Helps School Secretaries with Extracurricular Activities • Maximizing Your Mini Memory's 4K of RAM • Exploring Enhanced BASIC on the Compact Computer • The LOGO Tortoise Debates the BASIC Hare • A Pocket Program to Organize Data with Linked Lists • Mentally Handicapped Learners Team Up with the TI-99/4A • The Wonders of Diskette Storage • Beeline—a Multi-Screen Strategy Game • Lost Ruins—an Archeological Dig Game • 3-D Illusions with Sprites in Depth • Game reviews, Group Grapevine, and much, much more.

JUNE 1983 (Partial Contents)
• Children and Computers Make the 99/4A Connection • Tune Your Guitar with Our TI Tuning Fork • Talk to Your Computer—Voice Technology Is Here • Gameware Buffet's Eat or Be Eaten Aardvark Game • Protect Your Station in the Space Zapper Game • What Multiplan Can and Can't Do • Understanding Inputs and Outputs in Drive For Diskettes—Part 2 • Calculate Loan Schedules on the CC-40 • Go on a LOGO Vacation • Letters on LOGO • A Review of Upper Room Software's Programs for Special Learners • Construct an RS232/Joystick Interface • Group Grapevine • Shopping Bus • A Natural Language Interface for the Professional • Game Reviews • and much, much more.

JULY 1983 (Partial Contents)
• The Evolution of Home Computer Graphics Comes Alive in Graphics Grows Up • Five Data Organizers in Need Out of Sorts • TI & 99'er at the Consumer Electronics Show • WarGames: The Movie and the Book • Editing with Multiplan • The LOGO Logician Presents To Model Is to Learn • LOGO Mosaic Designs Fill the Screen • Your Speech Synthesizer as a Spelling and Foreign Language Teacher • Software for Your Low-cost Printer Port • Gameware Buffet's Treasure Island and the Colorful Switch-A-Row • A Book Review of Learn BASIC for CC-40 Users • 3-D Animation with the TMS 9918A Video Chip • Games Reviews • Group Grapevine • and much, much more.

AUGUST, 1983 (Partial Contents)
• The Home Computer Goes To Work • Bit One, Part Two at the Fashion Factory • Better Business Bar Graphs in Graphic Persuasion • An Ensemble of Assemblers • Cashflow Helps Money Management • Keystrokes for Thrifty Folks—a Review of Typewriter • Game Reviews of Cavern Quest and Starprobe 99 • Counting Fun for Preschoolers • Peripheral Vision 99 • Mean Machines and Small Potatoes • Multiplan Medium Grabs Cells into Rectangles • Turtle Text, a LOGO Word Processor • Group Grapevine • Hello, Little Brother—CC-40 Speaks to 99/4A • Plato's Progress Looks at Student Assessment • Pocket Sunrise • Public Investigator • Gameware Buffet's Jungle Jim, Success Formula, and much more!

SEPTEMBER, 1983 (Partial Contents)
• Adding on to Your Home Computer • 99'er Directory of Commercially Available Software, Accessories and Peripherals • Peripheral Vision 99 Hardware Reviews • Pocket Sunrise, Part Two in Extended BASIC • Byte Lightning—Mini Memory Plays a Game • TI-WRITER Tutorial • LOGO's Functions, Sets and Turtles • The CC-40 and 99/4A Take the Data, and RUN • PLATO's Progress Looks at Geometry Courseware and the Shape of Things to Come • Gameware Buffet's Challenge of Camelot and The Fly • Game Reviews of Nit Wit and Crime and Punishment • Group Grapevine • 99'er Hall of Fame • 99'er Digest Update on New Products, and much, much more!

OCTOBER, 1983 (Partial Contents)
• Adventure and Fantasy with Your Home Computer • Do-It-Yourself Adventure • Picoprocessor Emulator, a 4-Bit Microprocessor • Have No Fear: Assembly Language Won't Byte, Part 1 • Make Your Mark • Les Izmore and Debug Computing Cartoon • Once Upon a Tortoise Shell—a LOGO Adventure • Turtle Shell • Number Nibbler for Children • Lots of Plots on Your CC-40 and HX1000 Four-Color Printer/Plotter • Multiplan Medium Solves a Grisly Murder Mystery • Bear Hunt • Escape From Wizard's Keep in Extended BASIC • Game Reviews • Shopping Bus • and much, much more.

NOVEMBER, 1983 (Partial Contents)
• Education with your Home Computer • Five Creative Learning Activities for Children • Let's Build America • Have No Fear: Assembly Language Won't Byte, Part 2 • Squeezing the Most Out of TI BASIC • TI-WRITER Tutorial • LOGO Lexicon • Interview with Dale Osborn • TI-WRITER: At Home in the Office • PLATO's Progress • Les Izmore and Debug • The Multiplan Medium • Computer Arrested Instruction—99'er Interviews the Kids • Gameware Buffet: Taco Man and Robo Chase • Game Reviews of Jail Break and Arithmetres • Hall of Fame • 99'er Digest • and much, much more.

HISTORICAL NOTE
99'er Magazine (founded in December, 1980) was the forerunner of Home Computer Magazine.

THE BEST OF 99'er™

SUPER CLOSE-OUT SPECIAL for TI-99/4A USERS

A Giant Home Computer Compendium™ for the Texas Instruments 99/4A

The largest, most comprehensive collection of programs and articles ever assembled for the TI Home Computer

VOLUME 1

- Over 200 thoroughly tested key-in-and-RUN programs and sub-programs typeset in a grid format for maximum clarity.
- Programming instruction in 4 languages—learn to use BASIC, Extended BASIC, LOGO and Assembly Language—for everything from record keeping and money management to arcade-quality action games.
- A selection of sensational game software featuring full-color graphics, animation, and sound effects.
- Beyond the owner's manual—tips and techniques for getting the most out of your computer system.
- Computer-Assisted Instruction—The home computer becomes your private tutor.
- Page after page of innovative applications—transforming your computer into a home productivity center.

Regular (Pre-Close-out) Prices:

Best Of 99'er (Book alone)	$19.95	+ $3.00 shipping
Best Of 99'er (Tape Set alone)	$35.00	+ $2.50 shipping

SPECIAL OFFER ▶ Buy the Tape Set for ONLY $35. And Get the Book FREE + FREE SHIPPING (Canada add $3.00)

FREE BONUS WHILE SUPPLIES LAST

ORDER THE BOOK & TAPE PACKAGE AND RECEIVE A Simon's Saucer AND A 99'er Programmer's Guide **ABSOLUTELY FREE!** This Additional $18 gift IS YOURS IF YOU ACT TODAY!

PLUS

FREE BONUS WHILE SUPPLIES LAST

THE BEST OF 99'er ON TAPE

A Choice Selection of 37 Full-length Programs On 5 Quality Cassette Tapes

★ Save Typing Time and Frustrating Key-in Errors.
★ Own the Most Comprehensive Software Library for the TI-99/4A
★ Enjoy Hundreds of Hours of Exciting Computer Activity.

People Who Know the Home Computer Best™

HOME COMPUTER magazine™

AN EMERALD VALLEY BOOK

EVP

Subscription & Product Order Form

Enclose payment or credit card information & mail with completed form to:
Emerald Valley Publishing Co. ● P.O. Box 70288 ● Eugene, OR 97401
Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:
1-800-828-2212 (Minimum $10. Order) In Oregon, Alaska, Hawaii Tel. (503) 485-8796

For more information see Shopping List on inside page.
Defective media gladly exchanged. NO REFUNDS on media.
Offer & Prices Subject To Change Without Notice.

☐ Check or Money Order Enclosed **(MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK)**

Charge my ☐ VISA ☐ MasterCard Date Expires _____ Tel. No. _____

Account No. _____ Signature _____

PURCHASER — SHIPPING ADDRESS [A] *Please Print* **SHIPPING ADDRESS** [B]

Name _____
Address _____
TO INSURE FASTER DELIVERY, STREET ADDRESS IS ESSENTIAL
City _____ State _____ Zip _____

Name _____
Address _____
TO INSURE FASTER DELIVERY, STREET ADDRESS IS ESSENTIAL
City _____ State _____ Zip _____

MAGAZINE & PROGRAM SUBSCRIPTIONS: (FOR OFFICE USE ONLY)
New Mag.: _____ CODE Old Mag.: _____ CODE New Program: _____ CODE Old Program: _____ CODE

If any portion of your order is to be shipped to address "B," please indicate your preference by checking box A or B for each item.

1 ON DISK or ON TAPE
(CURRENT ISSUE)
$4.95 + $1.00 ship. & handl.
$6.95 Canada — $10.95 Foreign Airmail
(Non-purchaser, non-subscriber price: $12.95)

(Please indicate your type of computer media)

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

Volume	Issue	Price good Through
5	5	Oct. 15, 1985

AMOUNT $ SHIP TO: A ADDRESS B

2 HOME COMPUTER MAGAZINE SUBSCRIPTION: NEW ☐ RENEWAL ☐ GIFT ☐

	USA Surface	CANANDA Surface	FOREIGN Surface
1-YEAR (10 issue)	$25	$32	$46
2-YEAR (20 issue)	$45	$59	N/A
3-YEAR (30 issue)	$63	$84	N/A

(Please indicate type of computer media for your 2 premium issues)

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

Your Subscription will begin with the up-coming new issue. ☐ Please send a raincheck for the free software.

Office Use Only
Sub.:
Media:
$ SHIP TO: A ADDRESS B

3 HOME COMPUTER MAGAZINE'S ON DISK or ON TAPE PROGRAM SUBSCRIPTION

(Please mark your choice)

	USA	CANADA
8 ISSUES	$44.95	$49.95
10 ISSUES	$49.95	$54.95

(Please indicate your type of computer media)

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

$ SHIP TO: A ADDRESS B

ADDITIONAL SPACE FOR ORDERING MERCHANDISE (See Shopping List On Inside)

ITEM NO.	QTY	Description	UNIT PRICE (Each)	SHIP. & Handl. (Each)	SHIPPING INSTRUCTIONS	
M E 0 1 0 1 0 6	1	**Music & Electronics Magazine** (6-Issue Charter Subscription) Includes Bonus—M & E Cassette Tape (a $5.95 Premium Value)	$18 (U.S) $23 (CANADA)	INCLUDED	A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$
					A SHIP TO: ADDRESS B	$

BACK ISSUES: HOME COMPUTER MAGAZINE and MEDIA*

(Please mark your choice)

(Please indicate your type of computer media)

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

ISSUE	4-1	4-2	AUG 4-3	4-4	4-5	5-1	5-2	5-3	5-4	U.S. Surface	CANADA	FOREIGN Surface	FOREIGN Air	QTY.
Magazine										3.95	5.95	5.50	7.50	× =
Media										6.95	7.95	N/A	10.95	× =
Magazine/Media Set										9.90	12.95	13.95	17.50	× =

$ SHIP TO: A ADDRESS B

BACK ISSUES: 99'er HOME COMPUTER MAGAZINE and MEDIA (COVERS ONLY TI COMPUTERS—THE FORERUNNER OF HCM)

(Please mark your choice) (Minimum Order of 3)

ISSUE	1-6	NOV 2-1	DEC 2-2	JAN 2-3	FEB 2-4	MAR 2-5	APR 2-6	MAY 2-7	JUN 2-8	JUL 2-9	AUG 2-10	SEP 2-11	OCT 2-12	NOV 2-13
Magazine														
Media														
Magazine/Media Set														

QTY	MAGAZINE PRICES* U.S.	Canada	Foreign‡	MAG/MEDIA SET PRICES U.S.	Canada	Foreign‡	MEDIA PRICES U.S.	Canada	Foreign‡
3	$7.95	$10.95	$10.95	$15.95	$19.95	$21.95	$10.95	$12.95	$14.95
6	$12.95	$18.95	$18.95	$29.50	$37.95	$39.95	$20.95	$22.95	$25.95
12	$21.95	$29.95	$29.95	$49.95	$59.95	$63.95	$39.95	$44.95	$47.95

$ SHIP TO: A ADDRESS B

FOR OFFICE USE ONLY

A	B		A	B	
☐	☐	Air Mail	☐	☐	2nd Class
☐	☐	First Class	☐	☐	3rd Class
☐	☐	UPS (Standard)	☐	☐	Air Canada
☐	☐	UPS (Blue Label)	☐	☐	4th Class
☐	☐	Other _____	☐	☐	Foreign Surface

NOTES:
‡ Magazines Shipped to Foreign Countries via Surface
* Atari coverage commenced with Volume 5, No. 5.
- Merchandise shipped via UPS—requires street address.
- Satisfaction Guaranteed—or the unfilled portion of your subscription will be refunded, less the cost of any premiums you have received.
- Machines covered: Apple II family, Atari 800/800-XL/130-XE, Commodore 64/128, IBM PC/PCjr, TI-99/4A.

APO/FPO Include $3.00 Extra Per Total Merchandise Order $

SUBTOTAL $

Priority Order ADD $2.00 SPECIAL SHIPPING $

TOTAL $

ORDER "A" DATE SHIPPED: _____ BY: _____
ORDER "B" DATE SHIPPED: _____ BY: _____
BACK ORDER DATE SHIPPED: _____ BY: _____

COMMENTS: _____
ACCT: _____
DATE: _____

SPO1550885

For Apple II Family, C-64, IBM PC & PCjr Users

ON DISK Revue™ Volume 1

An Attractive Library-Storage Slipcase

- 10 Complete Programs On A Ready-To-Run Disk
- Exciting Activities For The Entire Family—At An Affordable Price
- Entertainment, Productivity & Education Plus Valuable Programming Secrets

CONTENTS

1. Boolean Brain
2. Electronic Home Secretary
3. Savings Planner
4. Snap-Calc
5. Tablut
6. Wild Kingdom
7. Musical Mystery Words
8. Market Madness
9. Tower of Hanoi
10. Missile Math

64-page Full-Color Book

Valuable Proof-of-Purchase Coupon For Gift Redemption

Disk Stored in Re-usable Protective Pouch

ONLY $14.95 + SHIPPING & HANDLING

AS LOW AS $1.83 EACH MAGAZINE!
See Order Card on Other side of this page
ABOUT $4 PER SET!

A Back-Issue/Software Bonanza of 99'er HOME COMPUTER magazine™ At Unbeatable Prices

The original 99'er Magazine and 99'er Home Computer Magazine were the forerunners of the present-day Home Computer Magazine. Each of these magazine back issues—exclusively covering the Texas Instruments TI-99/4A—is now available with your choice of either a floppy disk or a cassette tape that contains all the programs in that issue.

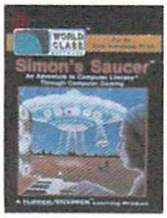
For Magazine Contents See "Shopping List" On Opposite Page

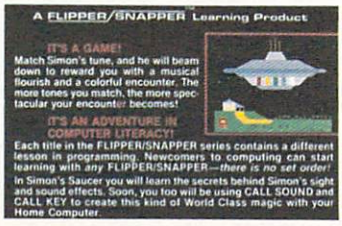
12 MAGAZINE & MEDIA SETS
SAVE UP TO $44

✶ SPECIAL MIX OR MATCH BONUS ✶
—WHILE SUPPLIES LAST—

You will receive a FREE Simon's Saucer™ package —A $12.95 Premium Value— when your order includes at least 12 items from this Back Issue offer (either 12 media, 12 magazines, or 6 of each).

- A quality, ready-to-run game on cassette tape.
- A durable and attractive ring-binder collector's case for your software library.
- A complete, easy-to-use programming lesson on a deck of colorful flip cards.





MAGAZINES OR MEDIA MAY BE ORDERED SEPARATELY
To Order, Use Order Card On Other Side Of This Page.

Offer & Prices Subject To Change Without Notice.

SENSATIONAL SOFTWARE GIVEAWAY

3 FANTASTIC OFFERS

1 YOUR CHOICE OF THIS ISSUE'S PROGRAMS AT A REMARKABLE PRICE—

ONLY $4.95!

plus $1.00 shipping & handling

Price will change to $6.95 upon becoming a "Back Issue."

FOR SUBSCRIBERS & NEWSSTAND PURCHASERS ONLY*

To participate in our monthly Software Giveaway, you need to be a bonafide purchaser of ***Home Computer Magazine***. You will receive all the programs ON TAPE™ or ON DISK™ (which have versions for your selected machine) whose listings appear in this issue.

IMPORTANT: *Please complete the order form located in the center of this magazine. Enclose it in an envelope along with $5.95 ($6.95 in Canada, $10.95 Foreign Airmail). Payment must be made by check, money order, or VISA/MasterCard. Proof of purchase (subscriber label number, sales receipt or any reasonable facsimile thereof) must also accompany this form.*

**Non-subscriber and non-purchaser price is $12.95 in the U.S.*

SORRY, WE CANNOT ACCEPT TELEPHONE ORDERS FOR THIS SERVICE.

*FREE SOFTWARE!

When Subscribing To

2 HOME COMPUTER magazine

Subscribe or Renew today, and with your paid subscription you will receive **FREE** software—**ON TAPE™** or **ON DISK™**. With a 1-year subscription to ***Home Computer Magazine*** you get **2 FREE issues** (a $11.90 Premium Value) of **ON TAPE™** or **ON DISK™**. Subscribe for 2 years and receive **4 issues** (a $23.80 Premium Value). And with a 3-year subscription we'll give you **6 full issues** (a $35.70 Premium Value) of this convenient software on cassette tape or floppy disk.

***DON'T HAVE A COMPUTER? TAKE A RAINCHECK!**

We'll give you a raincheck for the FREE software so that when you buy a computer, we will send you your choice of ON TAPE™ or ON DISK™ as a FREE BONUS as stated in this offer.

And You Save Up To 40% Off The Single-Copy Price of the Magazine!

SAVE EVEN MORE!

3 AND ENJOY THE CONVENIENCE OF A PROGRAM SUBSCRIPTION*

By subscribing to **ON TAPE™** or **ON DISK™** you will save money off the single-copy price **and** receive the same high-quality programs published in each issue of the magazine—**delivered right to your door!**

This cassette tape or floppy disk program service is the convenient, accurate, and affordable way to save hundreds of typing hours.

The Perfect Addition To Your Magazine Subscription!

- Program Subscribers Get Disks & Tapes First To Coincide With Magazine Arrival.
- Program Subscribers Get Special Updates As Available.

*DOES NOT INCLUDE MAGAZINE

TO ORDER: Please review these 3 Fantastic Offers and use the ORDER FORM located in the center of this magazine.

1 **RECEIVE ALL** the programs in this issue on magnetic media (see below). Indicate your choice of computer media on the order form and enclose $5.95 ($6.95 in Canada, $10.95 Foreign Airmail).
THIS PRICE GOOD FOR THE CURRENT ISSUE ONLY
Proof of purchase must be included with order.

2 **FREE Software** with your subscription to ***Home Computer Magazine*** for the terms listed below:

- 1-yr (10 issues) $25 ***PLUS 2 FREE*** ON TAPE or ON DISK
- 2-yr (20 issues) $45 ***PLUS 4 FREE*** ON TAPE or ON DISK
- 3-yr (30 issues) $63 ***PLUS 6 FREE*** ON TAPE or ON DISK

Canada add $7 per year; Foreign Surface add $21 for 1-yr magazine subscription. Free software offer available in U.S. & Canada only.

3 **SAVE MONEY** and time with a **Program Subscription*** to **ON TAPE™** or **ON DISK™** for the terms listed below:

- 8 ISSUES—*ONLY $44.95* "An Extension To Your 2 Free Issues of Software"
- 10 ISSUES—*ONLY $49.95* "A Full Year Of Program Convenience"

*Does not include magazine.

Canada add $5 for software subscription.
Software subscription not available in other countries at this time.

PLEASE LOCATE THIS BOX ON THE CENTER BOUND-IN ORDER FORM AND INDICATE YOUR CHOICE OF COMPUTER MEDIA

DISK VERSION						CASSETTE VERSION		
APPLE	ATARI	C-64	IBM PC	IBM PCjr	TI	ATARI	C-64	TI

Or ask for a raincheck for your FREE software.

SGV1550885

ALL PROGRAMS IN THIS MAGAZINE

ONLY $4.95* ON DISK OR TAPE!

The same high-quality Apple, Atari, Commodore, IBM, and Texas Instruments programs with type-in-and-RUN listings in this issue are now available ON DISK™ or ON TAPE™ to newsstand purchasers or subscribers of this magazine.

For only $4.95* plus $1.00 for shipping & handling, you receive all the programs for your particular brand of computer—rushed right to your door by First-Class mail.
—Truly A "Software Giveaway!"

For More Information, See ① Inside Rear Cover.

To Order, Use The Bound-In Order Form At Center Of Magazine. Offer & Prices Subject To Change Without Notice.

* Current Single-Issue Price Only — See Center Bound-In Order Form For Back-Issue Prices & Special Program Subscription Offer.